Linear-time factorization of covariance matrices

Kenneth L. Ho

Stanford University

SIAM UQ 2014

Introduction

- Covariance matrices are central to statistical modeling and UQ
  - *Example*: Gaussian processes/random fields
- Many covariance functions of interest are **not** compactly supported

Exponential (λ large):	$C(r;\lambda) = \exp(-r/\lambda)$
Matérn (ν small or λ large):	$C(r;\nu,\lambda) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}r}{\lambda}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}r}{\lambda}\right)$
Rational quadratic:	$C(r;\alpha,\lambda) = \left(1 + \frac{r^2}{2\alpha\lambda^2}\right)^{-\alpha}$

- Costs of common operations with dense covariance matrices

$y = Ax$	$\mathcal{O}(N^2)$
$x = (A + \sigma^2 I)^{-1} b$	$\mathcal{O}(N^3)$
$A = BB^{\mathsf{T}}$	$\mathcal{O}(N^3)$
$\Delta = \log\det A$	$\mathcal{O}(N^3)$

Introduction

- Covariance matrices are central to statistical modeling and UQ
  - *Example*: Gaussian processes/random fields
- Many covariance functions of interest are **not** compactly supported

Exponential (λ large):	$C(r;\lambda) = \exp(-r/\lambda)$
Matérn (ν small or λ large):	$C(r;\nu,\lambda) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}r}{\lambda}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}r}{\lambda}\right)$
Rational quadratic:	$C(r;\alpha,\lambda) = \left(1 + \frac{r^2}{2\alpha\lambda^2}\right)^{-\alpha}$

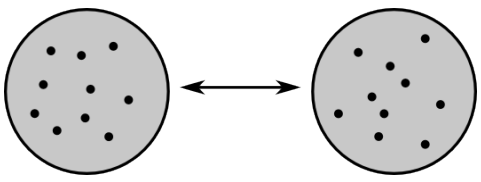
- Costs of common operations with dense covariance matrices

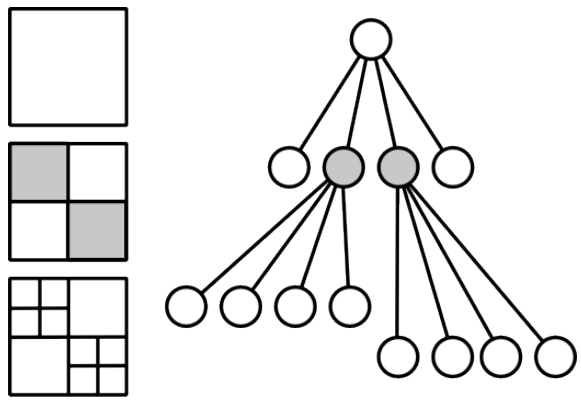
$y = Ax$	$\mathcal{O}(N^2)$	$\to \mathcal{O}(N)$
$x = (A + \sigma^2 I)^{-1} b$	$\mathcal{O}(N^3)$	$\to \mathcal{O}(N)$
$A = BB^{\mathsf{T}}$	$\mathcal{O}(N^3)$	$\to \mathcal{O}(N)$
$\Delta = \log\det A$	$\mathcal{O}(N^3)$	$\to \mathcal{O}(N)$

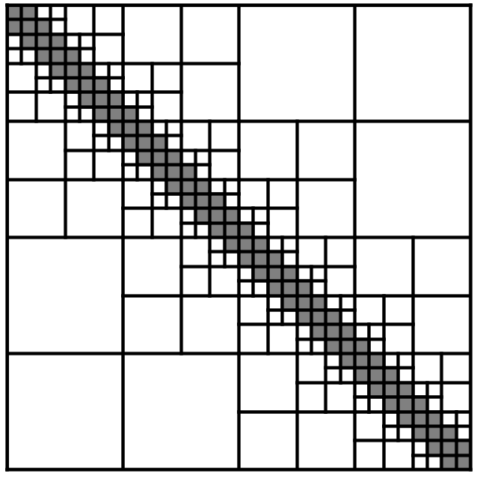
- Our goal is to accelerate these to **linear** complexity

Main observation

- Covariance matrix is dense but structured
- Far field is smooth $\Longrightarrow$ low-rank off-diagonal blocks
- Decompose and compress hierarchically
- Similar in flavor to fast multipole methods and treecodes







Overview

Problem setting:

- Matrix can be low-rank but best if rank is not too small
  - Otherwise just use low-rank techniques (random sampling)
- Low ambient dimensionality: think time or space
- Fixed-domain asymptotics ($N \to \infty$ with λ fixed)

Results:

- Generalized Cholesky factorization via recursive skeletonization
  - Originally developed for solving integral equations for elliptic PDEs
  - *Martinsson, Rokhlin (2005); Gillman, Young, Martinsson (2012); Ho, Greengard (2012); Ho, Ying (2013)*
- Optimal $\mathcal{O}(N)$ complexity with small constants
- Kernel-independent: depends weakly on specific covariance function
- Reformulation of previous methods in terms of sparsification

Overview

Problem setting:

- Matrix can be low-rank but best if rank is not too small
  - Otherwise just use low-rank techniques (random sampling)
- Low ambient dimensionality: think time or space
- Fixed-domain asymptotics ($N \to \infty$ with λ fixed)

Results:

- Generalized Cholesky factorization via recursive skeletonization
  - Originally developed for solving integral equations for elliptic PDEs
  - *Martinsson, Rokhlin (2005); Gillman, Young, Martinsson (2012); Ho, Greengard (2012); Ho, Ying (2013)*
- Optimal $\mathcal{O}(N)$ complexity with small constants
- Kernel-independent: depends weakly on specific covariance function
- Reformulation of previous methods in terms of sparsification

Tools: block elimination, interpolative decomposition, skeletonization

Block elimination

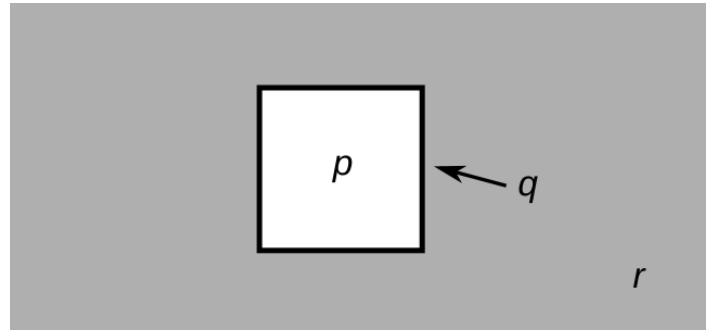
Let A be SPD with

$$A = \begin{bmatrix} A_{pp} & A_{pq} & \\ A_{qp} & A_{qq} & A_{qr} \\ & A_{rq} & A_{rr} \end{bmatrix}$$

(think of A as a sparse matrix) and define

$$S_p = \begin{bmatrix} I & -A_{pp}^{-1} A_{pq} & \\ & I & \\ & & I \end{bmatrix} \quad \Longrightarrow \quad S_p^{\mathsf{T}} A S_p = \begin{bmatrix} A_{pp} & & \\ & * & A_{qr} \\ & A_{rq} & A_{rr} \end{bmatrix}.$$

- DOFs p have been eliminated
- Interactions involving r are unchanged



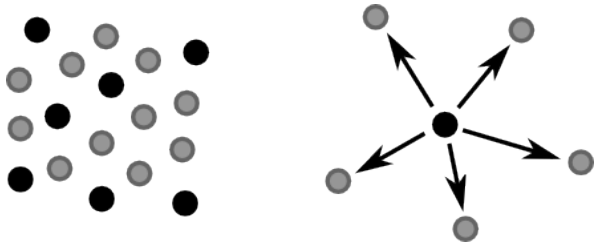
Interpolative decomposition

If $A_{:,q}$ is numerically low-rank, then there exist

- skeleton ($\hat{q}$) and redundant ($\check{q}$) columns partitioning $q = \hat{q} \cup \check{q}$
- an interpolation matrix T_q

such that

$$A_{:,\check{q}} \approx A_{:,\hat{q}} T_q.$$



- Essentially a pivoted QR written slightly differently:

$$A_{:,(\hat{q},\check{q})} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix} \approx Q_1 \begin{bmatrix} R_{11} & R_{12} \end{bmatrix}$$

$$\implies A_{:,\check{q}} \approx Q_1 R_{12} = \underbrace{Q_1 R_{11}}_{A_{:,\hat{q}}} \underbrace{\left(R_{11}^{-1} R_{12}\right)}_{T_q}$$

- Can be computed adaptively to any specified precision
- Fast randomized algorithms are available

Skeletonization

- Efficient elimination of redundant DOFs from dense matrices
- Let $A = \begin{bmatrix} A_{pp} & A_{pq} \\ A_{qp} & A_{qq} \end{bmatrix}$ with $A_{qp} = A_{pq}^{\mathsf{T}}$ low-rank
- Apply ID to A_{qp}: $A_{q\check{p}} \approx A_{q\hat{p}} T_p$
- Reorder $A = \begin{bmatrix} A_{\check{p}\check{p}} & A_{\check{p}\hat{p}} & A_{\check{p}q} \\ A_{\hat{p}\check{p}} & A_{\hat{p}\hat{p}} & A_{\hat{p}q} \\ A_{q\check{p}} & A_{q\hat{p}} & A_{qq} \end{bmatrix}$, define $Q_p = \begin{bmatrix} I & & \\ -T_p & I & \\ & & I \end{bmatrix}$
- **Sparsify** via ID: $Q_p^{\mathsf{T}} A Q_p \approx \begin{bmatrix} * & * & \\ * & A_{\hat{p}\hat{p}} & A_{\hat{p}q} \\ & A_{q\hat{p}} & A_{qq} \end{bmatrix}$
- Block eliminate: $S_p^{\mathsf{T}} Q_p^{\mathsf{T}} A Q_p S_p \approx \begin{bmatrix} * & & \\ & * & A_{\hat{p}q} \\ & A_{q\hat{p}} & A_{qq} \end{bmatrix}$

Algorithm: recursive skeletonization

Build tree.
for each level $\ell = 0, 1, 2, \ldots, L$ from finest to coarsest **do**
    Let C_ℓ be the set of all cells on level ℓ.
    for each cell $c \in C_\ell$ **do**
        Skeletonize remaining DOFs in c.
    end for
end for

Algorithm: recursive skeletonization

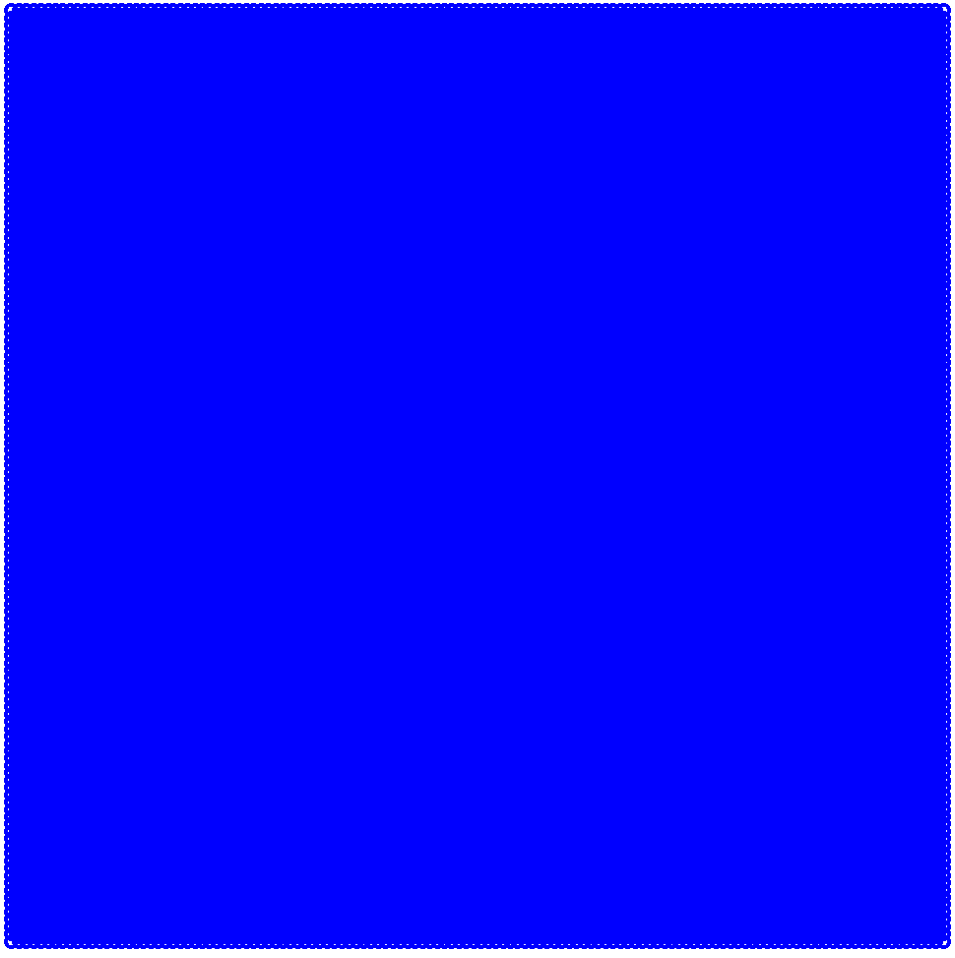
Build tree.
for each level $\ell = 0, 1, 2, \ldots, L$ from finest to coarsest **do**
    Let C_ℓ be the set of all cells on level ℓ.
    for each cell $c \in C_\ell$ **do**
        Skeletonize remaining DOFs in c.
    end for
end for

Example. Matérn ($\nu = 3/2$) on the unit square:

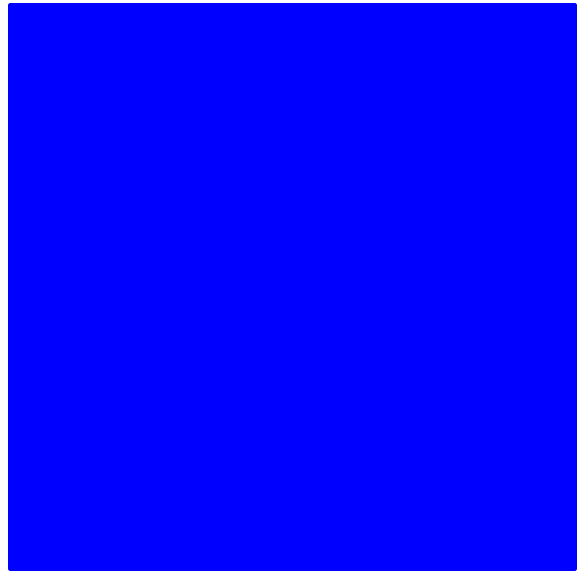
$$C(r; \lambda) = \left(1 + \frac{\sqrt{3}r}{\lambda}\right) \exp\left(-\frac{\sqrt{3}r}{\lambda}\right), \quad \lambda = \frac{1}{4}.$$

Approximate to relative precision $\epsilon = 10^{-6}$: $N = 16384 \to 543$.

RS in 2D: level 0

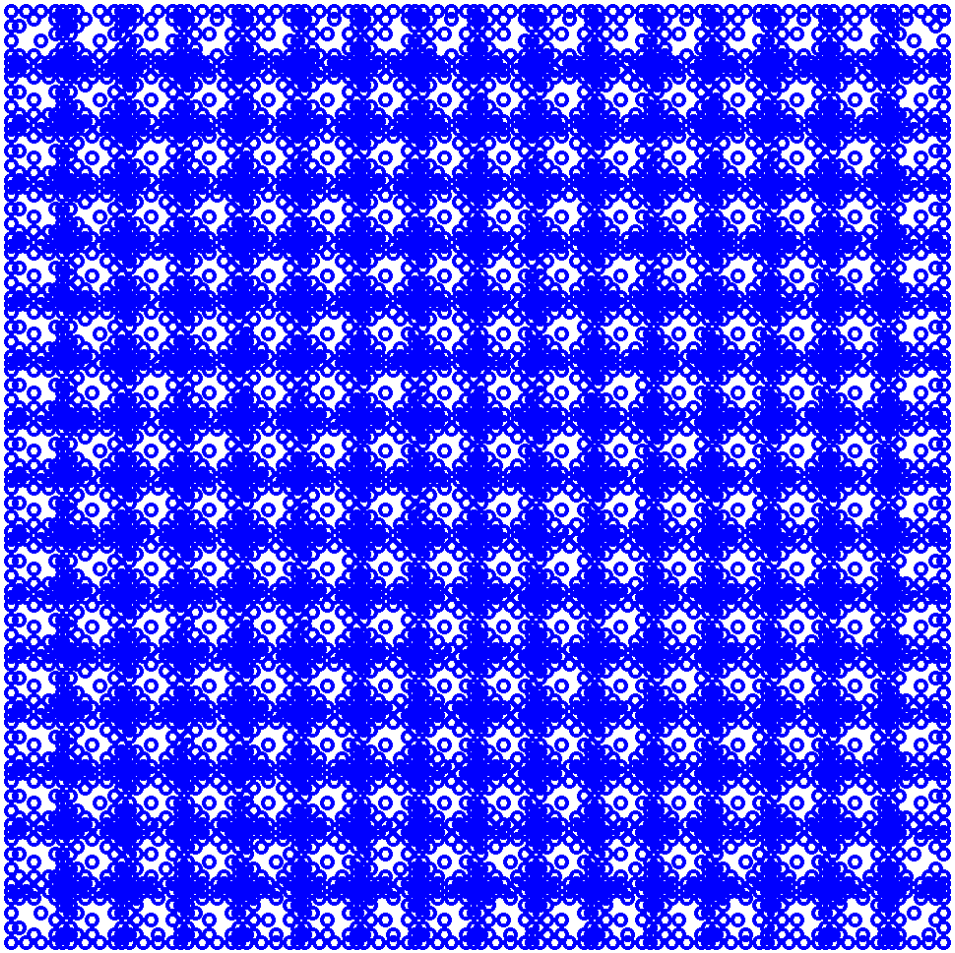


domain

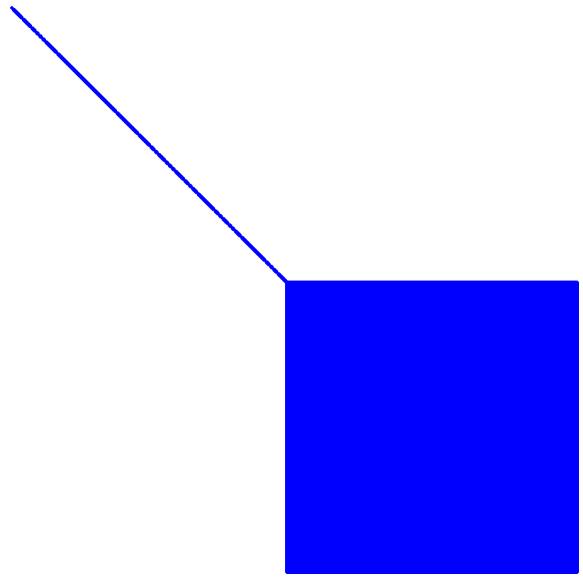


matrix

RS in 2D: level 1

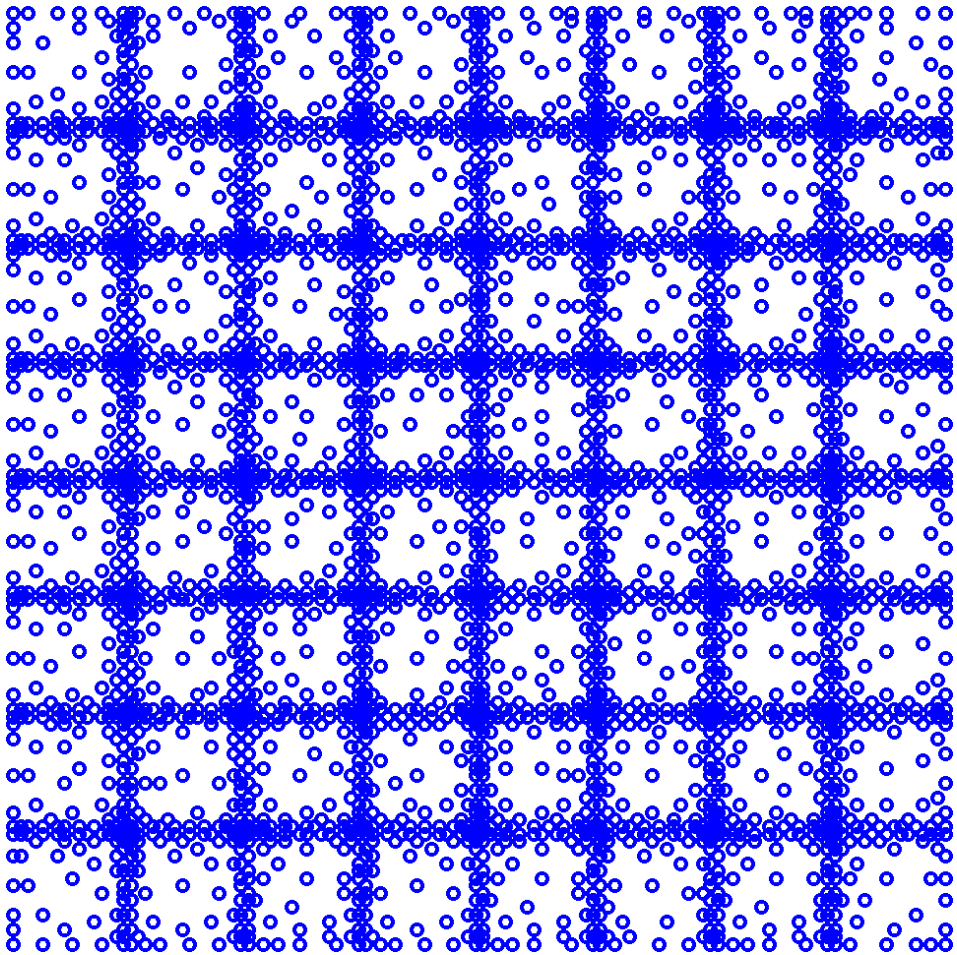


domain

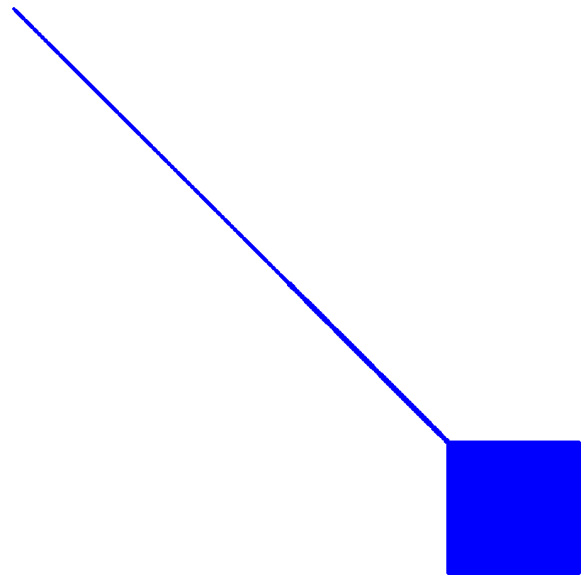


matrix

RS in 2D: level 2

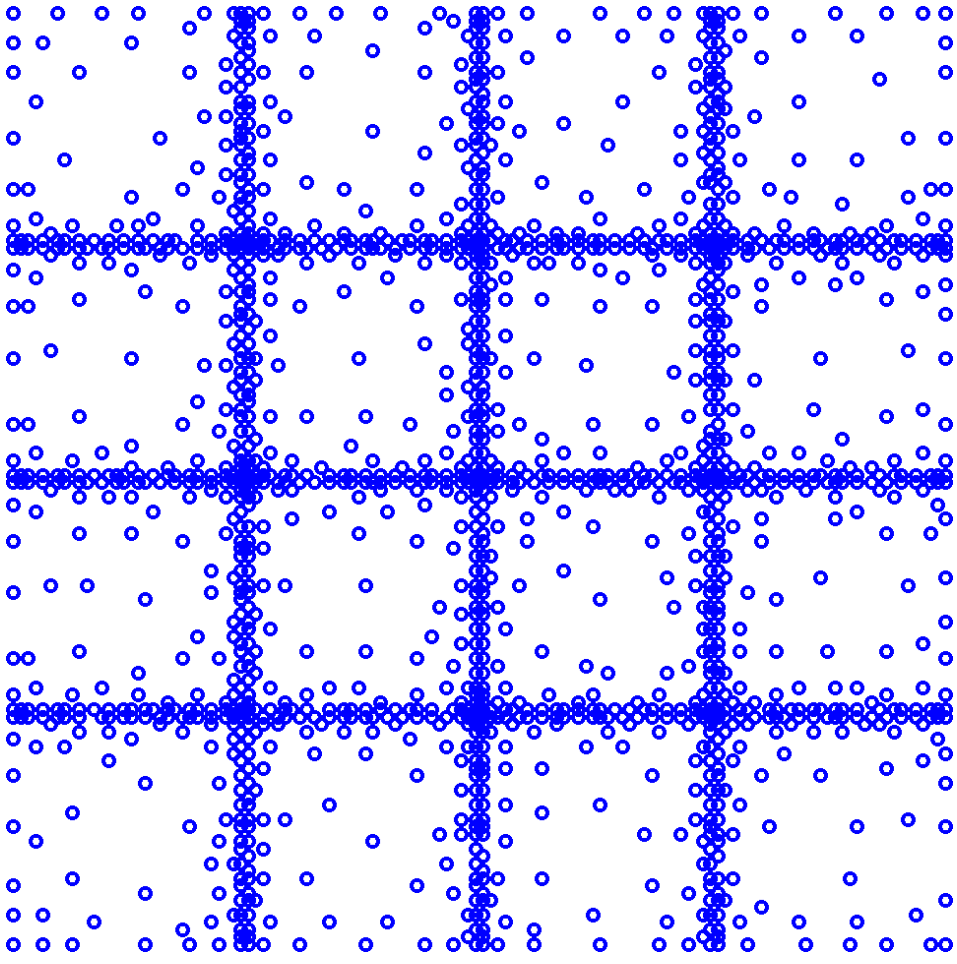


domain

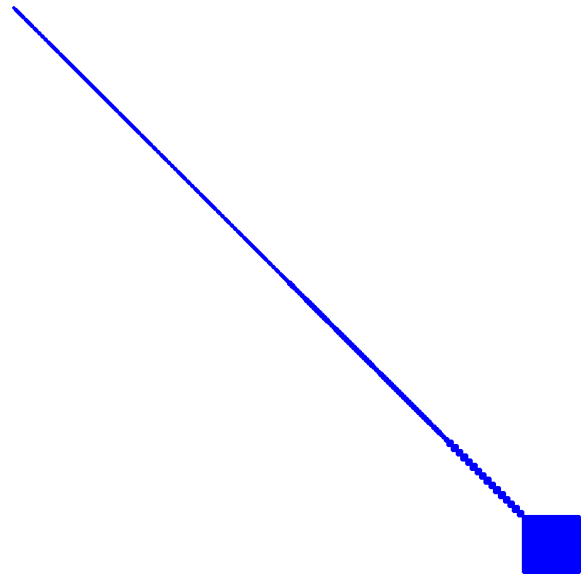


matrix

RS in 2D: level 3

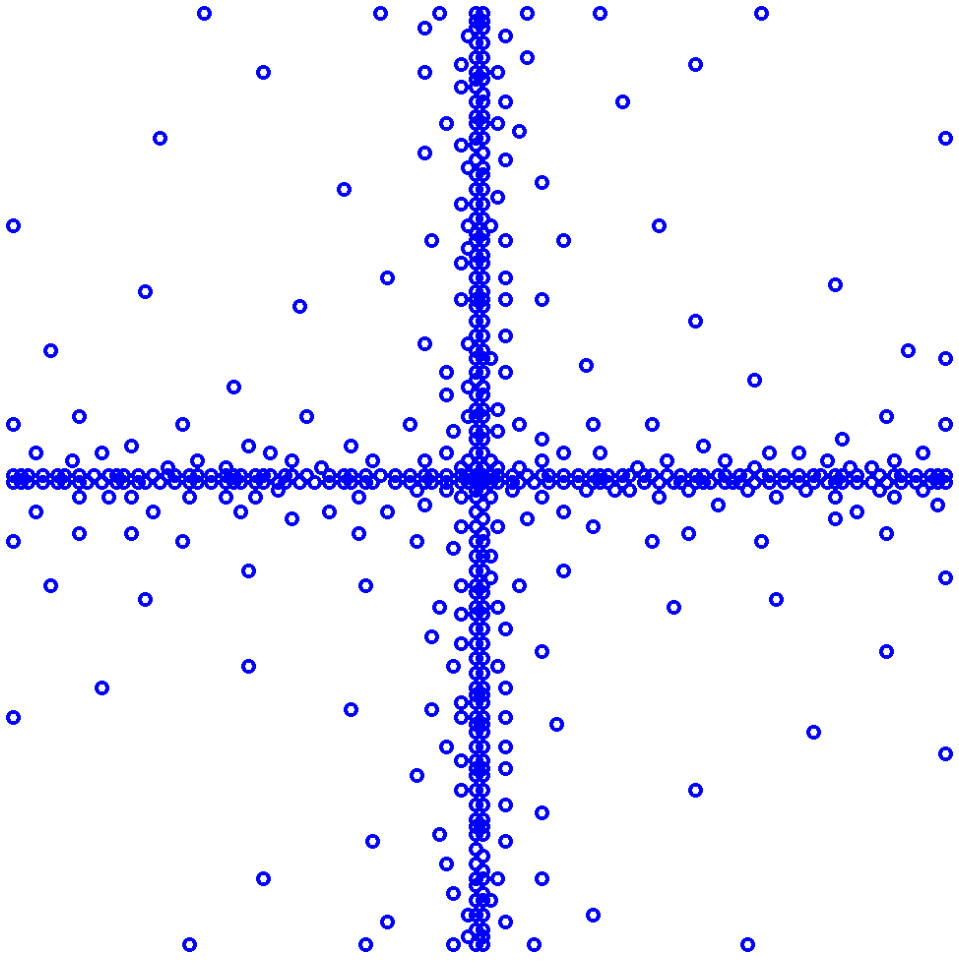


domain

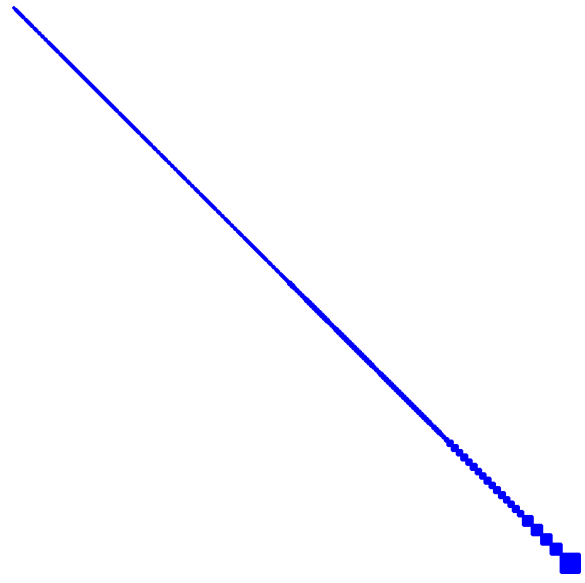


matrix

RS in 2D: level 4



domain



matrix

Matrix factorization

- Skeletonization operators:

$$U_\ell = \prod_{c \in C_\ell} Q_c S_c, \qquad Q_p = \begin{bmatrix} I & & \\ * & I & \\ & & I \end{bmatrix}, \quad S_p = \begin{bmatrix} I & * & \\ & I & \\ & & I \end{bmatrix},$$

- Symmetric block diagonalization:

$$D \approx U_{L-1}^{\mathsf{T}} \cdots U_0^{\mathsf{T}} A U_0 \cdots U_{L-1}$$

- Generalized Cholesky/LDL^{T} decomposition:

$$A \approx U_0^{-\mathsf{T}} \cdots U_{L-1}^{-\mathsf{T}} D U_{L-1}^{-1} \cdots U_0^{-1}$$
$$A^{-1} \approx U_0 \cdots U_{L-1} D^{-1} U_L^{\mathsf{T}} \cdots U_0^{\mathsf{T}}$$

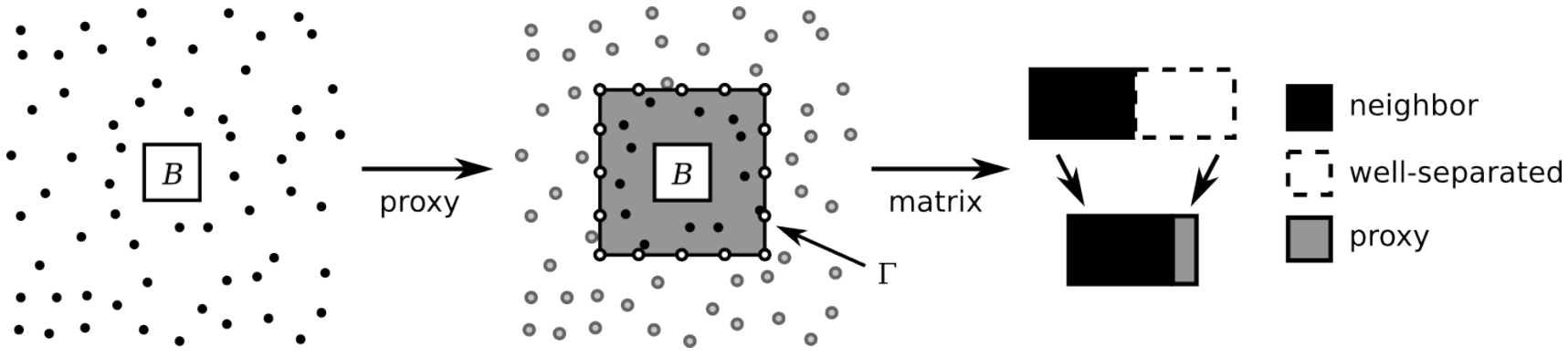
- Cholesky square root: take half the factorization
- Determinant: $\det A = \det D$
- All operations are **very cheap** once the factorization has been constructed

Accelerated compression

- Main cost of algorithm is computing IDs
- Each ID requires a tall-and-skinny QR (on $A_{p^C,p}$)
- Naive compression is global and therefore at least $\mathcal{O}(N^2)$

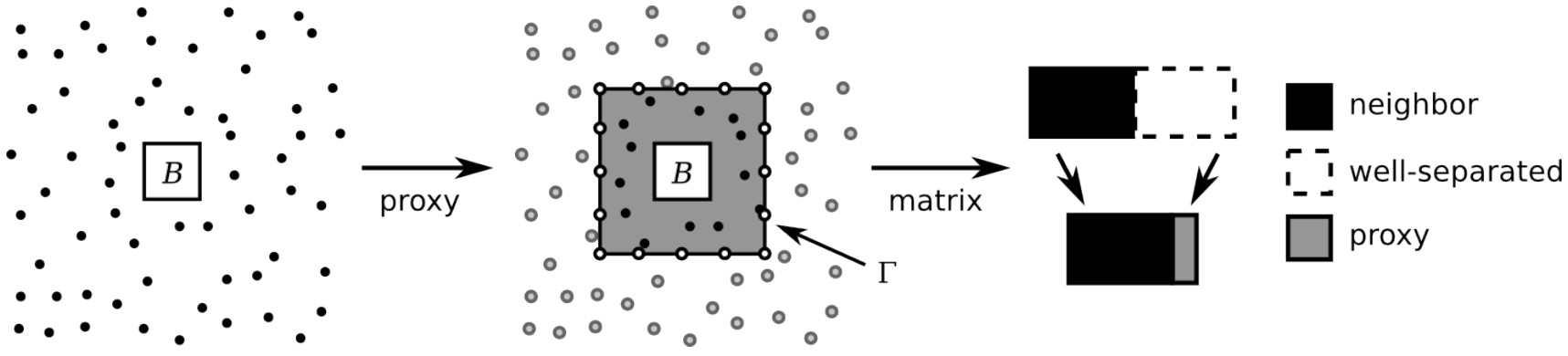
Accelerated compression

- Main cost of algorithm is computing IDs
- Each ID requires a tall-and-skinny QR (on $A_{p^C,p}$)
- Naive compression is global and therefore at least $\mathcal{O}(N^2)$
- Idea from elliptic PDEs: use Green's theorem
  - Capture well-separated interactions via a local proxy surface
  - Keep neighbors explicitly



Accelerated compression

- Main cost of algorithm is computing IDs
- Each ID requires a tall-and-skinny QR (on $A_{p^C,p}$)
- Naive compression is global and therefore at least $\mathcal{O}(N^2)$
- Idea from elliptic PDEs: use Green's theorem
  - Capture well-separated interactions via a local proxy surface
  - Keep neighbors explicitly



- In our case, no Green's theorem but should still be able to sample sparsely
- Use a few concentric rings of radius 1, 2, 4, 8, etc.
- **Not rigorous** but works well in many cases

Theorem

If the off-diagonal block rank is bounded, then constructing the approximate factorization requires $\mathcal{O}(N)$ operations.

Theorem

If the off-diagonal block rank is bounded, then constructing the approximate factorization requires $\mathcal{O}(N)$ operations.

- For fixed-domain asymptotics, interaction length scale is independent of N
- Therefore, number of "distinct" interactions is bounded
- Rank is bounded $\implies$ **linear** complexity
- Note: constant has the form $\mathcal{O}(2^d)$

Theorem

If the off-diagonal block rank is bounded, then constructing the approximate factorization requires $\mathcal{O}(N)$ operations.

- For fixed-domain asymptotics, interaction length scale is independent of N
- Therefore, number of "distinct" interactions is bounded
- Rank is bounded $\implies$ **linear** complexity
- Note: constant has the form $\mathcal{O}(2^d)$

What about increasing-domain asymptotics?

- Number of interactions grows as $1/\lambda \sim N^{1/d}$
- Cost becomes $\mathcal{O}\left(N^{3(1-1/d)}\right)$
- Must do additional work to recover linear complexity
  - Hierarchical interpolative factorization: *Ho, Ying (2013)*

Numerical benchmarks

Matérn ($\nu = 3/2$, $\lambda = 1/8$) with nugget effect of $\sigma^2 = 0.01$

- Point distributions: unit line (1D) or square (2D)

d	ϵ	N	r	t_f (s)	$t_{a/s}$ (s)	t_d (s)	e_a	e_d
1D	10^{-08}	262144	4	1.8e+1	4.8e−1	9.8e−2	1.1e−08	1.2e−9
		524288	4	3.5e+1	1.1e+0	2.0e−1	4.3e−07	1.8e−7
		1048576	4	7.0e+1	2.0e+0	3.9e−1	4.7e−07	1.1e−7
1D	10^{-12}	262144	4	1.8e+1	4.7e−1	9.9e−2	2.1e−13	—
		524288	4	3.5e+1	9.3e−1	2.0e−1	2.8e−13	—
		1048576	4	7.0e+1	1.9e+0	4.0e−1	3.0e−13	—
2D	10^{-06}	256^2	214	7.4e+0	1.2e−1	1.8e−2	5.8e−07	2.6e−6
		512^2	219	2.8e+1	4.1e−1	7.3e−2	1.8e−06	4.1e−6
		1024^2	220	1.1e+2	1.6e+0	2.9e−1	1.7e−06	8.0e−6
2D	10^{-09}	256^2	1081	3.2e+1	2.1e−1	1.8e−2	5.4e−10	—
		512^2	1227	6.7e+1	5.9e−1	7.4e−2	1.1e−09	—
		1024^2	1301	1.7e+2	1.9e+0	3.0e−1	4.0e−09	—

Example: GP regression and conditional sampling

- **Unknown** function $f(x)$ on $[0, 1]$
- Prior: zero mean, Matérn covariance $C(x, x')$ with $\nu = 3/2$ and $\lambda = 1/8$
- Measurements $y_1 = f(x_1) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2)$, at N uniform random points
- Estimate values of $y_2 = f(x_2)$ at N equispaced points:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \right), \qquad \begin{aligned} A_{11} &= C(x_1, x_1) + \sigma^2 I \\ A_{21} &= C(x_2, x_1) \\ A_{22} &= C(x_2, x_2) \end{aligned}$$

$$\implies y_2 \mid y_1 \sim \mathcal{N}(\mu_{\text{post}}, A_{\text{post}}), \qquad \begin{aligned} \mu_{\text{post}} &= A_{21} A_{11}^{-1} y_1 \\ A_{\text{post}} &= A_{22} - A_{21} A_{11}^{-1} A_{12} \end{aligned}$$

- $N \sim 10^6$, $\sigma^2 = 0.01$: 273 s to compute μ_{post} to precision 10^{-5}
- Generate conditional samples via $\hat{y}_2 = z_2 - A_{21} A_{11}^{-1} z_1$, where $z \sim \mathcal{N}(0, A)$
- Estimate posterior variance to precision 10^{-2} by sampling: $\sim$ 30 min

Summary

- Efficient factorization of covariance matrices
  - Apply, solve, square root, determinant, etc.
  - Extends to general structured matrices with low-rank off-diagonal blocks
- **Linear** complexity under fixed-domain asymptotics
  - Can extend to increasing domain asymptotics with some work
- Key idea: sparsification and elimination (skeletonization) via the ID
- Naturally parallelizable: independent for-loops up a tree
- However, effective only in low dimensions (great for time course data!)
  - High-dimensional setting will require new ideas

Summary

- Efficient factorization of covariance matrices
  - Apply, solve, square root, determinant, etc.
  - Extends to general structured matrices with low-rank off-diagonal blocks
- **Linear** complexity under fixed-domain asymptotics
  - Can extend to increasing domain asymptotics with some work
- Key idea: sparsification and elimination (skeletonization) via the ID
- Naturally parallelizable: independent for-loops up a tree
- However, effective only in low dimensions (great for time course data!)
  - High-dimensional setting will require new ideas

Thanks!