Sparse Cholesky Factorization

for solving PDEs with Gaussian processes

Yifan Chen

Applied and Computational Math, Caltech

ICERM, June 2023

The Paper

[Chen, Owhadi, Schäfer 2023]

Sparse Cholesky Factorization

for Solving Nonlinear PDEs via Gaussian Processes

Houman Owhadi
Caltech

Florian Schäfer
Georgia Tech

Link: `https://arxiv.org/abs/2304.01294`

Gaussian processes (GPs) and kernel methods

GPs and kernel methods are widely used
in scientific computing and scientific machine learning

- Spatial statistics
- Surrogate modeling
- Experimental design and Bayes optimization
- Solving PDEs and inverse problems
- ...

Gaussian processes (GPs) and kernel methods

GPs and kernel methods are widely used
in scientific computing and scientific machine learning

- Spatial statistics
- Surrogate modeling
- Experimental design and Bayes optimization
- Solving PDEs and inverse problems
- ...

Focus of this talk: fast algorithms for the methodology

Computations in GPs and Kernel Methods

Dense kernel matrices: for example

$$\Theta = k(\mathbf{X}, \mathbf{X})$$

where $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \subset \mathbb{R}^d$ and $k(\mathbf{X}, \mathbf{X}) \in \mathbb{R}^{n \times n}$

Computations in GPs and Kernel Methods

Dense kernel matrices: for example

$$\Theta = \begin{pmatrix} k(\mathbf{X},\mathbf{X}) & \Delta_{\mathbf{y}} k(\mathbf{X},\mathbf{X}) \\ \Delta_{\mathbf{x}} k(\mathbf{X},\mathbf{X}) & \Delta_{\mathbf{x}}\Delta_{\mathbf{y}} k(\mathbf{X},\mathbf{X}) \end{pmatrix} \in \mathbb{R}^{N\times N}$$

where $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \subset \mathbb{R}^d$ and $k(\mathbf{X},\mathbf{X}) \in \mathbb{R}^{n\times n}$

Computations in GPs and Kernel Methods

Dense kernel matrices: for example

$$\Theta = \begin{pmatrix} k(\mathbf{X},\mathbf{X}) & \Delta_{\mathbf{y}} k(\mathbf{X},\mathbf{X}) \\ \Delta_{\mathbf{x}} k(\mathbf{X},\mathbf{X}) & \Delta_{\mathbf{x}}\Delta_{\mathbf{y}} k(\mathbf{X},\mathbf{X}) \end{pmatrix} \in \mathbb{R}^{N\times N}$$

where $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \subset \mathbb{R}^d$ and $k(\mathbf{X},\mathbf{X}) \in \mathbb{R}^{n\times n}$

Derivative entries, e.g., $\Delta_{\mathbf{x}} k$ arise naturally in PDE problems where we have derivative information

e.g., **[Chen, Hosseni, Owhadi, Stuart 2021]**

Computations in GPs and Kernel Methods

Dense kernel matrices: for example

$$\Theta = \begin{pmatrix} k(\mathbf{X},\mathbf{X}) & \Delta_{\mathbf{y}} k(\mathbf{X},\mathbf{X}) \\ \Delta_{\mathbf{x}} k(\mathbf{X},\mathbf{X}) & \Delta_{\mathbf{x}}\Delta_{\mathbf{y}} k(\mathbf{X},\mathbf{X}) \end{pmatrix} \in \mathbb{R}^{N\times N}$$

where $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \subset \mathbb{R}^d$ and $k(\mathbf{X},\mathbf{X}) \in \mathbb{R}^{n\times n}$

Derivative entries, e.g., $\Delta_{\mathbf{x}} k$ arise naturally in PDE problems where we have derivative information

e.g., **[Chen, Hosseni, Owhadi, Stuart 2021]**

Cubic bottleneck $O(N^3)$: Θ is a dense matrix

Fast Algorithms

Many approximate methods:

- Nyström approximation, inducing points, sparse GPs, random features, covariance tapering, divide-and-conquer, structured kernel interpolation, hierarchical matrices, wavelets based methods, sparse Cholesky factorization ...

Fast Algorithms

Many approximate methods:

- Nyström approximation, inducing points, sparse GPs, random features, covariance tapering, divide-and-conquer, structured kernel interpolation, hierarchical matrices, wavelets based methods, sparse Cholesky factorization ...
- Based on low-rank/sparse ideas and their multiscale variants

Fast Algorithms

Many approximate methods:

- Nyström approximation, inducing points, sparse GPs, random features, covariance tapering, divide-and-conquer, structured kernel interpolation, hierarchical matrices, wavelets based methods, sparse Cholesky factorization ...
- Based on low-rank/sparse ideas and their multiscale variants
- Mostly developed when **there is no derivatives of k**

Fast Algorithms

Many approximate methods:

- Nyström approximation, inducing points, sparse GPs, random features, covariance tapering, divide-and-conquer, structured kernel interpolation, hierarchical matrices, wavelets based methods, sparse Cholesky factorization ...
- Based on low-rank/sparse ideas and their multiscale variants
- Mostly developed when **there is no derivatives of k**

The goal: advance methods for **kernel matrices with derivatives**

Fast Algorithms

Many approximate methods:

- Nyström approximation, inducing points, sparse GPs, random features, covariance tapering, divide-and-conquer, structured kernel interpolation, hierarchical matrices, wavelets based methods, sparse Cholesky factorization ...
- Based on low-rank/sparse ideas and their multiscale variants
- Mostly developed when **there is no derivatives of k**

The goal: advance methods for **kernel matrices with derivatives**

A new approach [Chen, Owhadi, Schäfer 2023]

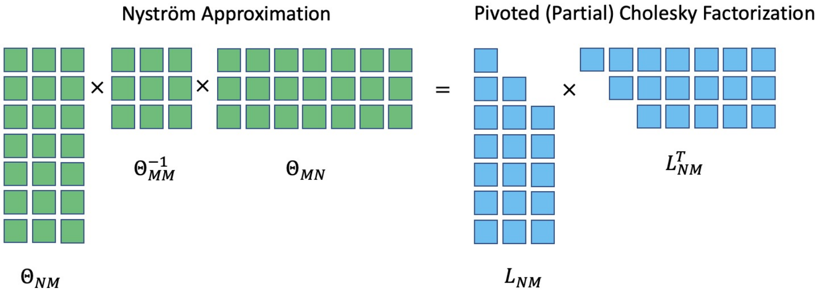
A provable near-linear complexity algorithm

even when **there are derivatives of k**

Outline

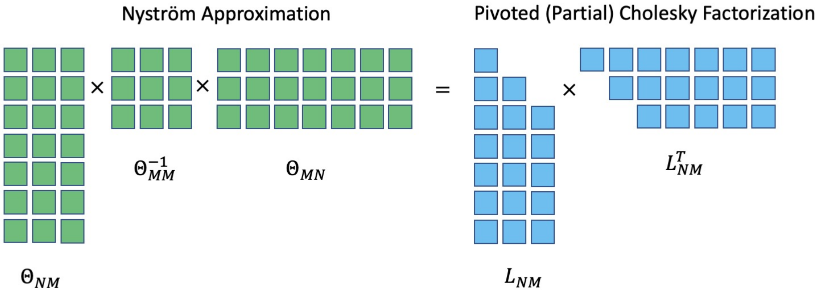
Outline

Warm-up: Nyström Approximation



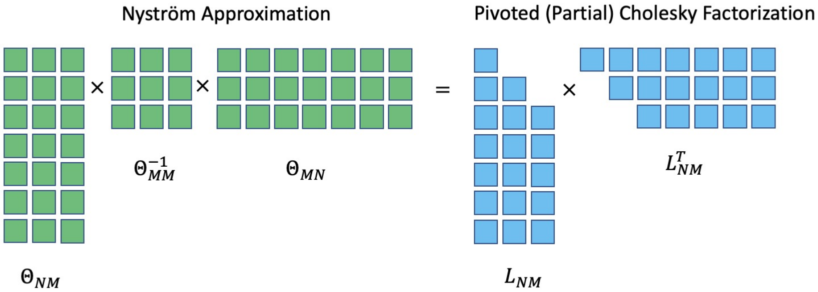
- $\Theta \approx \Theta_{NM}\Theta_{MM}^{-1}\Theta_{MN}$, with complexity $O(NM^2)$

Warm-up: Nyström Approximation



- $\Theta \approx \Theta_{NM}\Theta_{MM}^{-1}\Theta_{MN}$, with complexity $O(NM^2)$
- Applied to kernel matrices with derivatives [Eriksson, Dong, Lee, Bindel 2018], [Yang, Li, Rana, Gupta, Venkatesh 2018], [Meng, Yang 2022]

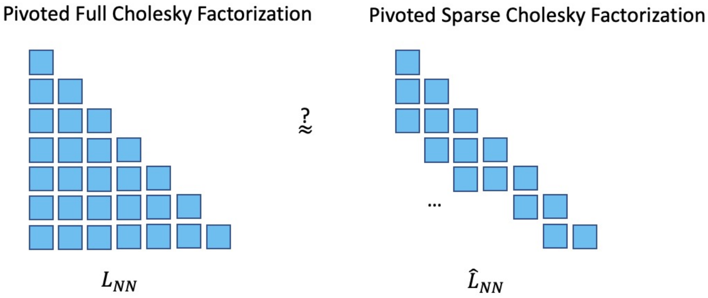
Warm-up: Nyström Approximation



- $\Theta \approx \Theta_{NM}\Theta_{MM}^{-1}\Theta_{MN}$, with complexity $O(NM^2)$
- Applied to kernel matrices with derivatives [Eriksson, Dong, Lee, Bindel 2018], [Yang, Li, Rana, Gupta, Venkatesh 2018], [Meng, Yang 2022]

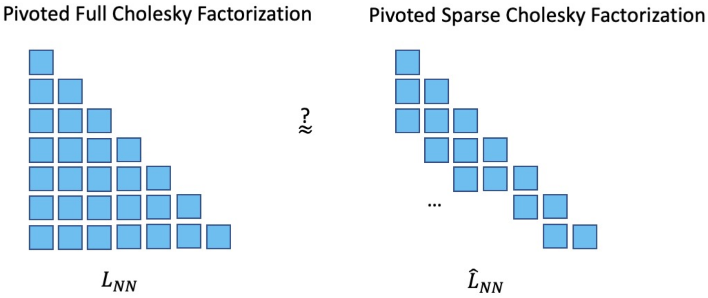
Nevertheless, for **high accuracy** in scientific/PDE problems,
low rank approximation may not be enough

Increase the Rank $\Rightarrow$ Full Cholesky Factorization



- Not computationally affordable: complexity $O(N^3)$ again

Increase the Rank $\Rightarrow$ Full Cholesky Factorization



- Not computationally affordable: complexity $O(N^3)$ again

$\Rightarrow$ **Approach: Sparse Cholesky factorization**

Sketch of the Result

[Chen, Owhadi, Schäfer 2023]

For Θ with derivative entries, we present a sparse Cholesky factorization algorithm with the state-of-the-art complexity

- $O(N \log^d(N/\epsilon))$ in space; and
- $O(N \log^{2d}(N/\epsilon))$ in time

The algorithm outputs

- a permutation matrix P_{perm}; and
- a upper triangular matrix U with $O(N \log^d(N/\epsilon))$ nonzeros

such that

$$\|\Theta^{-1} - P_{\mathrm{perm}}^T U U^T P_{\mathrm{perm}}\|_{\mathrm{Fro}} \leq \epsilon$$

where $\|\cdot\|_{\mathrm{Fro}}$ is the Frobenius norm

Assumptions on k to get rigorous results: see the paper

How? Probabilistic Interpretation of Cholesky Factorization

Connection between linear algebra and probability

Let $\Theta \in \mathbb{R}^{N \times N}$, and $X \sim \mathcal{N}(0, \Theta)$

- Lower-triangular Cholesky factor of $\Theta = LL^T$

$$\frac{L_{ij}}{L_{jj}} = \frac{\mathrm{Cov}[X_i, X_j | X_{1:j-1}]}{\mathrm{Var}[X_j | X_{1:j-1}]} \qquad (i \geq j)$$

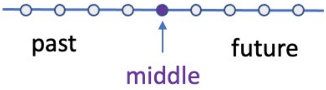
- Upper-triangular Cholesky factor of $\Theta^{-1} = UU^T$

$$\frac{U_{ij}}{U_{jj}} = (-1)^{i \neq j} \frac{\mathrm{Cov}[X_i, X_j | X_{1:j-1 \setminus \{i\}}]}{\mathrm{Var}[X_j | X_{1:j-1 \setminus \{i\}}]} \qquad (i \leq j)$$

Proof by mathematical induction on the value of j

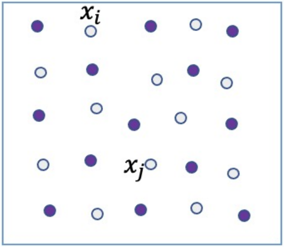
Conditioning, Screening Effects, and Sparsity

Screening effects [Stein 2002] (when no derivative entries ...)



$$k(x, y) = \exp(-|x - y|)$$

Cov [past, future | middle] = 0



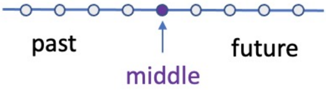
Matérn's kernel

Cov [fine x_i, fine x_j | coarse] << 1

if x_i and x_j are well separated by coarse points

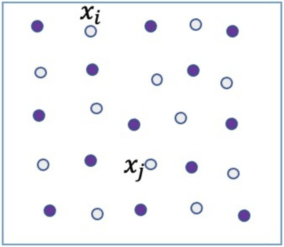
Conditioning, Screening Effects, and Sparsity

Screening effects [Stein 2002] (when no derivative entries ...)



$$k(x,y) = \exp(-|x-y|)$$

Cov [past, future | middle] = 0



Matérn's kernel

Cov [fine x_i, fine x_j | coarse] << 1

if x_i and x_j are well separated by coarse points

Cholesky factors ≈ sparse if points ordered from **coarse to fine**

[Schäfer, Sullivan, Owhadi 2021], [Schäfer, Katzfuss, Owhadi 2021]

How to Order From Coarse to Fine?

Max-min ordering

The next ordered point is the **farthest** to **points selected before**

$$\mathbf{x}_k = \mathsf{argmax}_{\mathbf{x}_i} \mathrm{dist}(\mathbf{x}_i, \{\mathbf{x}_j, 1 \leq j < k\})$$

with its lengthscale defined by

$$l_k = \mathrm{dist}(\mathbf{x}_k, \{\mathbf{x}_j, 1 \leq j < k\})$$

How to Order From Coarse to Fine?

Max-min ordering

The next ordered point is the **farthest** to **points selected before**

$$\mathbf{x}_k = \mathrm{argmax}_{\mathbf{x}_i} \mathrm{dist}(\mathbf{x}_i, \{\mathbf{x}_j, 1 \leq j < k\})$$

with its lengthscale defined by

$$l_k = \mathrm{dist}(\mathbf{x}_k, \{\mathbf{x}_j, 1 \leq j < k\})$$

- Lead to developments of rigorous sparse Cholesky factorization algorithm for **kernel matrices without derivative entries**!

  [Schäfer, Sullivan, Owhadi 2021], [Schäfer, Katzfuss, Owhadi 2021]

How to Order From Coarse to Fine?

Max-min ordering

The next ordered point is the **farthest** to **points selected before**

$$\mathbf{x}_k = \mathsf{argmax}_{\mathbf{x}_i} \mathrm{dist}(\mathbf{x}_i, \{\mathbf{x}_j, 1 \leq j < k\})$$

with its lengthscale defined by

$$l_k = \mathrm{dist}(\mathbf{x}_k, \{\mathbf{x}_j, 1 \leq j < k\})$$

- Lead to developments of rigorous sparse Cholesky factorization algorithm for **kernel matrices without derivative entries**!

  [Schäfer, Sullivan, Owhadi 2021], [Schäfer, Katzfuss, Owhadi 2021]

What is missing: the case when derivative entries exist

Existence of Sparse Factors In the Case of Derivative Entries

A new coarse-to-fine ordering [Chen, Owhadi, Schäfer 2023]

Order the pointwise entries by **max-min ordering** of the points, then followed with **arbitrary order** of derivative entries

i.e., derivative entries treated as **finer scales** *than pointwise ones*

Existence of Sparse Factors In the Case of Derivative Entries

A new coarse-to-fine ordering [Chen, Owhadi, Schäfer 2023]

Order the pointwise entries by **max-min ordering** of the points, then followed with **arbitrary order** of derivative entries

i.e., derivative entries treated as **finer scales** *than pointwise ones*

Theorem [Chen, Owhadi, Schäfer 2023]

Consider the upper triangular inverse Cholesky factors of the reordered matrix $\Theta^{-1}_{\text{reordered}} = U^\star U^{\star T}$. For $1 \leq i \leq j \leq N$,

$$|U^\star_{ij}| \leq C l_j^\alpha \exp\left(-\frac{\mathrm{dist}(\mathbf{x}_{P(i)}, \mathbf{x}_{P(j)})}{C l_j}\right)$$

where C, α are generic constants. Here $\mathbf{x}_{P(i)}$ is the physical point corresponding to the ith ordered entry

- Proof assumes k: Green function of psd differential operators

Computing Sparse Factors In the Case of Derivative Entries

Sparsity pattern: entries outside

$$S_{l,\rho} = \{1 \leq i \leq j \leq N : \mathrm{dist}(\mathbf{x}_{P(i)}, \mathbf{x}_{P(j)}) \leq \rho l_j\}$$

is **exponentially small** regarding ρ

Computing Sparse Factors In the Case of Derivative Entries

Sparsity pattern: entries outside

$$S_{l,\rho} = \{1 \leq i \leq j \leq N : \mathrm{dist}(\mathbf{x}_{P(i)}, \mathbf{x}_{P(j)}) \leq \rho l_j\}$$

is **exponentially small** regarding ρ

Algorithm: Given the sparsity pattern, **using optimization** to extract an optimal sparse factor U^ρ [Schäfer, Katzfuss, Owhadi 2021]

Computing Sparse Factors In the Case of Derivative Entries

Sparsity pattern: entries outside

$$S_{l,\rho} = \{1 \le i \le j \le N : \mathrm{dist}(\mathbf{x}_{P(i)}, \mathbf{x}_{P(j)}) \le \rho l_j\}$$

is **exponentially small** regarding ρ

Algorithm: Given the sparsity pattern, **using optimization** to extract an optimal sparse factor U^ρ [Schäfer, Katzfuss, Owhadi 2021]

- Sparse set: $\mathcal{S}_{l,\rho} = \{A \in \mathbb{R}^{N\times N} : A_{ij} \neq 0 \Rightarrow (i,j) \in S_{l,\rho}\}$

Computing Sparse Factors In the Case of Derivative Entries

Sparsity pattern: entries outside

$$S_{l,\rho} = \{1 \le i \le j \le N : \mathrm{dist}(\mathbf{x}_{P(i)}, \mathbf{x}_{P(j)}) \le \rho l_j\}$$

is **exponentially small** regarding ρ

Algorithm: Given the sparsity pattern, **using optimization** to extract an optimal sparse factor U^ρ [Schäfer, Katzfuss, Owhadi 2021]

- Sparse set: $\mathcal{S}_{l,\rho} = \{A \in \mathbb{R}^{N\times N} : A_{ij} \neq 0 \Rightarrow (i,j) \in S_{l,\rho}\}$
- $U^\rho = \mathrm{argmin}_{U \in \mathcal{S}_{l,\rho}} \mathrm{KL}\left(\mathcal{N}(0, \Theta_{\text{reordered}}) \parallel \mathcal{N}(0, (UU^T)^{-1})\right)$

Computing Sparse Factors In the Case of Derivative Entries

Sparsity pattern: entries outside

$$S_{l,\rho} = \{1 \leq i \leq j \leq N : \operatorname{dist}(\mathbf{x}_{P(i)}, \mathbf{x}_{P(j)}) \leq \rho l_j\}$$

is **exponentially small** regarding ρ

Algorithm: Given the sparsity pattern, **using optimization** to extract an optimal sparse factor U^ρ [Schäfer, Katzfuss, Owhadi 2021]

- Sparse set: $\mathcal{S}_{l,\rho} = \{A \in \mathbb{R}^{N\times N} : A_{ij} \neq 0 \Rightarrow (i,j) \in S_{l,\rho}\}$
- $U^\rho = \operatorname{argmin}_{U \in \mathcal{S}_{l,\rho}} \operatorname{KL}\left(\mathcal{N}(0, \Theta_{\text{reordered}}) \parallel \mathcal{N}(0, (UU^T)^{-1})\right)$
  - Explicit solution formula [Vecchia 1988],[Marzouk et al. 2016], [Schäfer, Katzfuss, Owhadi 2021]

Computing Sparse Factors In the Case of Derivative Entries

Sparsity pattern: entries outside

$$S_{l,\rho} = \{1 \leq i \leq j \leq N : \mathrm{dist}(\mathbf{x}_{P(i)}, \mathbf{x}_{P(j)}) \leq \rho l_j\}$$

is **exponentially small** regarding ρ

Algorithm: Given the sparsity pattern, **using optimization** to extract an optimal sparse factor U^ρ [Schäfer, Katzfuss, Owhadi 2021]

- Sparse set: $\mathcal{S}_{l,\rho} = \{A \in \mathbb{R}^{N \times N} : A_{ij} \neq 0 \Rightarrow (i,j) \in S_{l,\rho}\}$
- $U^\rho = \mathrm{argmin}_{U \in \mathcal{S}_{l,\rho}} \mathrm{KL}\left(\mathcal{N}(0, \Theta_{\text{reordered}}) \parallel \mathcal{N}(0, (UU^T)^{-1})\right)$
  - Explicit solution formula [Vecchia 1988],[Marzouk et al. 2016], [Schäfer, Katzfuss, Owhadi 2021]
  - Can be implemented $O(N\rho^d)$ in space and $O(N\rho^{2d})$ time

Computing Sparse Factors In the Case of Derivative Entries

Sparsity pattern: entries outside

$$S_{l,\rho} = \{1 \leq i \leq j \leq N : \mathrm{dist}(\mathbf{x}_{P(i)}, \mathbf{x}_{P(j)}) \leq \rho l_j\}$$

is **exponentially small** regarding ρ

Algorithm: Given the sparsity pattern, **using optimization** to extract an optimal sparse factor U^ρ [Schäfer, Katzfuss, Owhadi 2021]

- Sparse set: $\mathcal{S}_{l,\rho} = \{A \in \mathbb{R}^{N\times N} : A_{ij} \neq 0 \Rightarrow (i,j) \in S_{l,\rho}\}$
- $U^\rho = \mathrm{argmin}_{U \in \mathcal{S}_{l,\rho}} \mathrm{KL}\left(\mathcal{N}(0, \Theta_{\text{reordered}}) \parallel \mathcal{N}(0, (UU^T)^{-1})\right)$
  - Explicit solution formula [Vecchia 1988],[Marzouk et al. 2016], [Schäfer, Katzfuss, Owhadi 2021]
  - Can be implemented $O(N\rho^d)$ in space and $O(N\rho^{2d})$ time
  - Parallizable: decoupled problems for each column of U

Computing Sparse Factors In the Case of Derivative Entries

Sparsity pattern: entries outside

$$S_{l,\rho} = \{1 \leq i \leq j \leq N : \mathrm{dist}(\mathbf{x}_{P(i)}, \mathbf{x}_{P(j)}) \leq \rho l_j\}$$

is **exponentially small** regarding ρ

Algorithm: Given the sparsity pattern, **using optimization** to extract an optimal sparse factor U^ρ [Schäfer, Katzfuss, Owhadi 2021]

- Sparse set: $\mathcal{S}_{l,\rho} = \{A \in \mathbb{R}^{N\times N} : A_{ij} \neq 0 \Rightarrow (i,j) \in S_{l,\rho}\}$
- $U^\rho = \mathrm{argmin}_{U\in\mathcal{S}_{l,\rho}} \mathrm{KL}\left(\mathcal{N}(0, \Theta_{\text{reordered}}) \parallel \mathcal{N}(0, (UU^T)^{-1})\right)$
  - Explicit solution formula [Vecchia 1988],[Marzouk et al. 2016], [Schäfer, Katzfuss, Owhadi 2021]
  - Can be implemented $O(N\rho^d)$ in space and $O(N\rho^{2d})$ time
  - Parallizable: decoupled problems for each column of U
- Theory: $\rho = O(\log(N/\epsilon)) \Rightarrow \|\Theta_{\text{reordered}}^{-1} - U^\rho(U^\rho)^T\|_{\text{Fro}} \leq \epsilon$

Outline

Nonlinear Elliptic Equations

- 2D Example: nonlinear elliptic equation with $\tau(u) = u^3$

$$-\Delta u + \tau(u) = f \quad \text{w/ Dirichlet's boundary condition}$$

- $\Omega = [0,1]^2$. Collocation points uniformly distributed

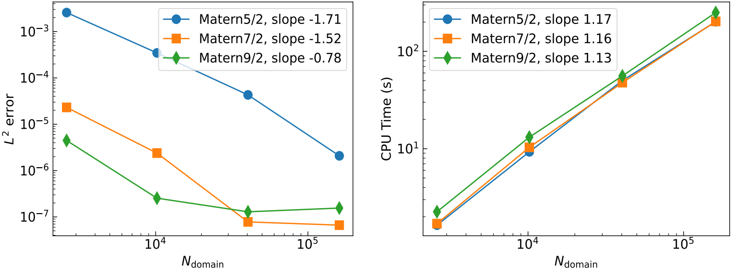
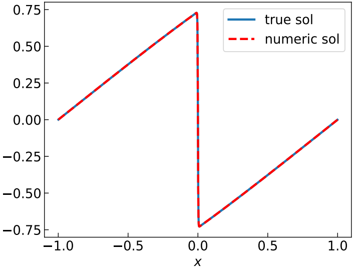


Figure: Run 3 linearization steps with initialization as a zero function. Accuracy floor due to finite $\rho = 4.0$

Burgers' Equation

- $\partial_t u + u\partial_x u - 0.001\partial_x^2 u = 0, \quad \forall (x,t) \in (-1,1) \times (0,1]$
- $\Delta t = 0.02, \rho = 4$, solve to $t = 1$



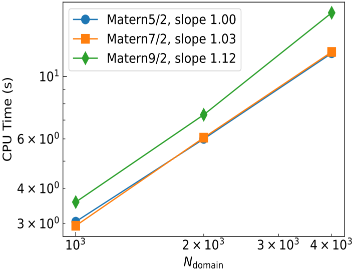


Figure: Run 2 linearization steps at each time step

Monge-Ampère Equation

- Equation: $\det(D^2 u) = f$ in $(0,1)^2$
- Truth $u(\mathbf{x}) = \exp\big(0.5((x_1 - 0.5)^2 + (x_2 - 0,5)^2)\big)$
- Matérn kernel with $\nu = 5/2$, lengthscale 0.3

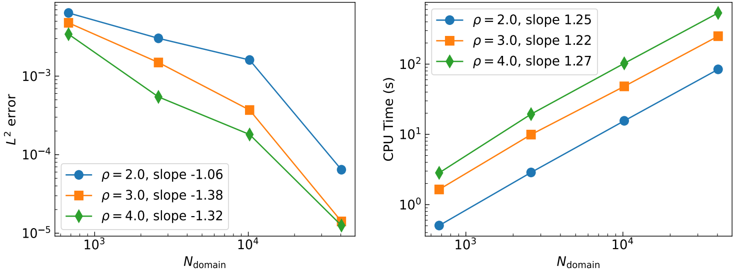


Figure: Run 3 linearization steps with initial guess $1/2\|\mathbf{x}\|^2$. Accuracy floor due to finite ρ

Outline

Summary

Near-linear complexity sparse Cholesky factorization

- Order entries from coarse to fine, with derivative entries treated as finer scales compared to pointwise entries
- This ordering leads to approximately sparse factors
- Compute the inverse Cholesky factors via optimization

Summary

Near-linear complexity sparse Cholesky factorization

- Order entries from coarse to fine, with derivative entries treated as finer scales compared to pointwise entries
- This ordering leads to approximately sparse factors
- Compute the inverse Cholesky factors via optimization

Near-linear complexity GP/kernel solver for nonlinear PDEs

- Apply the factorization algorithm into the GP solver
- Each linearization in the solver is of near-linear complexity
  $\Rightarrow$ a machine learning based near-linear complexity solver for general nonlinear PDEs (assuming the linearizations converge)

Summary

Near-linear complexity sparse Cholesky factorization

- Order entries from coarse to fine, with derivative entries treated as finer scales compared to pointwise entries
- This ordering leads to approximately sparse factors
- Compute the inverse Cholesky factors via optimization

Near-linear complexity GP/kernel solver for nonlinear PDEs

- Apply the factorization algorithm into the GP solver
- Each linearization in the solver is of near-linear complexity $\Rightarrow$ a machine learning based near-linear complexity solver for general nonlinear PDEs (assuming the linearizations converge)

Further directions

- Fast algorithms for high dimensional problems: when d is large **[Chen, Epperly, Tropp, Webber 2022]**
- Optimize the ordering and sparsity patterns?

Thank You

[Chen, Owhadi, Schäfer 2023]

Sparse Cholesky Factorization
for Solving Nonlinear PDEs via Gaussian Processes

Link: `https://arxiv.org/abs/2304.01294`