Identification of unknown parameters and prediction of missing values. Comparison of approaches.

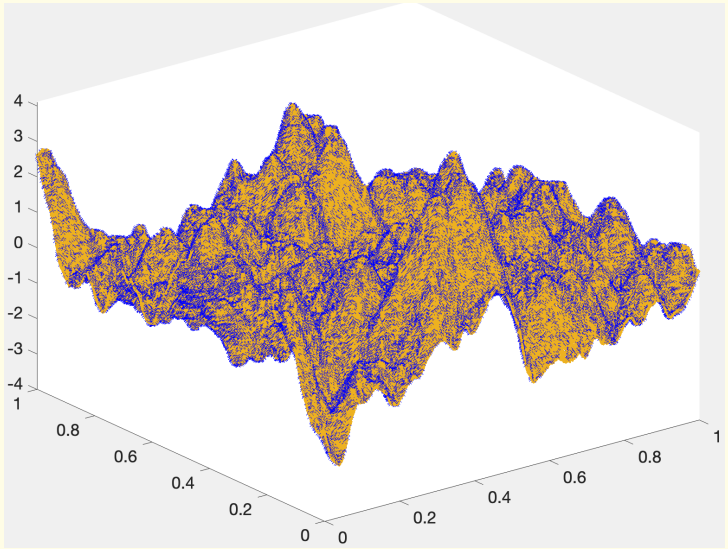
Alexander Litvinenko[1],

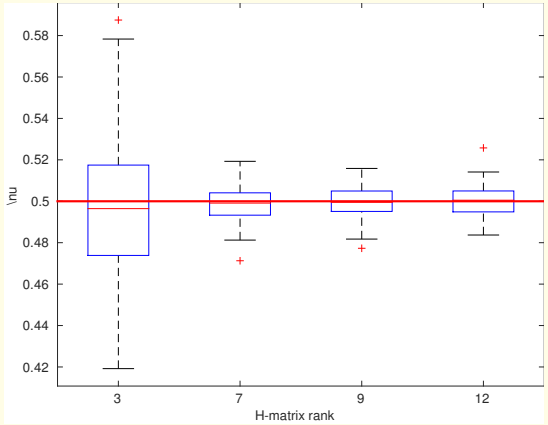
(joint work with V. Berikov[2], R. Kriemann[3] and KAUST stat. people [4]

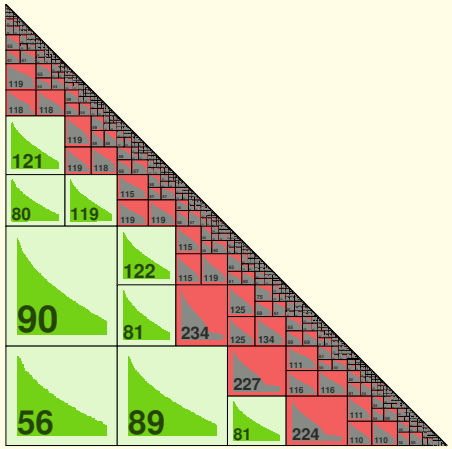
[1]RWTH Aachen, Germany, [2]Sobolev Institute of Mathematics, [3]MPI for Mathematics in the Sciences in Leipzig, Germany, [4] KAUST

Analysis of large datasets and prediction

Goals: identify unknown parameters in a large dataset and prediction
MLE deals with dense cov. matrix $\mathbf{C}$ of size $2,000,000 \times 2,000,000$





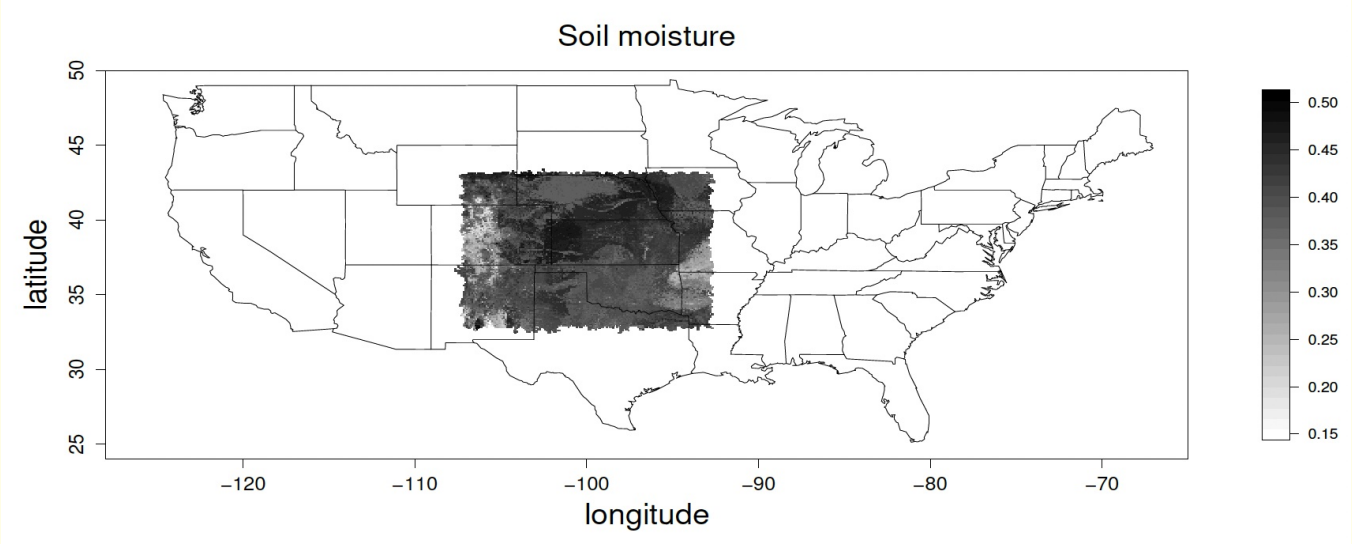


We show how to:

1. reduce storage cost from 32TB to 16 GB
2. approximate Cholesky factorisation of $\mathbf{C}$, its determinant, inverse in 8 minutes on modern desktop computer.
3. make prediction

Predict moisture, estimate covariance parameters

Grid: $1830 \times 1329 = 2,432,070$ locations with 2,153,888 observations and 278,182 missing values.



High-resolution daily soil moisture data at the top layer of the Mississippi basin, U.S.A., 01.01.2014 (Chaney et al., in review). Important for agriculture, defense.

The structure of the talk

1. Motivation: improve statistical models, data analysis, prediction
2. Identification of unknown parameters via maximizing Gaussian log-likelihood (MLE)
3. Tools: Hierarchical matrices [Hackbusch 1999]
4. Matérn covariance function, joint Gaussian log-likelihood
5. Error analysis
6. Prediction at new locations
7. Comparison with machine learning methods

Identifying unknown parameters via MLE method

Given:
Let $\mathbf{s}_1, \ldots, \mathbf{s}_n$ be locations.
$\mathbf{Z} = \{Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_n)\}^\top$, where $Z(\mathbf{s})$ is a Gaussian random field indexed by a spatial location $\mathbf{s} \in \mathbb{R}^d$, $d \geq 1$.

Assumption: $\mathbf{Z}$ has mean zero and stationary parametric covariance function, e.g. Matérn:

$$\mathbf{C}(\boldsymbol{\theta}) = \frac{2\sigma^2}{\Gamma(\nu)} \left(\frac{r}{2\ell}\right)^\nu K_\nu \left(\frac{r}{\ell}\right) + \tau^2 \mathbf{I}, \quad \boldsymbol{\theta} = (\sigma^2, \nu, \ell, \tau^2).$$

To identify: unknown parameters $\boldsymbol{\theta} := (\sigma^2, \nu, \ell, \tau^2)$.

Identifying unknown parameters via MLE method

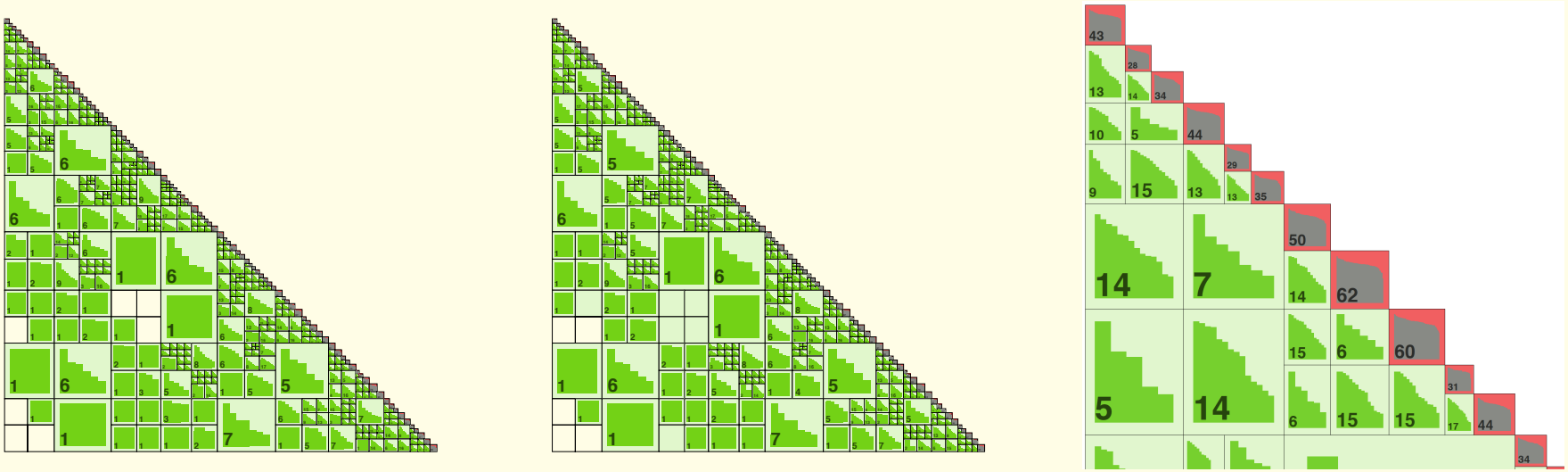
Statistical inference about $\boldsymbol{\theta}$ is then based on the Gaussian log-likelihood function:

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

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).

$\mathcal{H}$-matrix approximations of $\mathbf{C}$, $\mathbf{L}$ and $\mathbf{C}^{-1}$



$\mathcal{H}$-matrix approximations of the exponential covariance matrix (left), its hierarchical Cholesky factor $\tilde{\mathbf{L}}$ (middle), and the zoomed upper-left corner of the matrix (right), $n = 4000$, $\ell = 0.09$, $\nu = 0.5$, $\sigma^2 = 1$.

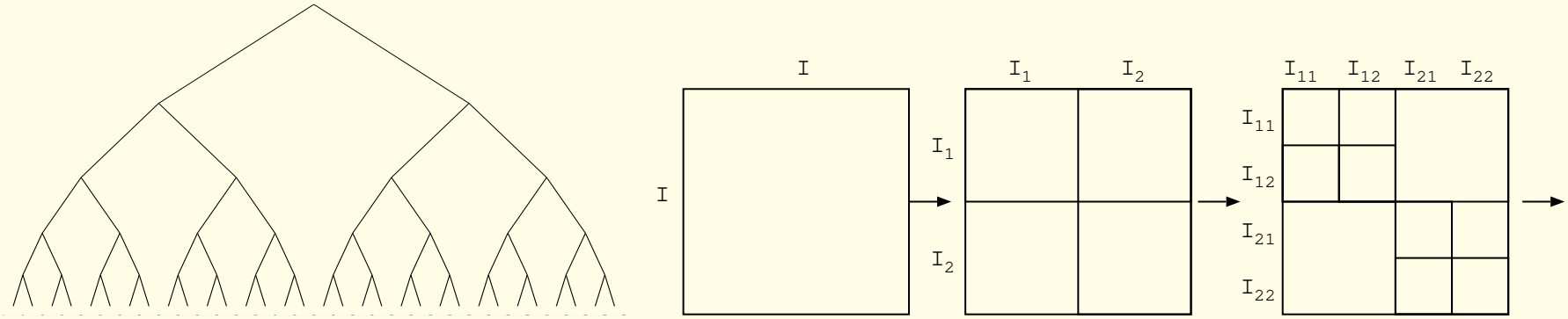
What will change after applying the $\mathcal{H}$-matrix approximation?

Approximate $\mathbf{C}$ by $\mathbf{C}^{\mathcal{H}}$

1. How $\det(\mathbf{C})$ differs from $\det(\mathbf{C}^{\mathcal{H}})$?
2. How $\mathbf{L}$ differs from $\mathbf{L}^{\mathcal{H}}$? [Mario Bebendorf et al]
3. How $\mathbf{C}^{-1}$ differs from $(\mathbf{C}^{\mathcal{H}})^{-1}$? [Mario Bebendorf et al]
4. How $\tilde{\mathcal{L}}(\theta, k)$ differs from $\mathcal{L}(\theta)$?
5. What is optimal $\mathcal{H}$-matrix rank?
6. How $\theta^{\mathcal{H}}$ differs from θ?

For theory, estimates for the rank and accuracy see works of Bebendorf, Grasedyck, Le Borne, Hackbusch,...

$\mathcal{H}$-matrix approximations of C, L and C^{-1}



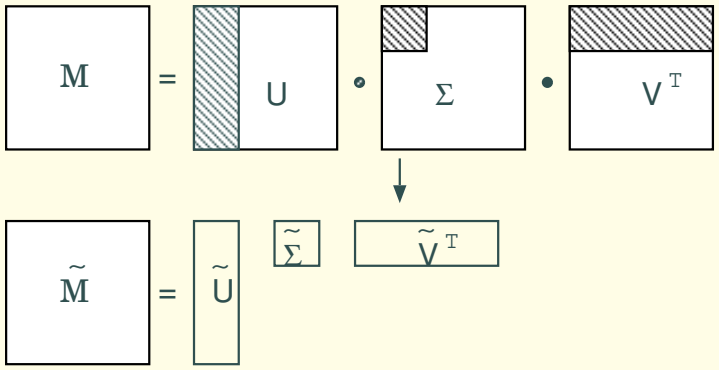
Examples of a cluster tree T_I (left) and a block cluster tree $T_{I\times I}$ (right). The decomposition of the matrix into sub-blocks is defined by $T_{I\times I}$ and the admissibility condition.

Low-rank (rank-k) matrices

How do we compute green blocks?

$M \in \mathbb{R}^{n \times m}$, $U \approx \widetilde{U} \in \mathbb{R}^{n \times k}$, $V \approx \widetilde{V} \in \mathbb{R}^{m \times k}$, $k \ll \min(n, m)$.

The storage $\widetilde{M} = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^T$ is $k(n+m)$ instead of $n \cdot m$ for M represented in the full matrix format.

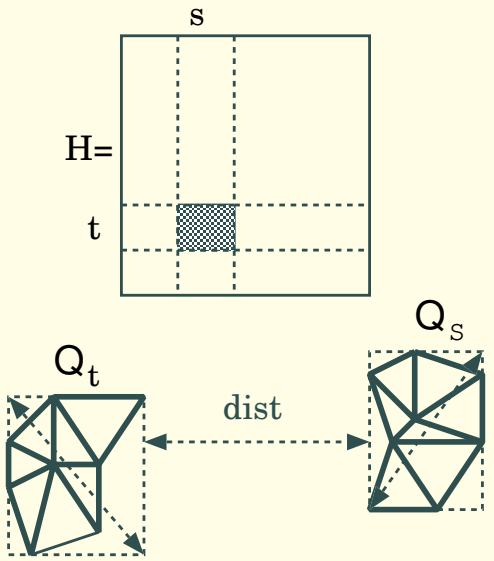


Reduced SVD, only k biggest singular values are taken.

Admissible condition

For each $(t \times s) \in T_{I\times I}$, $t, s \in T_I$, check admissibility condition
$\min\{diam(Q_t), diam(Q_s)\} \leq \eta \cdot dist(Q_t, Q_s)$.

if(adm=true) then $M|_{t\times s}$ is a rank-k matrix block
if(adm=false) then divide $M|_{t\times s}$ further or define as a dense matrix block, if small enough.



All steps: Grid $\rightarrow$ cluster tree (T_I) + admissibility condition $\rightarrow$ block cluster tree $(T_{I\times I}) \rightarrow \mathcal{H}$-matrix $\rightarrow \mathcal{H}$-matrix arithmetics.

$\mathcal{H}$-matrix storage and complexity (p-proc. on shared mem.)

Operation	Sequential Compl.	Parallel Complexity (R.Kriemann 2005)
$building(M)$	$N = \mathcal{O}(n\log n)$	$\frac{N}{p} + \mathcal{O}(\lvert V(T)\backslash\mathcal{L}(T)\rvert)$
$storage(M)$	$N = \mathcal{O}(kn\log n)$	N
Mx	$N = \mathcal{O}(kn\log n)$	$\frac{N}{p} + \frac{n}{\sqrt{p}}$
$\alpha M' \oplus \beta M''$	$N = \mathcal{O}(k^2 n\log n)$	$\frac{N}{p}$
$\alpha M' \odot M'' \oplus \beta M$	$N = \mathcal{O}(k^2 n\log^2 n)$	$\frac{N}{p} + \mathcal{O}(C_{sp}(T)\lvert V(T)\rvert)$
M^{-1}	$N = \mathcal{O}(k^2 n\log^2 n)$	$\frac{N}{p} + \mathcal{O}(nn_{min}^2)$
LU	$N = \mathcal{O}(k^2 n\log^2 n)$	N
$\mathcal{H}$-LU	$N = \mathcal{O}(k^2 n\log^2 n)$	$\frac{N}{p} + \mathcal{O}(\frac{k^2 n\log^2 n}{n^{1/d}})$

Details of the parameter identification

To maximize the log-likelihood function we use the Brent's method (combining bisection method, secant method and inverse quadratic interpolation) or any other.

1. $\mathbf{C}(\boldsymbol{\theta}) \approx \widetilde{\mathbf{C}}(\boldsymbol{\theta}, \varepsilon)$.
2. $\widetilde{\mathbf{C}}(\boldsymbol{\theta}, k) = \widetilde{\mathbf{L}}(\boldsymbol{\theta}, k)\widetilde{\mathbf{L}}(\boldsymbol{\theta}, k)^T$
3. $\mathbf{Z}^T\widetilde{\mathbf{C}}^{-1}\mathbf{Z} = \mathbf{Z}^T(\widetilde{\mathbf{L}}\widetilde{\mathbf{L}}^T)^{-1}\mathbf{Z} = \mathbf{v}^T \cdot \mathbf{v}$, where $\mathbf{v}$ is a solution of $\widetilde{\mathbf{L}}(\boldsymbol{\theta}, k)\mathbf{v}(\boldsymbol{\theta}) := \mathbf{Z}$.

$$\log\det\{\widetilde{\mathbf{C}}\} = \log\det\{\widetilde{\mathbf{L}}\widetilde{\mathbf{L}}^T\} = \log\det\{\prod_{i=1}^{n}\lambda_i^2\} = 2\sum_{i=1}^{n}\log\lambda_i,$$

$$\widetilde{\mathcal{L}}(\boldsymbol{\theta}, k) = -\frac{n}{2}\log(2\pi) - \sum_{i=1}^{n}\log\{\widetilde{\mathbf{L}}_{ii}(\theta, k)\} - \frac{1}{2}(\mathbf{v}(\boldsymbol{\theta})^T \cdot \mathbf{v}(\boldsymbol{\theta})). \quad (2)$$

Error analysis

Theorem (1)

Let $\widetilde{\mathbf{C}}$ *be an* $\mathcal{H}$*-matrix approximation of matrix* $\mathbf{C} \in \mathbb{R}^{n \times n}$ *such that*

$$\rho(\widetilde{\mathbf{C}}^{-1}\mathbf{C} - \mathbf{I}) \leq \varepsilon < 1.$$

Then

$$|log \det \mathbf{C} - log \det \widetilde{\mathbf{C}}| \leq -nlog(1 - \varepsilon), \tag{3}$$

Proof: See [Ballani, Kressner 14] and [Ipsen 05].
Remark: factor n is pessimistic and is not really observed numerically.

Error in the log-likelihood

Theorem (2)

Let $\widetilde{\mathbf{C}} \approx \mathbf{C} \in \mathbb{R}^{n\times n}$ *and* $\mathbf{Z}$ *be a vector,* $\|\mathbf{Z}\| \leq c_0$ *and* $\|\mathbf{C}^{-1}\| \leq c_1$. *Let* $\rho(\widetilde{\mathbf{C}}^{-1}\mathbf{C} - \mathbf{I}) \leq \varepsilon < 1$. *Then it holds*

$$
\begin{aligned}
|\widetilde{\mathcal{L}}(\boldsymbol{\theta}) - \mathcal{L}(\boldsymbol{\theta})| &= \frac{1}{2}(log|\mathbf{C}| - log|\widetilde{\mathbf{C}}|) + \frac{1}{2}|\mathbf{Z}^{\mathrm{T}}\left(\mathbf{C}^{-1} - \widetilde{\mathbf{C}}^{-1}\right)\mathbf{Z}| \\
&\leq -\frac{1}{2}\cdot nlog(1-\varepsilon) + \frac{1}{2}|\mathbf{Z}^{\mathrm{T}}\left(\mathbf{C}^{-1}\mathbf{C} - \widetilde{\mathbf{C}}^{-1}\mathbf{C}\right)\mathbf{C}^{-1}\mathbf{Z}| \\
&\leq -\frac{1}{2}\cdot nlog(1-\varepsilon) + \frac{1}{2}c_0^2\cdot c_1\cdot\varepsilon.
\end{aligned}
$$

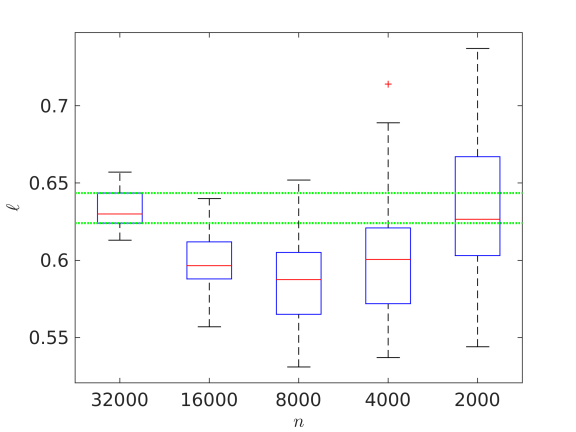
$\mathcal{H}$-matrix approximation

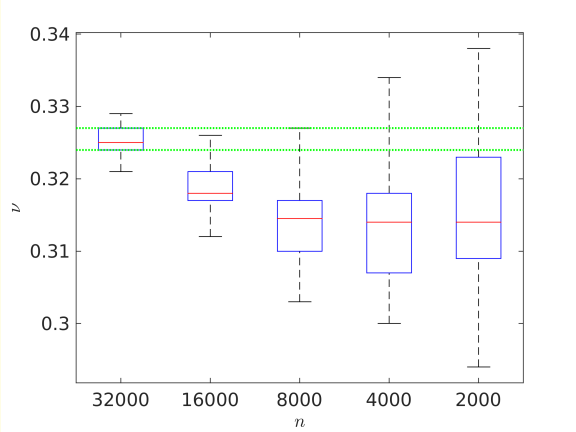
ε accuracy in each sub-block,
$n = 16641$, $\nu = 0.5$, c.r.=compression ratio.

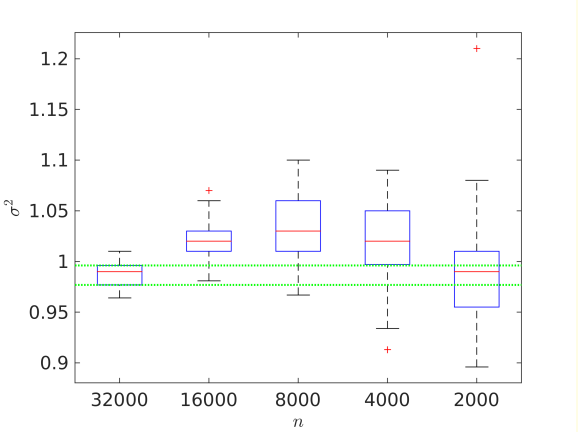
ε $\ell = 0.0334$	$\lvert\log\lvert\mathbf{C}\rvert - \log\lvert\widetilde{\mathbf{C}}\rvert\rvert$	$\lvert\frac{\log\lvert\mathbf{C}\rvert - \log\lvert\widetilde{\mathbf{C}}\rvert}{\log\lvert\widetilde{\mathbf{C}}\rvert}\rvert$	$\lVert\mathbf{C} - \widetilde{\mathbf{C}}\rVert_F$	$\frac{\lVert\mathbf{C} - \widetilde{\mathbf{C}}\rVert_2}{\lVert\mathbf{C}\rVert_2}$	$\lVert\mathbf{I} - (\widetilde{\mathbf{L}}\widetilde{\mathbf{L}}^\top)^{-1}\mathbf{C}\rVert_2$	c.r. in %
1e-1	3.2e-4	1.2e-4	7.0e-3	7.6e-3	2.9	9.16
1e-2	1.6e-6	6.0e-7	1.0e-3	6.7e-4	9.9e-2	9.4
1e-4	1.8e-9	7.0e-10	1.0e-5	7.3e-6	2.0e-3	10.2
1e-8	4.7e-13	1.8e-13	1.3e-9	6e-10	2.1e-7	12.7
$\ell = 0.2337$						
1e-4	9.8e-5	1.5e-5	8.1e-5	1.4e-5	2.5e-1	9.5
1e-8	1.45e-9	2.3e-10	1.1e-8	1.5e-9	4e-5	11.3

$\log|\mathbf{C}| = 2.63$ for $\ell = 0.0334$ and $\log|\mathbf{C}| = 6.36$ for $\ell = 0.2337$.

Boxplots for unknown parameters







Moisture data example. Boxplots for the 100 estimates of (ℓ, ν, σ_2), respectively, when $n = 32K, 16K, 8K, 4K, 2K$. $\mathcal{H}$-matrix with a fixed rank $k = 11$. Green horizontal lines denote 25% and 75% quantiles for $n = 32K$.

Time and memory for parallel $\mathcal{H}$-matrix approximation

Maximal # cores is 40, $\nu = 0.325$, $\ell = 0.64$, $\sigma^2 = 0.98$

n	$\tilde{\mathbf{C}}$			$\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top$		
	time sec.	size MB	kB/dof	time sec.	size MB	$\|\mathbf{I} - (\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top)^{-1}\tilde{\mathbf{C}}\|_2$
32.000	3.3	162	5.1	2.4	172.7	$2.4 \cdot 10^{-3}$
128.000	13.3	776	6.1	13.9	881.2	$1.1 \cdot 10^{-2}$
512.000	52.8	3420	6.7	77.6	4150	$3.5 \cdot 10^{-2}$
2.000.000	229	14790	7.4	473	18970	$1.4 \cdot 10^{-1}$

Dell Station, 20×2 cores, 128 GB RAM

Prediction

Let $\mathbf{Z} = (\mathbf{Z}_1, \mathbf{Z}_2)^\top$ has mean zero and a stationary covariance, $\mathbf{Z}_1$ - known, $\mathbf{Z}_2$ unknown.

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

We compute predicted values

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

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

2021 KAUST Competition

We participated in **2021 KAUST Competition on Spatial Statistics for Large Datasets**.

You can download the datasets and look the final results here https://cemse.kaust.edu.sa/stsds/2021-kaust-competition-spatial-statistics-large-dataset

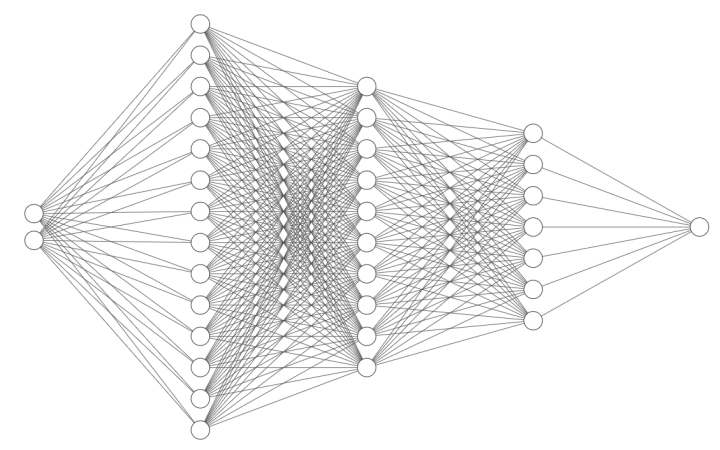
See more our details in https://www.eccomasproceedia.org/conferences/thematic-conferences/uncecomp-2021/8027

Machine learning methods to make prediction

k-nearest neighbours (kNN): for each x find its k nearest neighbors $x_1, \ldots, x_k$ with respect to some metrics, and set the prediction $\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i$, where y_i is the observed responses x_i. k is determined by cross-validation procedure. We tried $k \in \{1, \ldots, 20\}$.

Random Forest (RF): 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. We tried $\in \{100, \ldots, 150\}$ trees.

Deep Neural Network (DNN): fully connected neural network



Input layer consists of two neurons (input feature dimensionality), and output layer consists of one neuron (predicted feature dimensionality).
We examined few variants of FCNN architecture for $\{3, 4, \ldots, 10\}$ hidden layers and 50-100 neurons in each hidden layer.

Datasets:

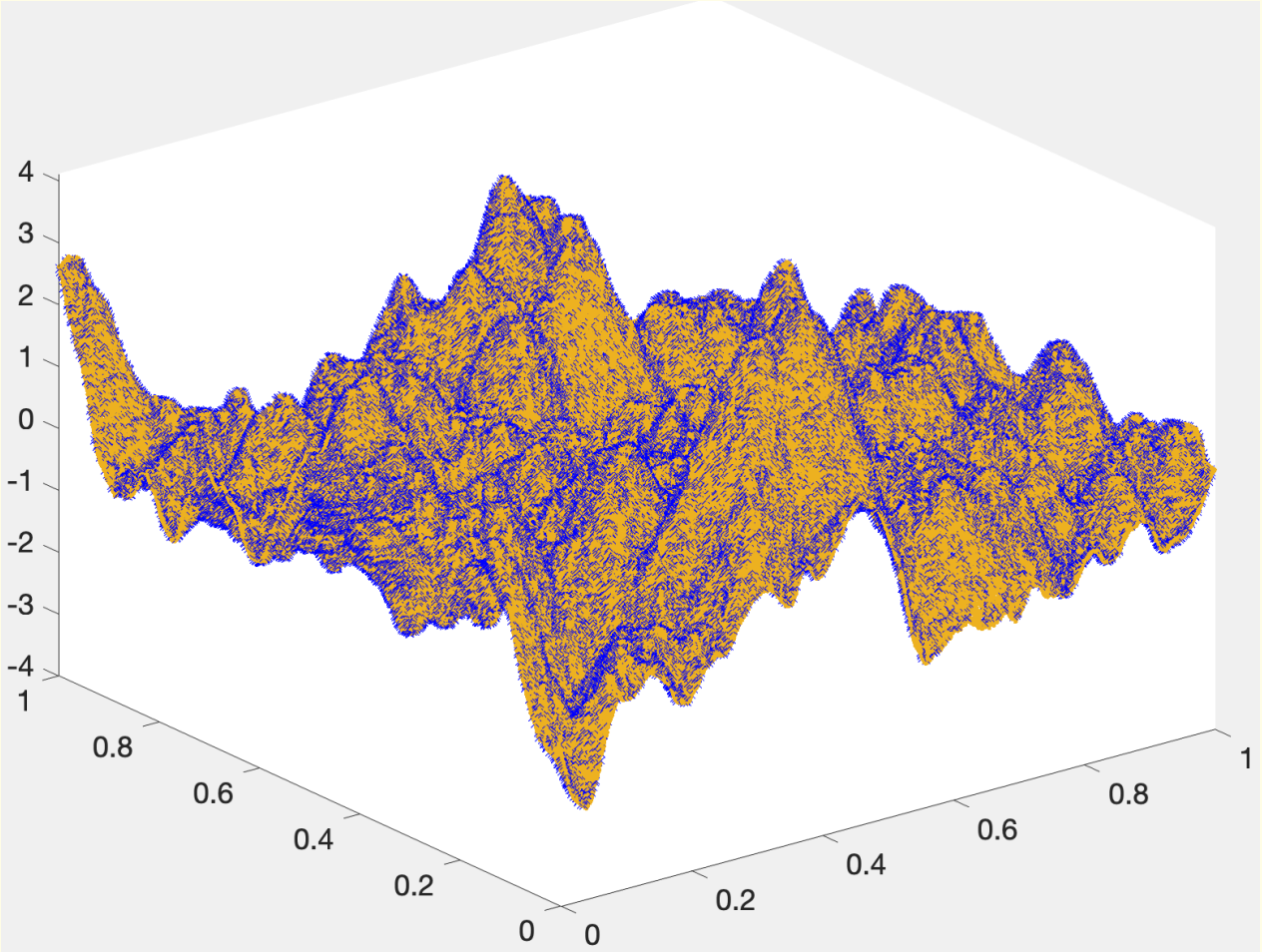
For all tests below we used datasets from the statistical competition
https://cemse.kaust.edu.sa/stsds/2021-kaust-competition-spatial-statistics-large-dataset
The spatial domain is $\mathcal{D} := [0.0, 1.0] \times [0.0, 1.0]$.
Each dataset includes 90% of training samples and 10% testing samples, where prediction should be made.

Comparison of kNN, decision forest and FCNN

kNN method showed the best MC cross-validation results (over 100 runs) in comparison with other ML methods.
We found out that $k = 3$ neighbours is optimal for Test 2a (moderate sample size 100K), and $k = 7$ for large Test 2b (1Mi samples).

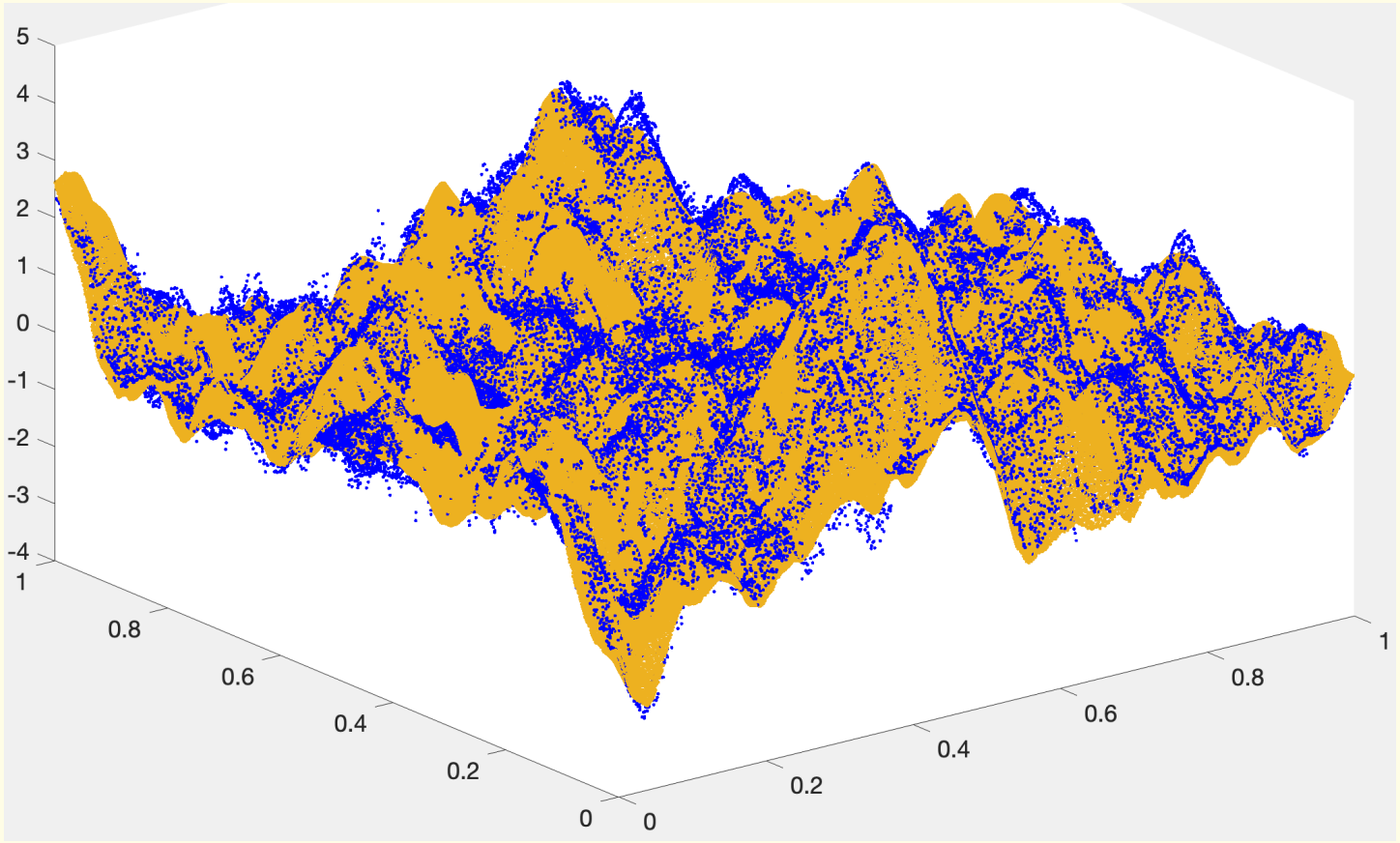
Trying different numbers of decision trees in the random forest, we defined that an ensemble of 120 regression trees is optimal.
FCNN architecture: 7 hidden layers with 100 neurons in each layer.
#training epochs = 500, batch size = 10000.
We use $\tanh(\cdot)$ activation function and Adam optimizer.
The average calculation time is
0.07 sec. for kNN,
12 sec. for RF
173 sec. for FCNN.
Because of its efficiency, we decided to run only kNN method for larger datasets.

Prediction by kNN



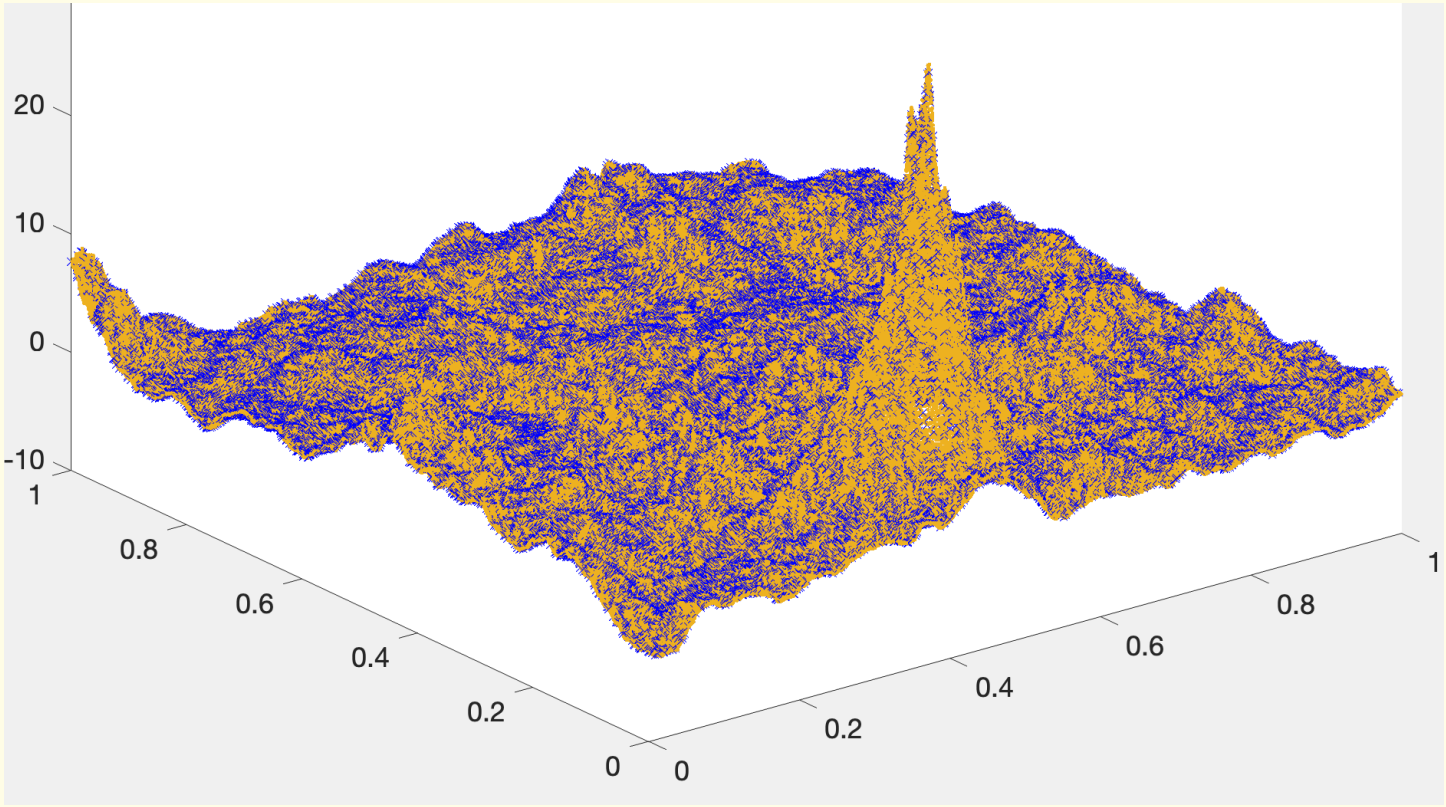
Test-2b, dataset1: The yellow points at 900.000 locations were used for training and the blue points were predicted at 100.000 new locations.

Less accurate prediction by $\mathcal{H}$-MLE method



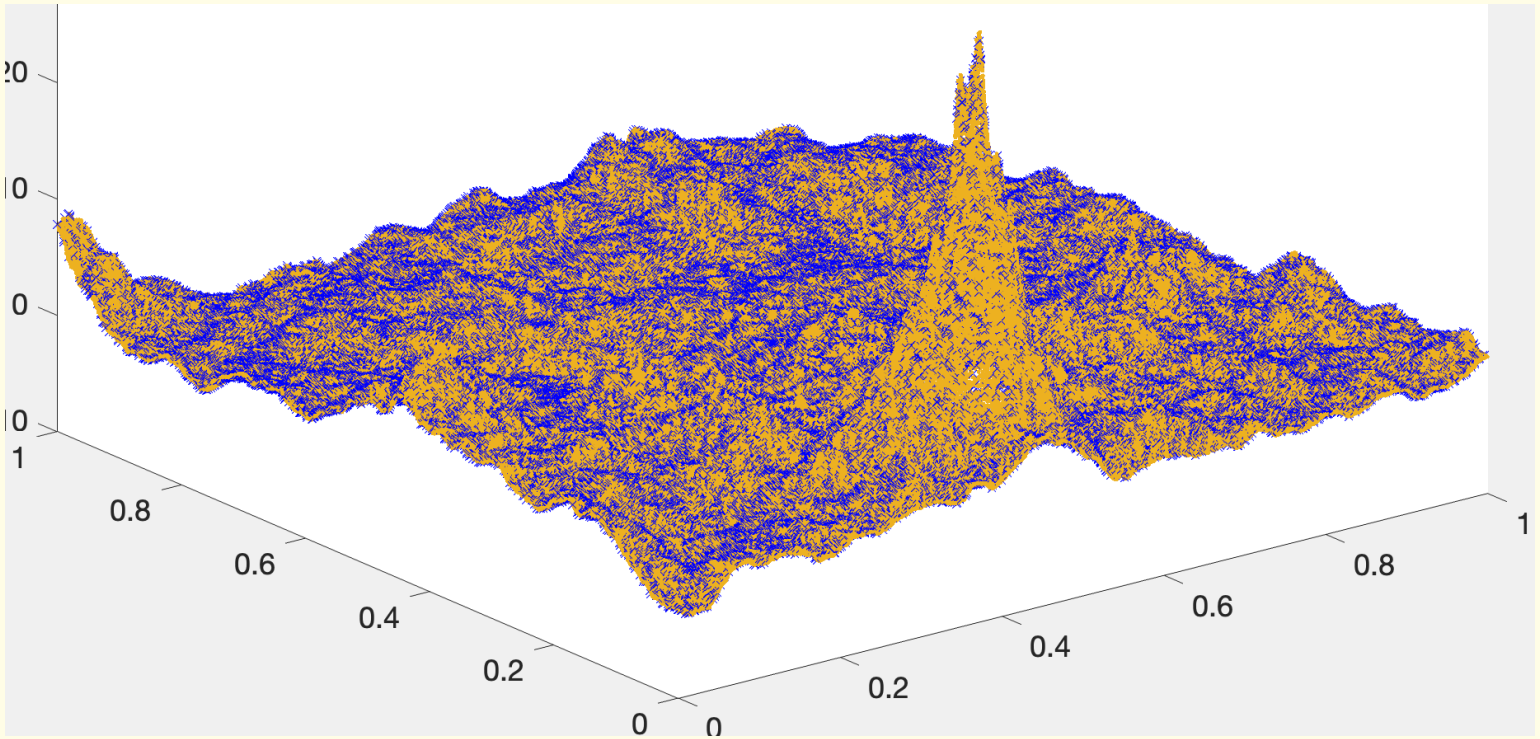
$\mathcal{H}$-MLE: Test 2b, dataset 1. The yellow points at 900.000 locations were used for training and the blue points were predicted at 100.000 new locations.

Prediction by kNN



Test-2b, dataset2: The yellow points at 900.000 locations were used for training and the blue points were predicted at 100.000 new locations.

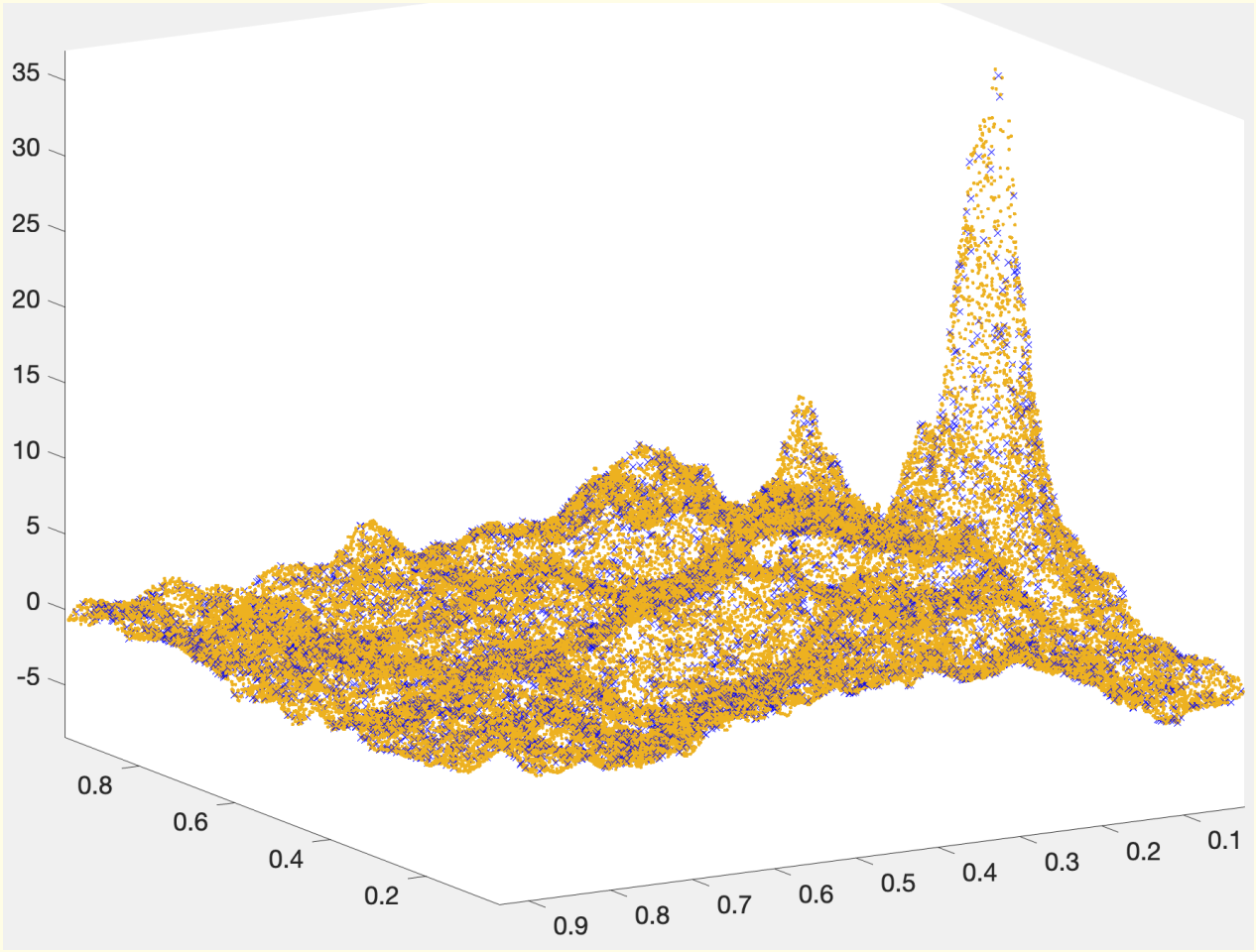
Prediction by the $\mathcal{H}$-MLE method



$\mathcal{H}$-MLE: Test 2b, dataset 2. The yellow points at 900.000 locations were used for training and the blue points were predicted at 100.000 new locations.

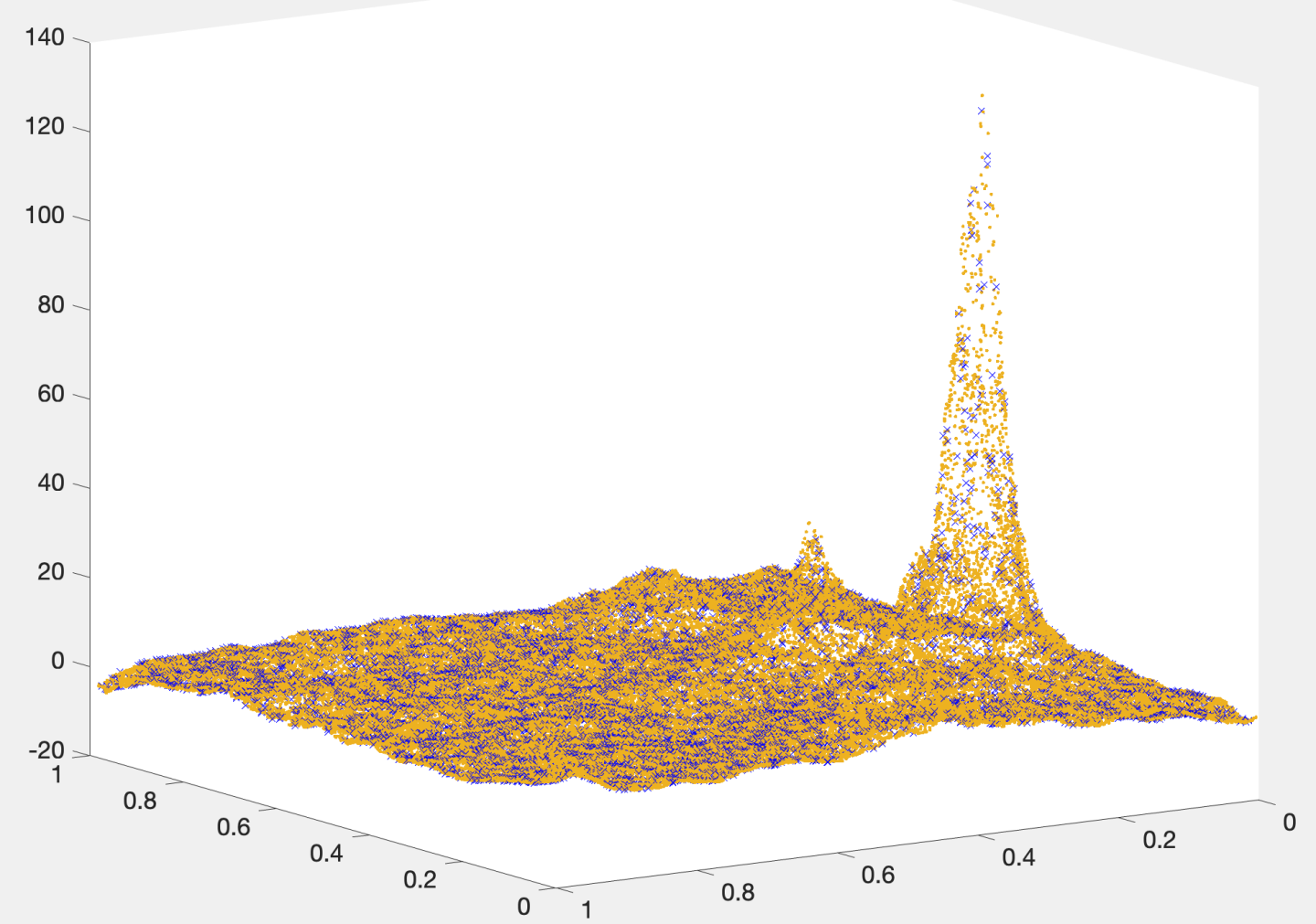
Prediction by the $\mathcal{H}$-MLE method

The yellow points at $90,000$ locations were used for training and $10,000$ values at new locations (blue points) were predicted.

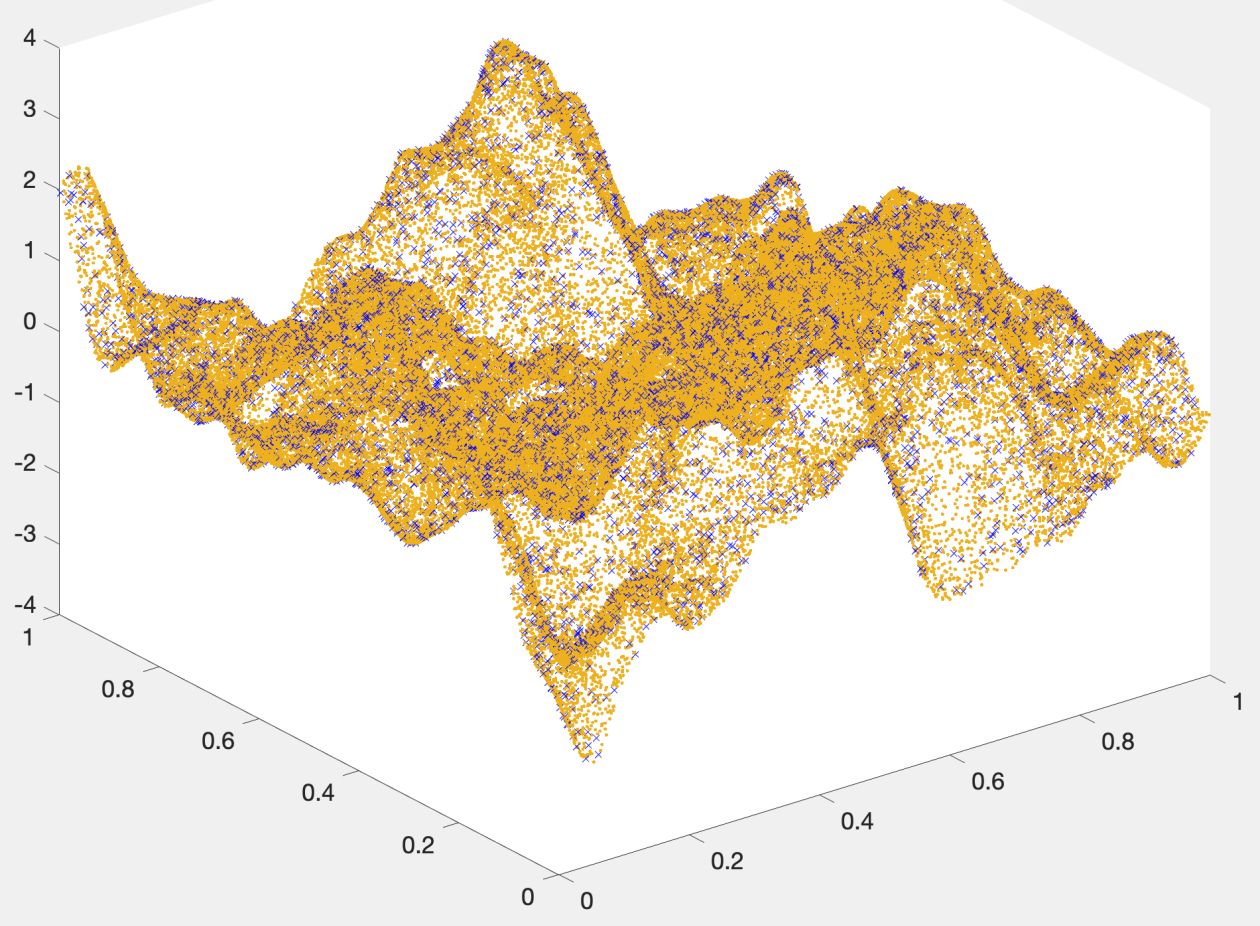


Prediction by the $\mathcal{H}$-MLE method

The yellow points at $90,000$ locations were used for training and $10,000$ values at new locations (blue points) were predicted.

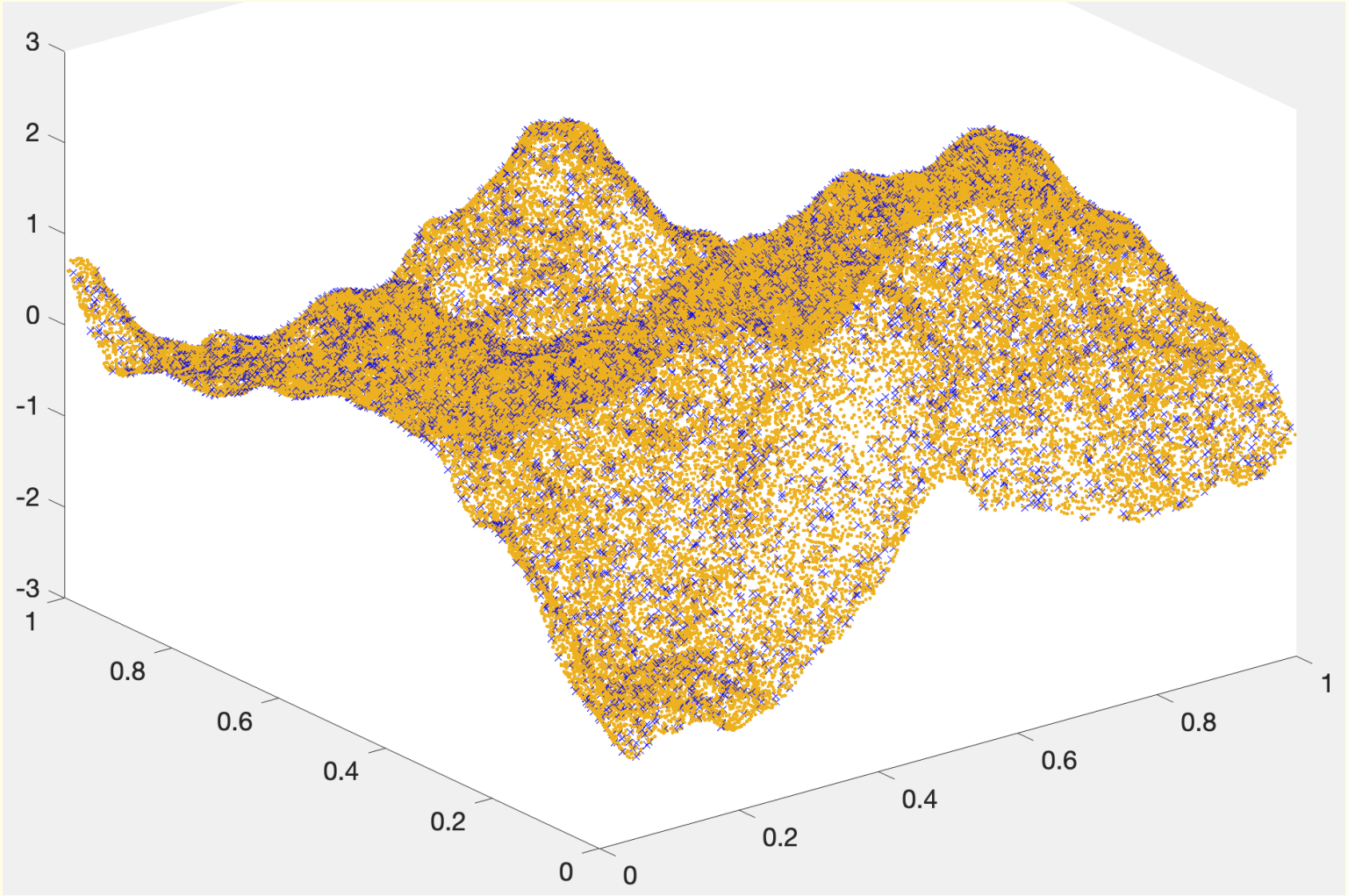


Prediction by the $\mathcal{H}$-MLE method



$\mathcal{H}$-MLE: Test 1b, dataset 4. The yellow points at $90,000$ locations were used for training and the blue points were predicted in $10,000$ locations.

Prediction by the $\mathcal{H}$-MLE method



$\mathcal{H}$-MLE: Test 1b, datasets 7. The yellow points at $90,000$ locations were used for training and the blue points were predicted in $10,000$ locations.

RMSE error: comparison for all methods

Root Mean Square Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{n_t}\sum_{i=1}^{n_t}\left(\hat{Z}(\mathbf{s}_i) - Z(\mathbf{s}_i)\right)^2}, \tag{4}$$

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.

	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	—	—

RMSE errors for $\mathcal{H}$-MLE, kNN, RF, and FCNN methods.

Conclusion

- With $\mathcal{H}$-matrices you can approximate covariance matrices (e.g., Matérn), Gaussian log-likelihoods, identify unknown parameters and make predictions
- MLE estimate and predictions depend on $\mathcal{H}$-matrix accuracy
- parameter identification problem has multiple solutions ($\boldsymbol{\theta}$)
- Investigated dependence of $\mathcal{H}$-matrix approximation error on the estimated parameters
- Each of ML methods needs fine-tuning stage to optimize its hyperparameters or architecture.
- kNN is easy to implement and it shows promising results

Open questions and TODOs

- Dependence of the Gaussian log-likelihood on a very large matrix
- How to skip/avoid redundant data?
- A good starting point for optimization is needed
- $\mathcal{H}$-matrices become expensive for large number of parameters to be identified

(All tests are reproducible
`https://github.com/litvinen/large_random_fields.git`

Likelihood in d-dimensions

Example: $N = 6000^3$. Using MATLAB on a MacBookPro with 16 GB RAM, the time required set up the matrices $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}_3$ is 11 seconds; it takes 4 seconds to compute $\mathbf{L}_1$, $\mathbf{L}_2$, and $\mathbf{L}_3$.

In previous work we used the $\mathcal{H}$-matrices to approximate $\mathbf{C}_i$ and its $\mathcal{H}$-Cholesky factor $\mathbf{L}_i$ for $n = 2 \cdot 10^6$ in 2 minutes. Here, assuming $\mathbf{C} = \mathbf{C}_1 \otimes \ldots \otimes \mathbf{C}_d$, we approximate $\mathbf{C}$ for $N = (2 \cdot 10^6)^d$ in $2d$ minutes.

$$\mathcal{L} \approx \tilde{\mathcal{L}} = -\frac{\prod_{\nu=1}^{d} n_\nu}{\log(2\pi)} - \sum_{j=1}^{d} \log\det \mathbf{C}_j \prod_{i=1, i\neq j}^{d} n_i - \sum_{i=1}^{r}\sum_{j=1}^{r}\prod_{\nu=1}^{d}(\mathbf{u}_{i,\nu}^T, \mathbf{u}_{j,\nu}).$$

Tensor approximation of Gaussian covariance

Let $\mathrm{cov}(x, y) = e^{-|x-y|^2}$ be the Gaussian covariance function, where $x = (x_1, .., x_d),\ y = (y_1, ..., y_d) \in [a, b]^d$. Let $\mathbf{C}$ be the Gaussian covariance matrix defined on a tensor grid in $[a, b]^d$, $d \geq 1$.
Then $\mathbf{C}$ can be written as Kronecker product of one-dimensional covariance matrices, i.e., $\mathbf{C} = \mathbf{C}_1 \otimes ... \otimes \mathbf{C}_d$.

Tensor Cholesky

If d Cholesky decompositions exist, i.e, $\mathbf{C}_i = \mathbf{L}_i \cdot \mathbf{L}_i^T$, $i = 1..d$. Then

$$\mathbf{C}_1 \otimes ... \otimes \mathbf{C}_d = (\mathbf{L}_1\mathbf{L}_1^T) \otimes ... \otimes (\mathbf{L}_d\mathbf{L}_d^T) \quad (5)$$

$$= (\mathbf{L}_1 \otimes ... \otimes \mathbf{L}_d) \cdot (\mathbf{L}_1^T \otimes ... \otimes \mathbf{L}_d^T) =: \mathbf{L} \cdot \mathbf{L}^T, \quad (6)$$

where $\mathbf{L} := \mathbf{L}_1 \otimes ... \otimes \mathbf{L}_d$, $\mathbf{L}^T := \mathbf{L}_1^T \otimes ... \otimes \mathbf{L}_d^T$ are also low- and upper-triangular matrices.

The computational complexity drops from $\mathcal{O}(N\log N)$, $N = n^d$, to $\mathcal{O}(dn\log n)$, where n is number of mesh points in one-dimensional problem. Further research is required for non-Gaussian covariance functions.

Tensor Inverse and determinant

If inverse matrices $\mathbf{C}_i^{-1}$, $i = 1..d$, exist, then

$$(\mathbf{C}_1 \otimes ... \otimes \mathbf{C}_d)^{-1} = \mathbf{C}_1^{-1} \otimes ... \otimes \mathbf{C}_d^{-1}. \tag{7}$$

$$\log\det(\mathbf{C}_1 \otimes \mathbf{C}_2 \otimes \mathbf{C}_3) = n_2 n_3 \log\det \mathbf{C}_1 + n_1 n_3 \log\det \mathbf{C}_2 + n_1 n_2 \log\det \mathbf{C}_3. \tag{8}$$

The cost drops again from $\mathcal{O}(N\log N)$, $N = n^d$, to $\mathcal{O}(dn\log n)$.
Example: Let $n = 6000$, $d = 3$, $N = 6000^3$. MATLAB time to setup matrices $\mathbf{C}_1$,$\mathbf{C}_2$ and $\mathbf{C}_3$ takes 11 second and to compute $\mathbf{L}_1$, $\mathbf{L}_2$ and $\mathbf{L}_3$ takes 4 seconds on usual MacBookPro with 16 GB RAM. Large matrices $\mathbf{C}$ and $\mathbf{L}$ are never constructed (i.e., the Kronecker product is never calculated).

det and trace

Let $\mathbf{C} \approx \tilde{\mathbf{C}} = \sum_{i=1}^{r} \bigotimes_{\mu=1}^{d} \mathbf{C}_{i\mu}$, then

$$\operatorname{diag}(\tilde{\mathbf{C}}) = \operatorname{diag}\left(\sum_{i=1}^{r} \bigotimes_{\mu=1}^{d} \mathbf{C}_{i\mu}\right) = \sum_{i=1}^{r} \bigotimes_{\mu=1}^{d} \operatorname{diag}\left(\mathbf{C}_{i\mu}\right), \tag{9}$$

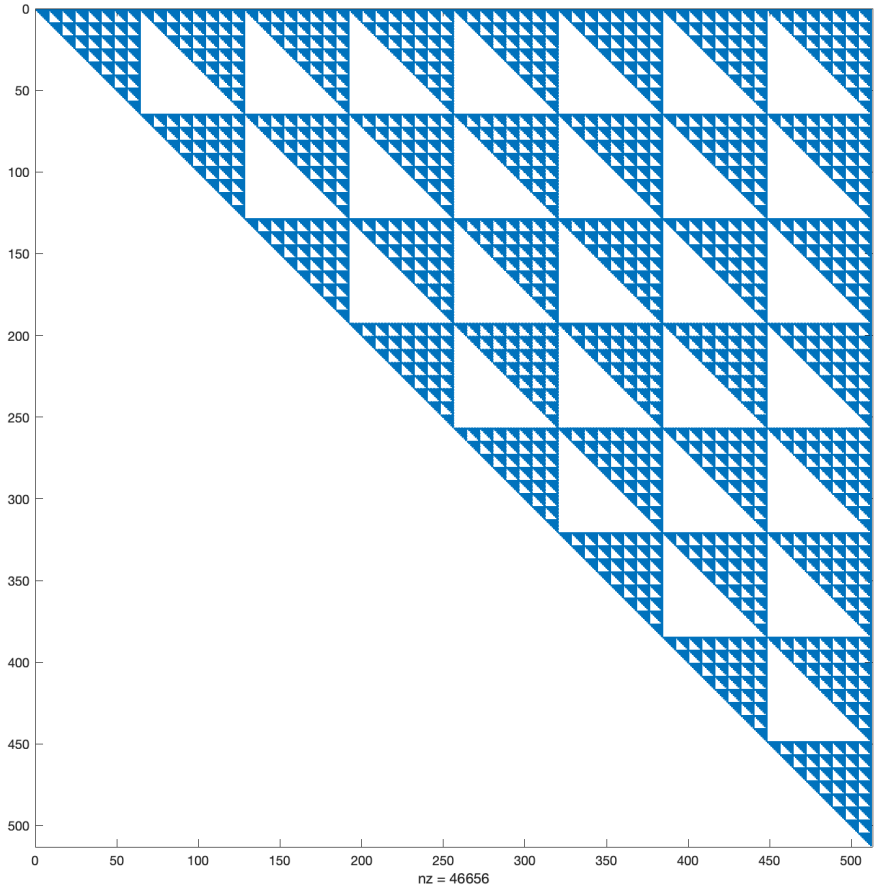
$$\operatorname{trace}(\tilde{\mathbf{C}}) = \operatorname{trace}\left(\sum_{i=1}^{r} \bigotimes_{\mu=1}^{d} \mathbf{C}_{i\mu}\right) = \sum_{i=1}^{r} \prod_{\mu=1}^{d} \operatorname{trace}(\mathbf{C}_{i\mu}). \tag{10}$$

$$\det(\mathbf{C}_1 \otimes \mathbf{C}_2) = \det(\mathbf{C}_1)^{n_2} \cdot \det(\mathbf{C}_2)^{n_1}$$

$$\begin{aligned} \log\det(\mathbf{C}_1 \otimes \mathbf{C}_2) &= \log(\det(\mathbf{C}_1)^{n_2} \cdot \det(\mathbf{C}_2)^{n_1}) \\ &= n_2 \log\det \mathbf{C}_1 + n_1 \log\det \mathbf{C}_2. \end{aligned}$$

$$\begin{aligned} \log\det(\mathbf{C}_1 \otimes \mathbf{C}_2 \otimes \mathbf{C}_3) = {} & n_2 n_3 \log\det \mathbf{C}_1 + n_1 n_3 \log\det \mathbf{C}_2 \\ & + n_1 n_2 \log\det \mathbf{C}_3. \end{aligned}$$

Kronecker tensor product of two Cholesky matrices



Future research

1. Domain decomposition techniques to approximate precision matrix

Domain Decomposition of Mexico. (Picture is taken from Hackbusch's paper).

DD for A

$$A = \begin{pmatrix} A_{11} & 0 & \dots & 0 & A_{1\Sigma} \\ 0 & A_{22} & \dots & 0 & A_{2\Sigma} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & A_{pp} & A_{p\Sigma} \\ A_{\Sigma 1} & A_{\Sigma 2} & \dots & A_{\Sigma p} & A_{\Sigma\Sigma} \end{pmatrix}. \tag{11}$$

The inverse for A

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & A_{pp}^{-1} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} A_{11}^{-1}A_{1\Sigma} \\ \vdots \\ A_{pp}^{-1}A_{p\Sigma} \\ -I \end{bmatrix} [S^{-1}A_{\Sigma 1}A_{11}^{-1}, \cdots, S^{-1}A_{\Sigma p}A_{pp}^{-1}, -S^{-1}].$$

where $S := A_{\Sigma\Sigma} - \sum_{i=1}^{p} A_{\Sigma i}A_{ii}^{-1}A_{i\Sigma}$.

Future research

2. Multiscale random fields

$$(\kappa(x) - \nabla \cdot (a(x)\nabla))^{\alpha/2} u(x) = \mathcal{W}$$

$a(x)$ oscillates rapidly,
$\kappa(x)$ varies across scales,
see more by H. Rue, F. Lindgren, D. Bolin

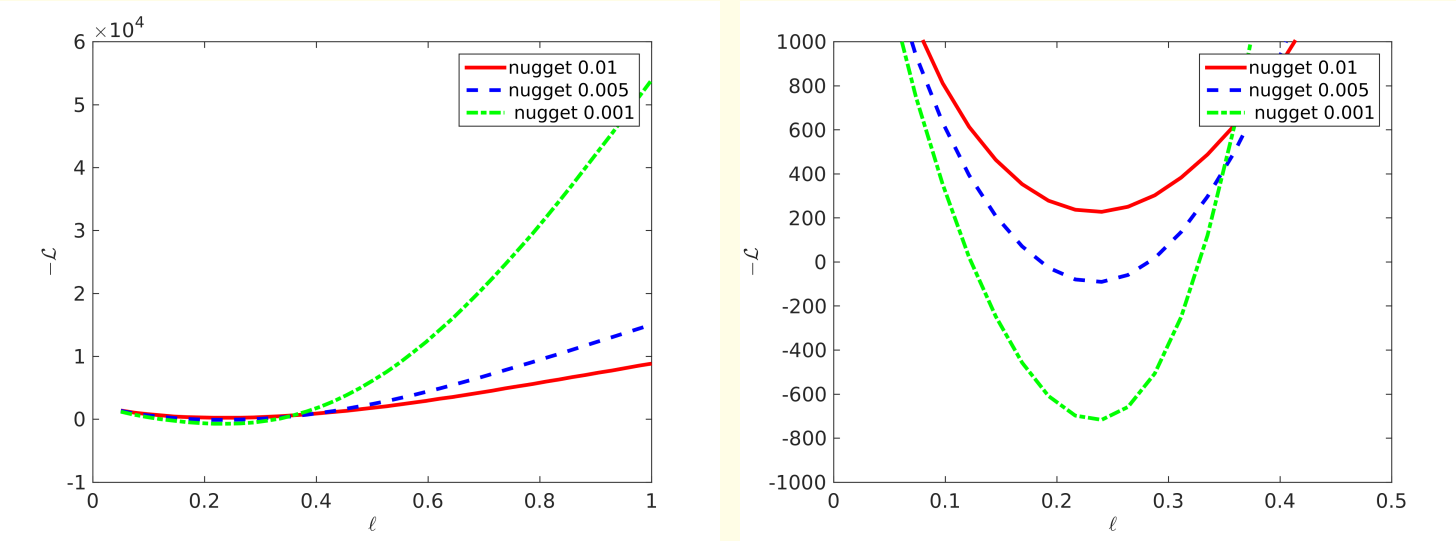
Literature

1. A. Litvinenko, R. Kriemann, V. Berikov, Identification of unknown parameters and prediction with hierarchical matrices, UNCECOMP 2021 Conf. Proc., Uncertainty Quantification in Computational Sciences and Engineering, M. Papadrakakis, V. Papadopoulos, G. Stefanou (Eds.), pp 129-144, https://2021.uncecomp.org, ISBN: 978-618-85072-6-5, 2021
2. A. Litvinenko, R. Kriemann, M.G. Genton, Y. Sun, D.E. Keyes, HLIBCov: Parallel hierarchical matrix approximation of large covariance matrices and likelihoods with applications in parameter identification, MethodsX 7, 100600, 2020
3. A. Litvinenko, Y. Sun, M.G. Genton, D.E. Keyes, Likelihood approximation with hierarchical matrices for large spatial datasets, Computational Statistics & Data Analysis 137, 115-132, 2019
4. M. Espig, W. Hackbusch, A. Litvinenko, H.G. Matthies, E. Zander, Iterative algorithms for the post-processing of high-dimensional data, Journal of Computational Physics 410, 109396, 2020
5. A. Litvinenko, D. Keyes, V. Khoromskaia, B.N. Khoromskij, H.G. Matthies, Tucker tensor analysis of Matérn functions in spatial statistics, Computational Methods in Applied Mathematics 19 (1), 101-122, 2019
6. M. Espig, W. Hackbusch, A. Litvinenko, H.G. Matthies, E. Zander, Efficient analysis of high dimensional data in tensor formats, Sparse Grids and Applications, 31-56, Springer, Berlin, 2013

Appendix

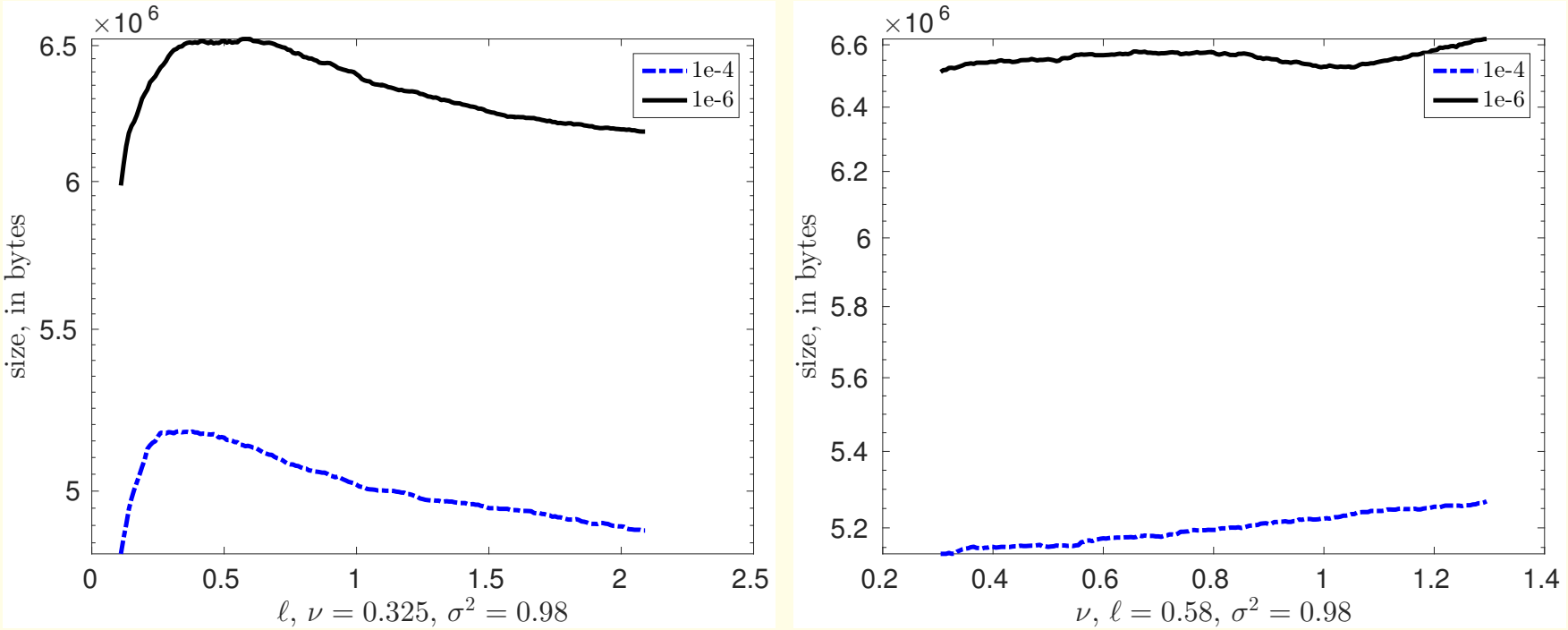
Remark: stabilization with nugget

To avoid instability in computing Cholesky, we add: $\widetilde{\mathbf{C}}_m = \widetilde{\mathbf{C}} + \tau^2 \mathbf{I}$.
Let λ_i be eigenvalues of $\widetilde{\mathbf{C}}$, then eigenvalues of $\widetilde{\mathbf{C}}_m$ will be $\lambda_i + \tau^2$,
$\log\det(\widetilde{\mathbf{C}}_m) = \log\prod_{i=1}^{n}(\lambda_i + \tau^2) = \sum_{i=1}^{n}\log(\lambda_i + \tau^2)$.



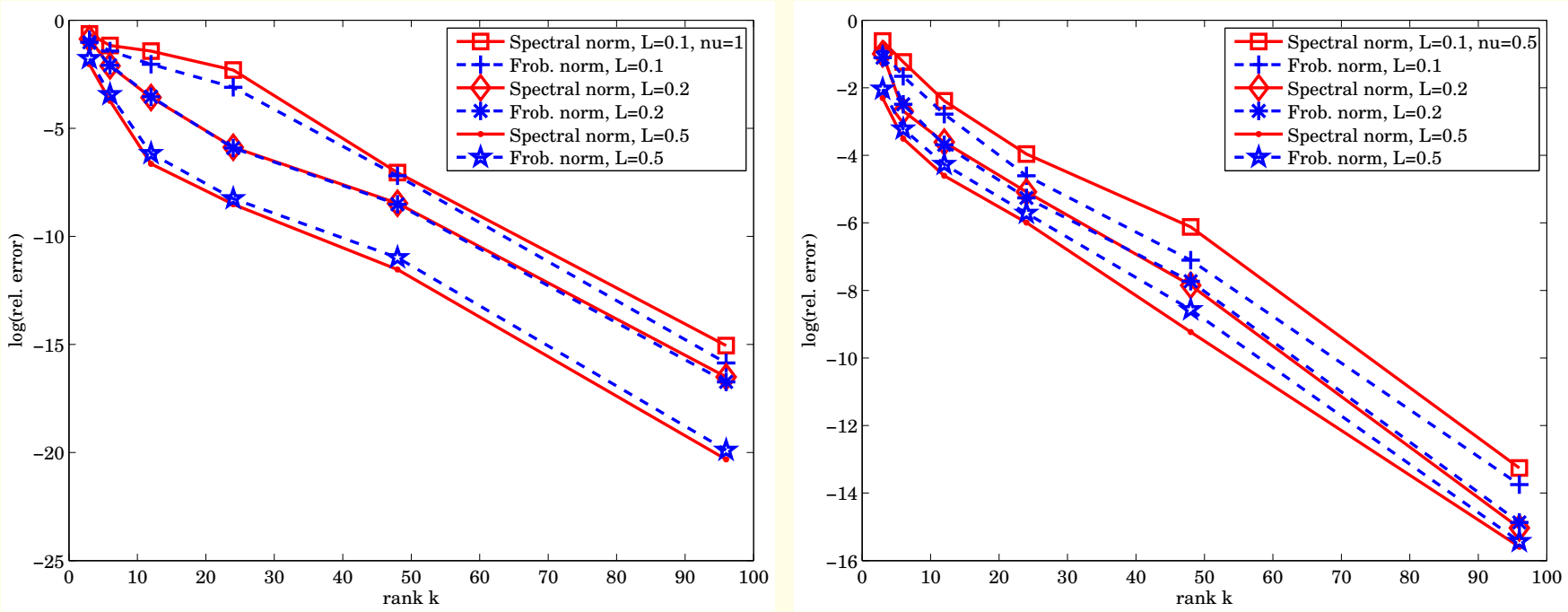
(left) Dependence of the negative log-likelihood on parameter ℓ with nuggets $\{0.01, 0.005, 0.001\}$ for the Gaussian covariance. (right) Zoom of the left figure near minimum; $n = 2000$ random points from moisture example, rank $k = 14$, $\tau^2 = 1$, $\nu = 0.5$.

How much memory do $\mathcal{H}$-matrices need?

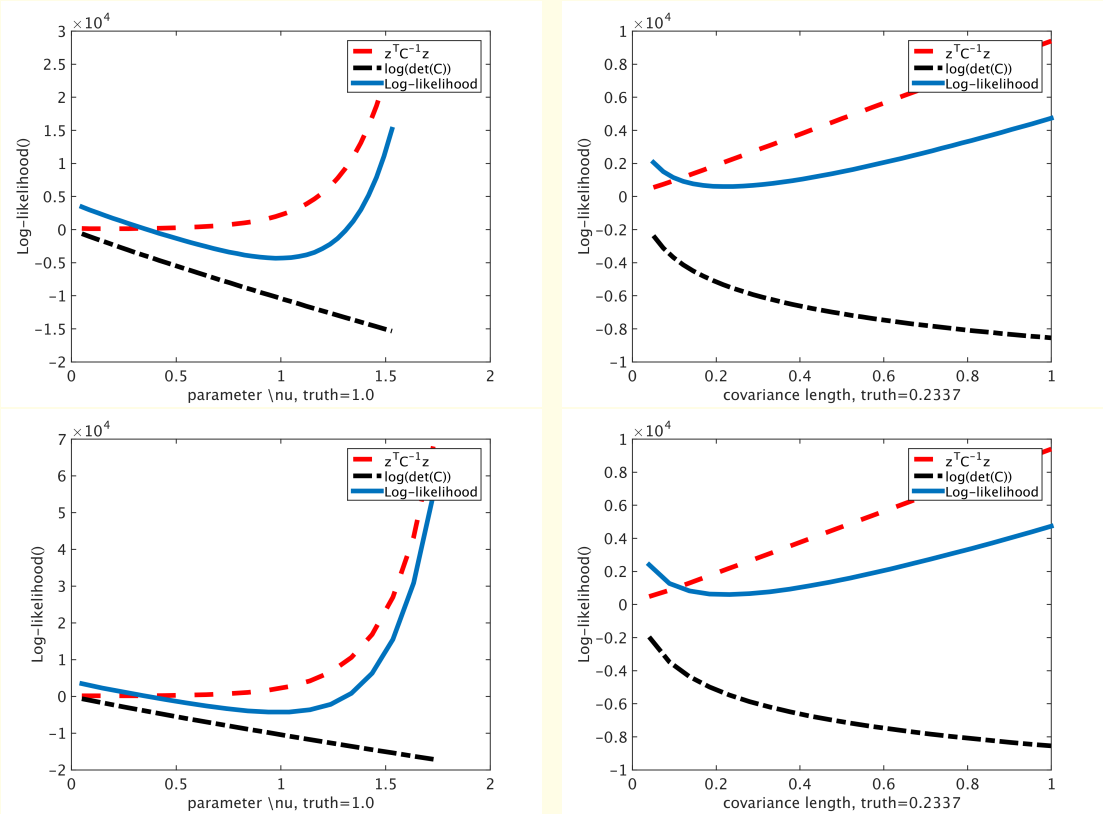


(left) Dependence of the matrix size on the covariance length ℓ, and (right) the smoothness ν for two different $\mathcal{H}$-accuracies $\varepsilon = \{10^{-4}, 10^{-6}\}$

$\mathcal{H}$-matrices : Error convergence

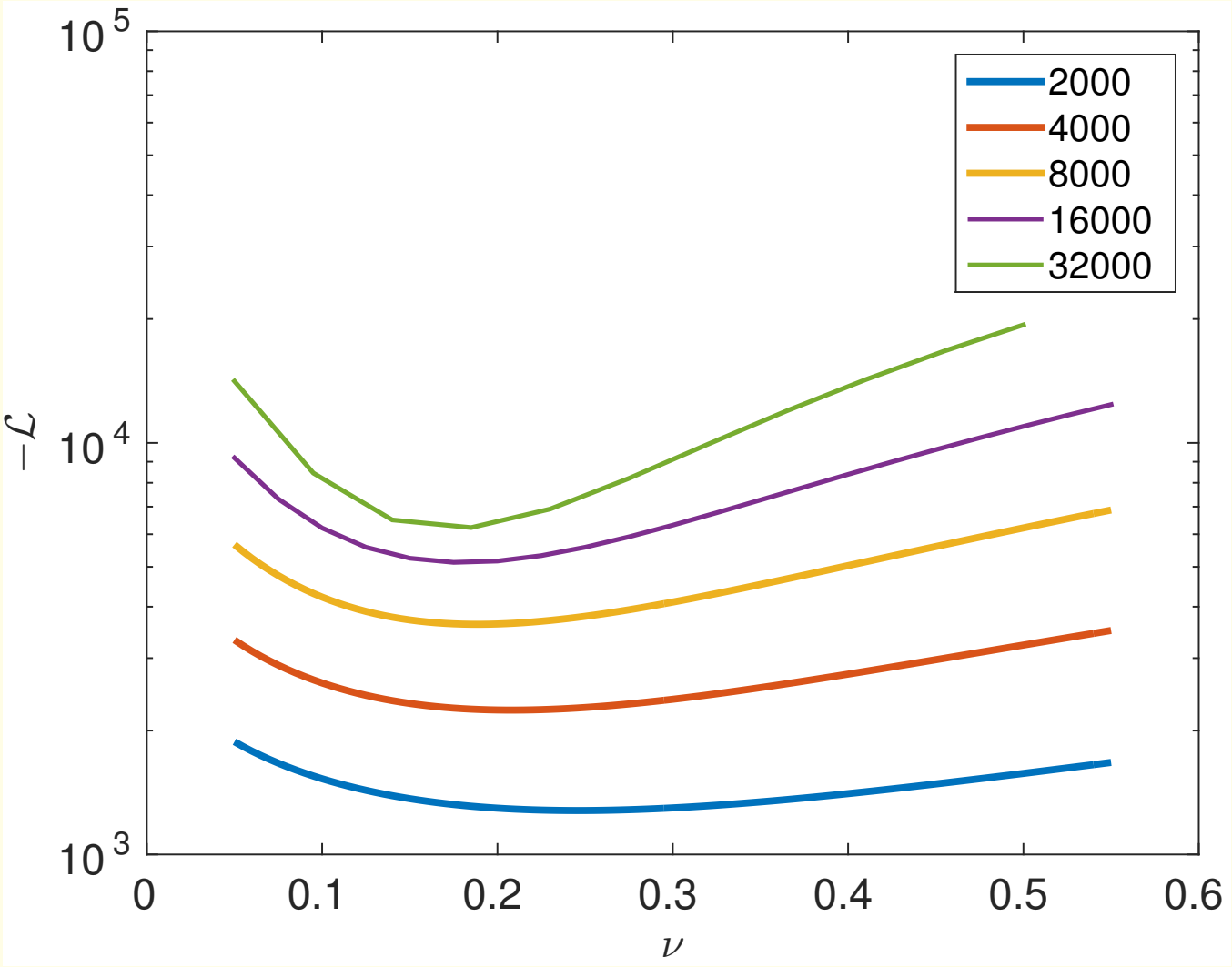


Convergence of the $\mathcal{H}$-matrix approximation errors for covariance lengths $\{0.1, 0.2, 0.5\}$; (left) $\nu = 1$ and (right) $\nu = 0.5$, computational domain $[0, 1]^2$.



Dependence of log-likelihood ingredients on parameters, $n = 4225$. $k = 8$ in the first row and $k = 16$ in the second.

How does log-likelihood depend on n?



Dependence of negative log-likelihood function on different number of locations $n = \{2000, 4000, 8000, 16000, 32000\}$ in log-scale.