Available online at www.eccomasproceedia.org
Eccomas Proceedia UNCECOMP (2021) 129-144

ECCOMAS
Proceedia

UNCECOMP 2021
4th ECCOMAS Thematic Conference on
Uncertainty Quantification in Computational Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou (eds.)
Streamed from Athens, Greece, 28 -30 June 2021

IDENTIFICATION OF UNKNOWN PARAMETERS AND PREDICTION WITH HIERARCHICAL MATRICES

A. Litvinenko[1*], R. Kriemann[2], and V. Berikov[3,4]

[1] RWTH Aachen, Aachen, Germany
e-mail: litvinenko@uq.rwth-aachen.de

[2] Max Planck Institute for Mathematics in the Sciences (MiS) in Leipzig
e-mail: rok@mis.mpg.de

[3]Sobolev Institute of Mathematics, Novosibirsk, Russia
e-mail: berikov@math.nsc.ru

[4]Novosibirsk State University, Novosibirsk, Russia

Abstract. *Statistical analysis of massive datasets very often implies expensive linear algebra operations with large dense matrices. Typical tasks are an estimation of unknown parameters of the underlying statistical model and prediction of missing values. We developed the $\mathcal{H}$-MLE procedure, which solves these typical tasks. The unknown parameters can be estimated by maximizing the joint Gaussian log-likelihood function, which depends on a covariance matrix. To decrease the high computational cost, we approximate the covariance matrix in the hierarchical ($\mathcal{H}$-) matrix format, which has only a log-linear computational cost. The $\mathcal{H}$-matrix technique allows inhomogeneous covariance matrices and almost arbitrary locations. Especially, $\mathcal{H}$-matrices can be applied in cases when the matrices under consideration are dense and unstructured.*

For validation purposes, we implemented three machine learning methods: the kNN, random forest, and deep neural network. The best results (for the given datasets) were obtained by the kNN method with three or seven neighbors depending on the dataset. The results computed with the $\mathcal{H}$-MLE method were compared with the results obtained by the kNN method.

The developed $\mathcal{H}$-matrix code and all datasets are freely available online.

Keywords: Computational statistics, parameter inference, prediction, hierarchical matrix, data analysis, Matérn covariance, random field, spatial statistics

ISSN:2623-3339 © 2021 The Authors. Published by Eccomas Proceedia.
Peer-review under responsibility of the organizing committee of UNCECOMP 2021.
doi: 10.7712/120221.8027.19161

Contents

1 Introduction **2**

2 $\mathcal{H}$-matrix approximation of covariance matrices and the log-likelihood **4**

3 Prediction Errors **5**

4 Machine learning methods to make predictions **5**

5 Numerical results, obtained by the $\mathcal{H}$-MLE method **6**

5.1 Tests 1a and 1b: Parameter identification and prediction, 8 datasets with $n = 90,000 + 10,000$ 7

5.2 Test-2a: Parameter identification and prediction, $n = 90,000 + 10,000$ 8

5.3 Test-2b: Parameter identification and prediction, $n = 900,000 + 100,000$ . . . 9

6 Numerical results, obtained by the machine learning methods **11**

7 Conclusion **12**

1 Introduction

The number of measurements that should be statistically analyzed increases from year to year. The involved statistical methods often contain intensive operations with large dense matrices. In case these measurements are distributed irregularly across the given domain, the efficient algorithms like the Fast Fourier Transformation and similar are not applicable. Therefore, new efficient methods are needed.

To make these expensive computations possible, we suggest to use the hierarchical matrix ($\mathcal{H}$-matrix) technique [19, 18, 29, 26]. We will demonstrate how to solve statistical inference and prediction tasks appearing very often in spatial statistics. This work is an extension of our previous research [32, 31]. The first novelty is that we simultaneously identify four unknown parameters and not three. This new parameter is the regularization term - the nugget τ^2. Another novelty is the new research on how well we can make statistical predictions with $\mathcal{H}$-matrix approximations. Finally, we compare our identified parameters and predicted values with the actual results and with results obtained by other methods. We summarize the strong and weak sides of the $\mathcal{H}$-matrix technique.

Assumptions. Let $(\mathbf{s}_1, \ldots, \mathbf{s}_n)$ be the set of locations. We model the set of measurements as a realization from a stationary Gaussian spatial random field. Specifically, we let $\mathbf{Z} = \{Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_n)\}^\top$, where $Z(\mathbf{s})$ is a Gaussian random field indexed by a spatial location $\mathbf{s} \in \mathbb{R}^d$, $d = 2, 3$. Then, we assume that $\mathbf{Z}$ has zero mean and a stationary parametric covariance function $C(\mathbf{h}; \boldsymbol{\theta}) = \mathrm{cov}\{Z(\mathbf{s}), Z(\mathbf{s} + \mathbf{h})\}$, where $\mathbf{h} \in \mathbb{R}^d$ is a spatial lag vector and $\boldsymbol{\theta} \in \mathbb{R}^4$ the unknown parameter vector of interest. Statistical inferences about $\boldsymbol{\theta}$ are often based on the Gaussian log-likelihood function:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{C}(\boldsymbol{\theta})| - \frac{1}{2}\mathbf{Z}^\top \mathbf{C}(\boldsymbol{\theta})^{-1}\mathbf{Z}, \tag{1}$$

where the covariance matrix $\mathbf{C}(\boldsymbol{\theta})$ has entries $C(\mathbf{s}_i - \mathbf{s}_j; \boldsymbol{\theta})$, $i, j = 1, \ldots, n$. The maximum likelihood estimator of $\boldsymbol{\theta}$ is the value $\widehat{\boldsymbol{\theta}}$ that maximizes (1). When the sample size n is large, the evaluation of (1) becomes challenging. Indeed, the storage of the n-by-n covariance matrix $\mathbf{C}$

requires $\mathcal{O}(n^2)$ units of memory. Computation of the inverse and log-determinant of $\mathbf{C}(\boldsymbol{\theta})$ cost $\mathcal{O}(n^3)$ FLOPs. Hence, parallel and scalable methods that can reduce this high cost are needed.

Similar works for the case when measurements are located on a rectangular grid can be resolved via the fast Fourier transformation (FFT) method [47, 9, 16, 45, 11] with the computing cost $\mathcal{O}(n \log n)$. However, the FFT method does not work for data measured at irregularly spaced locations or requires expensive, non-trivial modifications.

Other recent ideas include the low-tensor rank methods [30, 36, 22], covariance tapering [14, 24, 42, 43], likelihood approximations [44, 12], Gaussian Markov random-field approximations [13], Vecchia framework [46, 23], the nearest-neighbor Gaussian process models [10], the low-rank update [40], multiresolution Gaussian process models [37], equivalent kriging [27], and Bayesian-like approach [34, 35].

An $\mathcal{H}$-matrix approximation of covariance matrices was done in [28, 41, 4, 20, 3, 6, 39]. The inverse of the covariance matrix was approximated in [2, 3, 5]. The $\mathcal{H}$-matrix technique for the parameter estimation was proposed in [3, 2]. There are many implementations of $\mathcal{H}$-matrices exist: HLIB (`http://www.hlib.org/`), $\mathcal{H}^2$ (`https://github.com/H2Lib`), HLIBPro (`https://www.hlibpro.com/`), and some others. In this work, we are using the HLIBPro library. For extended details, we refer to our earlier works [31, 32]. The data, which we used in this work were generated in the ExaGeoStat library [1] (https://github.com/ecrc/exageostat) without using $\mathcal{H}$-matrices.

Matérn covariance functions: We consider the Matérn family [33], which has gained widespread interest in recent years [17]. The Matérn covariance depends only on the distance $\mathbf{h} := \|\mathbf{s}-\mathbf{s}'\|$, where $\mathbf{s}$ and $\mathbf{s}'$ are any two spatial locations:

$$C(\mathbf{h};\boldsymbol{\theta}) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}\left(\frac{h}{\ell}\right)^{\nu} K_\nu\left(\frac{h}{\ell}\right) + \tau^2\mathbf{I}, \tag{2}$$

with parameters $\boldsymbol{\theta} = (\sigma, \ell, \nu, \tau)^\top$. Here σ^2 is the variance, τ^2 the nugget, $\nu > 0$ controls the smoothness of the random field, with larger values of ν corresponding to smoother fields, and $\ell > 0$ the spatial range parameter that measures how quickly the correlation of the random field decays with distance. A larger ℓ corresponds to a faster decay. $\mathcal{K}_\nu$ denotes the modified Bessel function of the second kind of order ν.

Prediction: Estimating the unknown parameters $\boldsymbol{\theta}$ is only an intermediate step. Once it is done, the estimation $\widehat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}$ is used for prediction at new locations. Let $I_1 := (\mathbf{s}_1, \dots, \mathbf{s}_n)$ be locations with known values $\mathbf{Z}_1$, and $I_2 = (\mathbf{s}_{n+1}, \dots, \mathbf{s}_{n+m})$ be the new locations with unknown values $\mathbf{Z}_2 = \{Z(\mathbf{s}_{n+1}), \dots, Z(\mathbf{s}_{n+m})\}^\top$ to be predicted. Here $(Z(\mathbf{s}_1), \dots, Z(\mathbf{s}_n), Z(\mathbf{s}_{n+1}, \dots, Z(\mathbf{s}_{n+m}))^\top$ is a Gaussian random field indexed by spatial locations with indices from the index set (I_1, I_2). We assume that vector $(\mathbf{Z}_1, \mathbf{Z}_2)$ is zero mean and has a stationary parametric covariance function. After discretisation we can get the following block covariance matrix

$$\begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{bmatrix}, \tag{3}$$

where $\mathbf{C}_{11} \in \mathbb{R}^{n_1 \times n_1}$, $\mathbf{C}_{12} \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{C}_{21} \in \mathbb{R}^{n_2 \times n_1}$, and $\mathbf{C}_{22} \in \mathbb{R}^{n_2 \times n_2}$. Now, the unknown vector $\mathbf{Z}_2$ can be computed by the following formula [8]

$$\mathbf{Z}_2 = \mathbf{C}_{21}\mathbf{C}_{11}^{-1}\mathbf{Z}_1. \tag{4}$$

We can also say that $\mathbf{Z}_2$ has the conditional distribution with the mean value $\mathbf{C}_{21}\mathbf{C}_{11}^{-1}\mathbf{Z}_1$ and the covariance matrix $\mathbf{C}_{22} - \mathbf{C}_{21}\mathbf{C}_{11}^{-1}\mathbf{C}_{12}$.

2 $\mathcal{H}$-matrix approximation of covariance matrices and the log-likelihood

The $\mathcal{H}$-matrix technique is defined as a recursive partitioning of a given matrix into sub-blocks. The majority of these sub-blocks are approximated by low-rank matrices on the fly (without computing any dense sub-matrices). And only a minor number of sub-block are calculated as dense matrices without any approximation. Details about block partitioning and heuristic algorithms used for low-rank approximation are not so trivial. Therefore, we skip them here and refer to [26, 32, 31].

The $\mathcal{H}$-matrix approximation error depends on the type of the covariance matrix, its smoothness, covariance length, computational geometry, nugget, and the dimensionality of the problem. For some matrices, the problem may become ill-posed since even tiny perturbations in the covariance matrix $\mathbf{C}(\boldsymbol{\theta})$ may result in considerable perturbations in the log-determinant and the log-likelihood. The usage of $\tau^2\mathbf{I}$ regularisation helps partially to resolve this issue.

Storage and complexity. We let $\mathbf{C}(\boldsymbol{\theta}) \in \mathbb{R}^{n\times n}$ be approximated by an $\mathcal{H}$-matrix $\widetilde{\mathbf{C}}(\boldsymbol{\theta};k)$ or $\widetilde{\mathbf{C}}(\boldsymbol{\theta};\varepsilon)$. In the first case we fix the maximal rank k in each sub-block (the approximation accuracy will vary from sub-block to sub-block). In the second case we fix the accuracy ε in each sub-block (the ranks of sub-blocks will vary). The $\mathcal{H}$-Cholesky decomposition of $\widetilde{\mathbf{C}}(\boldsymbol{\theta};k)$ costs $\mathcal{O}(k^2 n\log^2 n)$. The solution of the linear system $\widetilde{\mathbf{L}}(\boldsymbol{\theta};k)\mathbf{v}(\boldsymbol{\theta}) = \mathbf{Z}$ costs $\mathcal{O}(k^2 n\log^2 n)$. The log-determinant $\log|\widetilde{\mathbf{C}}(\boldsymbol{\theta};k)| = 2\sum_{i=1}^n \log\{\widetilde{L}_{ii}(\boldsymbol{\theta};k)\}$ is available for free. The cost of computing the log-likelihood function $\widetilde{\mathcal{L}}(\boldsymbol{\theta};k)$ is $\mathcal{O}(k^2 n\log^2 n)$ and the cost of computing the MLE $\widehat{\boldsymbol{\theta}}$ in m iterations is $\mathcal{O}(mk^2 n\log^2 n)$.

Maximization of the log-likelihood. To maximize $\widetilde{\mathcal{L}}(\boldsymbol{\theta};k) \approx \mathcal{L}(\boldsymbol{\theta})$ we use the Brent-Dekker method [7, 38]. It is implemented in the GNU Scientific library `https://www.gnu.org/software/gsl/`. The Brent-Dekker algorithm first uses the fast-converging secant method or inverse quadratic interpolation to maximize $\widetilde{\mathcal{L}}(\boldsymbol{\theta};\cdot)$. If those do not work, then it returns to the more robust bisection method. In the following we will call this optimization procedure $\mathcal{H}$-MLE. It iteratively computes parameter $\boldsymbol{\theta}$ where the maximum of $\widetilde{\mathcal{L}}(\boldsymbol{\theta};\cdot)$ is achieved.

Additionally to the $\mathcal{H}$-Cholesky factorisation $\mathbf{C}(\boldsymbol{\theta}) = \mathbf{L}(\boldsymbol{\theta})\mathbf{L}(\boldsymbol{\theta})^\top$, we implemented a more stable factorisation $\mathbf{L}(\boldsymbol{\theta})\mathbf{D}(\boldsymbol{\theta})\mathbf{L}^\top(\boldsymbol{\theta})$, which avoids extracting square roots of diagonal elements. Both factorizations are connected via $\mathbf{LDL}^\top = (\mathbf{LD}^{1/2})(\mathbf{LD}^{1/2})^\top$. Very small negative diagonal elements can appear due to, e.g., the rounding off error.

The computation of $\widehat{\boldsymbol{\theta}}$ depends on the number of iterations in the optimization algorithm and the used threshold (10^{-4} in our experiments). The maximal number of iterations we used was 400. We may need more depending on the initial guess and the threshold. The running times are listed in Table 4.

$\mathcal{H}$-matrix approximation error analysis. For multiple numerical tests we refer to our earlier works [32, 31, 26]. There the reader can find numerical errors for $\mathbf{C}$, the Cholesky factor $\mathbf{L}$, and the log-likelihood $\mathcal{L}$. The $\mathcal{H}$-matrix approximation accuracy of the Cholesky factor and the inverse depends on the condition number of $\mathbf{C}$. The prediction accuracy can be estimated as follows

$$\begin{aligned}
\|\mathbf{Z}_2 - \widetilde{\mathbf{Z}}_2\| &= \|\mathbf{C}_{21}\mathbf{C}_{11}^{-1}\mathbf{Z}_1 - \widetilde{\mathbf{C}}_{21}\widetilde{\mathbf{C}}_{11}^{-1}\mathbf{Z}_1\| \\
&= \|\mathbf{C}_{21}\mathbf{C}_{11}^{-1}\mathbf{Z}_1 - \widetilde{\mathbf{C}}_{21}\mathbf{C}_{11}^{-1}\mathbf{Z}_1 + \widetilde{\mathbf{C}}_{21}\mathbf{C}_{11}^{-1}\mathbf{Z}_1 - \widetilde{\mathbf{C}}_{21}\widetilde{\mathbf{C}}_{11}^{-1}\mathbf{Z}_1\| \\
&\le \|\mathbf{C}_{21}\mathbf{C}_{11}^{-1}\mathbf{Z}_1 - \widetilde{\mathbf{C}}_{21}\mathbf{C}_{11}^{-1}\mathbf{Z}_1\| + \|\widetilde{\mathbf{C}}_{21}\mathbf{C}_{11}^{-1}\mathbf{Z}_1 - \widetilde{\mathbf{C}}_{21}\widetilde{\mathbf{C}}_{11}^{-1}\mathbf{Z}_1\| \\
&\le \|\mathbf{C}_{21} - \widetilde{\mathbf{C}}_{21}\| \cdot \|\mathbf{C}_{11}^{-1}\| \cdot \|\mathbf{Z}_1\| + \|\widetilde{\mathbf{C}}_{21}\| \cdot \|\mathbf{C}_{11}^{-1} - \widetilde{\mathbf{C}}_{11}^{-1}\| \cdot \|\mathbf{Z}_1\|
\end{aligned}$$

Now we see that the quality of the prediction depends on the quality of the $\mathcal{H}$-matrix approximation of matrices $\mathbf{C}_{21}$ and $\mathbf{C}_{11}^{-1}$, i.e. the norms $\|\mathbf{C}_{21} - \widetilde{\mathbf{C}}_{21}\|$ and $\|\mathbf{C}_{11}^{-1} - \widetilde{\mathbf{C}}_{11}^{-1}\|$. In our earlier works [32, 31, 26], we demonstrated the error decay for $\mathbf{C}$ and $\mathbf{C}^{-1}$.

3 Prediction Errors

We used the Mean Loss Efficiency (MLOE), the Mean Misspecification of the Mean Square Error (MMOM), and the Root Mean Square Error (RMSE) as in the 2021 KAUST Competition on Spatial Statistics for Large Datasets [21]:

$$\text{MLOE} := \frac{1}{M}\sum_{j=1}^{M}\left(\frac{\mathbb{E}_t\left((\hat{Z}_a(\mathbf{s}_j) - Z(\mathbf{s}_j))^2\right)}{\mathbb{E}_t\left((\hat{Z}_t(\mathbf{s}_j) - Z(\mathbf{s}_j))^2\right)} - 1\right), \tag{5}$$

$$\text{MMOM} := \frac{1}{M}\sum_{j=1}^{M}\left(\frac{\mathbb{E}_a\left((\hat{Z}_a(\mathbf{s}_j) - Z(\mathbf{s}_j))^2\right)}{\mathbb{E}_t\left((\hat{Z}_a(\mathbf{s}_j) - Z(\mathbf{s}_j))^2\right)} - 1\right). \tag{6}$$

Here $(\mathbf{s}_1, \ldots, \mathbf{s}_M) := \mathcal{J}$ is a fixed subset of $M < n$ randomly-chosen locations. For numerical purposes M was chosen to be equal 1000. $\hat{Z}_t(\mathbf{s}_j)$ and $\hat{Z}_a(\mathbf{s}_j)$ are respectively kriging prediction at $\mathbf{s}_j$ using the true and approximated model (plugging in the true parameters and estimated parameters in the covariance function), and $\mathbb{E}_t(\cdot)$ and $\mathbb{E}_a(\cdot)$ are respectively the expectation using the true and approximated model. We refer to [21] for more details.

MLOE gives us an understanding of the average loss of prediction efficiency when the approximated model is used to predict instead of the true model. MMOM presents the average misspecification of the mean square error when calculated under the approximated model.

The RMSE error was used to evaluate the prediction accuracy

$$\text{RMSE} = \sqrt{\frac{1}{n_t}\sum_{i=1}^{n_t}\left(\hat{Z}(\mathbf{s}_i) - Z(\mathbf{s}_i)\right)^2}, \tag{7}$$

where $\hat{Z}(\mathbf{s}_i)$ and $Z(\mathbf{s}_i)$ are respectively the predicted and true realization values at the location $\mathbf{s}_i$ in the testing dataset, and n_t is the total number of locations in the testing dataset.

4 Machine learning methods to make predictions

Machine learning is aimed at building a model of data automatically from the observations. The obtained predictions can be considered as a baseline for comparison with other forecasting methods in which some additional information on the studied process is used. In the following, we tried three methods:

k-nearest neighbours (kNN): This method belongs to classical non-parametric family of statistical machine learning methods and follows a simple idea: for each data point x for which one needs to predict its output $\hat{y}$, find its k nearest neighbors $x_1, \ldots, x_k$ with respect to some metrics, and set $\hat{y} = \frac{1}{k}\sum_{i=1}^{k} y_i$, where y_i is the observed value of the response for point x_i. The value of k should be determined in the best way, for example, by cross-validation procedure or using an independent test sample for error estimation.

Random Forest (RF): Random Forest (RF) is another popular machine learning method in which a large number of decision (or regression) trees are generated independently on random sub-samples of data. The final decision for x is calculated over the ensemble of trees by averaging the predicted outcomes. The method is theoretically well-substantiated and gives state-of-the-art results in many practical tasks, especially in the presence of many irrelevant features describing the observed data.

Deep Neural Network (DNN): Methods of this broad class are based on the artificial neural network paradigm, which models the functioning of neurons in the brain. In our study, we use a fully connected neural network (FCNN) which includes several fully connected layers, i.e., connecting each neuron in a layer to every neuron in the next layer (see an example in Figure 1). Mathematically speaking, the input feature vector transformation performed with each layer can be presented as a matrix-vector multiplication, where the matrix elements are neuron connection weights. Each layer is followed by a non-linear activation unit. FCNN training procedure consists of finding neurons' connection weights for which the quality metric takes the best value (usually by gradient descent technique).

Each ML method needs a fine-tuning stage to optimize its hyperparameters or architecture.

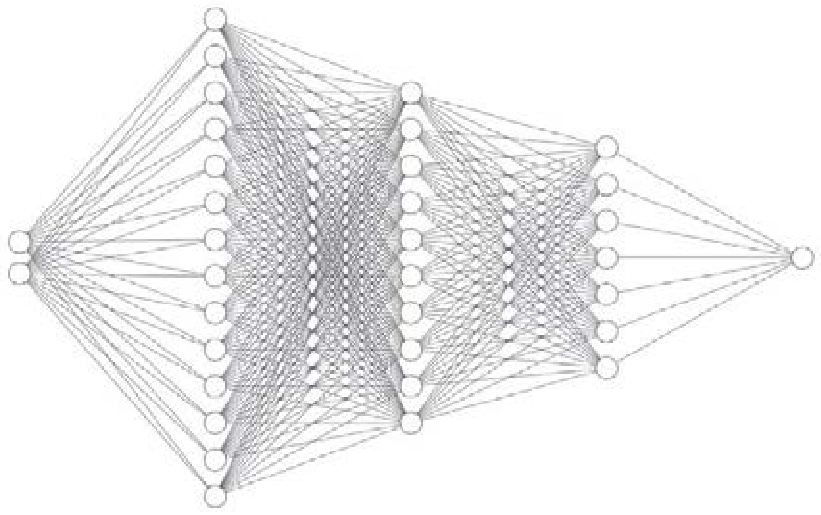

Figure 1: Example of FCNN architecture. The input layer consists of two neurons (input feature dimensionality), and the output layer consists of one neuron (predicted feature dimensionality). Three hidden layers with different numbers of neurons are presented.

The best value of k and the distance metric should be determined for kNN. For RF, one needs to set the number of trees in the ensemble, tree complexity, and splitting criterion. For FCNN, one needs to optimize the number of layers and neurons in each layer. For some other DNN (kNN and RF) parameters, we use default Matlab, Scikit-learn, or Tensorflow settings. For other hyperparameters (such as k for kNN, the number of trees for RF, or the number of layers and hidden units for FCNN), we minimized the root mean squared error (RMSE) metric using a validation sample repeatedly obtained by random sub-sampling of data in the proportion 1:9. We used a candidate set of k values in the interval $\{1, \dots, 20\}$, number of trees in the interval $\{100, \dots, 150\}$ and examined some variants of FCNN architecture for different number of hidden layers in the interval [3, 10] (having 50-100 neurons in each hidden layer).

5 Numerical results, obtained by the $\mathcal{H}$-MLE method

Datasets: For all tests below we used datasets from the statistical competition [21]. Three hidden layers with different numbers of neurons The spatial domain is the unit square $\mathcal{D} := [0.0, 1.0] \times [0.0, 1.0]$. Each dataset includes 90% of training samples and 10% testing samples, where prediction should be made.

In the following, we will perform the following numerical tests: 1a, 1b, 2a, and 2b.

Tests 1a, 1b, and 2a contain $100,000$ locations, and Test 2b contains $1,000,000$ locations. The training datasets and datasets for predictions are taken from [21].

Hardware: For the $\mathcal{H}$-MLE method we used a parallel cluster with two Intel Xeon Gold 6144 processors. Each processor has 8 cores (16 threads) with 3.5GHz and 384GB RAM in total.

Software: All $\mathcal{H}$-MLE numerical results are reproducible. We invite the reader to install HLIBPro-2.9 (from `www.hlibpro.com`), download our code from `https://github.com/litvinen/large_random_fields.git` and play with it.

Parameters identified in Test 1a are used for the prediction in Test 1b. Parameters identified in Tests 2a and 2b are used for prediction in Tests 2a and 2b, respectively.

After we identified all parameters and did all predictions, we uploaded all these data to the competition webpage[21] and the organisers of that competition computed for us the approximation errors. These errors are listed in Tables 1 and 5.

5.1 Tests 1a and 1b: Parameter identification and prediction, 8 datasets with $n = 90,000 + 10,000$

In the Test-1a, there are 16 given datasets from different zero-mean stationary isotropic Gaussian random fields with a Matérn covariance. The training dataset consists of $90,000$ randomly distributed locations and associated observations at these locations. The task is to infer four unknown parameters of the Matérn covariance function shown in (2) for each dataset.

To avoid negative intermediate values for these parameters, in the following we assume that:

$$\sigma = \frac{2.0}{1.1^{\sigma_0}}, \quad \ell = \frac{1.0}{1.5^{\ell_0}}, \quad \nu = \frac{1.0}{1.2^{\nu_0}}, \quad \tau = \frac{1.0}{2.0^{\tau_0}},$$

where σ_0, ℓ_0, ν_0, τ_0 the new parameters to be identified by the optimization algorithm. As the initial guess, we took $(\sigma_0, \ell_0, \nu_0, \tau_0) = (2, 2, 1, 15)$. If we saw that we were wrong with these values (too many iterations were needed), we rerun the optimization algorithm with some new values. The advantage of this "log"-representation is that the auxiliary values σ_0, ℓ_0, ν_0, τ_0 are allowed to take negative values, whereas σ, ℓ, ν, τ not. Negative values may appear during iterations in the MLE optimization procedure.

Table 1 contains 8 solutions for 8 given datasets [21]. The 1st column contains the dataset index, columns 2,3,4 and 5 contain values of $(\sigma^2, \ell, \nu, \tau^2)$ respectively. The column 6,7,8, and 9 contain the true values $(\hat{\sigma}^2, \hat{\ell}, \hat{\nu}, \hat{\tau}^2)$ respectively. The columns 10 and 11 contain the MLOE and MMOM errors. The 12th columns contains the RMSE error (as defined in Eq. 7). One can see that in some rows the estimated parameter values are very close to the true values, but in some not. The reason is that the derivative of the log-likelihood function at the point $(\sigma, \ell, \nu, \tau)$ is almost zero (is equal to our threshold 10^{-4}), and our optimization algorithm indicates this point as the maximum. To improve the estimate, we should iterate longer. Later, in Test-1b, the estimated parameters are used for the prediction, and one can see that our predictions are reasonable.

One can see that for some datasets (e.g., 4,5,7), the MLOE and MMOM errors are large. We would not say that the $\mathcal{H}$-matrix method failed since the optimization algorithm's accuracy to compute the MLE estimate was 10^{-4}. We think that the problem is ill-posed and contains multiple solutions, i.e., there are many points $\boldsymbol{\theta}$ where the derivative of $\mathcal{L}$ is almost zero. Here "almost" means smaller than 10^{-4}.

dataset	σ^2	ℓ	ν	τ^2	$\hat{\sigma}^2$	$\hat{\ell}$	$\hat{\nu}$	$\hat{\tau}^2$	MLOE	MMOM	RMSE
1	0.29	0.0106	2.471	2.5e-14	1.5	0.0175	2.3	0	2.2e-2	4.8e-1	4e-3
2	1.762	0.0223	1.501	1.1e-14	1.5	0.0211	1.5	0	1.8e-4	8.8e-2	2.4e-2
3	1.478	0.0305,	0.600	1.0e-10	1.5	0.031	0.6	0	2.0e-6	8.0e-3	0.23
4	1.09	0.0176,	1.522	7.0e-14	1.5	0.0526	2.3	0	1.8	3566	5.6e-4
5	0.95	0.0781	0.714	1.3e-13	1.5	0.0632	1.5	0	2.2e-1	100	5.4e-3
6	1.32	0.0826	0.601	1.22e-27	1.5	0.0928	0.6	0	5.4e-4	5.3e-2	0.12
7	2.38	0.4370	0.795	2.47e-8	1.5	0.1686	1.5	0	2.5e-1	164	2e-3
8	1.2	0.2043	0.601	4.1e-17	1.5	0.2475	0.6	0	2.8e-3	6.4e-2	6.6e-2

Table 1: $\mathcal{H}$-MLE method. Comparison of the identified (columns 2-5) and true parameters (columns 6-9).

Equation 4 was used to do prediction at the new $10,000$ locations. Figure 2 (left and right) visualize datasets 4 and 7 (see the 4th and 7th rows in Table 1), where the estimated parameters are far away from the true values (see also the last three columns in Table 1). As we can see $90,000$ given measurements (yellow points) and $10,000$ predicted (blue points) for both datasets are very good aligned.

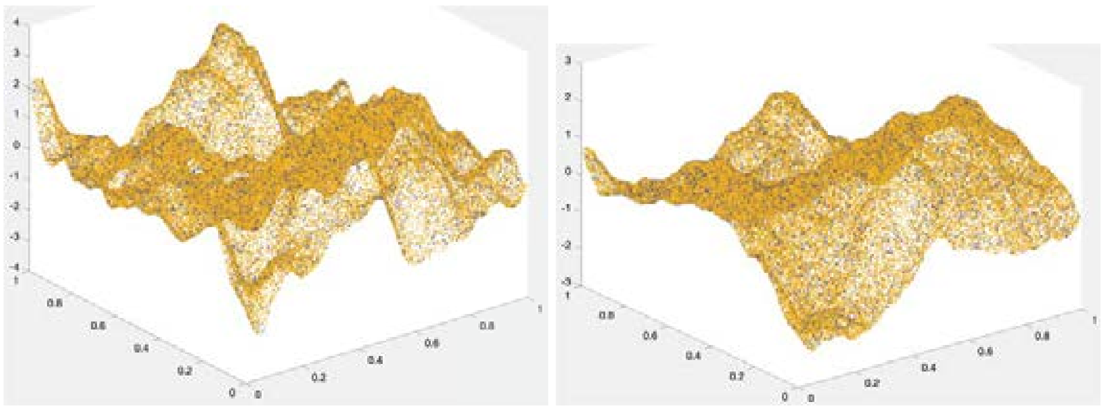

Figure 2: Test-1b, datasets 4 and 7: Prediction obtained by the $\mathcal{H}$-MLE method. The yellow points at $90,000$ locations were used for training and the blue points were predicted in $10,000$ locations. Although the identified parameters for these datasets were far away from the true values (see rows 4 and 7 in Table 1), one can still observe a very good alignment of yellow and blue points.

5.2 Test-2a: Parameter identification and prediction, $n = 90,000 + 10,000$

There are two datasets in this experiment. Each dataset contains $90,000$ measurements to identify unknown parameters (training of the statistical model). Later this model is used to predict unknown values in new $10,000$ locations. The identified parameters for both datasets are listed in Table 2. The columns 2,3,4,5 contain the values $(\sigma^2, \ell, \nu, \tau^2)$ respectively, the 6th column the value $\mathcal{L}(\sigma^2, \ell, \nu, \tau^2)$, and columns 7,8,9,10 contain the true values $(\hat{\sigma}^2, \hat{\ell}, \hat{\nu}, \hat{\tau}^2)$ respectively. These true values were obtained from organisators after the competition [21] finished.

Parameters ν and τ were identified well, but σ^2 and ℓ not. We see two possible reasons for this. The first reason is that the true model was not Gaussian. The second reason is the insufficient threshold 10^{-4} in the MLE optimization algorithm. Initially, the organizers did not provide any additional information about the utilized model. Here, we actually tested how the Gaussian model approximates the Tukey g-and-h random model [48]. Meaning, that both datasets in Task-2a were univariate non-Gaussian spatial datasets, which were generated by the Tukey g-and-h random fields. These fields generalize Gaussian random fields $Z(\mathbf{s})$. The Tukey g-and-h random process $T(\mathbf{s})$ is defined by marginal transformation at each location s as follows:

$$T(\mathbf{s}) := \xi + \omega \cdot \frac{\exp(g \cdot Z(\mathbf{s})) - 1}{g} \cdot \exp\left(\frac{hZ^2(\mathbf{s})}{2}\right), \tag{8}$$

where ξ and ω are the location and scale parameters. The parameter g defines the skewness and $h \geq 0$ the tail-heaviness. Two different pairs of (g, h) were chosen, which simulate medium and strong deviation from Gaussian random fields. These parameters ξ, ω, g, h used for generating both datasets are listed in Table 2.

data	σ^2	ℓ	ν	τ^2	$\mathcal{L}$	$\hat{\sigma}^2$	$\hat{\ell}$	$\hat{\nu}$	$\hat{\tau}^2$	ξ	ω	g	h
1	7.7	0.07	1.037	$1.6e-15$	$9.6e+4$	1	0.1	1	0	1	2	0.2	0.2
2	31	0.047	1.066	$4.0e-14$	$0.5e+4$	1	0.1	1	0	1	2	0.5	0.3

Table 2: $\mathcal{H}$-MLE method. Comparison of the obtained parameter values with the true values for Test-2a, $n = 90,000$.

Figures 3 (left and right) show predictions obtained by the $\mathcal{H}$-MLE method. The yellow points at $90,000$ locations were used for training and the blue points were predicted in $10,000$ new locations. One can see a very good alignment of yellow and blue points on both pictures.

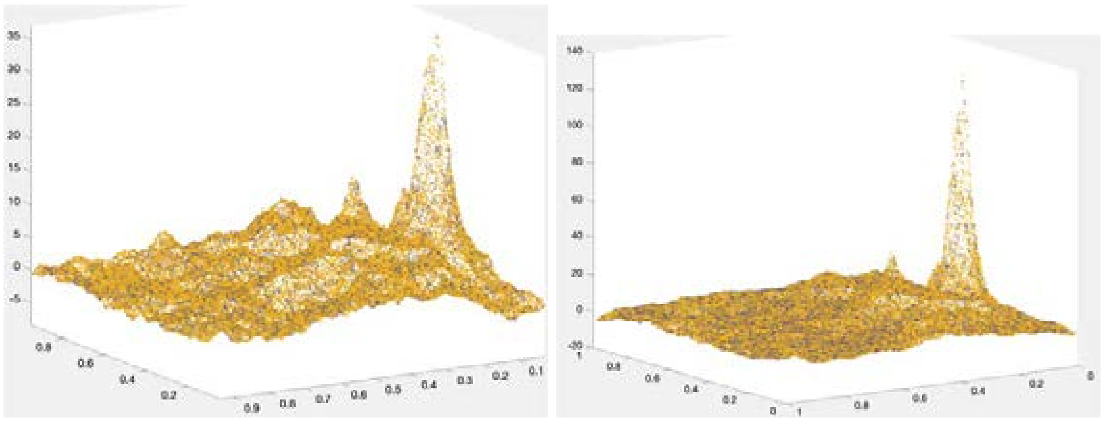

Figure 3: Test-2a, datasets 1 and 2: Prediction obtained by the $\mathcal{H}$-MLE method. The yellow points at $90,000$ locations were used for training and the blue points were predicted in $10,000$ locations. One can see a very good alignment of yellow and blue points on both figures.

5.3 Test-2b: Parameter identification and prediction, $n = 900,000 + 100,000$

There are two datasets in this experiment. Each dataset contains $900,000$ measurements to identify unknown parameters (training of the statistical model). Later this model is used to predict unknown values in new $100,000$ locations. The identified parameters are listed in Table 3 for two datasets. The columns 2,3,4,5 contain the values $(\sigma^2, \ell, \nu, \tau^2)$ respectively, the 6th column the value $\mathcal{L}(\sigma^2, \ell, \nu, \tau^2)$, and columns 7,8,9,10 contain the true values $(\hat{\sigma}^2, \hat{\ell}, \hat{\nu}, \hat{\tau}^2)$ respectively. These true values were obtained from organisators after the competition [21] finished.

Table 4 summarizes the computational times for the $\mathcal{H}$-MLE method. We note that this computing time varies a lot for the parameter identification task because it depends on the

data	σ^2	ℓ	ν	τ^2	$\mathcal{L}$	$\hat{\sigma}^2$	$\hat{\ell}$	$\hat{\nu}$	$\hat{\tau}^2$
1	3.72	1.143830	0.94636	4.4e-3	3.6e+5	1.5	0.0632	1.5	0
2	0.92	0.012496	1.30867	8.5e-9	1.8e+6	1	0.1	1	0

Table 3: $\mathcal{H}$-MLE method. Comparison of the obtained parameter values with the true values for Test-2b, $n = 900,000$.

initial guess in the optimization algorithm. If the initial guess lies very close to the (unknown) true value of $\boldsymbol{\theta}$, then only a few iterations are needed. If not, then a few hundred iterations and the computing time may increase by a factor of 10.

Datasets/ Tasks	1a param. infer.	1b pred.	2a param. infer., pred.	2b param. infer., pred.
$\mathcal{H}$-MLE comp. time (sec.)	360-3600	60	180-3600, 120	3600-36000, 600

Table 4: Computing time for the parameter inference and for the prediction, $\mathcal{H}$-MLE method.

Figures 4 (left and right) show predictions obtained by the $\mathcal{H}$-MLE method. The yellow points at 900.000 locations were used for training and the blue points were predicted at 100.000 new locations. One can see a very good alignment of yellow and blue points (on the right) and slightly different values on the left.

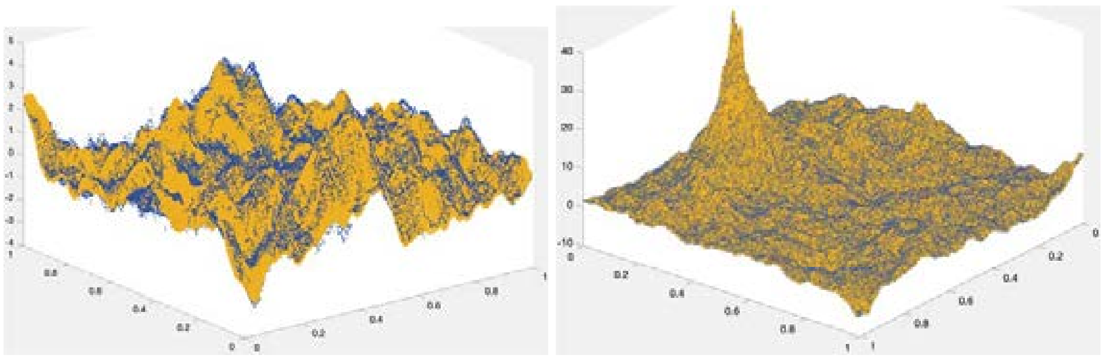

Figure 4: Test-2b, datasets 1 and 2: Prediction obtained by the $\mathcal{H}$-MLE method. The yellow points at 900.000 locations were used for training and the blue points were predicted in 100.000 locations. One can see a very good alignment of yellow and blue points (on the right) and slightly different values on the left.

6 Numerical results, obtained by the machine learning methods

To run the kNN method we used a usual notebook with Intel i5-9300 CPU, 2.40GHz and 8 GB RAM. We perform Monte-Carlo simulations, in which we repeatedly split the given dataset on training and testing subsamples and average the obtained prediction error estimates over all runs. In our experiments on both datasets in Test-2a, the kNN method has shown the best Monte-Carlo cross-validation results (over 100 runs) in comparison with other used ML methods. Predictions for datasets 1 and 2 from Test-2a are shown in Fig. 5.

Running the kNN method with different k, we found out that $k = 3$ is optimal for Test-2a, and $k = 7$ for Test 2b. Trying different numbers of trees in the random forest method, we defined that an ensemble of 120 regression trees is optimal. Further, we have designed the FCNN architecture with 7 hidden layers with 100 neurons in each layer. The number of training epochs is 500, and the batch size equals 10000. We use tanh$(\cdot)$ activation function and Adam optimizer. The average calculation time is 0.07 sec. for kNN (k-d tree was used to speed up calculations), 12 sec. for RF and 173 sec. for FCNN. Because of its efficiency, we decided to run only the kNN method for the prediction in Test-2b.

Table 5 contains the RMSE errors for all methods (defined in Eq. 7). Note that the dataset2 from Test-2b is sampled from the same random field as the dataset1 from Test-2a. The dataset1 from Test-2b is sampled from the same random field as the dataset5 from Test-1a. Remarkable is that RMSE for the dataset5 $(100.000$ locations) in Test-1a (5th row in Table 1) is equal $5.4e-3$, whereas RMSE for similar dataset1 $(1.000.000$ locations) from Test-2b is equal 0.25. We can explain this with 1) the fact that the MLE approach faces difficulties with large matrices, since the condition number of $\mathbf{C}$ is increasing, and 2) the number of needed iterations in the MLE optimization procedure increases. We did not run RF and FCNN methods on Test-2b because the computing time is much larger than for the kNN time. The kNN time for Test-2b is 1.23 sec.

Predictions for datasets 1 and 2 from Test-2b are shown in Fig. 6. The training datasets are depicted by yellow points and kNN predictions by blue points. We can see that the kNN method provides very good results.

	Test-2a		Test-2b	
dataset	1	2	1	2
$\mathcal{H}$-MLE	0.057	0.14	0.25	0.021
kNN	0.129	0.357	0.007	0.04
RF	0.226	0.607	—	—
FCNN	0.243	0.74	—	—

Table 5: Comparison of RMSE errors for $\mathcal{H}$-MLE, kNN, RF, and FCNN methods.

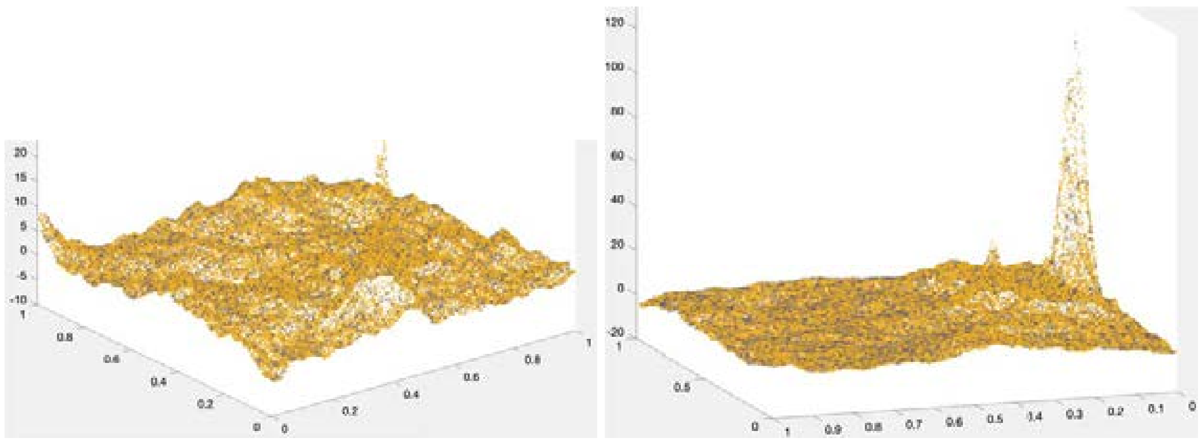

Figure 5: Test-2a, datasets 1 and 2: Prediction obtained by the kNN method. The yellow points at 90.000 locations were used for training and the blue points were predicted at 10.000 new locations. One can see a very good alignment of both.

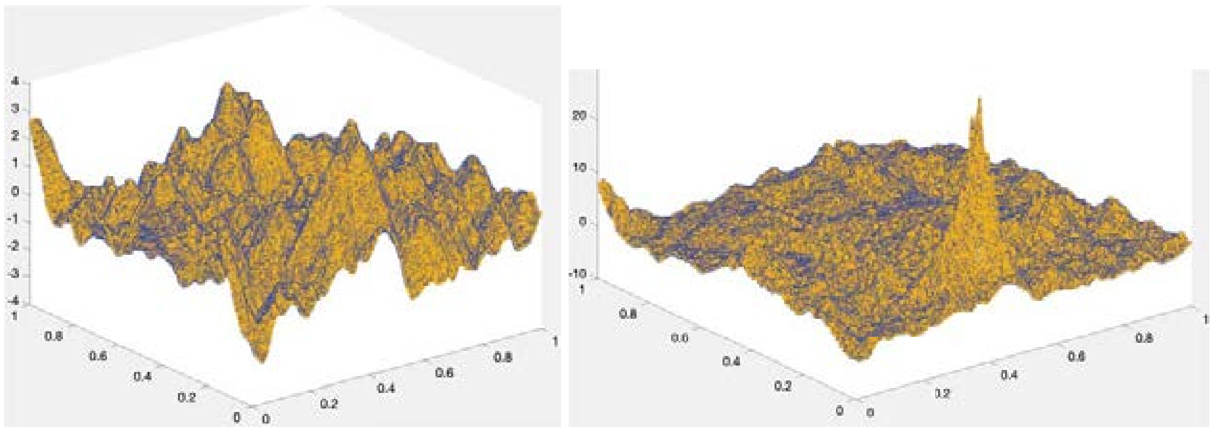

Figure 6: Test-2b, datasets 1(left) and 2(right): Prediction obtained by the kNN method. The yellow points at 900.000 locations were used for training and the blue points were predicted at 100.000 new locations. One can see a very good alignment of both.

7 Conclusion

We developed the $\mathcal{H}$-MLE procedure to estimate unknown parameters and to make statistical predictions. In order to make computations faster, we approximated the joint Gaussian log-likelihood function in the $\mathcal{H}$-matrix format. In the numerical section, we considered $8+2+2 = 12$ datasets: 8 in Tests 1a and 1b, 2 in Test-2a, and 2 in Test-2b. All datasets in Tests 1a, 1b and 2a contain $90,000$ locations for training and $10,000$ for testing (prediction). Both datasets in Test-2b contained $900,000$ locations for training and $100,000$ for prediction.

The $\mathcal{H}$-matrix technique drastically reduces the required memory and computing time, making it possible to work with larger sets of observations obtained on unstructured meshes. The main drawback of using the $\mathcal{H}$-matrix technique is that too many linear algebra operations are required to estimate just four scalar unknown parameters. For example, for some datasets with the unlikely chosen initial guess, we needed 400 iterations. On each iteration, we computed one $\mathcal{H}$-Cholesky factorization, one scalar product and solved a linear system. In total, it can take up to 8 hours on a modern parallel node for the dataset with $900,000$ locations. A possible remedy is to precompute the initial guess. It will significantly reduce the required number of iterations. This could be done, for instance, on a smaller subset of observations. Another drawback is that the $\mathcal{H}$-matrix approximation of $\mathbf{C}$ and $\mathbf{L}$ was recomputed entirely on every iteration for the new values of τ and σ. It would be a lot cheaper to add a new diagonal or scale the existing $\mathbf{C}$. It is indeed possible to modify the optimization algorithm, but the whole procedure will become more complicated. And the last drawback is that the total complexity depends on the matrix size and the number of parameters. For one to four parameters, the total computing time is acceptable, but it will be too large for five or more parameters. To tackle problems with large number of parameters we suggest to use low-rank tensor methods [30, 15, 25].

Among all implemented ML methods (kNN, random forest, deep neural network), the best results (for given datasets) were obtained by the kNN method with three or seven neighbors depending on the dataset. The results computed with the $\mathcal{H}$-MLE method were compared with the results obtained by the kNN method. For Test-2a, the $\mathcal{H}$-MLE method showed a smaller RMSE error than the kNN method, whereas, for Test-2b, the kNN method was better. To conclude, it is not surprising that our $\mathcal{H}$-MLE method worked fine on most datasets. We also understand that we can improve the $\mathcal{H}$-MLE results simply by taking a smaller threshold and more accurate $\mathcal{H}$-matrix arithmetics. What surprised us is that the well-known and straightforward kNN method performed very good and very fast. Since we did not make any theoretical comparison and compared $\mathcal{H}$-MLE and ML methods only numerically on given datasets, we can not in general conclude which method is better. We also remind that we used kNN only for the prediction.

Acknowledgment The study was carried out within the framework of the state contract of the Sobolev Institute of Mathematics (project no 0314-2019-0015). The work was partly supported by RFBR grant 19-29-01175. A. Litvinenko was supported by funding from the Alexander von Humboldt Foundation.

REFERENCES

[1] 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*, 29:2771–2784, 2018.

[2] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. OŃeil. Fast direct methods for gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):252–265, Feb 2016.

[3] S. Ambikasaran, J. Y. Li, P. K. Kitanidis, and E. Darve. Large-scale stochastic linear inversion using hierarchical matrices. *Computational Geosciences*, 17(6):913–927, 2013.

[4] J. Ballani and D. Kressner. Sparse inverse covariance estimation with hierarchical matrices. *http://sma.epfl.ch/∼anchpcommon/publications/quic_ballani_kressner_2014.pdf*, 2015.

[5] M. Bebendorf and W. Hackbusch. Existence of H-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients. *Numerische Mathematik*, 95(1):1–28, 2003.

[6] S. Börm and J. Garcke. Approximating Gaussian processes with H^2-matrices. In Joost N. Kok, Jacek Koronacki, Ramon Lopez de Mantaras, Stan Matwin, Dunja Mladen, and Andrzej Skowron, editors, *Proceedings of 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007. ECML 2007*, volume 4701, pages 42–53, 2007.

[7] R. P. Brent. Chapter 4: An algorithm with guaranteed convergence for finding a zero of a function, algorithms for minimization without derivatives. *Englewood Cliffs, NJ: Prentice-Hall*, 1973.

[8] N. Cressie and Chr. Wikle. *Statistics For Spatio-Temporal Data*, volume 465. 01 2011.

[9] R. Dahlhaus and H. Künsch. Edge effects and efficient parameter estimation for stationary random fields. *Biometrika*, 74(4):877–882, 1987.

[10] A. Datta, S. Banerjee, A. O. Finley, and A. E. Gelfand. Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111(514):800–812, 2016.

[11] C.R. Dietrich and G. N. Newsam. Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix. 18(4):1088–107, 1997.

[12] M. Fuentes. Approximate likelihood for large irregularly spaced spatial data. *Journal of the American Statistical Association*, 102:321–331, 2007.

[13] G.-A. Fuglstad, D. Simpson, F. Lindgren, and H. Rue. Does non-stationary spatial data always require non-stationary random fields? *Spatial Statistics*, 14:505–531, 2015.

[14] R. Furrer, M. G. Genton, and D. Nychka. Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, 15(3):502–523, 2006.

[15] L. Grasedyck, D. Kressner, and Chr. Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[16] J. Guinness and M. Fuentes. Circulant embedding of approximate covariances for inference from gaussian data on large lattices. *Journal of Computational and Graphical Statistics*, 26(1):88–97, 2017.

[17] P. Guttorp and T. Gneiting. Studies in the history of probability and statistics XLIX: On the Matérn correlation family. *Biometrika*, 93:989–995, 2006.

[18] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. I. Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[19] W. Hackbusch. *Hierarchical matrices: Algorithms and Analysis*, volume 49 of *Springer Series in Comp. Math.* Springer, 2015.

[20] H. Harbrecht, M. Peters, and M. Siebenmorgen. Efficient approximation of random fields for numerical applications. *Numerical Linear Algebra with Applications*, 22(4):596–617, 2015.

[21] H. Huang, S. Abdulah, M. G. Genton, Y. Sun, H. Ltaief, and D. E. Keyes. 2021 KAUST Competition on Spatial Statistics for Large Datasets, 2020. `https://cemse.kaust.edu.sa/stsds/2021-kaust-competition-spatial-statistics-large-datasets`.

[22] H. Huang and Y. Sun. Hierarchical low rank approximation of likelihoods for large spatial datasets. *Journal of Computational and Graphical Statistics*, 27(1):110–118, 2018.

[23] M. Katzfuss and J. Guinness. A general framework for Vecchia approximations of Gaussian processes. *ArXiv e-prints*, August 2017.

[24] C. G. Kaufman, M. J. Schervish, and D. W. Nychka. Covariance tapering for likelihood-based estimation in large spatial datasets. *Journal of the American Statistical Association*, 103(484):1545–1555, 2008.

[25] B. Khoromskij. *Tensor Numerical Methods in Scientific Computing*. De Gruyter, 2018.

[26] B. N Khoromskij, A. Litvinenko, and H. G. Matthies. Application of hierarchical matrices for computing the Karhunen–Loève expansion. *Computing*, 84(1-2):49–67, 2009.

[27] William Kleiber and Douglas W Nychka. Equivalent kriging. *Spatial Statistics*, 12:31–49, 2015.

[28] J. Y. Li, S. Ambikasaran, E. F. Darve, and P. K. Kitanidis. A Kalman filter powered by H2 matrices for quasi-continuous data assimilation problems. *Water Resources Research*, 50(5):3734–3749, 2014.

[29] A. Litvinenko. Application of hierarchical matrices for solving multiscale problems, 2006. PhD Dissertation, Leipzig University, `https://publications.rwth-aachen.de/record/754296/files/754296.pdf`.

[30] A. Litvinenko, D. Keyes, V. Khoromskaia, B. N. Khoromskij, and H. G. Matthies. Tucker Tensor analysis of Matern functions in spatial statistics. *Computational Methods in Applied Mathematics*, November 2018.

[31] A. Litvinenko, R. Kriemann, M. G. Genton, Y. Sun, and D. E. Keyes. Hlibcov: Parallel hierarchical matrix approximation of large covariance matrices and likelihoods with applications in parameter identification. *MethodsX*, 7:100600, 2020.

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

[33] Bertil Matérn. *Spatial Variation*, volume 36 of *Lecture Notes in Statistics*. Springer-Verlag, Berlin; New York, second edition edition, 1986.

[34] H. G. Matthies, E. Zander, O Pajonk, B. V. Rosić, and A. Litvinenko. Inverse problems in a Bayesian setting. In *Computational Methods for Solids and Fluids Multiscale Analysis, Probability Aspects and Model Reduction Editors: Ibrahimbegovic, Adnan (Ed.), ISSN: 1871-3033*, pages 245–286. Springer, 2016.

[35] H. G. Matthies, E. Zander, B. V. Rosić, and A. Litvinenko. Parameter estimation via conditional expectation: a bayesian inversion. *Advanced Modeling and Simulation in Engineering Sciences*, 3(1):24, 2016.

[36] W Nowak and A Litvinenko. Kriging and spatial design accelerated by orders of magnitude: combining low-rank covariance approximations with FFT-techniques. *Mathematical Geosciences*, 45(4):411–435, 2013.

[37] D. Nychka, S. Bandyopadhyay, D. Hammerling, F. Lindgren, and S. Sain. A multiresolution Gaussian process model for the analysis of large spatial datasets. *Journal of Computational and Graphical Statistics*, 24(2):579–599, 2015.

[38] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Section 9.3. Van Wijngaarden-Dekker-Brent Method. Numerical Recipes: The Art of Scientific Computing*, volume 3rd ed. New York: Cambridge University Press., 2007.

[39] A. Saibaba and P. Kitanidis. Efficient methods for large-scale linear inversion using a geostatistical approach. *Water Resources Research*, 48(5).

[40] A. Saibaba and P. Kitanidis. Fast computation of uncertainty quantification measures in the geostatistical approach to solve inverse problems. *Advances in Water Resources*, 82:124 – 138, 2015.

[41] A.K. Saibaba, S. Ambikasaran, J Yue Li, P. K. Kitanidis, and E.F. Darve. Application of hierarchical matrices to linear inverse problems in geostatistics. *Oil & Gas Science and Technology–Rev. IFP Energies Nouvelles*, 67(5):857–875, 2012.

[42] 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)*, 74(1):111–132, 2012.

[43] M. L. Stein. Statistical properties of covariance tapers. *Journal of Computational and Graphical Statistics*, 22(4):866–885, 2013.

[44] M. L. Stein, J. Chen, and M. Anitescu. Stochastic approximation of score functions for gaussian processes. *Ann. Appl. Stat.*, 7(2):1162–1191, 06 2013.

[45] J. R. Stroud, M. L. Stein, and S. Lysen. Bayesian and maximum likelihood estimation for gaussian processes on an incomplete lattice. *Journal of Computational and Graphical Statistics*, 26(1):108–120, 2017.

[46] A. V. Vecchia. Estimation and model identification for continuous spatial processes. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):297–312, 1988.

[47] P. Whittle. On stationary processes in the plane. *Biometrika*, 41(3-4):434–449, 1954.

[48] G. Xu and M. G. Genton. Tukey g-and-h random fields. *Journal of the American Statistical Association*, 112(519):1236–1249, 2017.