Skeleton preconditioner for coarse-graining integral equations

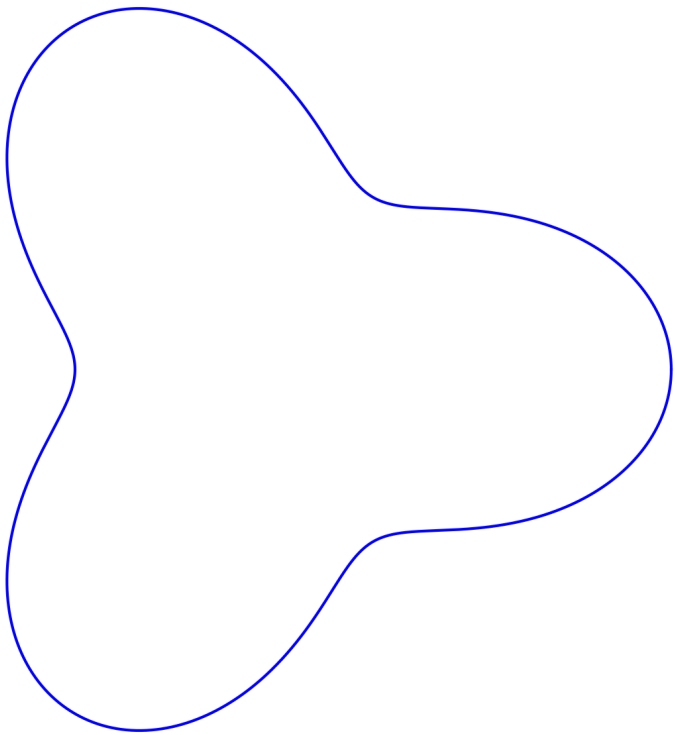
Kenneth L. Ho

Stanford University

SIAM CSE 2013

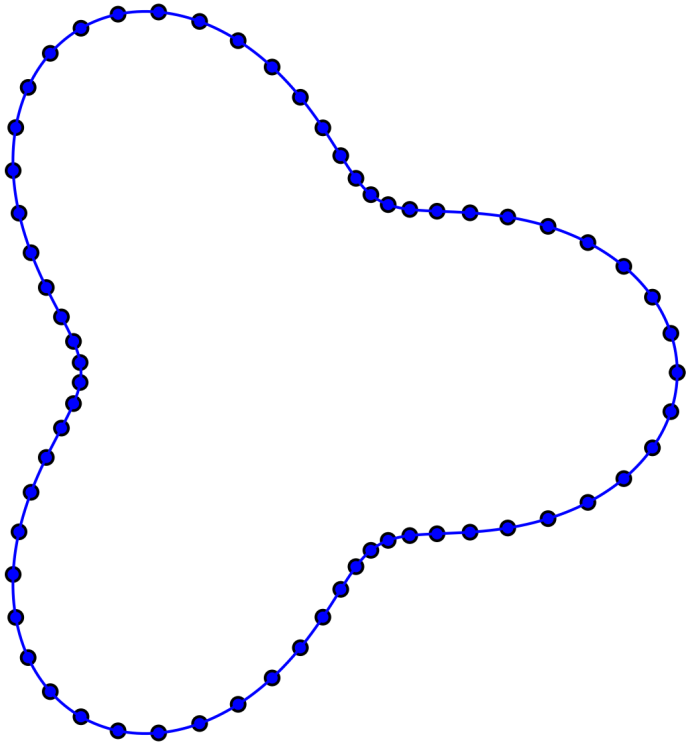
Motivation

An easy problem:



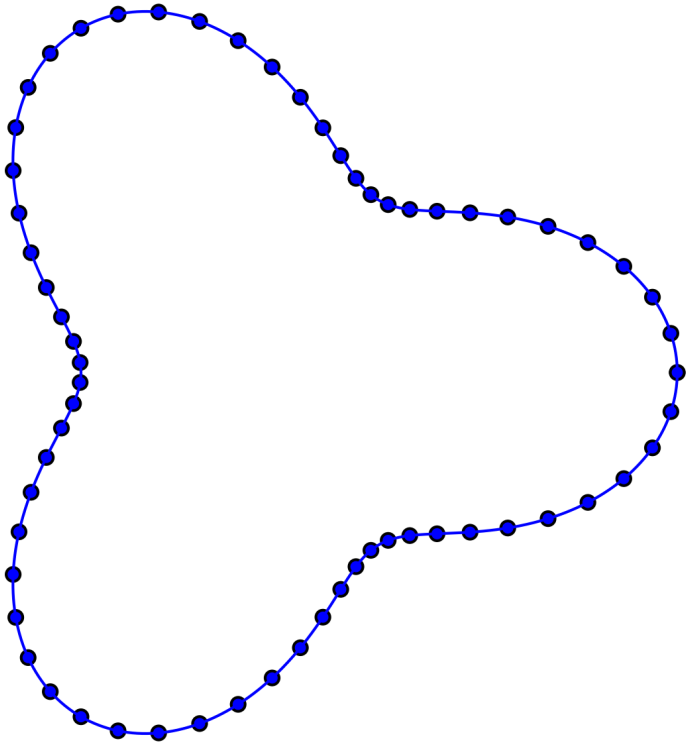
Motivation

An easy problem:

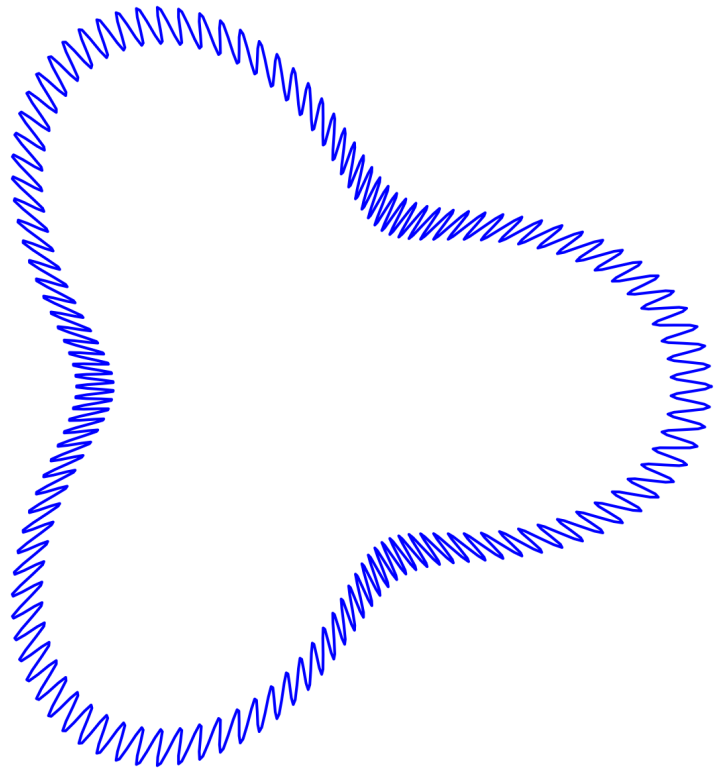


Motivation

An easy problem:

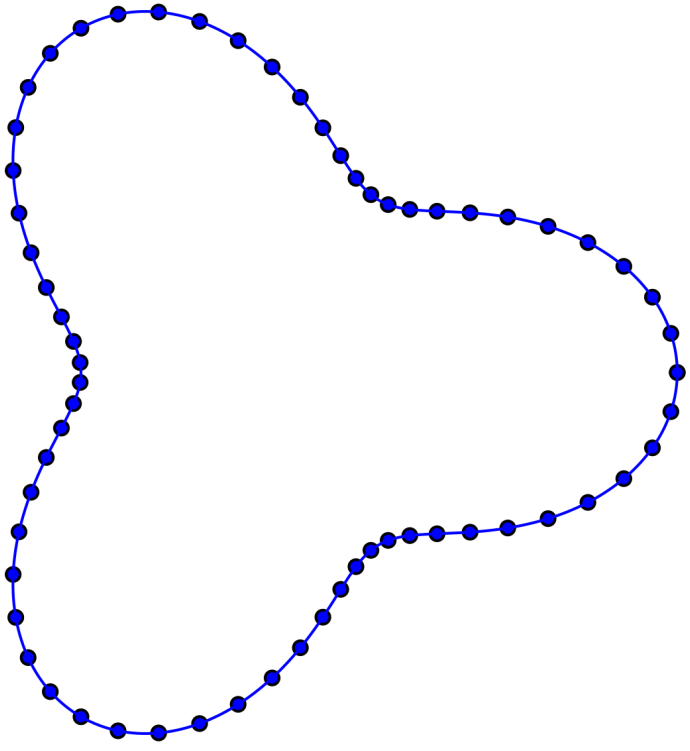


A harder problem:

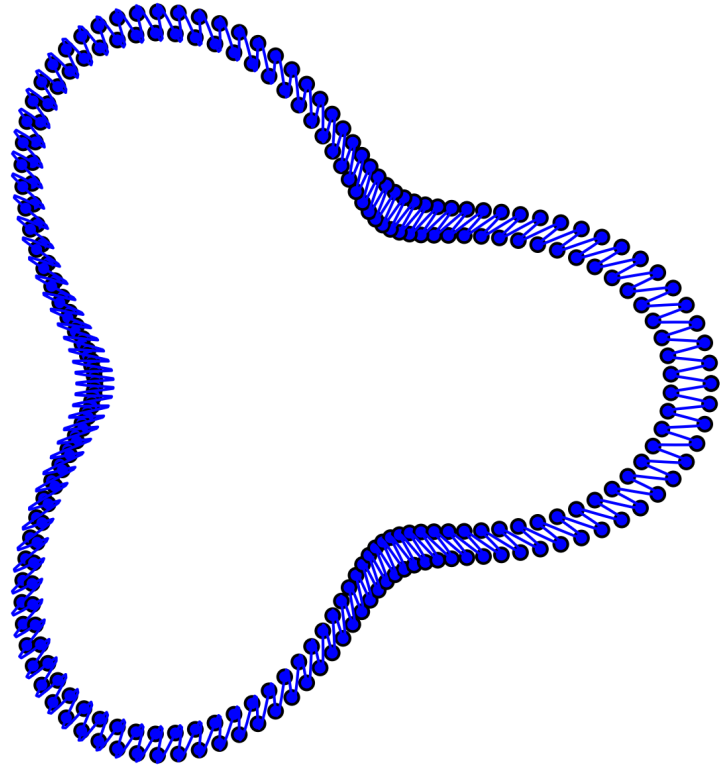


Motivation

An easy problem:

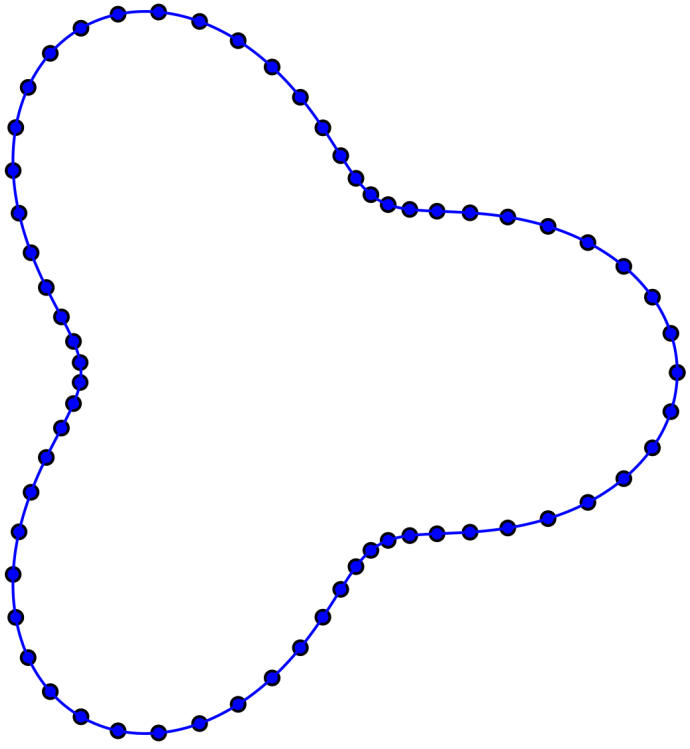


A harder problem:

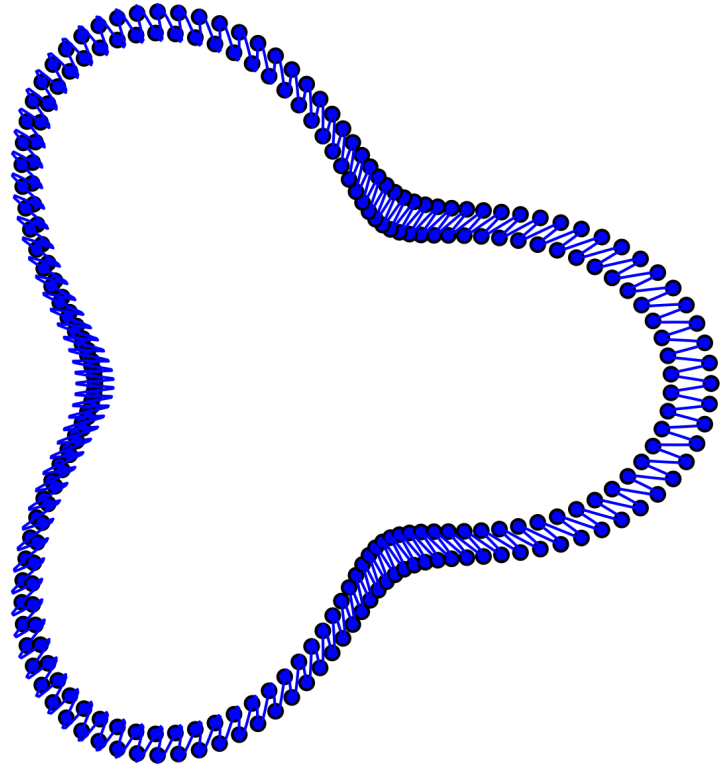


Motivation

An easy problem:



A harder problem:



Forced to iterate on a system of high dimension...

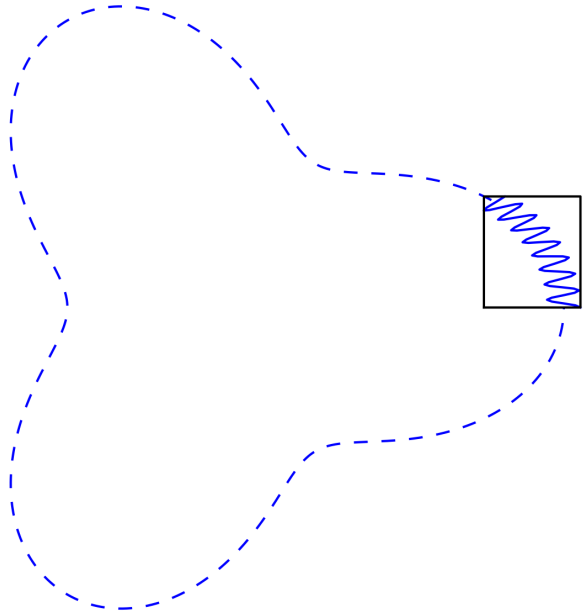
Motivation

Can we do better? Inspired by RCIP (Helsing and Ojala), the answer is yes.

Motivation

Can we do better? Inspired by RCIP (Helsing and Ojala), the answer is yes.

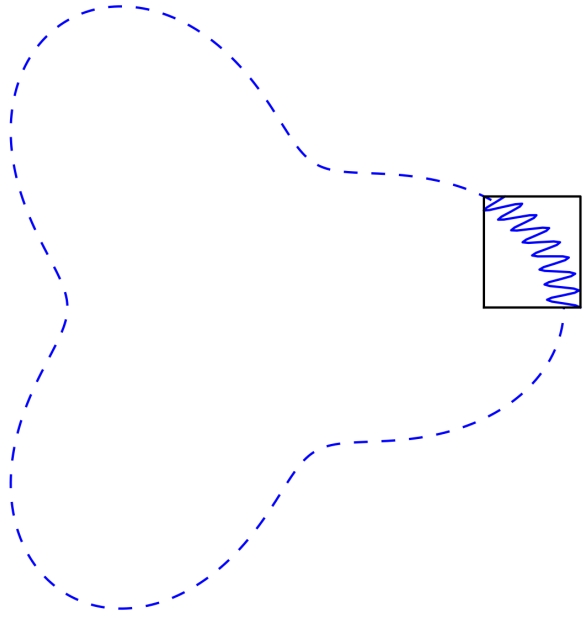
- Fine-scale structure only needs to be seen locally
- From far away, a coarse description is adequate (skeletons)
- Compute interactions between subdomains using skeletons
- Iterate on compressed skeleton system



Motivation

Can we do better? Inspired by RCIP (Helsing and Ojala), the answer is yes.

- Fine-scale structure only needs to be seen locally
- From far away, a coarse description is adequate (skeletons)
- Compute interactions between subdomains using skeletons
- Iterate on compressed skeleton system



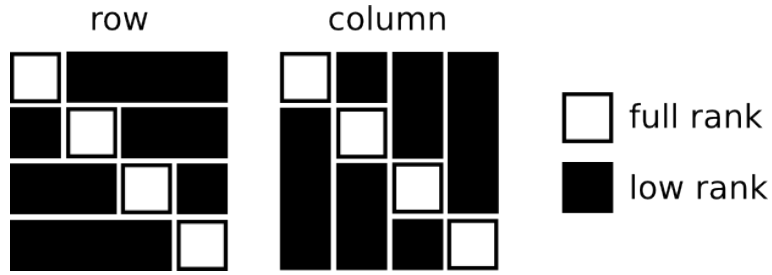
This talk in a nutshell: **RCIP in the language of skeletons**.

- All linear algebra; completely transparent; easy to apply, optimize, extend
- *Outline*: block separable matrix, interpolative decomposition, fast direct solver, skeleton preconditioner, numerical results

Block separable matrix

Low-rank off-diagonal block rows and columns:

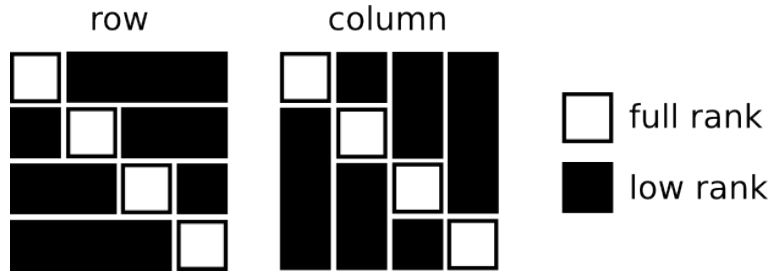
$$\underbrace{\begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix}}_{A_{ij}} = \underbrace{\begin{bmatrix} \times \\ \times \end{bmatrix}}_{U_i} \underbrace{\begin{bmatrix} \times \end{bmatrix}}_{S_{ij}} \underbrace{\begin{bmatrix} \times & \times \end{bmatrix}}_{V_j^*}, \quad i \neq j$$



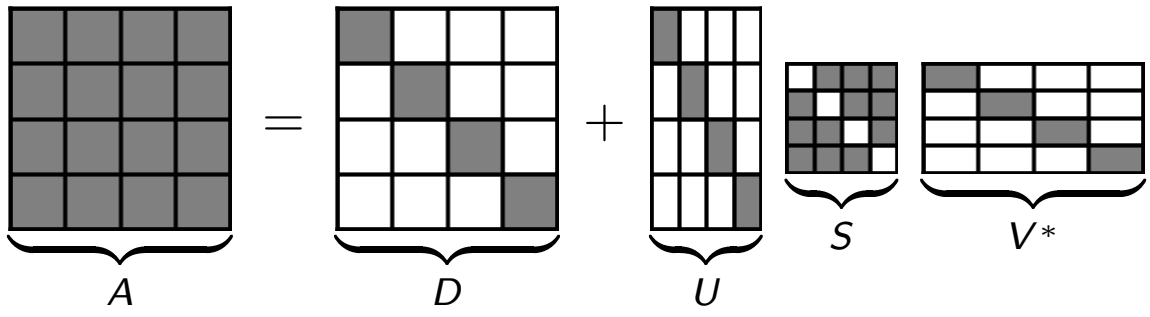
Block separable matrix

Low-rank off-diagonal block rows and columns:

$$\underbrace{\begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix}}_{A_{ij}} = \underbrace{\begin{bmatrix} \times \\ \times \end{bmatrix}}_{U_i} \underbrace{\begin{bmatrix} \times \end{bmatrix}}_{S_{ij}} \underbrace{\begin{bmatrix} \times & \times \end{bmatrix}}_{V_j^*}, \quad i \neq j$$



Matrix decomposition:



To solve $Ax = b$, let $z = V^*x$, $y = Sz$, $\Lambda = \left(V^*D^{-1}U\right)^{-1}$:

$$(\Lambda + S)z = \Lambda V^* D^{-1} b, \quad x = D^{-1}(b - Uy)$$

Interpolative decomposition

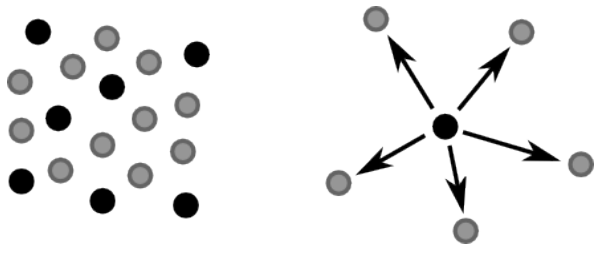
Factorization of low-rank matrix:

$$\underbrace{A}_{m\times n} = \underbrace{B}_{m\times k}\ \underbrace{P}_{k\times n}$$

- B is a column-submatrix of A
- P is an interpolation matrix (containing the $k \times k$ identity)

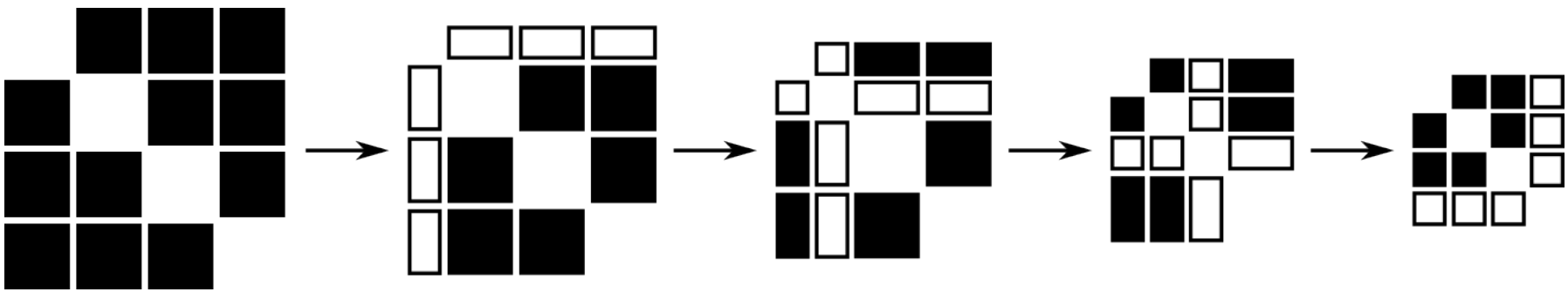
Essentially an RRQR written slightly differently

- ID compresses the column space; to compress the row space, apply to A^*
- Retained rows and columns: skeletons
- Can adaptively compute the ID to any specified precision



Matrix compression

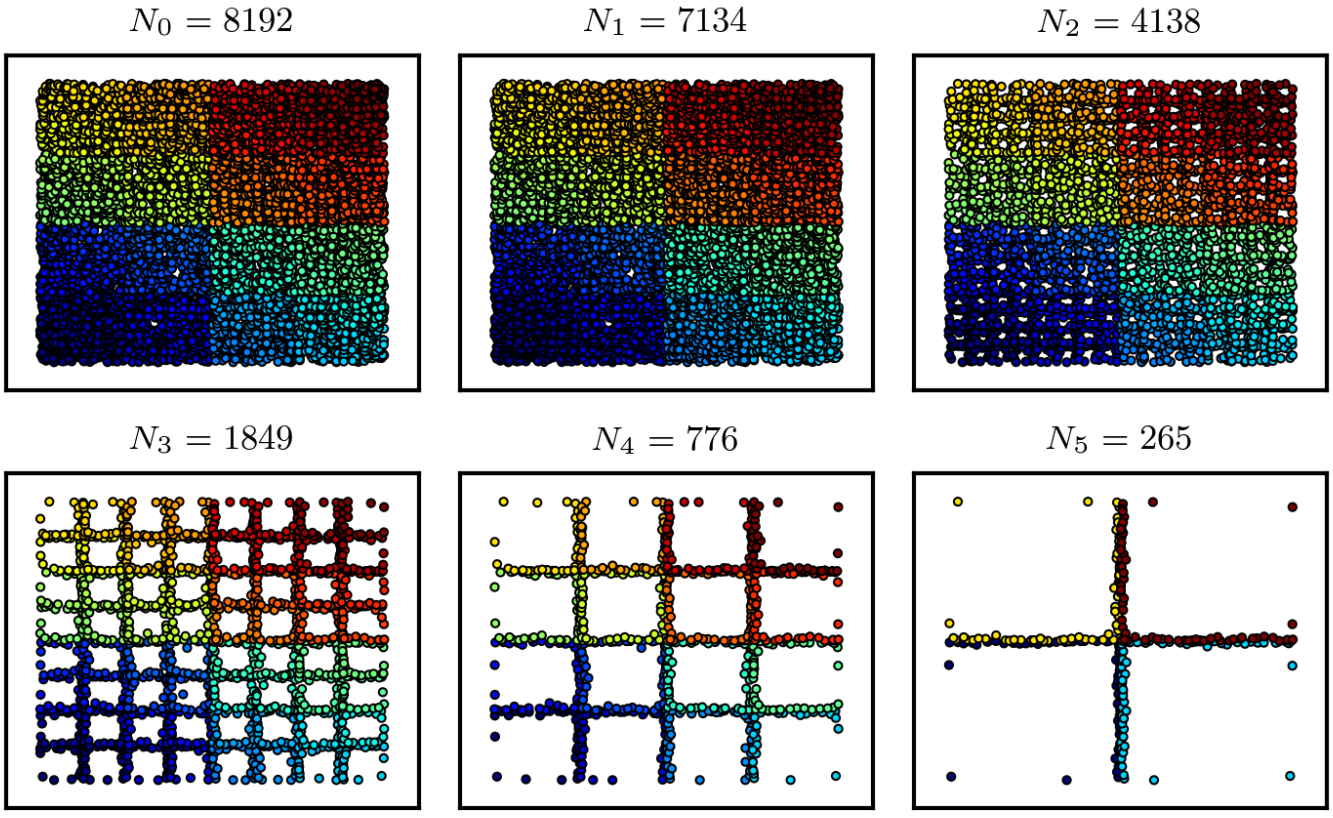
- Extract diagonal blocks D
- Compress row spaces of off-diagonal block rows $\longrightarrow U$
- Compress column spaces of off-diagonal block columns $\longrightarrow V^*$
- Subselect S from intersection of row and column skeletons
- Approximate off-diagonal blocks by $A_{ij} \approx U_i S_{ij} V_j^*$



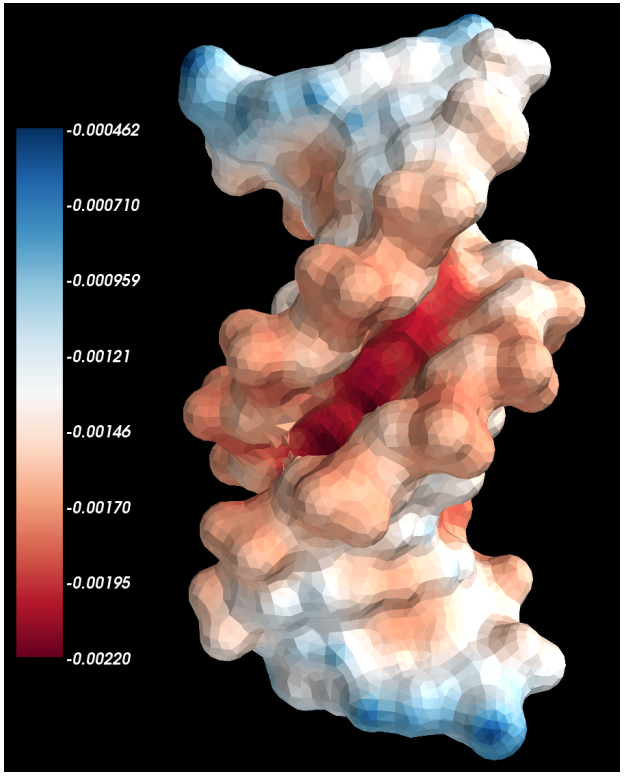
Skeletonization

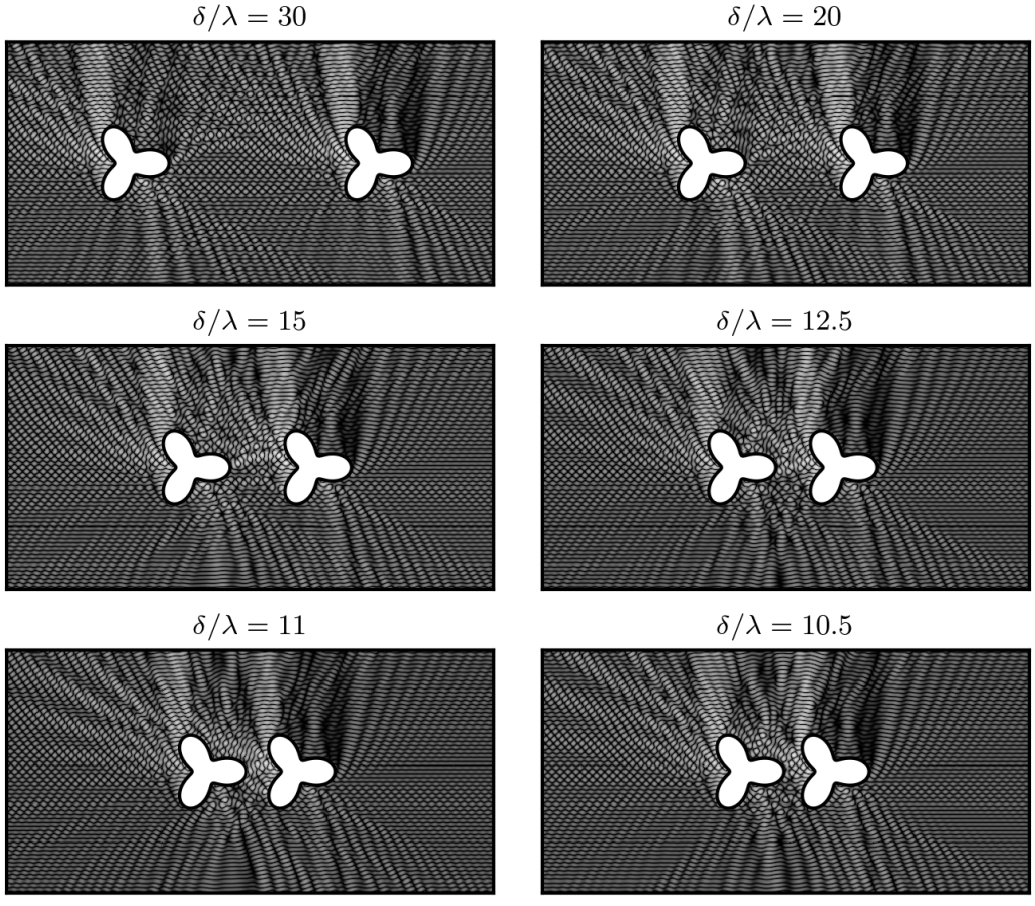
Fast direct solver

- Skeletonize matrix: $A = D + USV^*$
- Reduce $Ax = b$ to $(\Lambda + S)z = \Lambda V^* D^{-1} b$
- Since $\Lambda + S$ is "just like A", skeletonize again by working up a tree
- Repeat until at root, then solve densely



Fast direct solver





Remarks

- Many authors: Bremer, Gillman, Greengard, Martinsson, Rokhlin, ...
- Great for quasi-1D problems at low frequency (e.g., 2D Laplace BIE)
- Not so great in higher dimensions (yet) or at high frequency

In many cases, iterating is still the best approach.

Remarks

- Many authors: Bremer, Gillman, Greengard, Martinsson, Rokhlin, ...
- Great for quasi-1D problems at low frequency (e.g., 2D Laplace BIE)
- Not so great in higher dimensions (yet) or at high frequency

In many cases, iterating is still the best approach.

- But iterative solvers deal poorly with singularities
- Hybrid strategy: precondition with local direct solves
- Design goals:
  - Flexibility, generality, compatibility with fast algorithms
  - Iterate only on 'true' problem size

Skeleton preconditioner

- Block diagonal preconditioner:

$$D^{-1}Ax = D^{-1}b \quad \text{or} \quad (AD^{-1})Dx = b$$

  - Fast algorithms to invert D and apply A
  - But system has large dimension

Skeleton preconditioner

- Block diagonal preconditioner:

$$D^{-1}Ax = D^{-1}b \quad \text{or} \quad (AD^{-1})Dx = b$$

  - Fast algorithms to invert D and apply A
  - But system has large dimension

- Compressed skeleton system:

$$(\Lambda + S)z = \Lambda V^* D^{-1} b, \quad \Lambda = \left(V^* D^{-1} U\right)^{-1}$$

  - Reveals true problem size (charge basis)
  - Requires identical row/column skeletons, how to construct Λ efficiently?

Skeleton preconditioner

- Block diagonal preconditioner:

$$D^{-1}Ax = D^{-1}b \quad \text{or} \quad (AD^{-1})Dx = b$$

  - Fast algorithms to invert D and apply A
  - But system has large dimension

- Compressed skeleton system:

$$(\Lambda + S)z = \Lambda V^* D^{-1} b, \quad \Lambda = \left(V^* D^{-1} U\right)^{-1}$$

  - Reveals true problem size (charge basis)
  - Requires identical row/column skeletons, how to construct Λ efficiently?

- Skeleton preconditioner:

$$\left(I + V^* D^{-1} U S\right) z = V^* D^{-1} b$$

  - Small system size, fast algorithms, no skeleton restrictions
  - Diagonally preconditioned version of above

Skeleton preconditioner

- Block diagonal preconditioner:

$$D^{-1}Ax = D^{-1}b \quad \text{or} \quad (AD^{-1})Dx = b$$

  - Fast algorithms to invert D and apply A
  - But system has large dimension

- Compressed skeleton system:

$$(\Lambda + S)z = \Lambda V^* D^{-1} b, \quad \Lambda = \left(V^* D^{-1} U\right)^{-1}$$

  - Reveals true problem size (charge basis)
  - Requires identical row/column skeletons, how to construct Λ efficiently?

- Skeleton preconditioner:

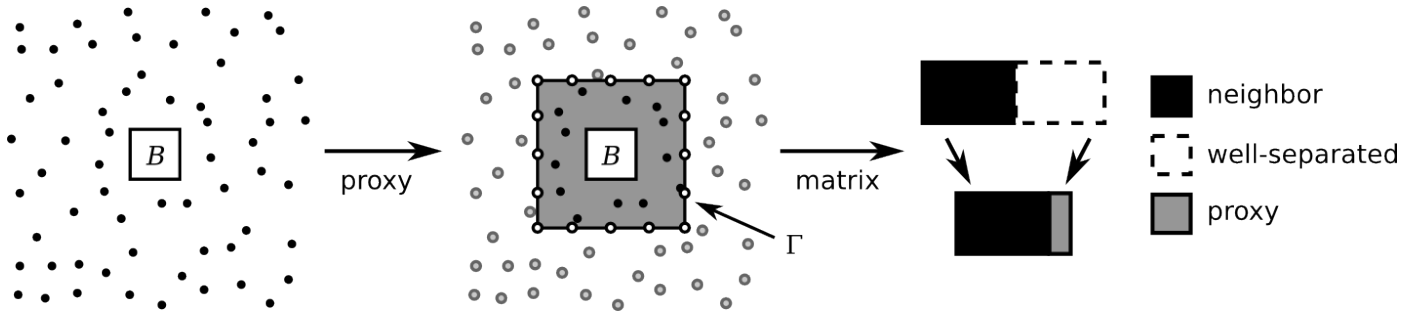
$$\left(I + V^* D^{-1} U S\right) z = V^* D^{-1} b$$

  - Small system size, fast algorithms, no skeleton restrictions
  - Diagonally preconditioned version of above

... and that's it!

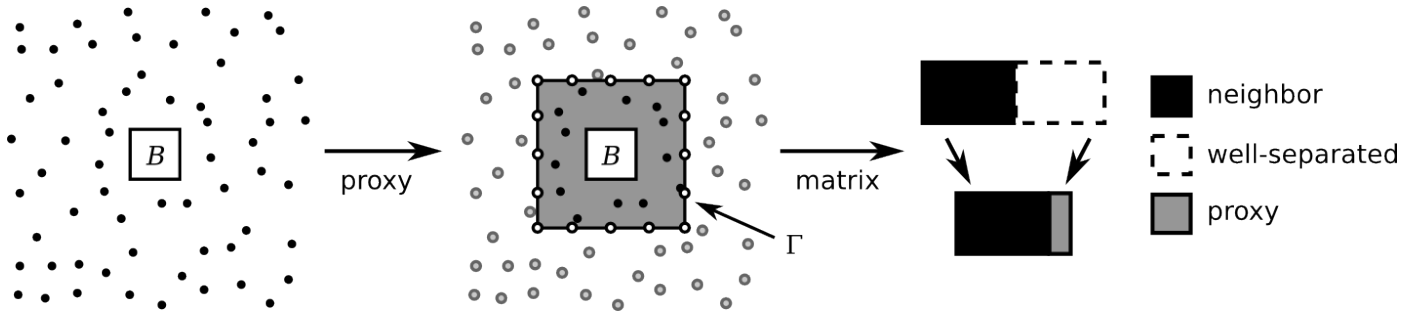
Algorithm

- Choose subdomains (requires scale separation) and skeletonize
  - Based on feature size, wavelength, etc.
  - Compute IDs hierarchically and use proxy trick:



Algorithm

- Choose subdomains (requires scale separation) and skeletonize
  - Based on feature size, wavelength, etc.
  - Compute IDs hierarchically and use proxy trick:



- Direct solve on each subdomain (fast direct solver, dense)
- Iterate on preconditioned skeleton system (fast multiplication)

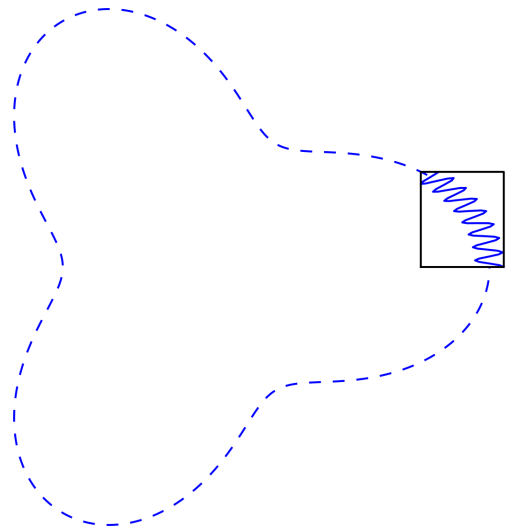
$$\left(I + V^* D^{-1} U S\right) z = V^* D^{-1} b$$

- Local solves to recover $x = D^{-1}(b - USz)$

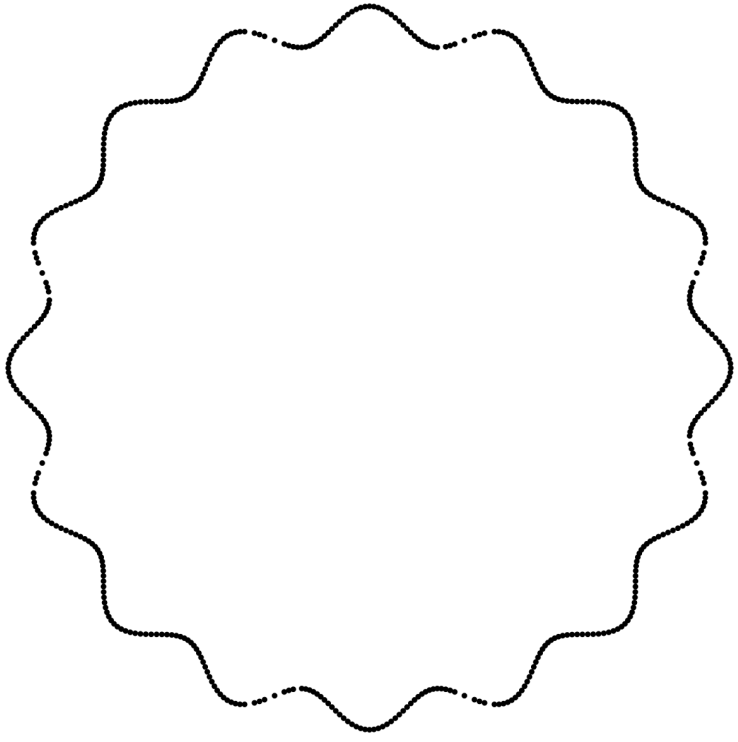
Remarks

- Coarse-grained description of subdomain interactions
  - x: fine variables
  - z: coarse variables, good outside of own subdomain
- Capture moments of solution (cf. moments of source in FMM)
- Evaluate field using coarse variables plus local 'corrections'
- If only far field needed (e.g., RCS), no further work!
- Explicit control over coarse basis during compression
- Example: Neumann problem
  - Single-layer representation: $u = \mathbb{S}\sigma$
  - Integral equation: $u' = \pm\frac{1}{2}\sigma + \mathbb{S}'\sigma$
  - Compress contributions from both $\mathbb{S}$ and $\mathbb{S}'$
- Same asymptotic complexity as iterative solver
- Just like RCIP but using

$$\left(I + V^* D^{-1} U S\right) z = V^* D^{-1} b$$



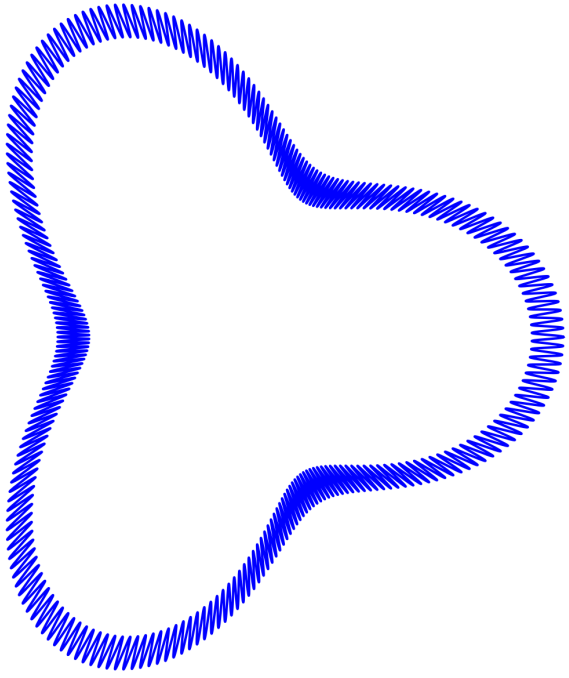
Numerical results

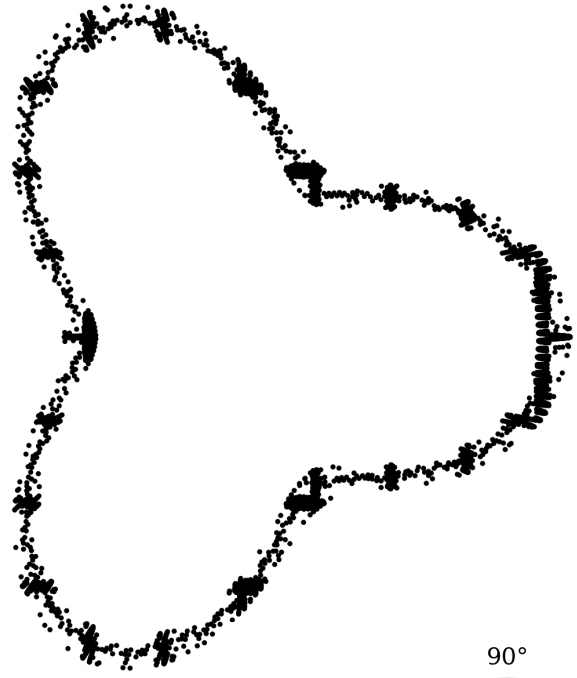


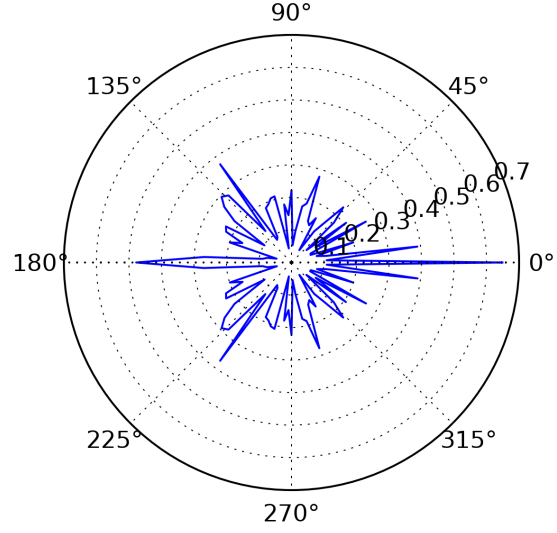
- Interior Dirichlet Helmholtz
- 16 bumps, 32–256 points each
- 8λ total, 2λ subdomains
- Tenth-order Kapur-Rokhlin
- Compress to $\epsilon = 10^{-14}$
- Dense linear algebra

p	N	K	$T^{\text{skel}}_{\text{comp}}$	$T^{\text{skel}}_{\text{dir}}$	$T^{\text{skel}}_{\text{iter}}$	$n^{\text{skel}}_{\text{iter}}$	$T^{\text{full}}_{\text{iter}}$	$n^{\text{full}}_{\text{iter}}$	error
32	512	484	0.24	0.01	0.32	155	1.0	373	8.4E−04
64	1024	859	0.80	0.03	0.80	138	4.1	475	3.9E−07
128	2048	977	1.86	0.10	0.91	107	5.2	221	3.0E−10
256	4096	757	4.24	0.90	1.61	101	13.5	145	7.3E−13

Numerical results







256 bumps, $\epsilon = 10^{-6}$

λ-size	p	N	K	T_{solv}	T_{rcs}
64 / 4	128	32768	4771	267	0.13
128 / 4	256	65536	11953	1419	0.34

Summary

- Skeleton preconditioner based on variant of fast direct solver reduction
- Related to RCIP (Helsing/Ojala) and charge bases (Bremer/Rokhlin/Sammis)
- Cleans up corners, boundary layers, sub-wavelength structures, etc.
- **Allows brute force discretization without much penalty**
- All overhead is local and parallelizable
- Compressed evaluation of far field
- Plug-and-play: fast direct solvers, fast multipole methods
- Very general: boundaries, volumes, whatever
- Formalizes philosophy of compress-then-iterate