Fast Hierarchical QR Factorization of Sparse Matrices

Abeynaya Gnanasekaran, Eric Darve

ICME, Stanford University

SIAM CSE21
March 5, 2021

Goal

We want to develop a fast, sparse QR factorization of A and use it to solve:

- Sparse linear systems

$$Ax = b$$

  where A is of size $N \times N$, sparse and unsymmetric

- Sparse linear least squares systems

$$\min_x \|Ax - b\|^2$$

  where A is $M \times N$, $M \geq N$, full column rank and sparse

Direct vs Iterative methods for sparse matrices

Direct methods	**Iterative methods**
Cholesky, LU, QR	CG, GMRES, CGLS, ...
Robust	Need domain knowledge
Accurate	No convergence guarantees
Generic	Specific
Expensive	Cheap

Direct vs Iterative methods for sparse matrices

Direct methods	**Iterative methods**
Cholesky, LU, QR	CG, GMRES, CGLS, ...
Robust	Need domain knowledge
Accurate	No convergence guarantees
Generic	Specific
Expensive	Cheap

Incomplete factorizations provide a compromise between the two

Incomplete Factorizations

Incomplete Factorization

Incomplete LU, Incomplete QR, Hierarchical methods

Tunable accuracy

Tunable cost

Use as direct method or preconditioner

Incomplete Factorizations

Incomplete Factorization

Incomplete LU, Incomplete QR, Hierarchical methods

Tunable accuracy

Tunable cost

Use as direct method or preconditioner

spaQR is an example of this kind and produces a fast, sparse, approximate QR factorization of a given matrix.

- Fast: $\mathcal{O}(M \log N)$ time to factorize and $\mathcal{O}(M)$ to apply/solve with a vector
- Accuracy: Controllable with parameter ϵ defined by the user

Motivation

Row/Column reordering

- Matrix is typically reordered to minimize fill-in entries

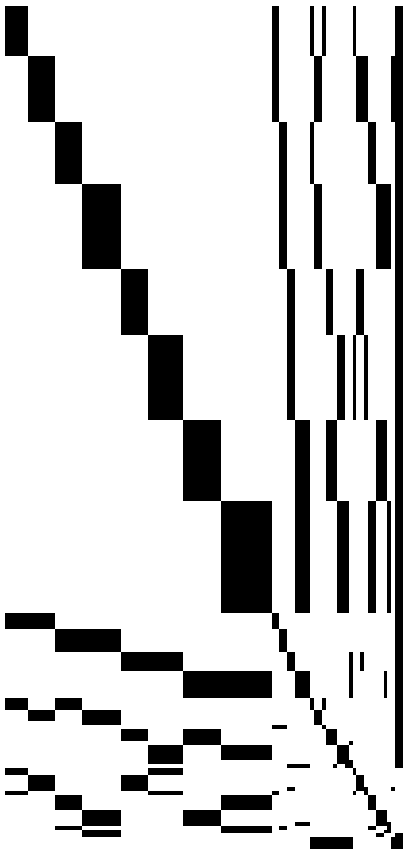
Row/Column reordering

- Matrix is typically reordered to minimize fill-in entries
- Column ordering: Nested Dissection ordering on the graph of $A^T A$
  - If $A = QR$, then R^T is also the Cholesky factor of $A^T A$

Row/Column reordering

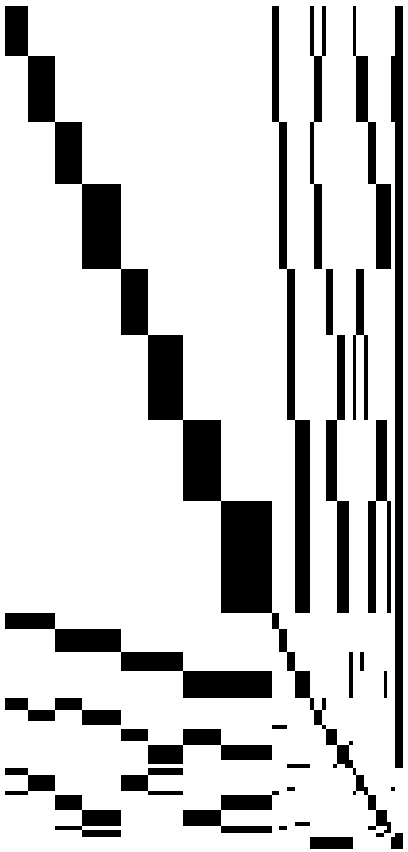
- Matrix is typically reordered to minimize fill-in entries
- Column ordering: Nested Dissection ordering on the graph of $A^T A$
  - If $A = QR$, then R^T is also the Cholesky factor of $A^T A$
- Row ordering: Done such that the diagonal blocks in the reordered matrix has a small condition number (based on heuristics)

Multifrontal sparse QR: Challenges



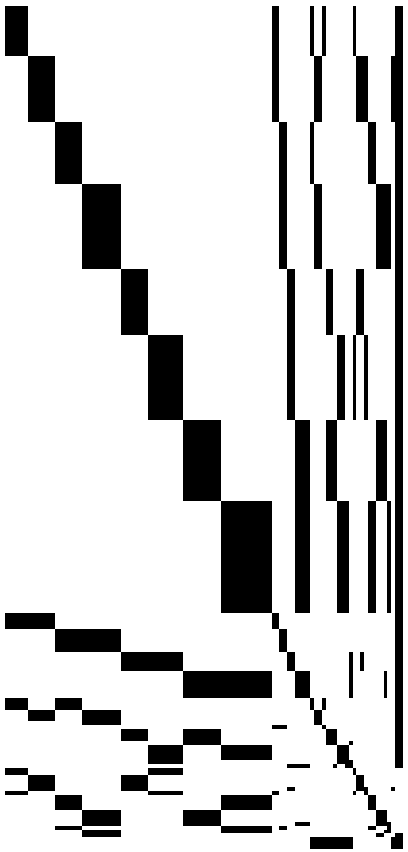
- Understand the cost of doing direct multifrontal QR

Multifrontal sparse QR: Challenges

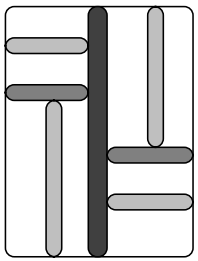


- Understand the cost of doing direct multifrontal QR
- Consider a simple 4 level Nested Dissection partitioning of a matrix

Multifrontal sparse QR: Challenges

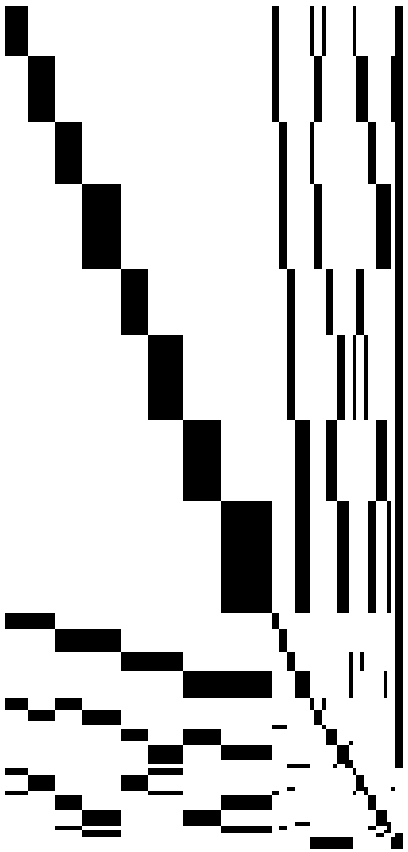


- Understand the cost of doing direct multifrontal QR
- Consider a simple 4 level Nested Dissection partitioning of a matrix
-

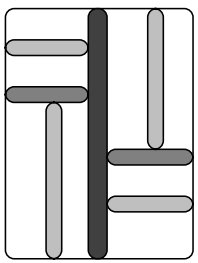


(a) separators

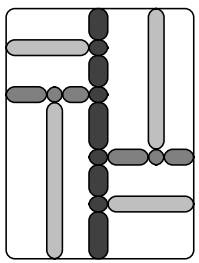
Multifrontal sparse QR: Challenges



- Understand the cost of doing direct multifrontal QR
- Consider a simple 4 level Nested Dissection partitioning of a matrix
-

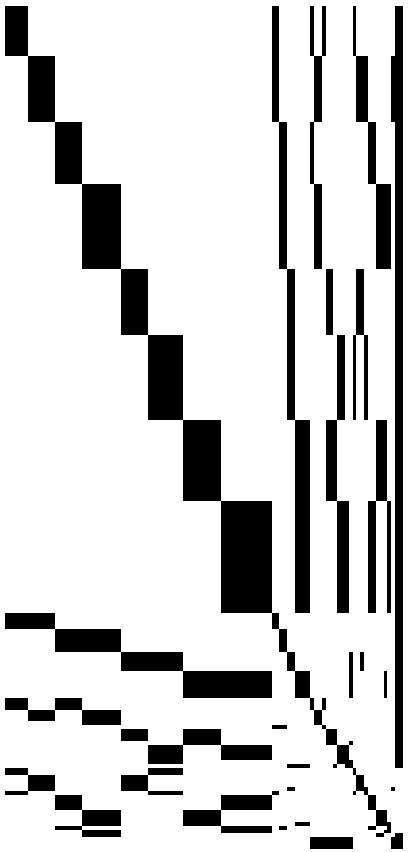


(a) separators

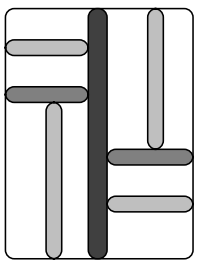


(b) interfaces

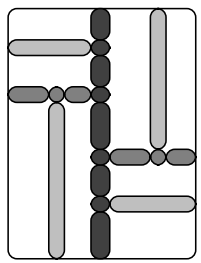
Multifrontal sparse QR: Challenges



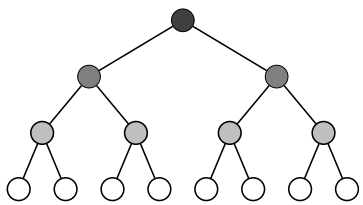
- Understand the cost of doing direct multifrontal QR
- Consider a simple 4 level Nested Dissection partitioning of a matrix
-



(a) separators

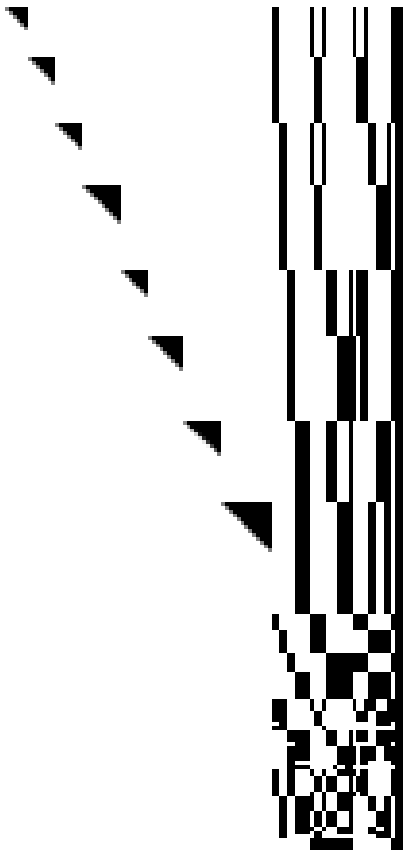


(b) interfaces

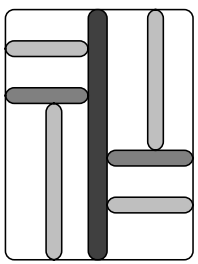


(c) elimination tree

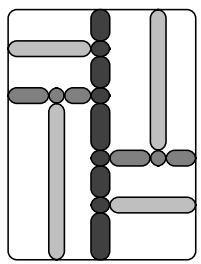
Multifrontal sparse QR: Challenges



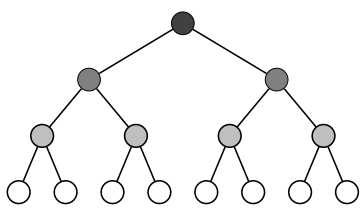
- Understand the cost of doing direct multifrontal QR
- Consider a simple 4 level Nested Dissection partitioning of a matrix
-



(a) separators

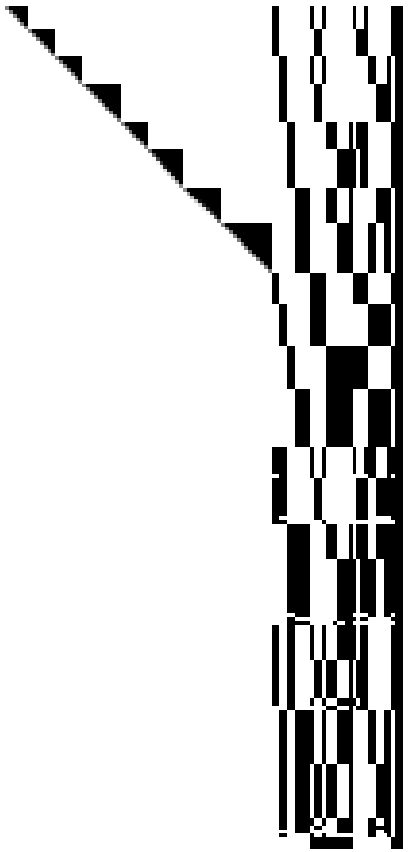


(b) interfaces

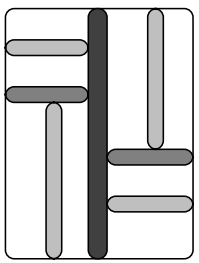


(c) elimination tree

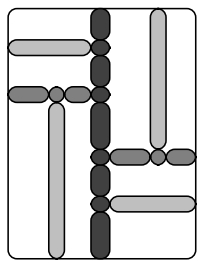
Multifrontal sparse QR: Challenges



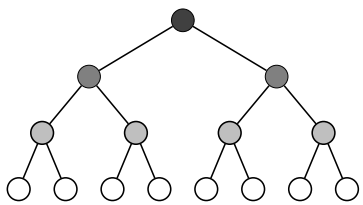
- Understand the cost of doing direct multifrontal QR
- Consider a simple 4 level Nested Dissection partitioning of a matrix
-



(a) separators

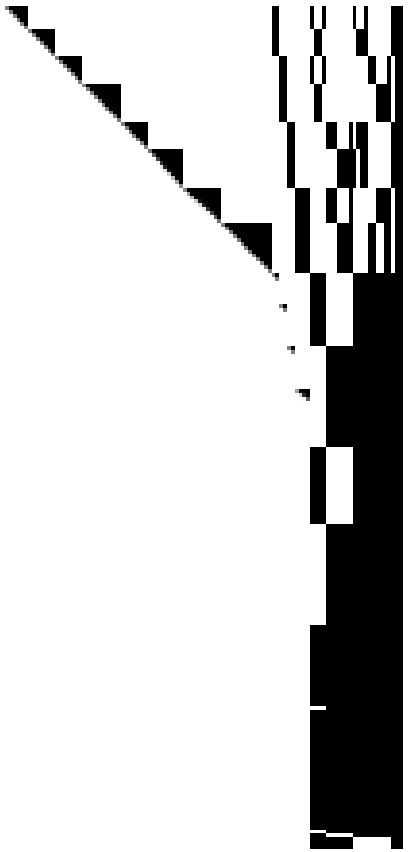


(b) interfaces

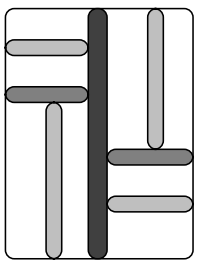


(c) elimination tree

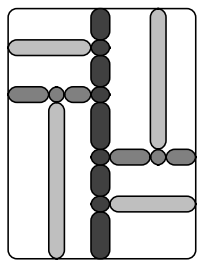
Multifrontal sparse QR: Challenges



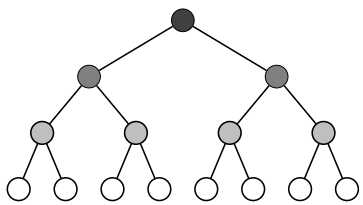
- Understand the cost of doing direct multifrontal QR
- Consider a simple 4 level Nested Dissection partitioning of a matrix
-



(a) separators

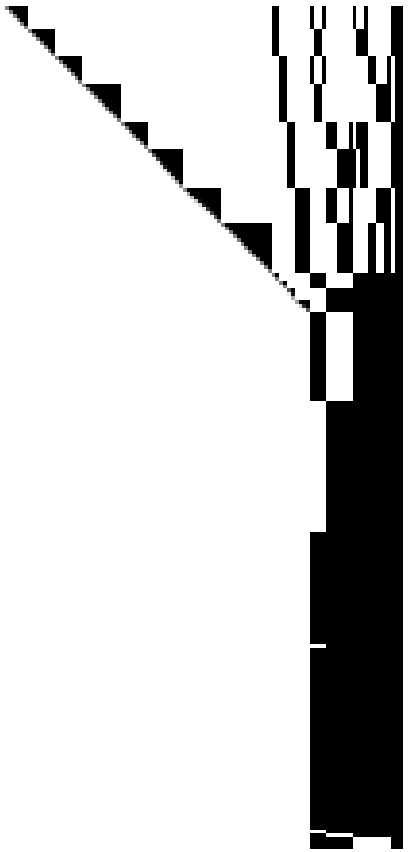


(b) interfaces

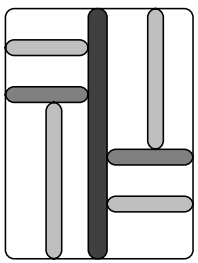


(c) elimination tree

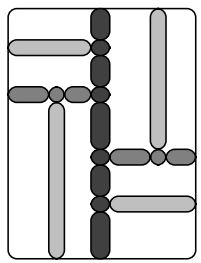
Multifrontal sparse QR: Challenges



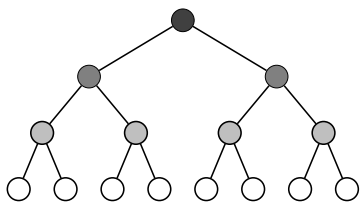
- Understand the cost of doing direct multifrontal QR
- Consider a simple 4 level Nested Dissection partitioning of a matrix
-



(a) separators

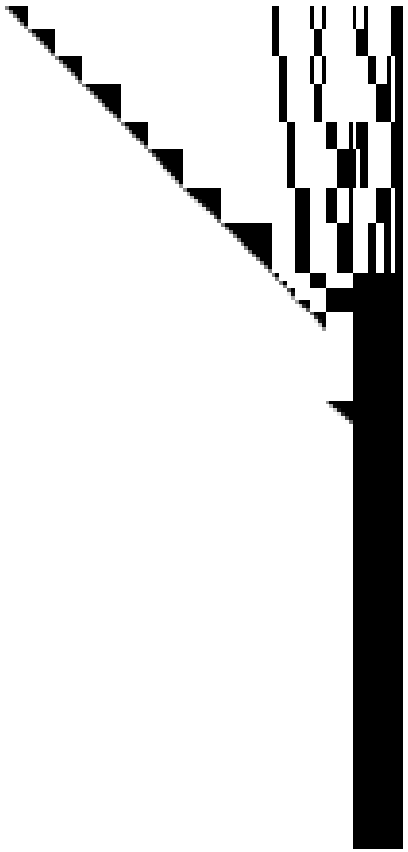


(b) interfaces

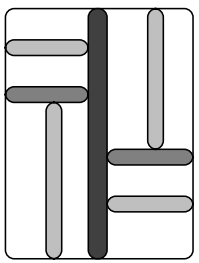


(c) elimination tree

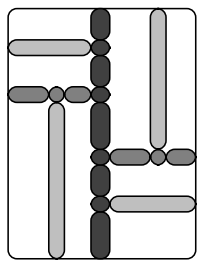
Multifrontal sparse QR: Challenges



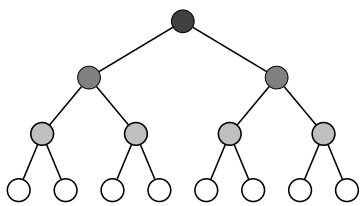
- Understand the cost of doing direct multifrontal QR
- Consider a simple 4 level Nested Dissection partitioning of a matrix
-



(a) separators

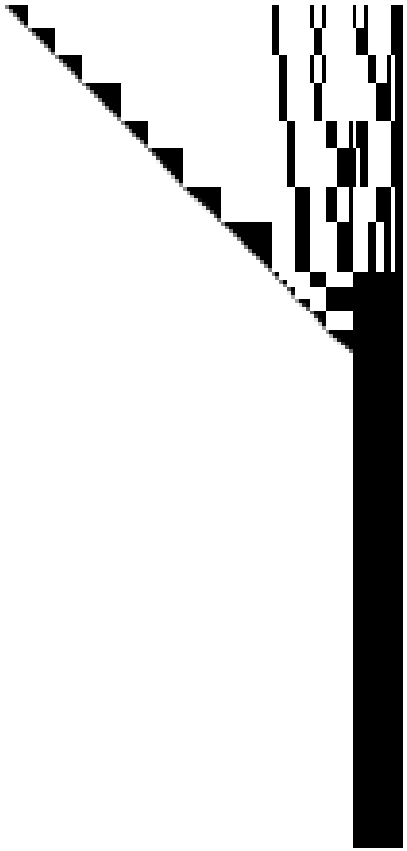


(b) interfaces

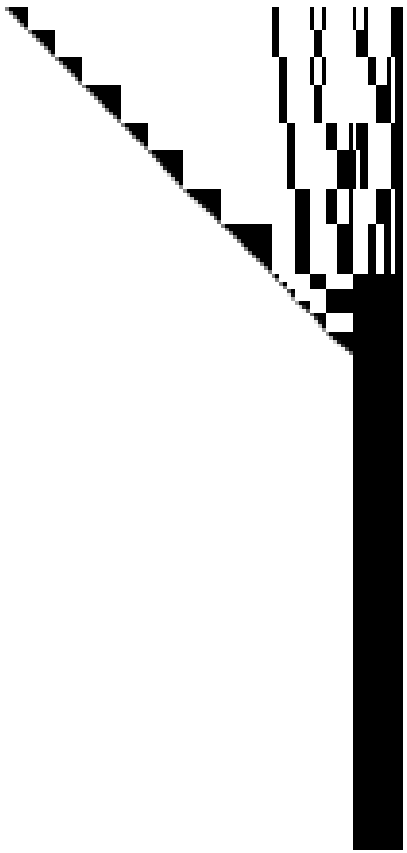


(c) elimination tree

Multifrontal sparse QR: Challenges

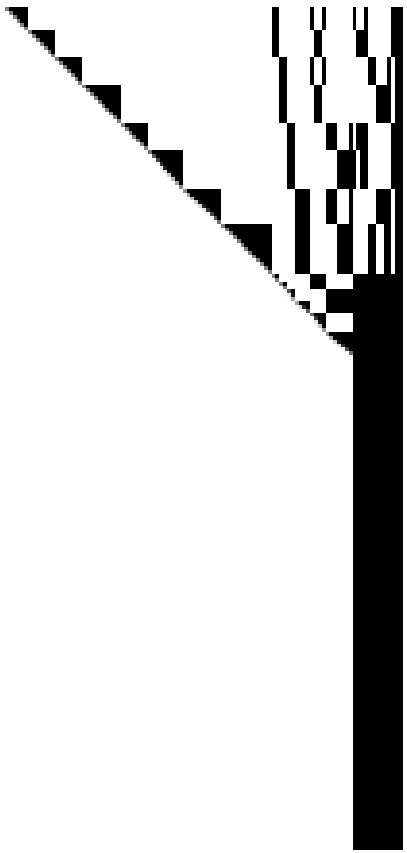


Multifrontal sparse QR: Challenges



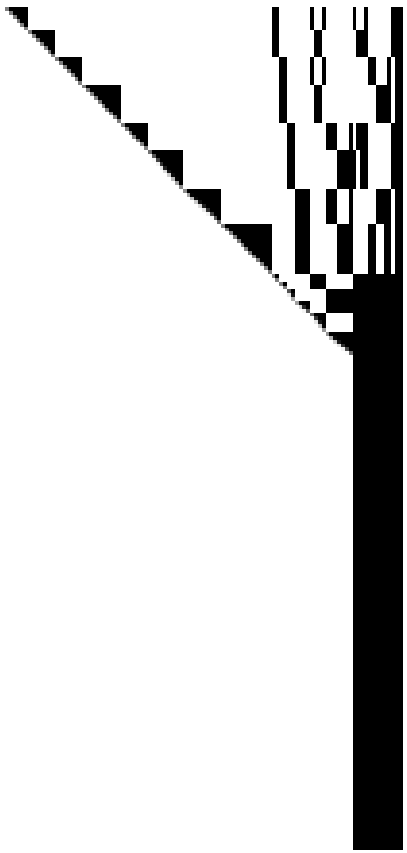
- The top separator block contains $\mathcal{O}(M - N)$ rows

Multifrontal sparse QR: Challenges



- The top separator block contains $\mathcal{O}(M - N)$ rows
- For 3D problems, the top separator is now a dense block with $\mathcal{O}(M - N)$ rows and $\mathcal{O}(N^{2/3})$ columns

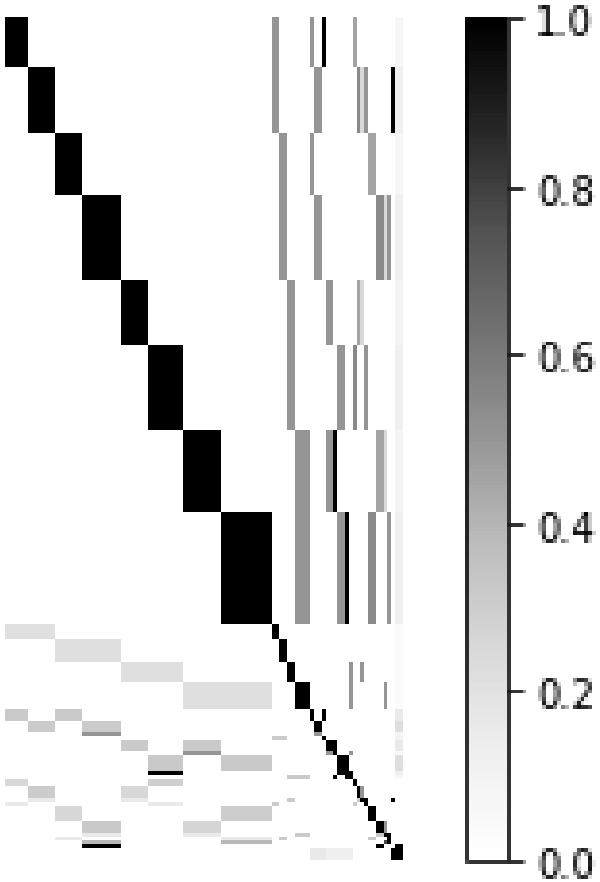
Multifrontal sparse QR: Challenges



- The top separator block contains $\mathcal{O}(M-N)$ rows
- For 3D problems, the top separator is now a dense block with $\mathcal{O}(M-N)$ rows and $\mathcal{O}(N^{2/3})$ columns
- Costs $\mathcal{O}\left((M-N)N^{4/3}\right)$ to factorize just this block! Expensive!

Ranks of off-diagonal blocks

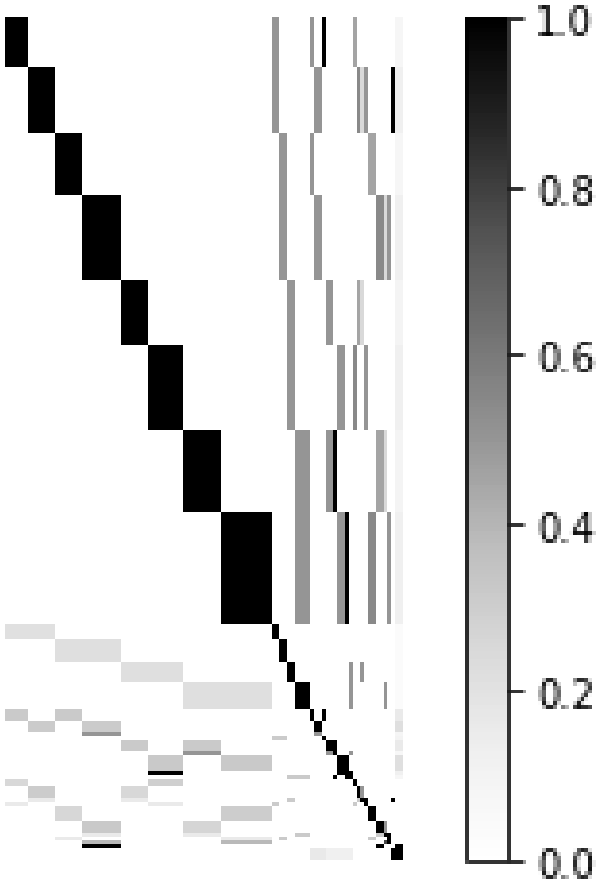
How can we improve?



- Color map indicates the fraction of singular values $\geq 10^{-2}$

Ranks of off-diagonal blocks

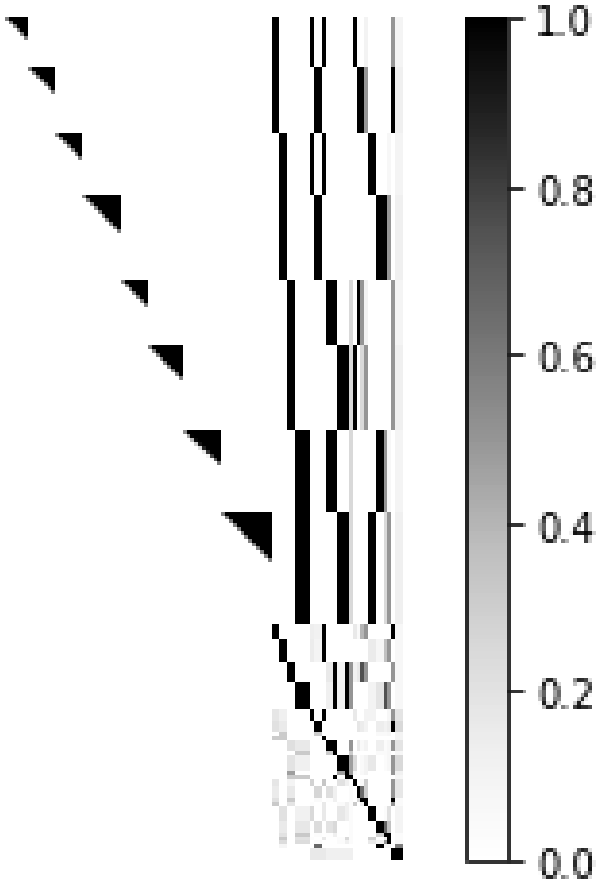
How can we improve?



- Color map indicates the fraction of singular values $\geq 10^{-2}$
- Lighter color $\implies$ low rank

Ranks of off-diagonal blocks

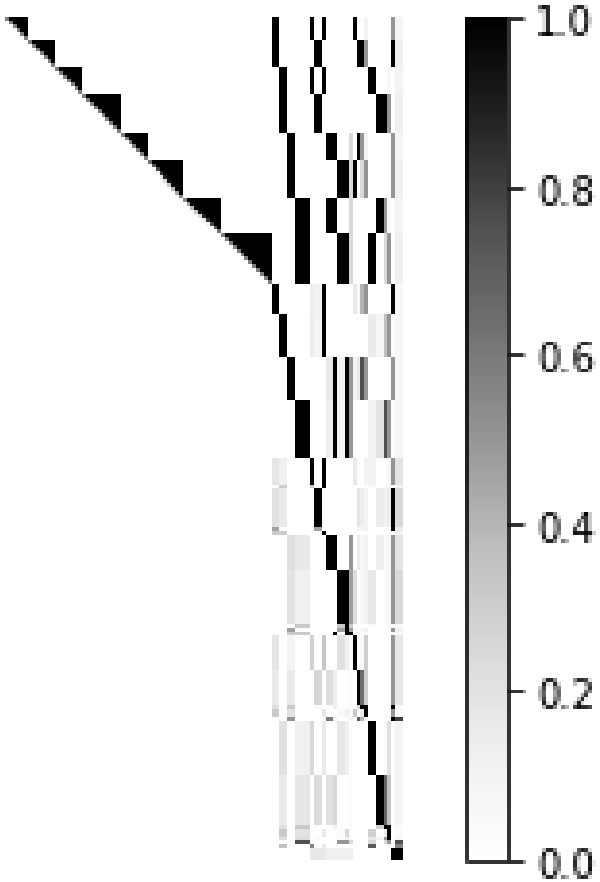
How can we improve?



- Color map indicates the fraction of singular values $\geq 10^{-2}$
- Lighter color $\implies$ low rank

Ranks of off-diagonal blocks

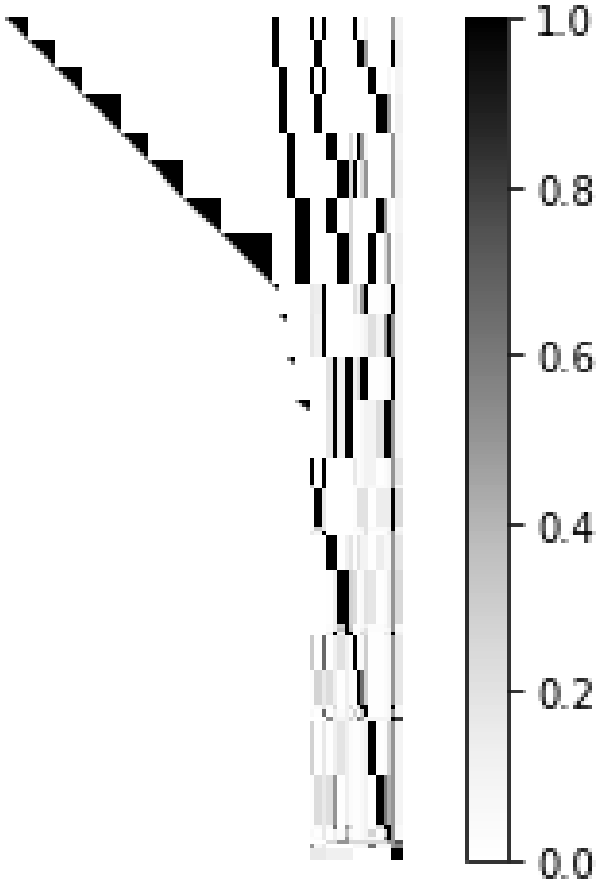
How can we improve?



- Color map indicates the fraction of singular values $\geq 10^{-2}$
- Lighter color $\implies$ low rank

Ranks of off-diagonal blocks

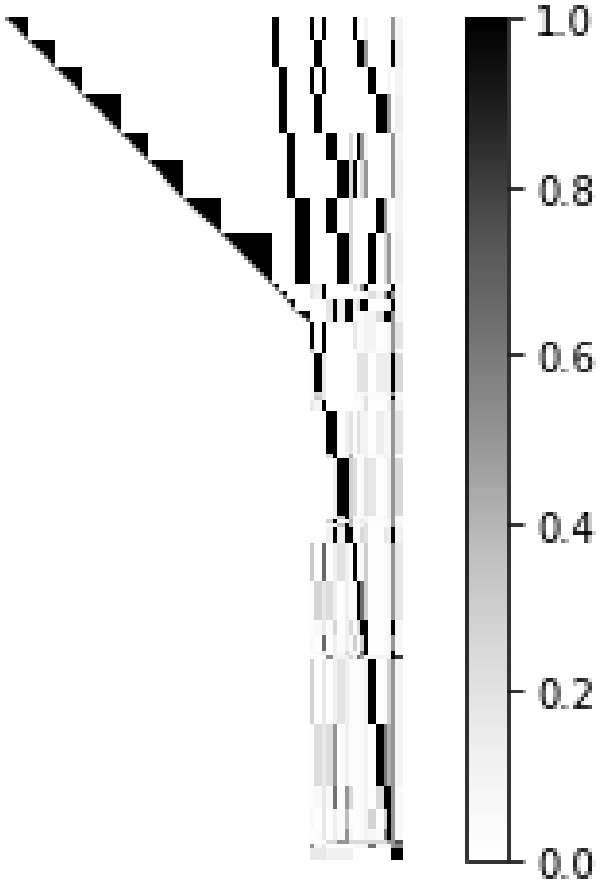
How can we improve?



- Color map indicates the fraction of singular values $\geq 10^{-2}$
- Lighter color $\implies$ low rank
- spaQR exploits this key observation to develop a fast factorization

Ranks of off-diagonal blocks

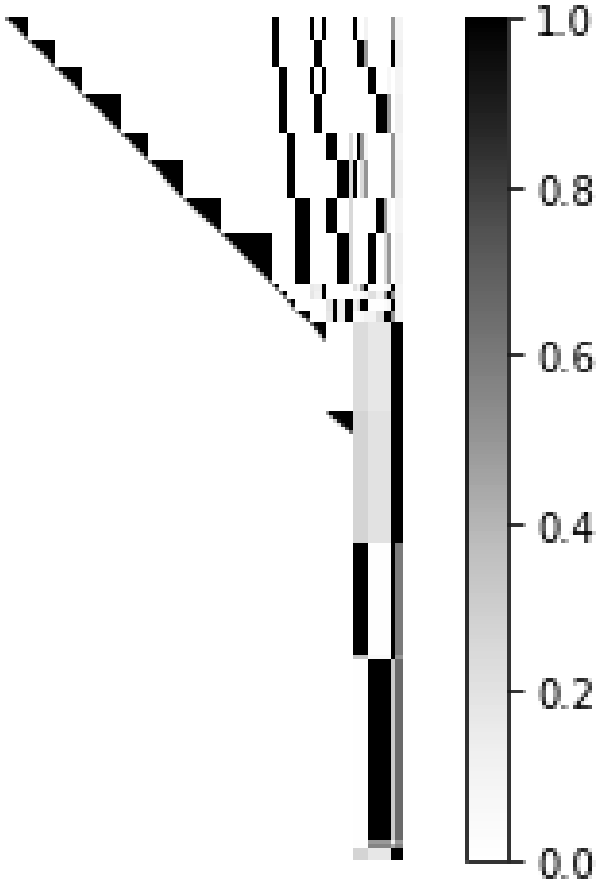
How can we improve?



- Color map indicates the fraction of singular values $\geq 10^{-2}$
- Lighter color $\implies$ low rank
- spaQR exploits this key observation to develop a fast factorization

Ranks of off-diagonal blocks

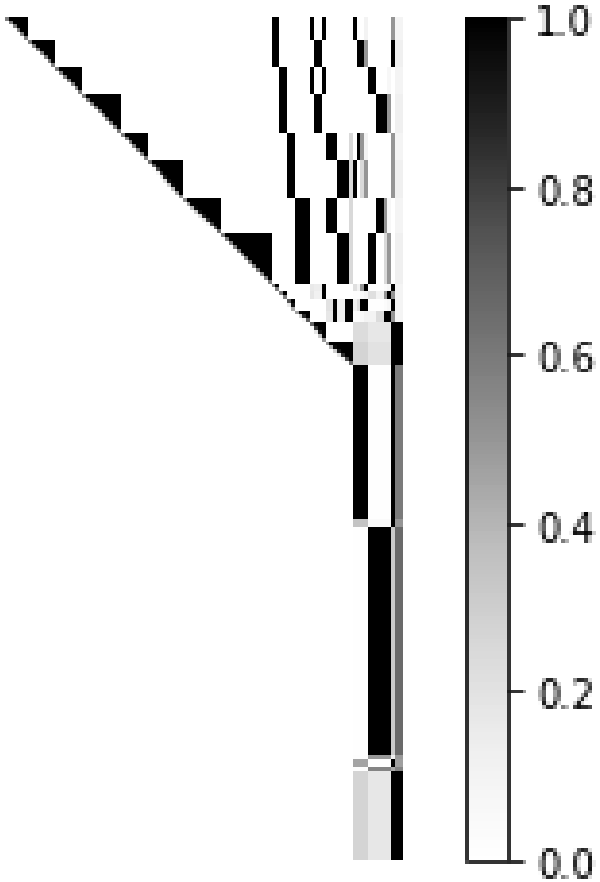
How can we improve?



- Color map indicates the fraction of singular values $\geq 10^{-2}$
- Lighter color $\implies$ low rank
- spaQR exploits this key observation to develop a fast factorization

Ranks of off-diagonal blocks

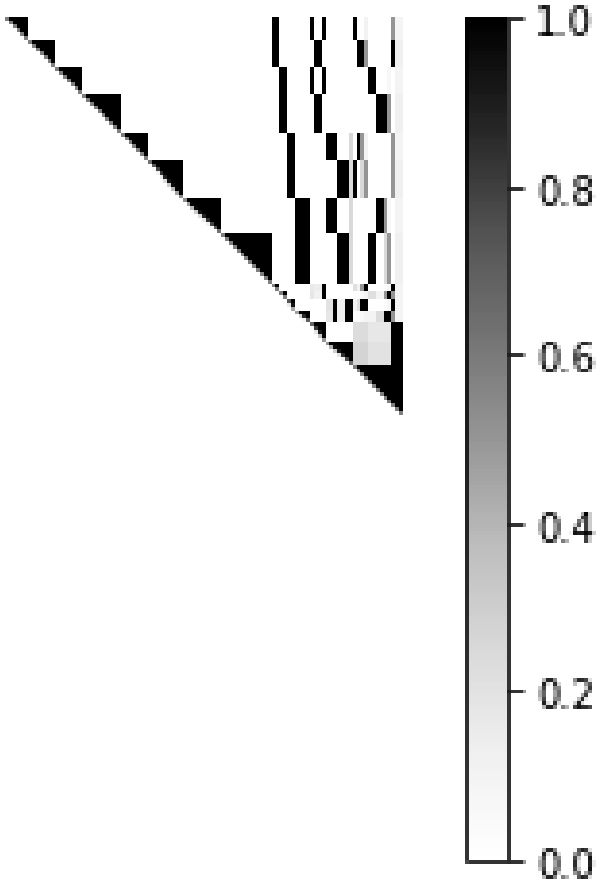
How can we improve?



- Color map indicates the fraction of singular values $\geq 10^{-2}$
- Lighter color $\implies$ low rank
- spaQR exploits this key observation to develop a fast factorization

Ranks of off-diagonal blocks

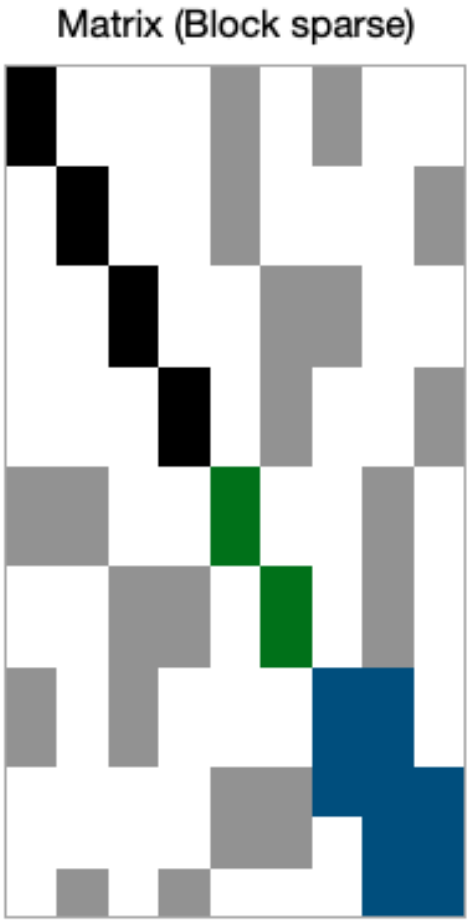
How can we improve?



- Color map indicates the fraction of singular values $\geq 10^{-2}$
- Lighter color $\implies$ low rank
- spaQR exploits this key observation to develop a fast factorization

Overview: spaQR algorithm

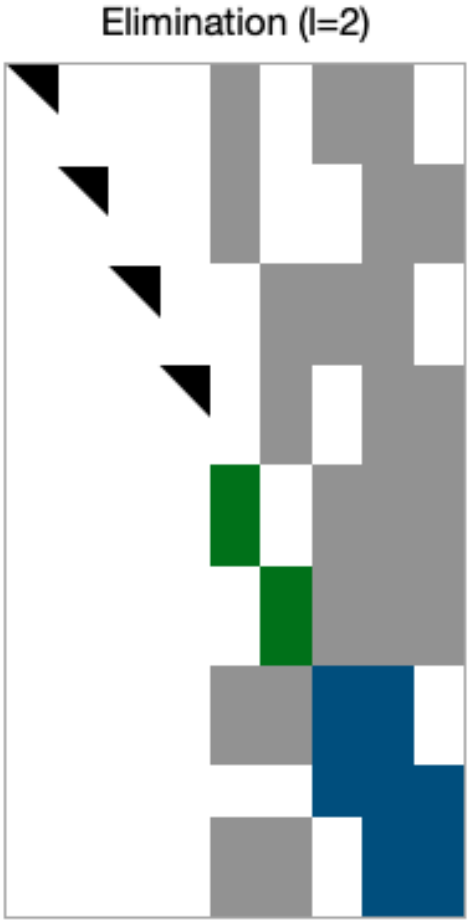
spaQR algorithm



Require: Matrix $A \in \mathbb{R}^{M \times N}$, $M \geq N$

Require: Tolerance ϵ

spaQR algorithm



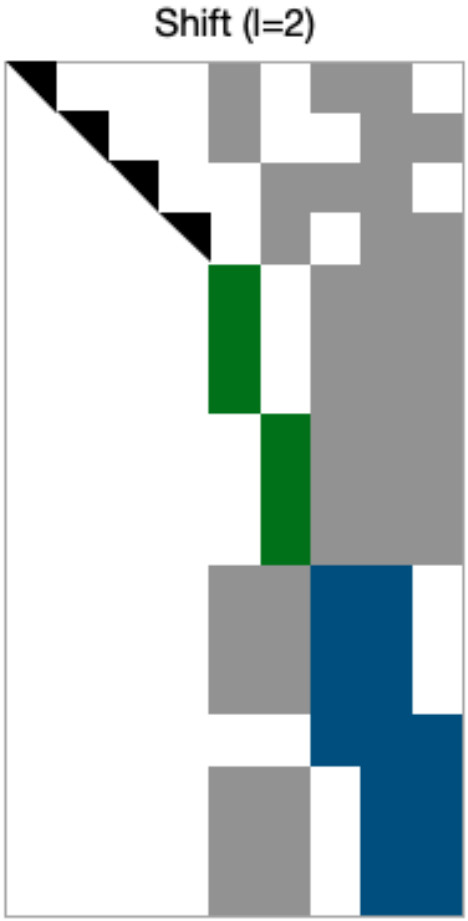
Require: Matrix $A \in \mathbb{R}^{M \times N}$, $M \geq N$

Require: Tolerance ϵ

for all $l = L-1, L-2, \ldots 0$ **do**

1. Factorize interiors at level l

spaQR algorithm



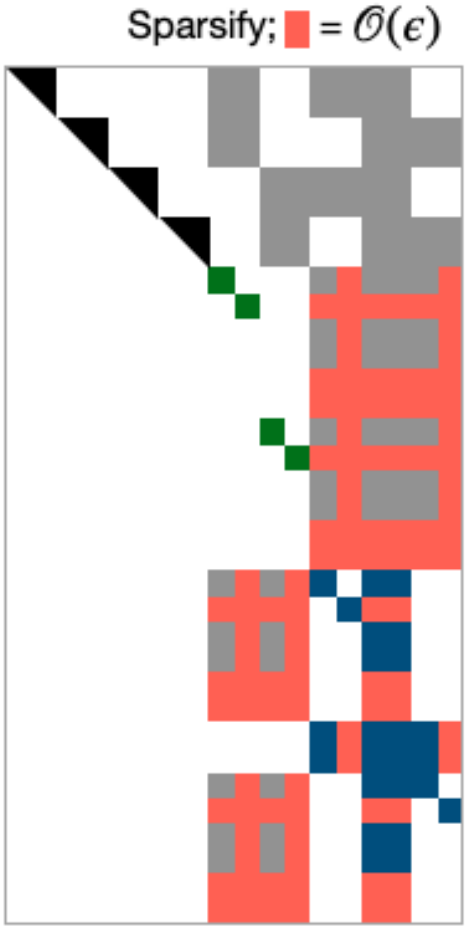
Require: Matrix $A \in \mathbb{R}^{M \times N}$, $M \geq N$

Require: Tolerance ϵ

for all $l = L-1, L-2, \ldots 0$ **do**

1. Factorize interiors at level l
2. Shift rows below diagonal of eliminated interiors

spaQR algorithm



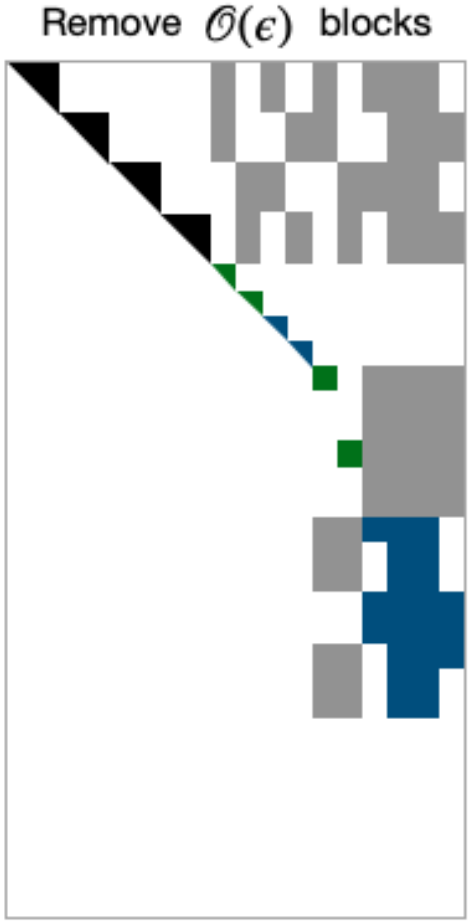
Require: Matrix $A \in \mathbb{R}^{M \times N}$, $M \geq N$

Require: Tolerance ϵ

for all $l = L-1, L-2, \ldots 0$ **do**

1. Factorize interiors at level l
2. Shift rows below diagonal of eliminated interiors
3. Scale, Sparsify interfaces at all levels $l' > l$

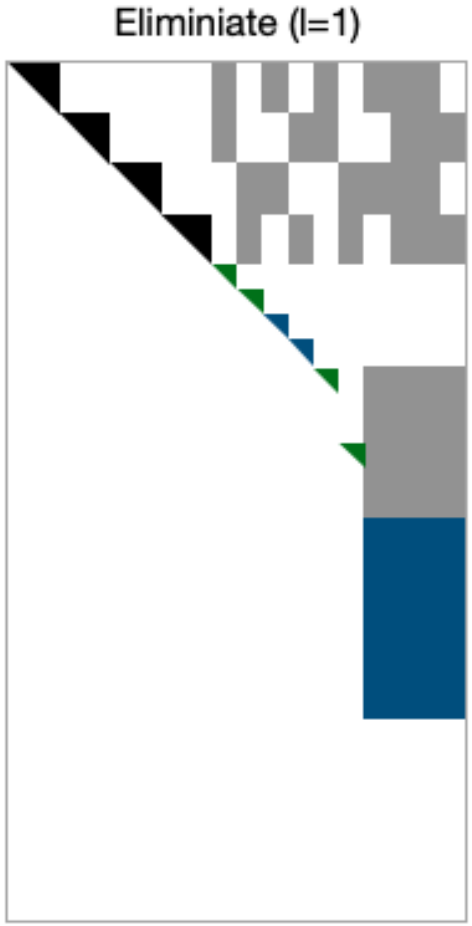
spaQR algorithm



Require: Matrix $A \in \mathbb{R}^{M \times N}$, $M \geq N$
Require: Tolerance ϵ
for all $l = L-1, L-2, \ldots 0$ **do**
1. Factorize interiors at level l
2. Shift rows below diagonal of eliminated interiors
3. Scale, Sparsify interfaces at all levels $l' > l$
4. Ignore $\mathcal{O}(\epsilon)$ blocks

end for

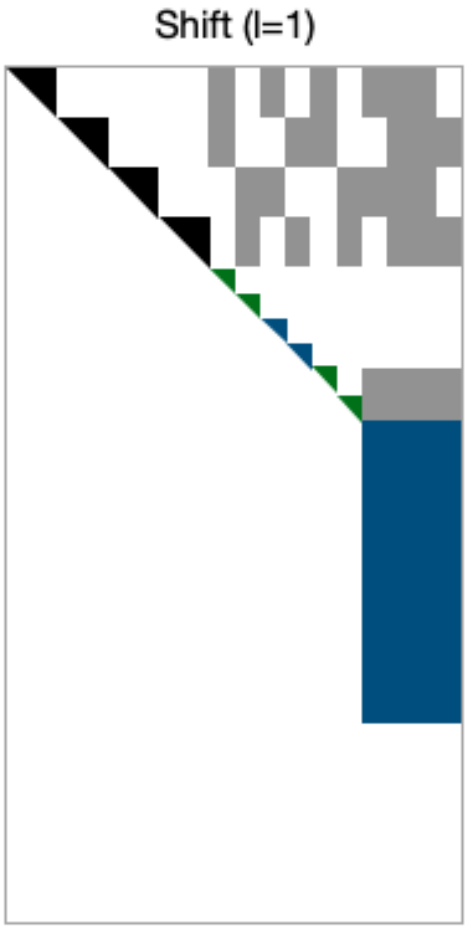
spaQR algorithm



Require: Matrix $A \in \mathbb{R}^{M \times N}$, $M \geq N$
Require: Tolerance ϵ
for all $l = L-1, L-2, \ldots 0$ **do**
1. Factorize interiors at level l
2. Shift rows below diagonal of eliminated interiors
3. Scale, Sparsify interfaces at all levels $l' > l$
4. Ignore $\mathcal{O}(\epsilon)$ blocks

end for

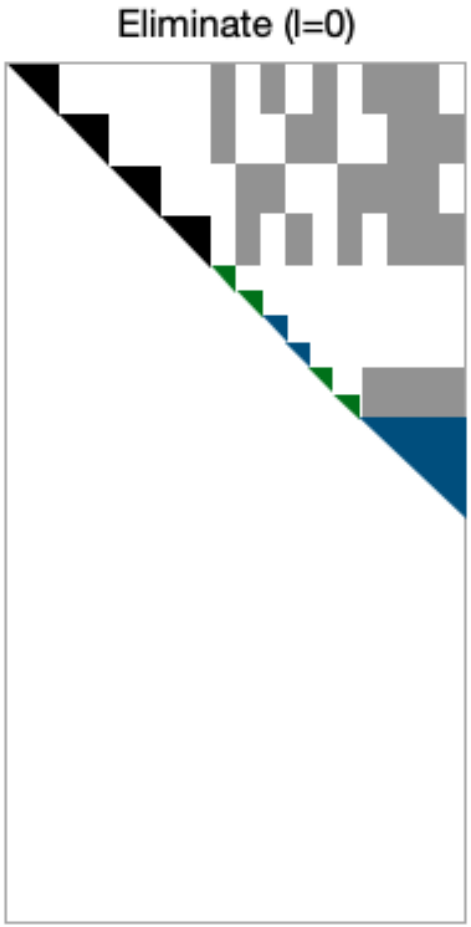
spaQR algorithm



Require: Matrix $A \in \mathbb{R}^{M \times N}$, $M \geq N$
Require: Tolerance ϵ
for all $l = L-1, L-2, \ldots 0$ **do**
1. Factorize interiors at level l
2. Shift rows below diagonal of eliminated interiors
3. Scale, Sparsify interfaces at all levels $l' > l$
4. Ignore $\mathcal{O}(\epsilon)$ blocks

end for

spaQR algorithm



Require: Matrix $A \in \mathbb{R}^{M \times N}$, $M \geq N$
Require: Tolerance ϵ
for all $l = L-1, L-2, \ldots 0$ **do**
1. Factorize interiors at level l
2. Shift rows below diagonal of eliminated interiors
3. Scale, Sparsify interfaces at all levels $l' > l$
4. Ignore $\mathcal{O}(\epsilon)$ blocks

end for

Sparsification

Sparsification in spaQR

- Key assumption: Off-diagonal blocks are low rank

Sparsification in spaQR

- Key assumption: Off-diagonal blocks are low rank
- Consider an interface p, the nodes n connected to p in the graph of $A^T A$

Sparsification in spaQR

- Key assumption: Off-diagonal blocks are low rank
- Consider an interface p, the nodes n connected to p in the graph of $A^T A$
- Consider a small piece of the matrix, focusing on the non-zero blocks for p and n,

$$A_p = \begin{bmatrix} A_{pp} & A_{pn} \\ A_{np} & A_{nn} \end{bmatrix} =$$

Step 1: Scaling

We first scale the diagonal blocks of each interface. For interface p, using the QR decomposition of $A_{pp} = U_{pp} \begin{bmatrix} R_{pp} \\ 0 \end{bmatrix}$

$$\bar{A}_p = \begin{bmatrix} U_{pp} & \\ & I \end{bmatrix}^T A_p \begin{bmatrix} R_{pp} & \\ & I \end{bmatrix}^{-1} = \begin{bmatrix} I & \bar{A}_{p_1 n} \\ & \bar{A}_{p_2 n} \\ \bar{A}_{np} & A_{nn} \end{bmatrix} =$$

Step 2: Compress rows $\bar{A}_{p_2 n}$

Consider a low rank approximation of $A_{p_2 n}$,

$$A_{p_2 n} = Q'_{pp} W'_{pn} = \begin{bmatrix} Q'_{pc} & Q'_{pf} \end{bmatrix} \begin{bmatrix} W'_{cn} \\ W'_{fn} \end{bmatrix} \quad \text{with} \quad \|W'_{fn}\|_2 = \mathcal{O}(\epsilon)$$

Step 2: Compress rows $\bar{A}_{p_2 n}$

Consider a low rank approximation of $A_{p_2 n}$,

$$A_{p_2 n} = Q'_{pp} W'_{pn} = \begin{bmatrix} Q'_{pc} & Q'_{pf} \end{bmatrix} \begin{bmatrix} W'_{cn} \\ W'_{fn} \end{bmatrix} \quad \text{with} \quad \|W'_{fn}\|_2 = \mathcal{O}(\epsilon)$$

Define, $Q'_p = \begin{bmatrix} I & & \\ & Q'_{pp} & \\ & & I \end{bmatrix}^T$

Step 2: Compress rows $\bar{A}_{p_2 n}$

Consider a low rank approximation of $A_{p_2 n}$,

$$A_{p_2 n} = Q'_{pp} W'_{pn} = \begin{bmatrix} Q'_{pc} & Q'_{pf} \end{bmatrix} \begin{bmatrix} W'_{cn} \\ W'_{fn} \end{bmatrix} \quad \text{with} \quad \| W'_{fn} \|_2 = \mathcal{O}(\epsilon)$$

Define, $Q'_p = \begin{bmatrix} I & & \\ & Q'_{pp} & \\ & & I \end{bmatrix}^T$

$$\bar{\bar{A}}_p = (Q'_p)^T \begin{bmatrix} I & \bar{A}_{p_1 n} \\ & \bar{A}_{p_2 n} \\ \bar{A}_{np} & A_{nn} \end{bmatrix} = \begin{bmatrix} I & A_{p_1 n} \\ & W'_{cn} \\ & W'_{fn} \\ A_{np} & A_{nn} \end{bmatrix} =$$

Step 2: Compress rows $\bar{A}_{p_2 n}$

Consider a low rank approximation of $A_{p_2 n}$,

$$A_{p_2 n} = Q'_{pp} W'_{pn} = \begin{bmatrix} Q'_{pc} & Q'_{pf} \end{bmatrix} \begin{bmatrix} W'_{cn} \\ W'_{fn} \end{bmatrix} \quad \text{with} \quad \|W'_{fn}\|_2 = \mathcal{O}(\epsilon)$$

Define, $Q'_p = \begin{bmatrix} I & & \\ & Q'_{pp} & \\ & & I \end{bmatrix}^T$

$$\bar{\bar{A}}_p = (Q'_p)^T \begin{bmatrix} I & \bar{A}_{p_1 n} \\ & \bar{A}_{p_2 n} \\ \bar{A}_{np} & A_{nn} \end{bmatrix} = \begin{bmatrix} I & A_{p_1 n} \\ & W'_{cn} \\ & W'_{fn} \\ A_{np} & A_{nn} \end{bmatrix} =$$

$\|W'_{fn}\| = \mathcal{O}(\epsilon)$ and the corresponding rows are removed

Step 2: Compress block $\begin{bmatrix} A_{np}^T & A_{p_1 n} \end{bmatrix}$

$$\begin{bmatrix} A_{np}^T & A_{p_1 n} \end{bmatrix} = Q_{pp} W_{pn} = \begin{bmatrix} Q_{pf} & Q_{pc} \end{bmatrix} \begin{bmatrix} W_{fn}^{(1)} & W_{fn}^{(2)} \\ W_{cn}^{(1)} & W_{cn}^{(2)} \end{bmatrix}$$

where $\| W_{fn}^{(1)} \|_2 \approx \| W_{fn}^{(2)} \|_2 = \mathcal{O}(\epsilon)$. Define, $Q_p = \begin{bmatrix} Q_{pp} & & \\ & I & \\ & & I \end{bmatrix}$

Step 2: Compress block $[A_{np}^T \quad A_{p_1 n}]$

$$[A_{np}^T \quad A_{p_1 n}] = Q_{pp} W_{pn} = [Q_{pf} \quad Q_{pc}] \begin{bmatrix} W_{fn}^{(1)} & W_{fn}^{(2)} \\ W_{cn}^{(1)} & W_{cn}^{(2)} \end{bmatrix}$$

where $\|W_{fn}^{(1)}\|_2 \approx \|W_{fn}^{(2)}\|_2 = \mathcal{O}(\epsilon)$. Define, $Q_p = \begin{bmatrix} Q_{pp} & & \\ & I & \\ & & I \end{bmatrix}$

$$Q_p^T \bar{\bar{A}}_p Q_p = \begin{bmatrix} I_f & & & W_{fn}^{(2)} \\ & I_c & & W_{cn}^{(2)} \\ & & & W'_{cn} \\ & & & W'_{fn} \\ W_{fn}^{(1)T} & W_{cn}^{(1)T} & & A_{nn} \end{bmatrix} =$$

Step 2: Compress block $[A_{np}^T \quad A_{p_1 n}]$

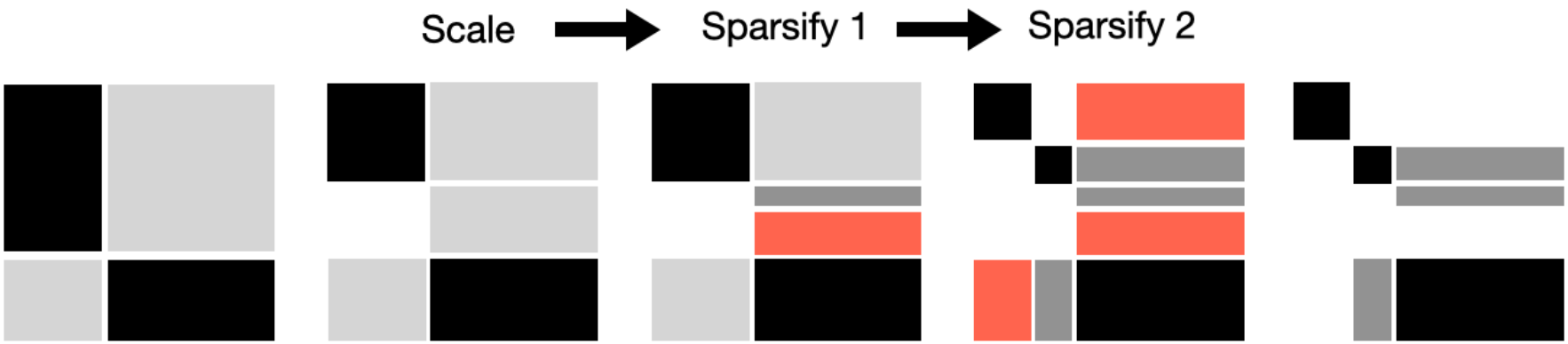
$$[A_{np}^T \quad A_{p_1 n}] = Q_{pp} W_{pn} = [Q_{pf} \quad Q_{pc}] \begin{bmatrix} W_{fn}^{(1)} & W_{fn}^{(2)} \\ W_{cn}^{(1)} & W_{cn}^{(2)} \end{bmatrix}$$

where $\|W_{fn}^{(1)}\|_2 \approx \|W_{fn}^{(2)}\|_2 = \mathcal{O}(\epsilon)$. Define, $Q_p = \begin{bmatrix} Q_{pp} & & \\ & I & \\ & & I \end{bmatrix}$

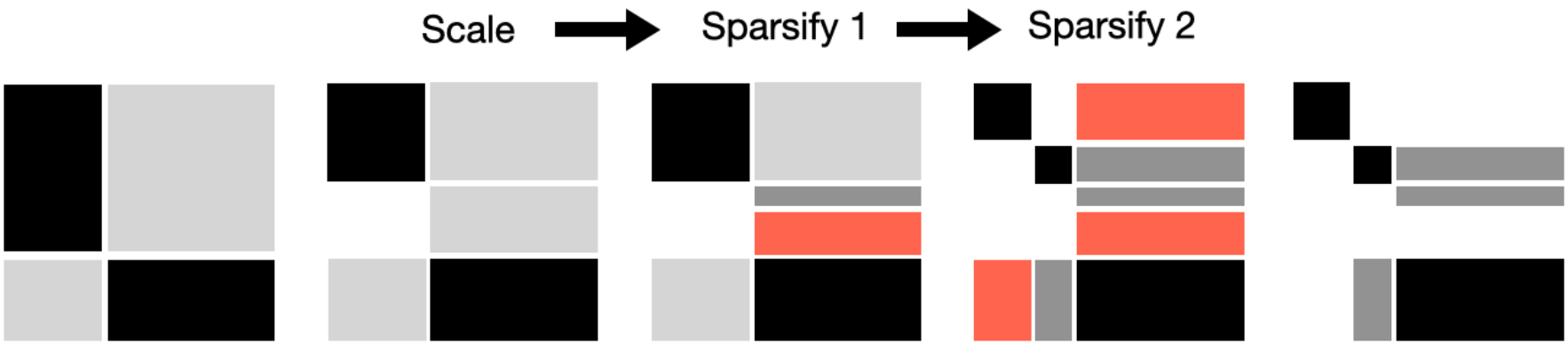
$$Q_p^T \bar{\bar{A}}_p Q_p = \begin{bmatrix} I_f & & W_{fn}^{(2)} \\ & I_c & W_{cn}^{(2)} \\ & & W'_{cn} \\ & & W'_{fn} \\ W_{fn}^{(1)T} & W_{cn}^{(1)T} & A_{nn} \end{bmatrix} =$$

Ignore the $\mathcal{O}(\epsilon)$ blocks

Summary



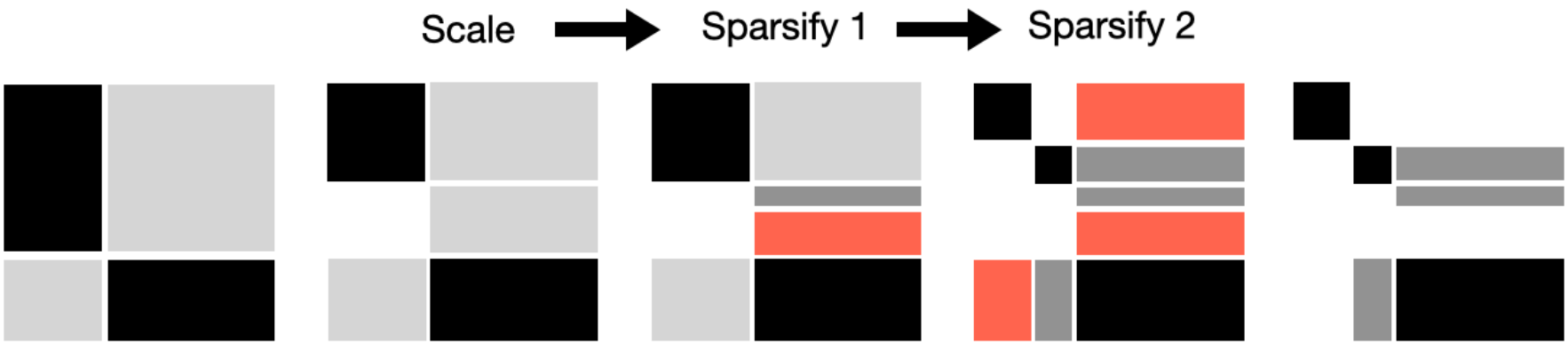
Summary



Sparsification,

- Reduces the size of the interfaces

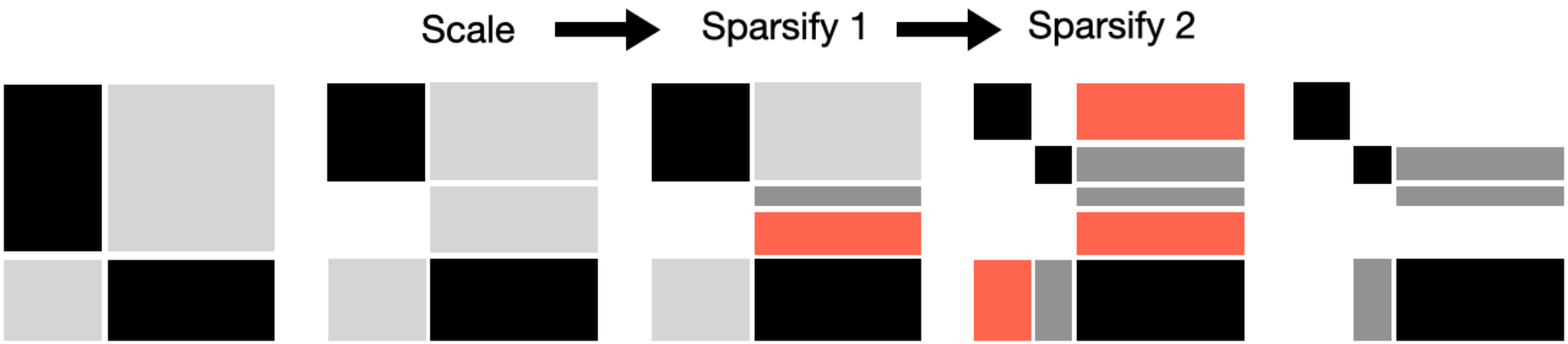
Summary



Sparsification,

- Reduces the size of the interfaces
- Does not introduce additional fill-in

Summary



Sparsification,

- Reduces the size of the interfaces
- Does not introduce additional fill-in
- **(Important)** Does not affect the elimination tree of $A^T A$, i.e., any two disjoint subtrees of the etree remain disjoint after sparsification of any interface.

Complexity

Complexity: Square matrices

3D PDE discretizations

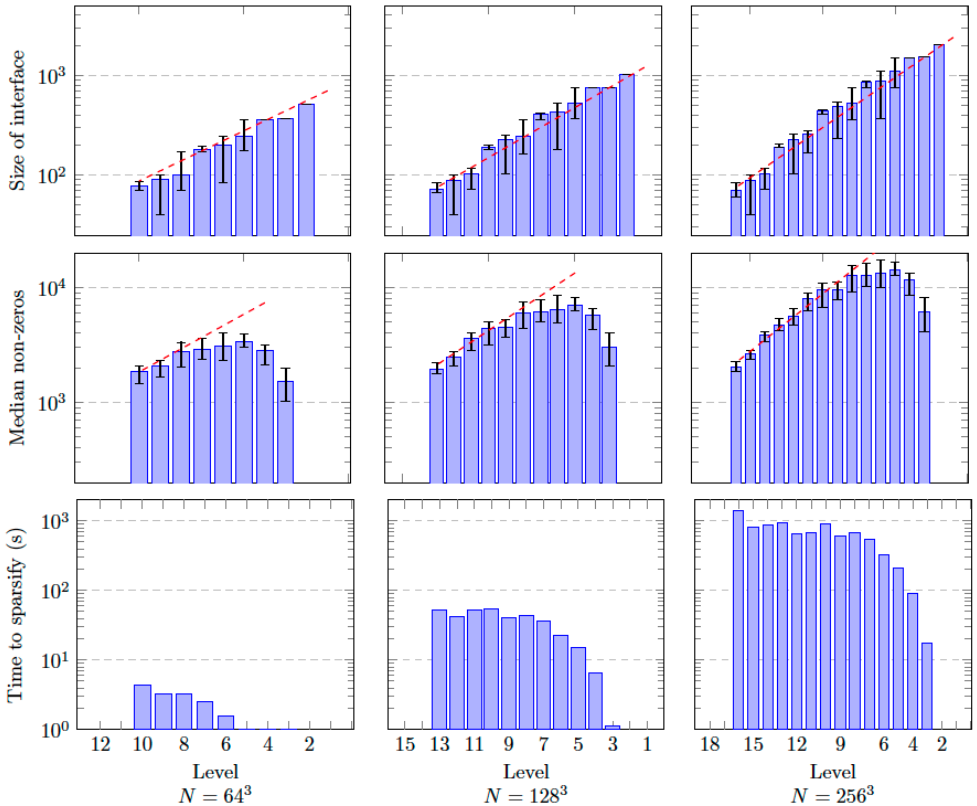


Figure: 3D Advection-Diffusion problem. Red dashed line indicates $2^{-l/3}$ scaling.

Complexity: Square matrices

3D PDE discretizations

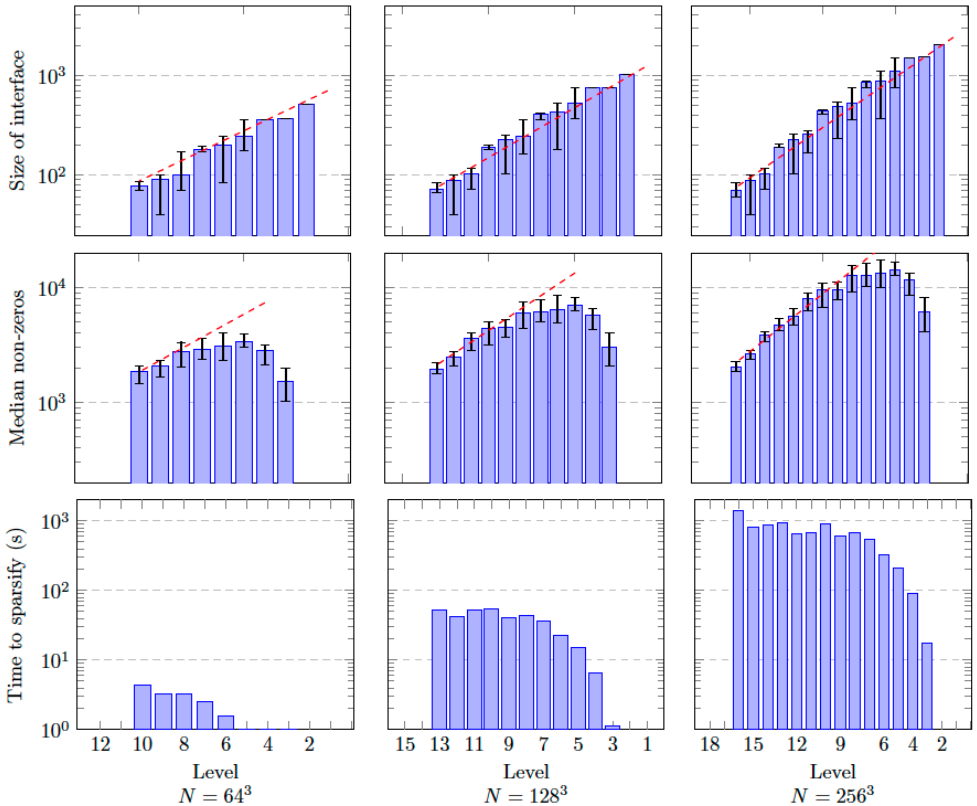


Figure: 3D Advection-Diffusion problem. Red dashed line indicates $2^{-l/3}$ scaling.

Assume that,

- Size of an interface,

$$c_l \in \mathcal{O}(2^{-l/3} N^{1/3})$$

- Number of non-zeros per row/column of an interface,

$$n_l \in \mathcal{O}(2^{-l/3} N^{1/3})$$

Complexity: Square matrices

3D PDE discretizations

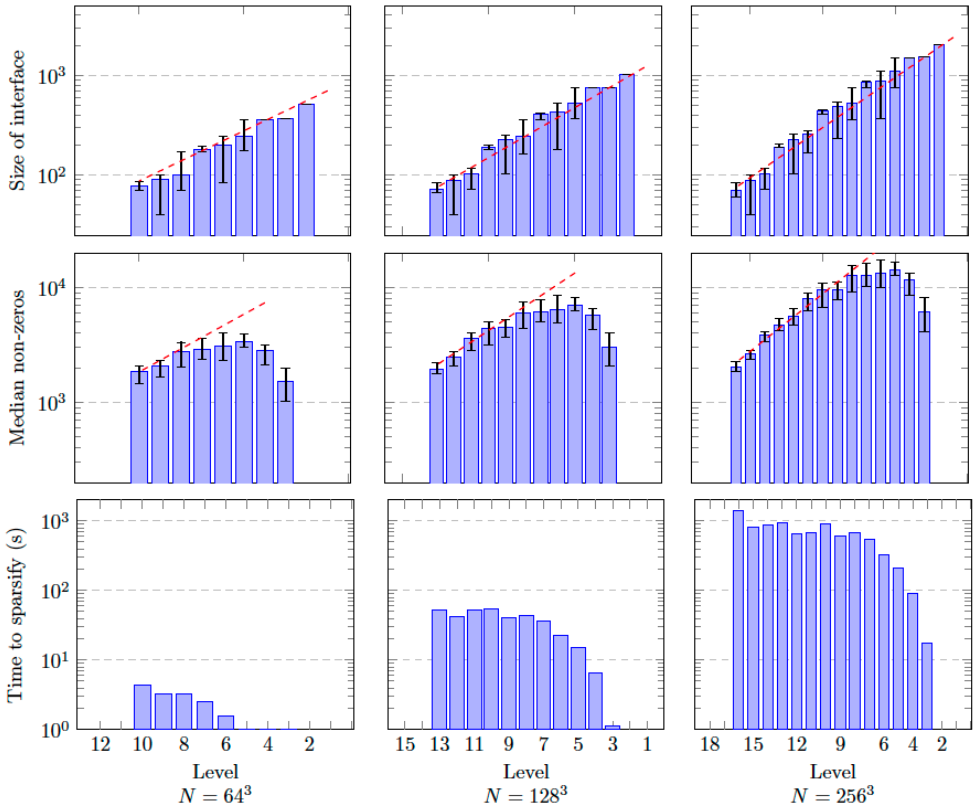


Figure: 3D Advection-Diffusion problem. Red dashed line indicates $2^{-l/3}$ scaling.

Assume that,

- Size of an interface,

$$c_l \in \mathcal{O}(2^{-l/3} N^{1/3})$$

- Number of non-zeros per row/column of an interface,

$$n_l \in \mathcal{O}(2^{-l/3} N^{1/3})$$

- Cost of interface spars/elmn: $\mathcal{O}(2^{-l} N)$

Complexity: Square matrices

3D PDE discretizations

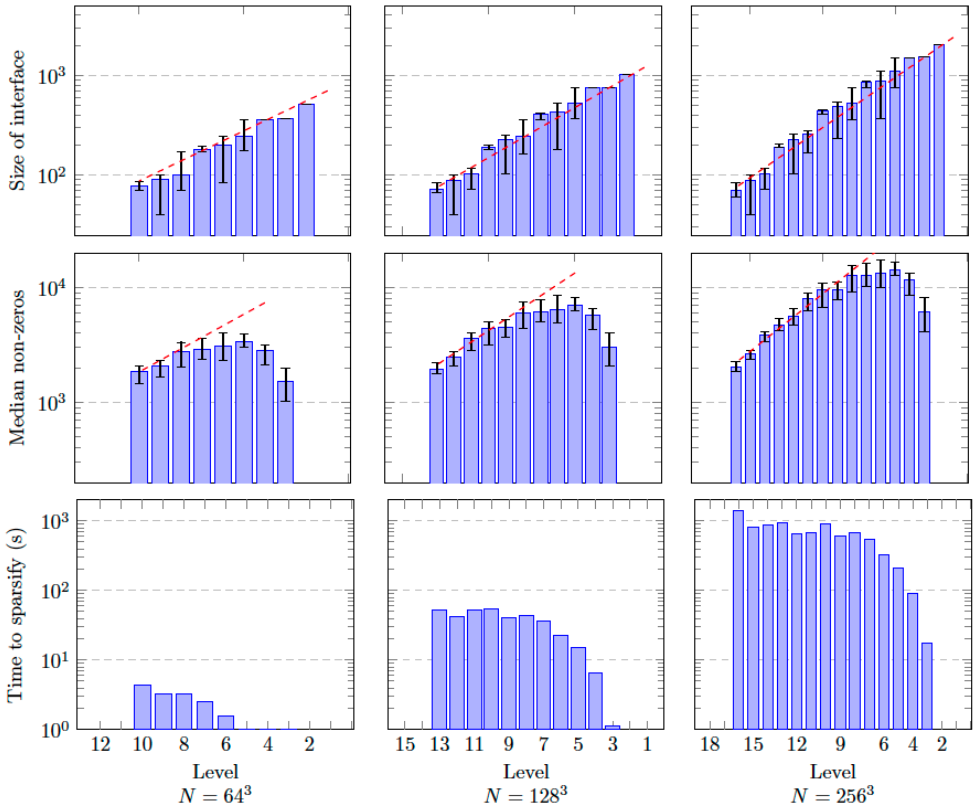


Figure: 3D Advection-Diffusion problem. Red dashed line indicates $2^{-l/3}$ scaling.

Assume that,

- Size of an interface,

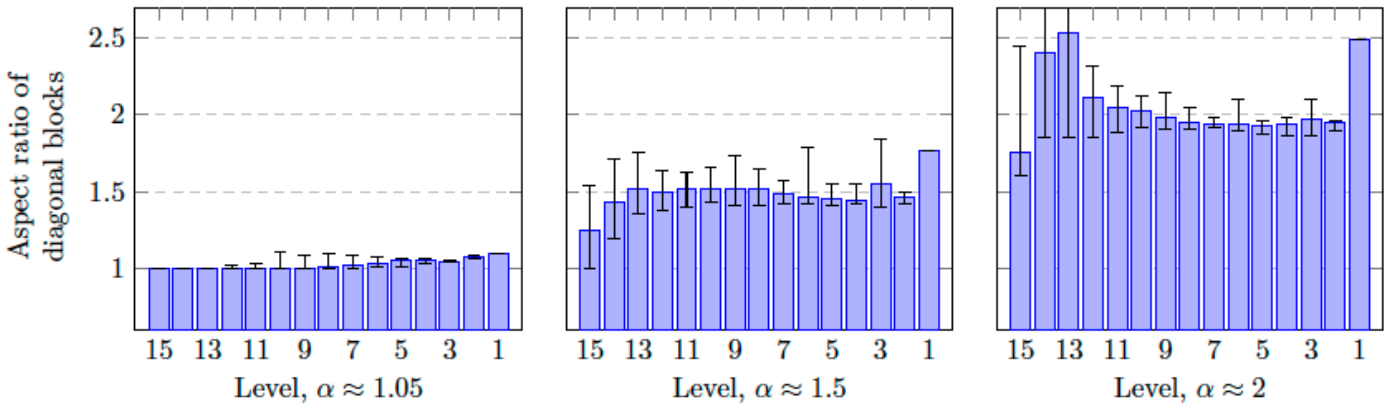
$$c_l \in \mathcal{O}(2^{-l/3} N^{1/3})$$

- Number of non-zeros per row/column of an interface,

$$n_l \in \mathcal{O}(2^{-l/3} N^{1/3})$$

- Cost of interface spars/elmn: $\mathcal{O}(2^{-l} N)$
- Cost at level l: $\mathcal{O}(2^l 2^{-l} N) = \mathcal{O}(N)$

Complexity: Rectangular matrices

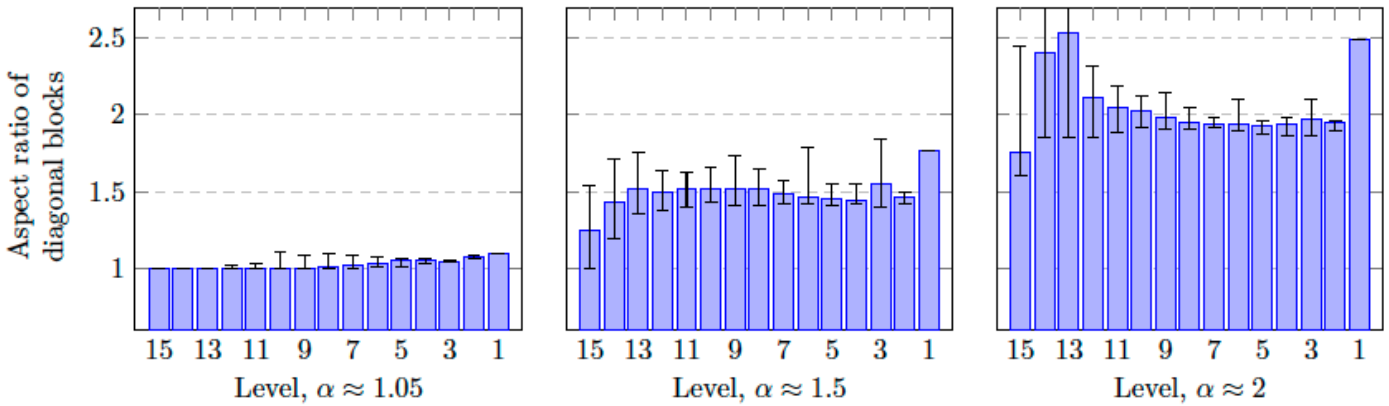


- Aspect ratio (# rows/ # cols) of each interface block is bounded, $\theta(\alpha)$, where $\alpha = M/N$

$$c_l \in \mathcal{O}(2^{-l/3} N^{1/3})$$

$$r_l \in \mathcal{O}(\alpha 2^{-l/3} N^{1/3})$$

Complexity: Rectangular matrices



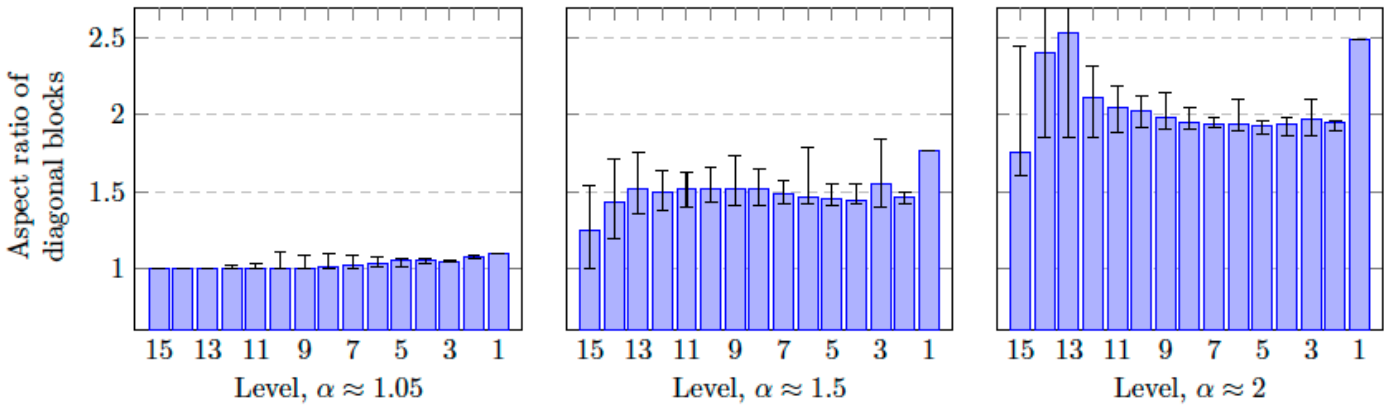
- Aspect ratio (# rows/ # cols) of each interface block is bounded, $\theta(\alpha)$, where $\alpha = M/N$

$$c_l \in \mathcal{O}(2^{-l/3} N^{1/3})$$

$$r_l \in \mathcal{O}(\alpha 2^{-l/3} N^{1/3})$$

- Similar analysis leads to a cost per level of $\mathcal{O}(\alpha N) = \mathcal{O}(M)$

Complexity: Rectangular matrices



- Aspect ratio (# rows/ # cols) of each interface block is bounded, $\theta(\alpha)$, where $\alpha = M/N$

$$c_l \in \mathcal{O}(2^{-l/3} N^{1/3})$$

$$r_l \in \mathcal{O}(\alpha 2^{-l/3} N^{1/3})$$

- Similar analysis leads to a cost per level of $\mathcal{O}(\alpha N) = \mathcal{O}(M)$
- We have $L = \log N$ levels, so total cost of $\mathcal{O}(M \log N)$

Numerical results

Results: Unsymmetric matrices

Advection-Diffusion Equation discretized using finite difference on a regular 3D grid

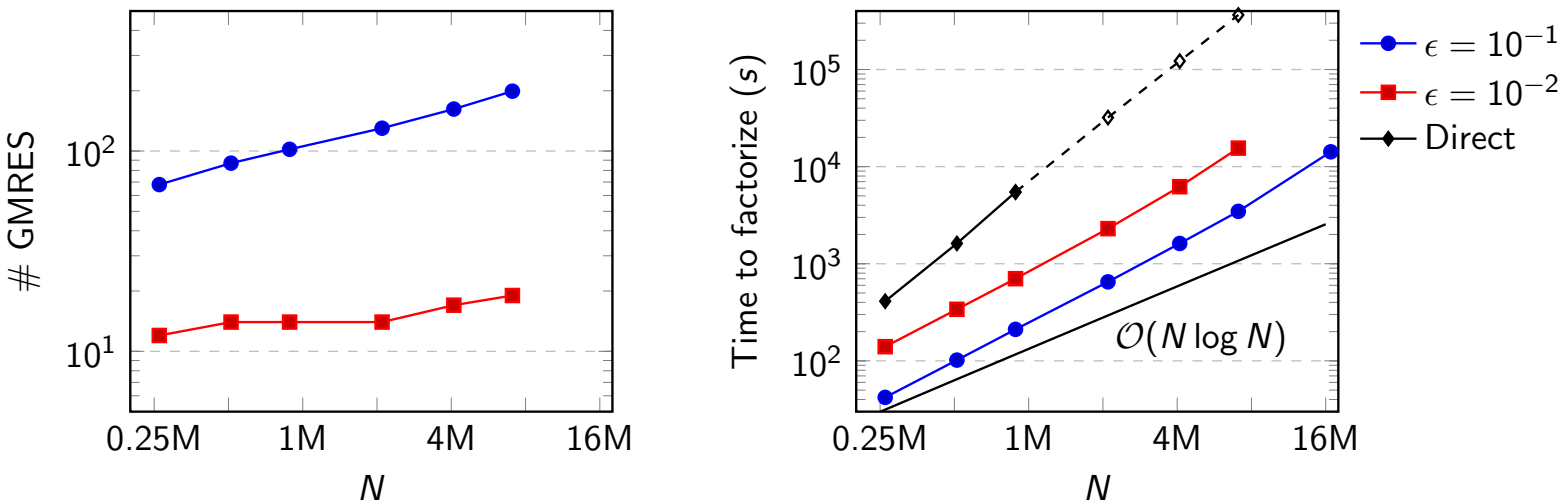


Figure: Variation in the number of GMRES iterations and time to factorize.

Results: Unsymmetric matrices

Advection-Diffusion Equation discretized using finite difference on a regular 3D grid

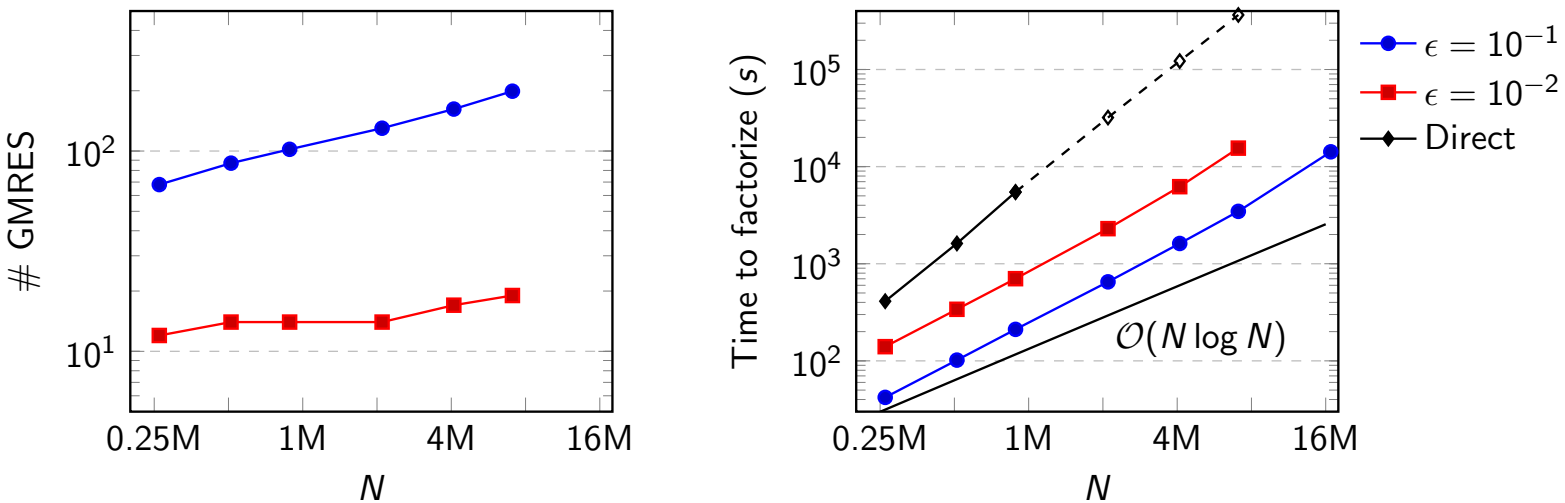


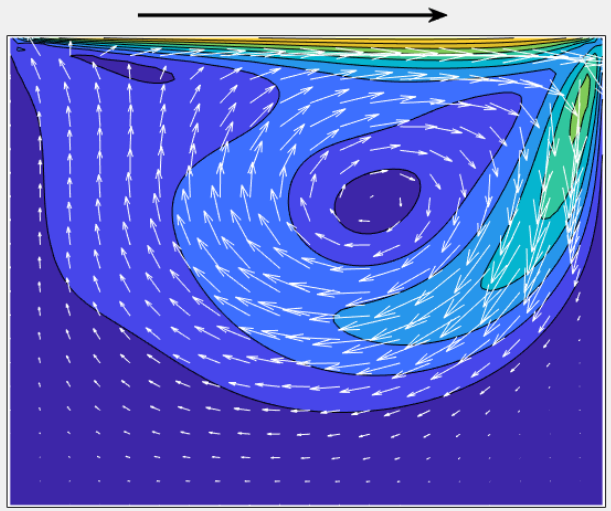
Figure: Variation in the number of GMRES iterations and time to factorize.

- Theoretical complexity is $\mathcal{O}(N \log N)$
- Empirically, the factorization time scales as $\mathcal{O}(N^{1.4})$ likely due to non-asymptotic effects

Results: Lid driven flow in a 2D square cavity

- Unsymmetric matrices
- Preconditioners based on LU factorization fail to converge

		# GMRES, $\epsilon = 10^{-5}$	
Matrix[1]	Re	# GMRES	Residual
E40R0000	0	6	1.9×10^{-14}
E40R0100	100	7	1.5×10^{-13}
E40R0500	500	6	4.1×10^{-13}
E40R1000	1000	11	3.3×10^{-14}
E40R2000	2000	23	7.9×10^{-14}
E40R3000	3000	19	1.5×10^{-13}
E40R4000	4000	36	1.6×10^{-12}
E40R5000	5000	21	5.9×10^{-13}



- spaQR stable and converges in less than 50 iterations for a wide range of Reynolds number

[1] Matrices taken from the SPARSKIT collection

PDE constrained optimization

$$\begin{aligned} \min_{u,z} \quad & \frac{1}{2}\int_\Omega (u-u_d)^2 dx + \frac{1}{2}\lambda \int_\Omega z^2 dx \\ \text{subject to} \quad & -\nabla \cdot (z\nabla u) = h \text{ in } \Omega \\ & u = 0 \text{ in } d\Omega \end{aligned}$$

- Recover u, z at every grid point, using available states u_d

PDE constrained optimization

$$\begin{aligned} \min_{u,z} \quad & \frac{1}{2}\int_\Omega (u-u_d)^2 dx + \frac{1}{2}\lambda \int_\Omega z^2 dx \\ \text{subject to} \quad & -\nabla \cdot (z\nabla u) = h \text{ in } \Omega \\ & u = 0 \text{ in } d\Omega \end{aligned}$$

- Recover u, z at every grid point, using available states u_d
- One approach involves solving the least squares problem

$$J^T x = b$$

where J is the Jacobian of the discretized equations

Results: 2D Inverse Poisson

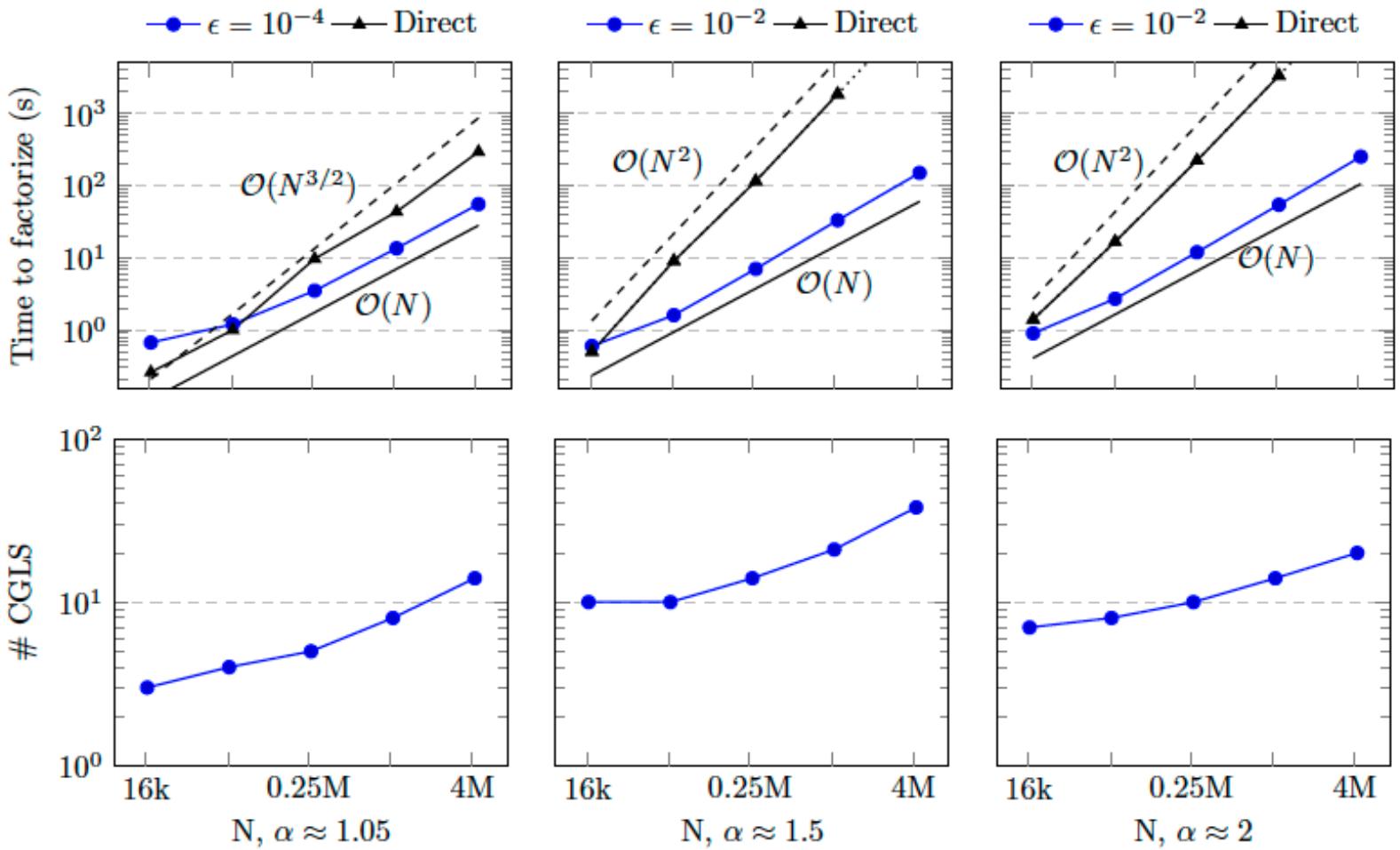


Figure: Results of 2D Inverse Poisson least squares problem for different values of aspect ratio $\alpha = M/N$

Results: 3D Inverse Poisson

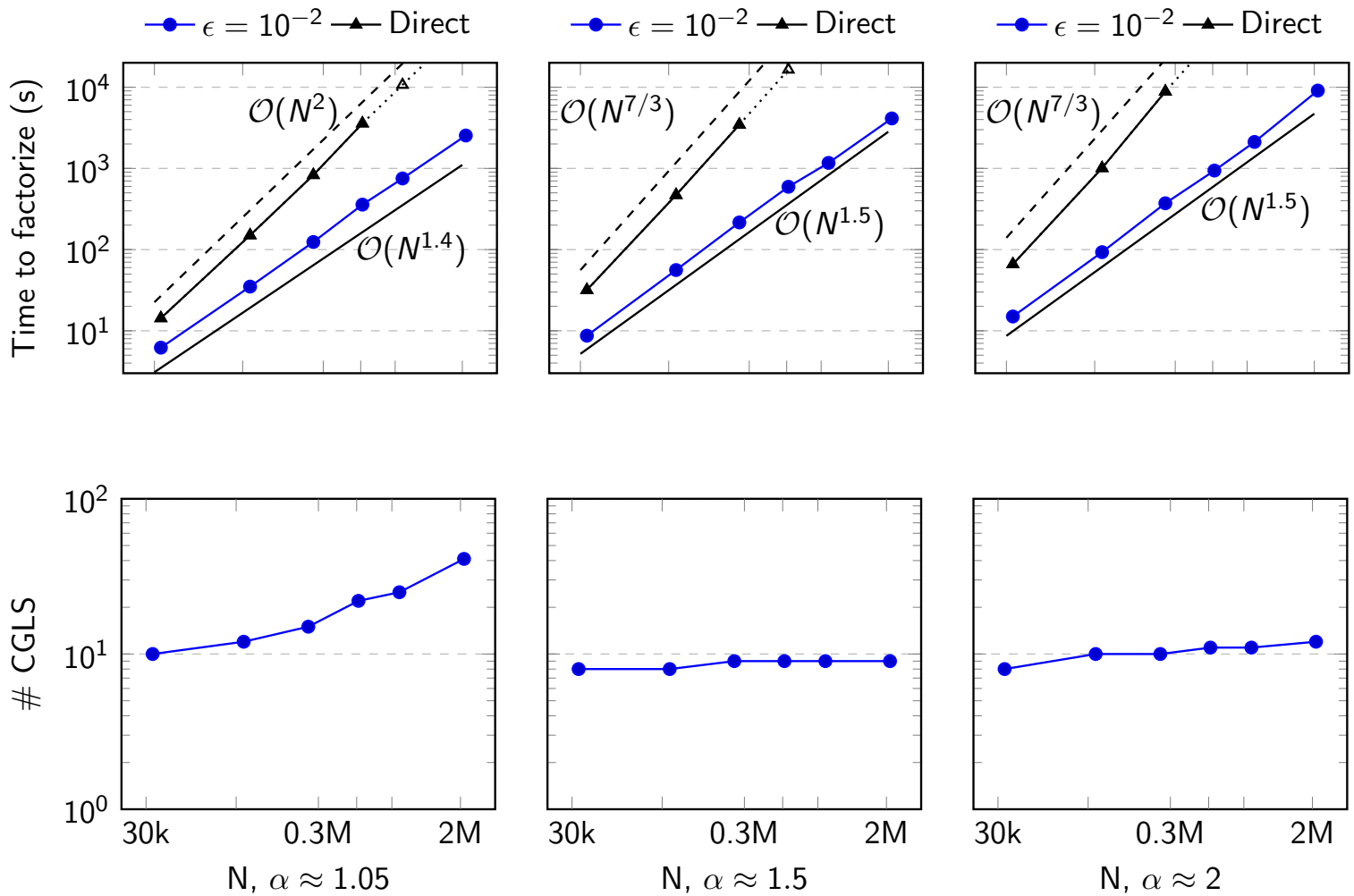


Figure: Results of 3D Inverse Poisson least squares problem for different values of aspect ratio, $\alpha = M/N$.

Conclusions

Conclusions

- spaQR is a hierarchical factorization that produces a fast, sparse, approximate QR factorization of large sparse matrices

Conclusions

- spaQR is a hierarchical factorization that produces a fast, sparse, approximate QR factorization of large sparse matrices
- Cost vs Accuracy compromise achieved with the parameter ϵ

Conclusions

- spaQR is a hierarchical factorization that produces a fast, sparse, approximate QR factorization of large sparse matrices
- Cost vs Accuracy compromise achieved with the parameter ϵ
- Structure of the spaQR algorithm naturally exhibits parallelism

Conclusions

- spaQR is a hierarchical factorization that produces a fast, sparse, approximate QR factorization of large sparse matrices
- Cost vs Accuracy compromise achieved with the parameter ϵ
- Structure of the spaQR algorithm naturally exhibits parallelism
- Useful for various applications in CFD, PDE constrained optimization and applications in computer graphics and vision

References & Funding

- References:
  - A. Gnanasekaran and E. Darve, Hierarchical orthogonal factorization: Sparse Square Matrices, 2020, `https://arxiv.org/abs/2010.06807`
  - A. Gnanasekaran and E. Darve, Hierarchical orthogonal factorization: Sparse Least Squares Problems, 2021, `https://arxiv.org/abs/2102.09878`
- spaQR code:
  - `https://github.com/Abeynaya/spaQR_public`
- Funding:
  - LDRD Research Grant from Sandia National Laboratories
  - NASA grant #80NSSC18M0152