Parallel Multilevel Low-Rank approximation preconditioners

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

ICME, Stanford

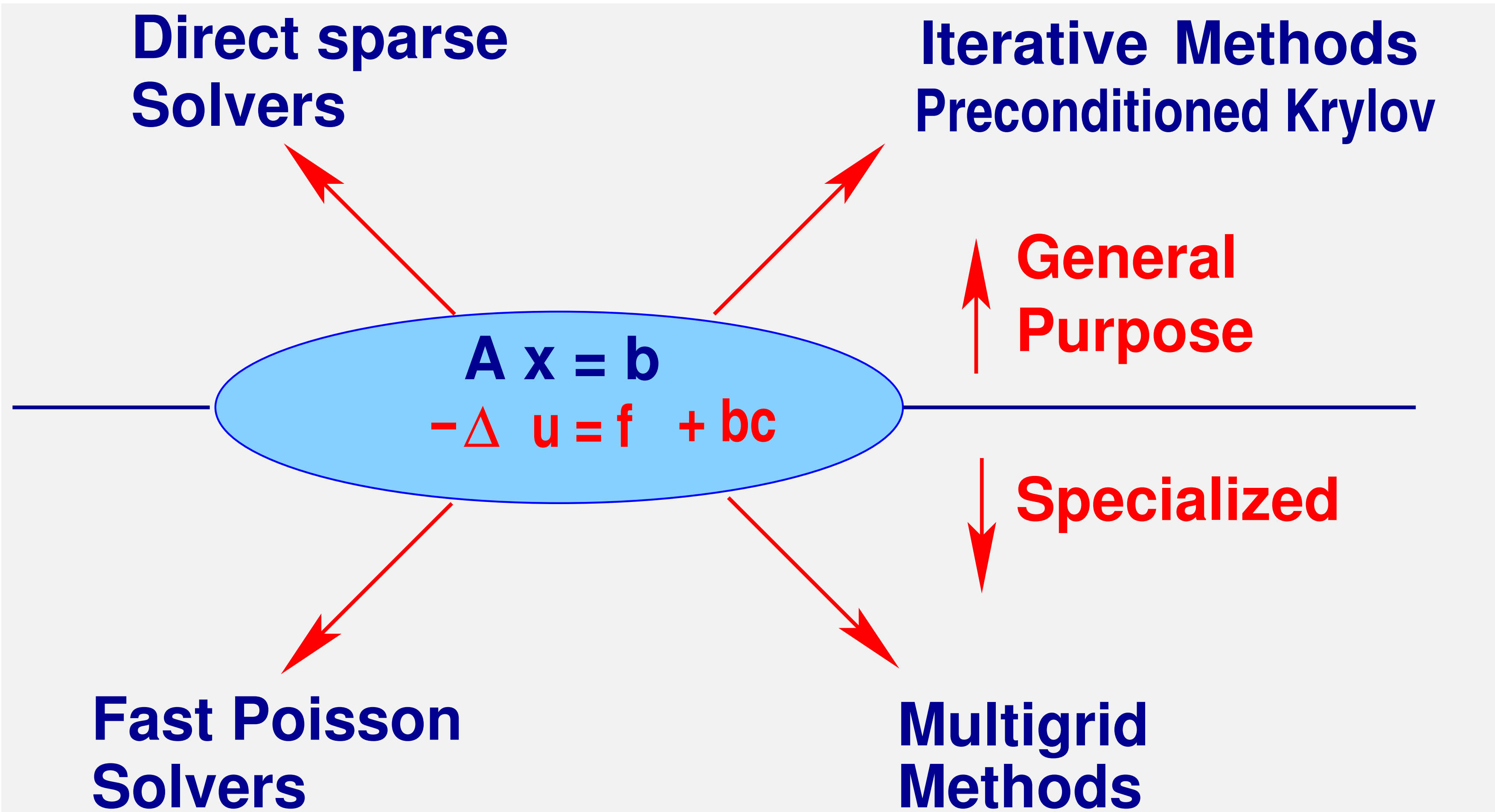
April 20th, 2015

First:

- Partly joint work with Ruipeng Li and Yuanzhe Xi

- Work supported by NSF-DMS

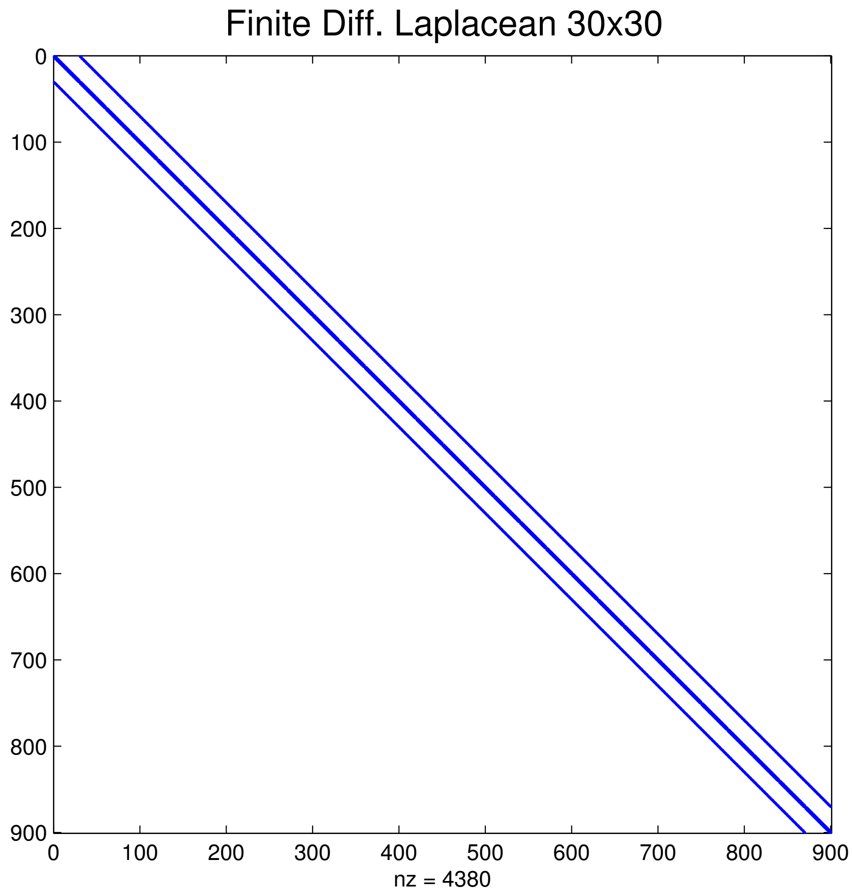
Introduction: Linear System Solvers

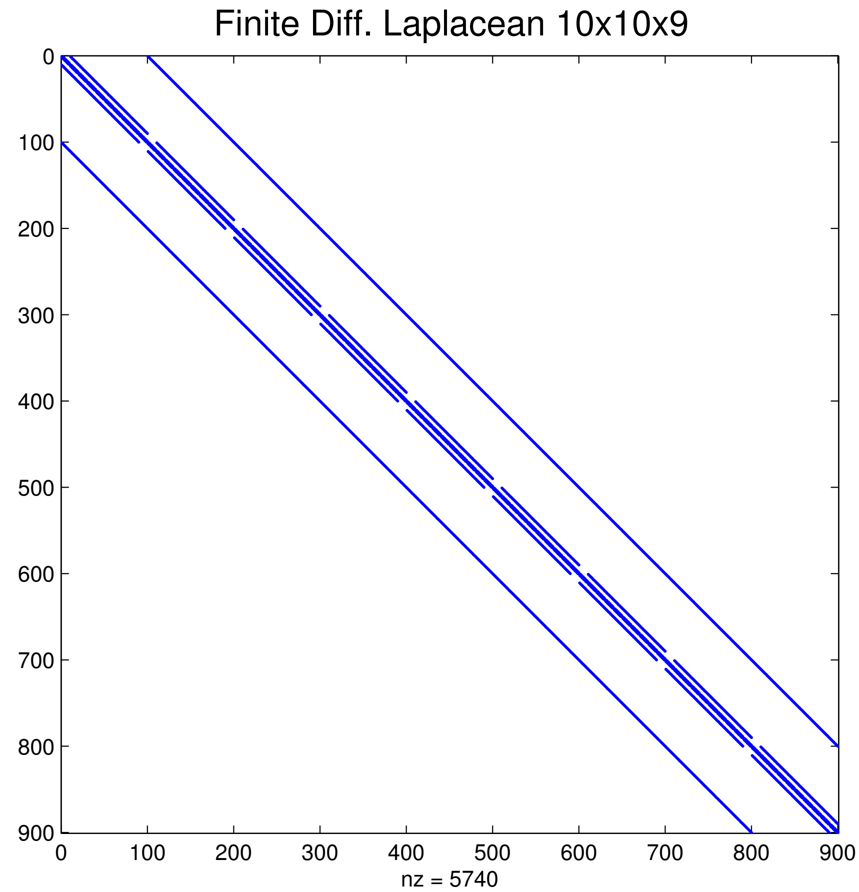


Long standing debate: direct vs. iterative

- Starting in the 1970's: huge progress of sparse direct solvers
- Iterative methods - much older - not designed for 'general systems'. Big push in the 1980s with help from 'preconditioning'
- General consensus now: Direct methods do well for 2-D problems and some specific applications [e.g., structures, ...]
- Usually too expensive for 3-D problems
- Huge difference between 2-D and 3-D case
- Test: Two Laplacean matrices of same dimension $n = 122,500$. First: on a 350×350 grid (2D); Second: on a $50 \times 50 \times 49$ grid (3D)

➤ Pattern of a similar [much smaller] coefficient matrix





Background: Preconditioned iterative solvers

Two ingredients:

- An accelerator: Conjugate gradient, BiCG, GMRES, BICGSTAB,.. [‘Krylov subspace methods’]
- A preconditioner: makes the system easier to solve by accelator, e.g. Incomplete LU factorizations; SOR/SSOR; Multigrid, ...

One viewpoint:

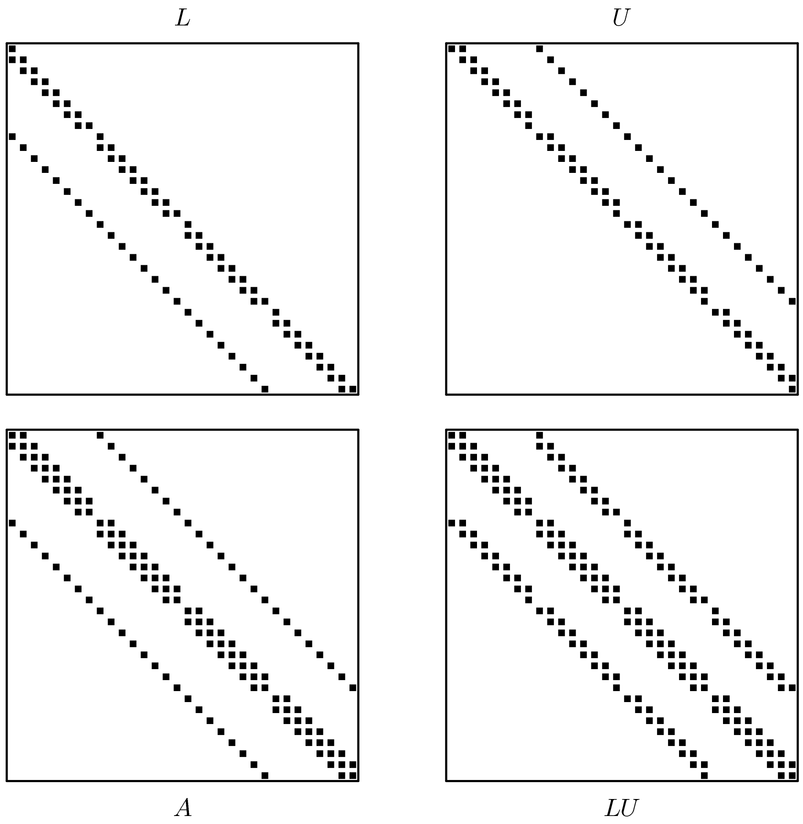
➤ Goal of accelerator: find best combination of basic iterates

➤ Goal of preconditioner: generate good basic iterates.. [Gauss-Seidel, ILU, ...]

Background: Incomplete LU (ILU) preconditioners

ILU: $A \approx LU$

Simplest Example: ILU(0) $\longrightarrow$



Common difficulties of ILUs:

Often fail for indefinite problems

Not too good for highly parallel environments

SPARSE MATRIX COMPUTATIONS ON GPUS

Sparse matrix computations with GPUs **

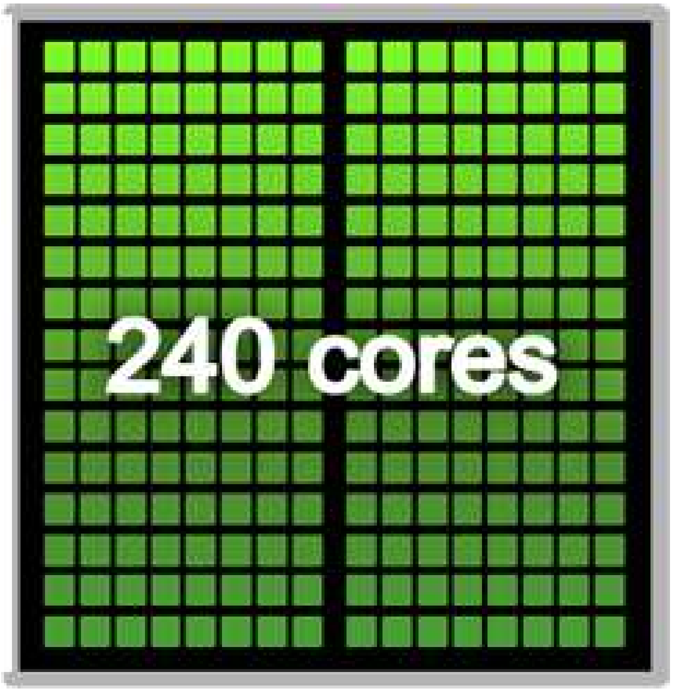
- ➤ GPUs Currently a very popular approach to: inexpensive supercomputing

- ➤ Can buy $\sim$ one Teraflop peak power for around a little more tham $1,000

Tesla C1060

** Joint work with Ruipeng Li

Tesla C 1060:



* 240 cores
* 4 GB memory
* Peak rate: 930 GFLOPS [single]
* Clock rate: 1.3 Ghz
* 'Compute Capability': 1.3 [allows double precision]

- ➤ Next: Fermi [48 cores/SM]— followed by: Kepler..

- ➤ Tesla K 80 : 2 $\times$ 2,496 $\rightarrow$ $\mathbf{4992}$ GPU cores. 24 GB Mem.; Peak: $\approx$ 2.91 TFLOPS double prec. [with clock Boost].

The CUDA environment: The big picture

➤ A host (CPU) and an attached device (GPU)

Typical program:

1. Generate data on CPU
2. Allocate memory on GPU

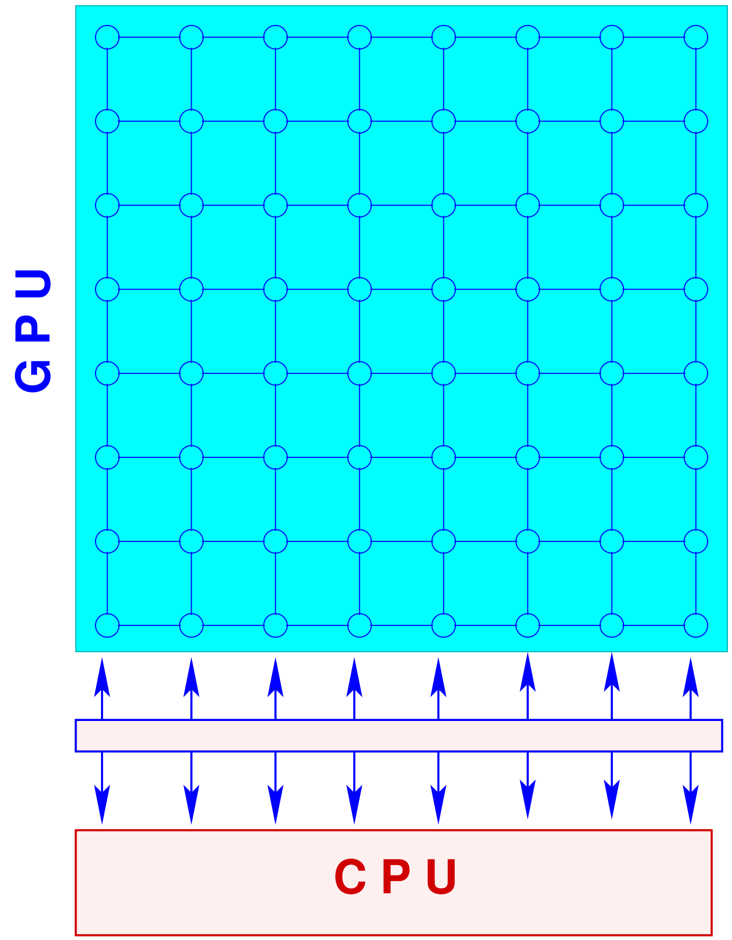
   `cudaMalloc(...)`
3. Send data Host $\rightarrow$ GPU

   `cudaMemcpy(...)`
4. Execute GPU 'kernel':

`kernel <<<(...)>>>(..)`

5. Copy data GPU $\rightarrow$CPU

   `cudaMemcpy(...)`



Sparse matrix computations on GPUs

Main issue in using GPUs for sparse computations:

- Huge performance degradation due to 'irregular sparsity'

➤ Matrices:

Matrix -name	N	NNZ
FEM/Cantilever	62,451	4,007,383
Boeing/pwtk	217,918	11,634,424

➤ Performance of Mat-Vecs on NVIDIA Tesla C1060

	Single Precision			*Double Precision*		
Matrix	CSR	JAD	DIA+	CSR	JAD	DIA+
FEM/Cantilever	9.4	10.8	25.7	7.5	5.0	13.4
Boeing/pwtk	8.9	16.6	29.5	7.2	10.4	14.5

- More recent tests: NVIDIA M2070 (Fermi), Xeon X5675
- Double precision in Gflops

MATRIX	Dim. N	CPU	CSR	JAD	HYB	DIA
rma10	46,835	3.80	10.19	12.61	8.48	-
cfd2	123,440	2.88	8.52	11.95	12.18	-
majorbasis	160,000	2.92	4.81	11.70	11.54	13.06
af_shell8	504,855	3.13	10.34	14.56	14.27	-
lap7pt	1,000,000	2.59	4.66	11.58	12.44	18.70
atmosmodd	1,270,432	2.09	4.69	10.89	10.97	16.03

- CPU SpMV: Intel MKL, parallelized using OpenMP
- HYB: from CUBLAS Library. [Uses ellpack+csr combination]

(*) Thanks: all matrices from the Univ. Florida sparse matrix collection

Sparse Forward/Backward Sweeps

➤ Next major ingredient of precond. Krylov subs. methods

➤ ILU preconditioning operations require L/U solves: $x \leftarrow U^{-1}L^{-1}x$

➤ Sequential outer loop.

```
for i=1:n
    for j=ia(i):ia(i+1)
        x(i) = x(i) - a(j)*x(ja(j))
    end
end
```

➤ Parallelism can be achieved with level scheduling:

- Group unknowns into levels
- Compute unknowns $x(i)$ of same level simultaneously
- $1 \leq nlev \leq n$

ILU: Sparse Forward/Backward Sweeps

- Very poor performance [relative to CPU]

Matrix	N	CPU Mflops	GPU-Lev #lev	GPU-Lev Mflops
Boeing/bcsstk36	23,052	627	4,457	43
FEM/Cantilever	62,451	653	2,397	168
COP/CASEYK	696,665	394	273	142
COP/CASEKU	208,340	373	272	115

Prec: miserable :-)

GPU Sparse Triangular Solve with Level Scheduling

➤ Very poor performance when #levs is large

➤ A few things can be done to reduce the # levels but perf. will remain poor

So...

...Either GPUs must go...

... or ILUs must go...

Alternatives to ILU preconditioners

- What would be a good alternative?

Wish-list: A preconditioner that

- Requires few 'irregular' computations
- Trades volume of computations for speed
- Is robust for indefinite problems

- Candidate:

- Multilevel Low-Rank (MLR) approximate inverse preconditioners

Related work:

- Work on H-matrices [Hackbusch, ...]

- Work on HSS matrices [e.g., JIANLIN XIA, SHIVKUMAR CHANDRASEKARAN, MING GU, AND XIAOYE S. LI, *Fast algorithms for hierarchically semiseparable matrices*, Numerical Linear Algebra with Applications, 17 (2010), pp. 953–976.]

- Work on 'balanced incomplete factorizations' (R. Bru et al.)

- Work on "sweeping preconditioners" by Engquist and Ying.

- Work on computing the diagonal of a matrix inverse [Jok Tang and YS (2010) ..]

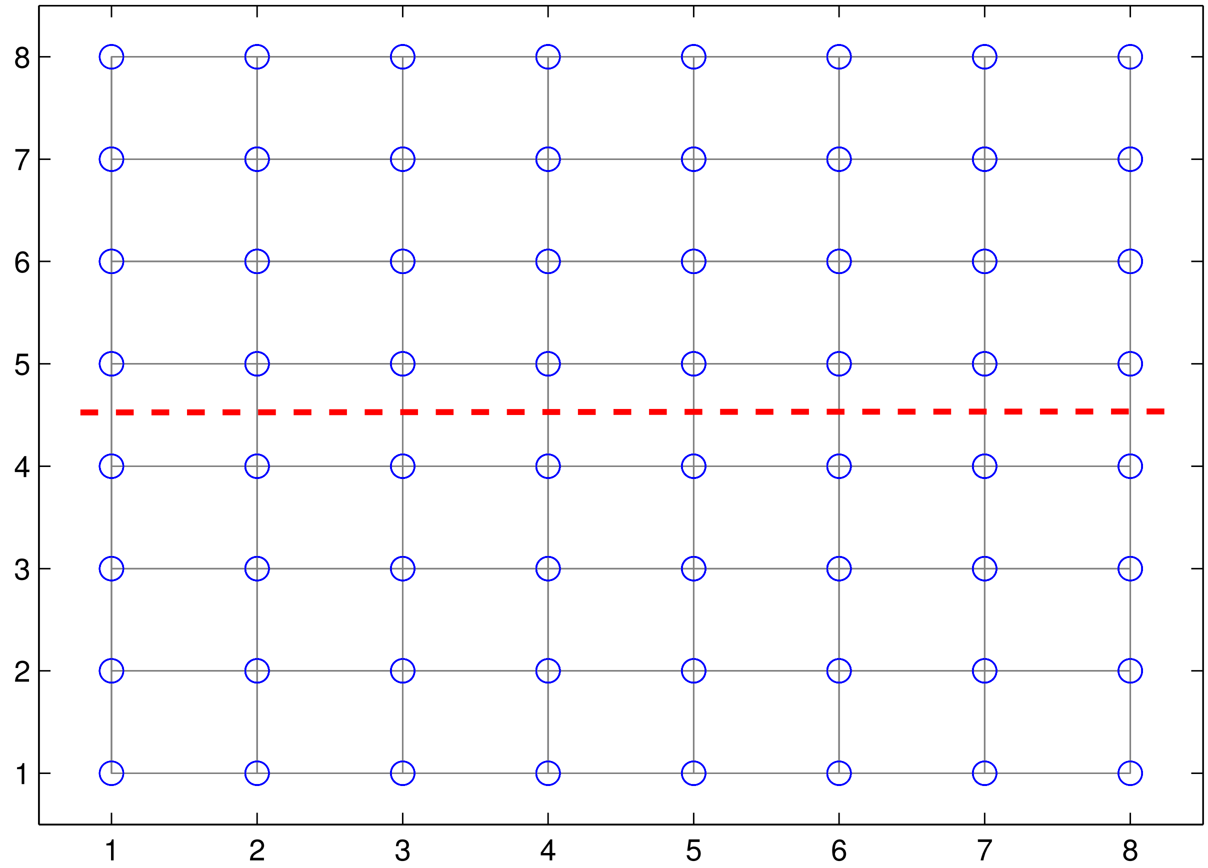
Low-rank Multilevel Approximations

➤ Starting point: symmetric matrix derived from a 5-point discretization of a 2-D Pb on $n_x \times n_y$ grid

$$A = \left(\begin{array}{cccc|ccc}
A_1 & D_2 & & & & & \\
D_2 & A_2 & D_3 & & & & \\
 & \ddots & \ddots & \ddots & & & \\
 & & D_\alpha & A_\alpha & D_{\alpha+1} & & \\
\hline
 & & & D_{\alpha+1} & A_{\alpha+1} & \ddots & \\
 & & & & \ddots & \ddots & \ddots \\
 & & & & & D_{n_y} & A_{n_y}
\end{array}\right)$$

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \equiv \begin{pmatrix} A_{11} & \\ & A_{22} \end{pmatrix} + \begin{pmatrix} & A_{12} \\ A_{21} & \end{pmatrix}$$

Corresponding splitting on FD mesh:



- $\boldsymbol{A}_{11} \in \mathbb{R}^{m\times m}$, $\boldsymbol{A}_{22} \in \mathbb{R}^{(n-m)\times(n-m)}$

- In the simplest case second matrix is:

$$\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_{11} & \\ & \boldsymbol{A}_{22} \end{pmatrix} + \left[\begin{array}{c|c} & -\mathbf{I} \\ \hline -\mathbf{I} & \end{array}\right]$$

- Write 2nd matrix as:

$$\left[\begin{array}{c|c} & -\mathbf{I} \\ \hline -\mathbf{I} & \end{array}\right] = \left[\begin{array}{c|c} +\mathbf{I} & \\ \hline & +\mathbf{I} \end{array}\right] - \underbrace{\left[\begin{array}{c|c} \mathbf{I} & \mathbf{I} \\ \hline \mathbf{I} & \mathbf{I} \end{array}\right]}_{\mathbf{E}\,\mathbf{E}^T}$$

$$\mathbf{E}^T = \left[\begin{array}{c|c} \mathbf{I} & \mathbf{I} \end{array}\right]$$

➤ Above splitting can be rewritten as

$$A = \underbrace{(A + EE^T)}_{B} - EE^T$$

$$A = B - EE^T,$$

$$B := \begin{pmatrix} B_1 & \\ & B_2 \end{pmatrix} \in \mathbb{R}^{n\times n}, \quad E := \begin{pmatrix} E_1 \\ E_2 \end{pmatrix} \in \mathbb{R}^{n\times n_x},$$

Note: $B_1 := A_{11} + E_1E_1^T, \quad B_2 := A_{22} + E_2E_2^T.$

➤ Next : Use Shermann - Morrison formula:

$$A^{-1} = B^{-1} + (B^{-1}E)X^{-1}(B^{-1}E)^T$$
$$X = I - E^T B^{-1} E$$

- Method: Use low-rank approx. for $B^{-1}E$ $\boxed{B^{-1}E \approx U_k V_k^T,}$ $U_k \in \mathbb{R}^{n\times k}$, $V_k \in \mathbb{R}^{n_x\times k}$,

- Replace $B^{-1}E$ by $U_k V_k^T$ in $X = I - (E^T B^{-1})E$:

  $X \approx G_k = I - V_k U_k^T E, \quad (\in \mathbb{R}^{n_x\times n_x})$ Leads to ...

Preconditioner

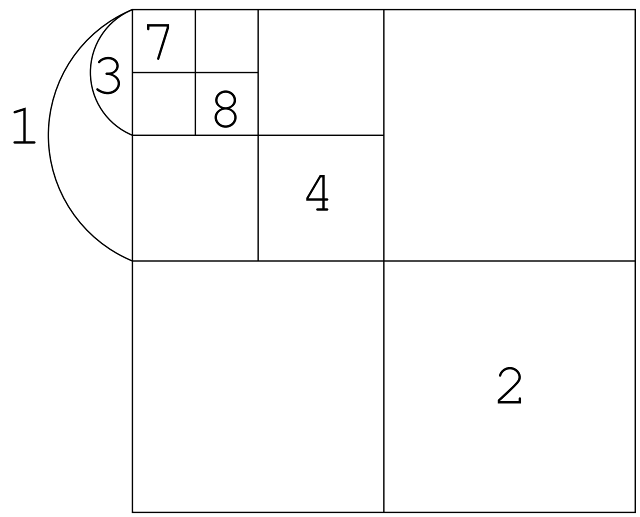
$$M^{-1} = B^{-1} + U_k H_k U_k^T, \quad H_k = V_k^T G_k^{-1} V_k$$

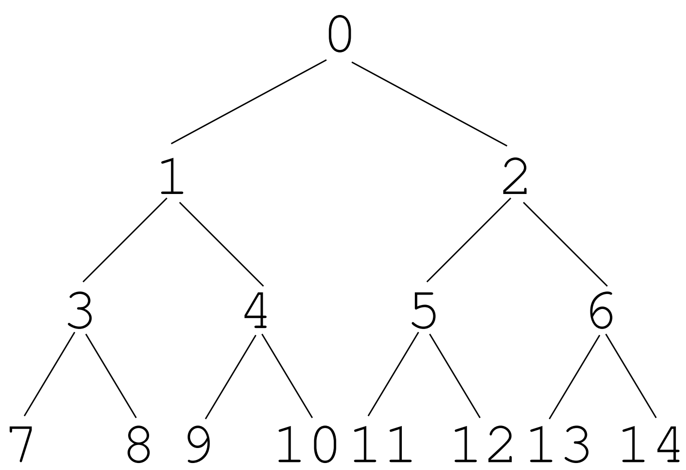
Use recursivity

- We can show : $H_k = (I - U_k^T E V_k)^{-1}$ and $H_k^T = H_k$

Recursive multilevel framework

- $A_i = B_i + E_i E_i^T$, $B_i \equiv \begin{pmatrix} B_{i_1} & \\ & B_{i_2} \end{pmatrix}$.
- Next level, set $A_{i_1} \equiv B_{i_1}$ and $A_{i_2} \equiv B_{i_2}$
- Repeat on A_{i_1}, A_{i_2}
- Last level, factor A_i (IC, ILU)
- Binary tree structure:





Theory: 2-level analysis for model problem

➤ Interested in eigenvalues γ_j of

$$A^{-1} - B^{-1} = B^{-1}EX^{-1}E^TB^{-1}$$

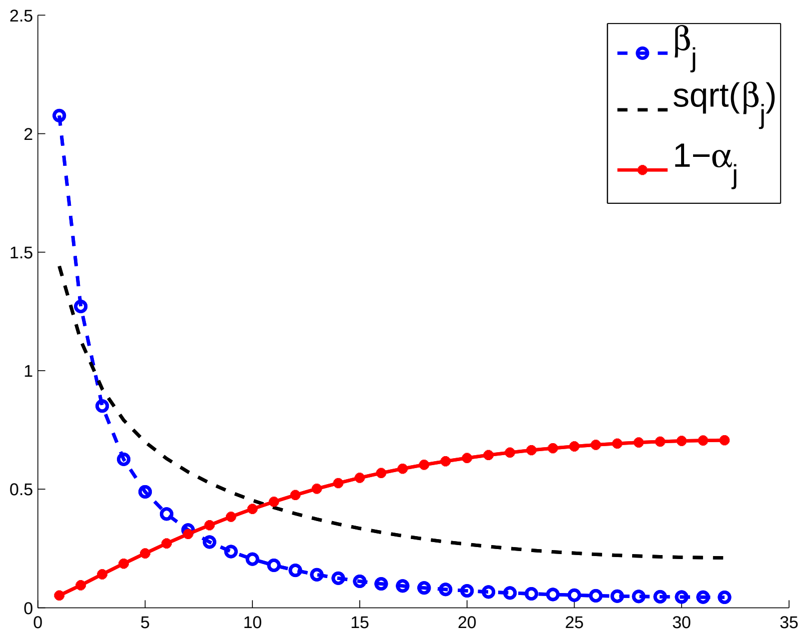
when A = Pure Laplacean .. They are:

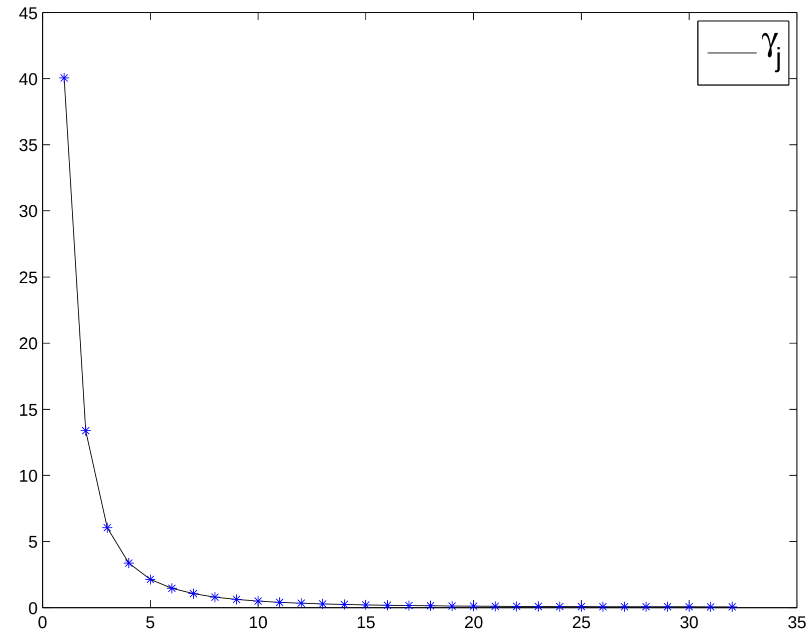
$$\gamma_j = \frac{\beta_j}{1-\alpha_j}, \quad j = 1, \cdots, n_x \quad \text{with:}$$

$$\beta_j = \sum_{k=1}^{n_y/2} \frac{\sin^2 \frac{n_y k\pi}{n_y+1}}{4\left(\sin^2 \frac{k\pi}{n_y+1} + \sin^2 \frac{j\pi}{2(n_x+1)}\right)^2},$$

$$\alpha_j = \sum_{k=1}^{n_y/2} \frac{\sin^2 \frac{n_y k\pi}{n_y+1}}{\sin^2 \frac{k\pi}{n_y+1} + \sin^2 \frac{j\pi}{2(n_x+1)}}.$$

➤ Decay of the γ_j's when $nx = ny = 32$.



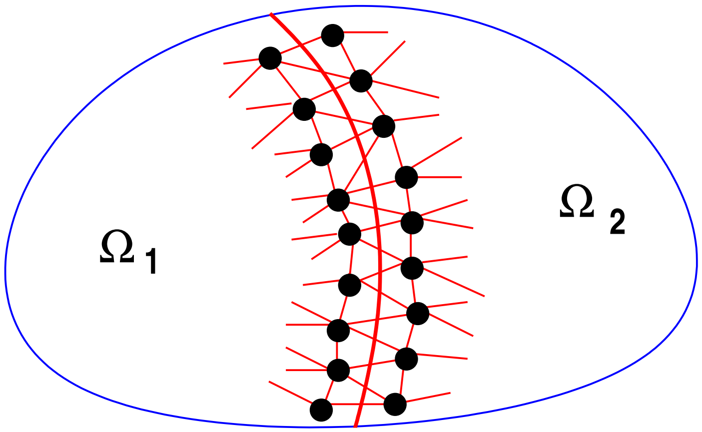


Note $\sqrt{\beta_j}$ are the singular values of $B^{-1}E$.

In this particular case 3 eigenvectors will capture 92 % of the inverse whereas 5 eigenvectors will capture 97% of the inverse.

Generalization: Domain Decomposition framework

Domain partitioned into 2 domains with an edge separator

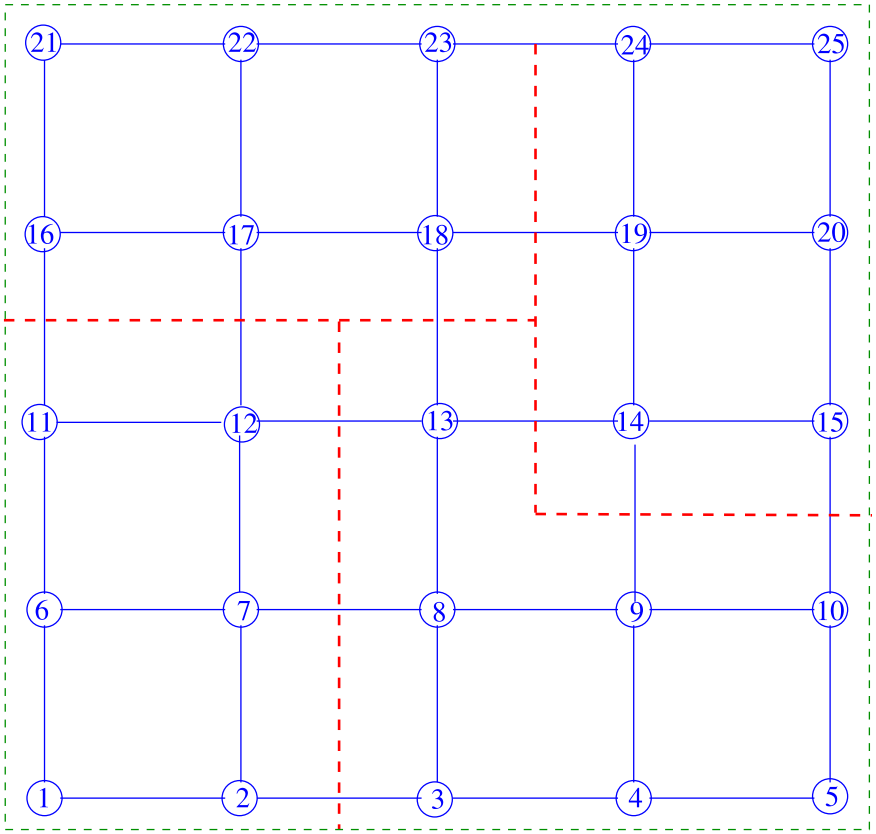


- Previous approach extended to this case –
- Issue with this general approach: recursive implementation

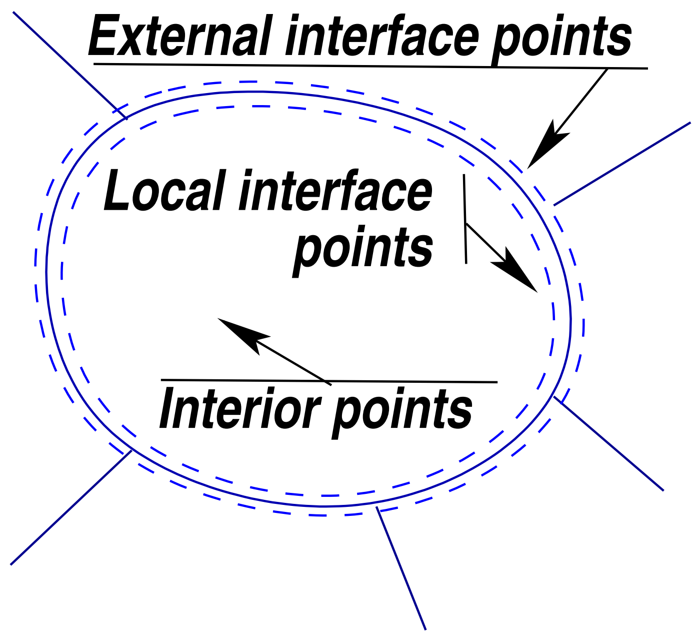
Avoiding recursivity: ‘standard’ DD framework

- Goal: avoid recursivity
- Consider a domain partitioned in p sub-domains using vertex- based partitioniong (edge-separator)

➤ Interface nodes in each domain are listed last.



Local view:

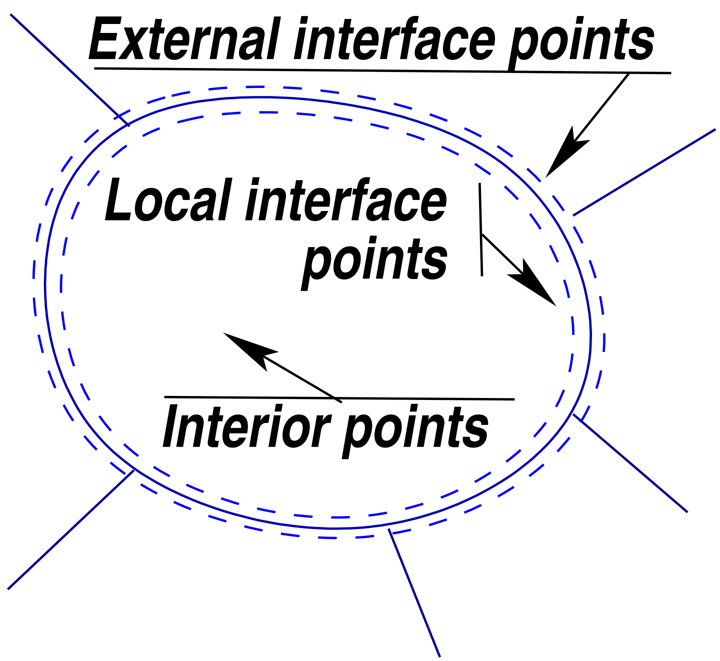


$$\begin{pmatrix} B_i & F_i \\ E_i^T & C_i \end{pmatrix} \begin{pmatrix} u_i \\ y_i \end{pmatrix} + \begin{pmatrix} 0 \\ \sum_{j \in N_i} E_{ij} y_j \end{pmatrix} = \begin{pmatrix} f_i \\ g_i \end{pmatrix}$$

The global system: Global view

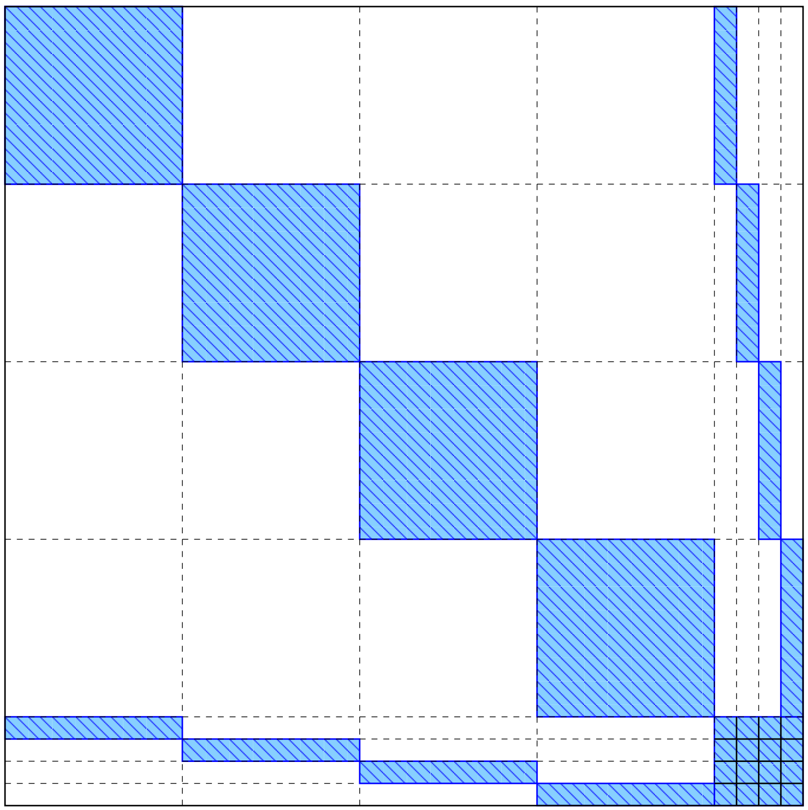
- Global system can be permuted to the form $\rightarrow$
- u_i's internal variables
- y interface variables

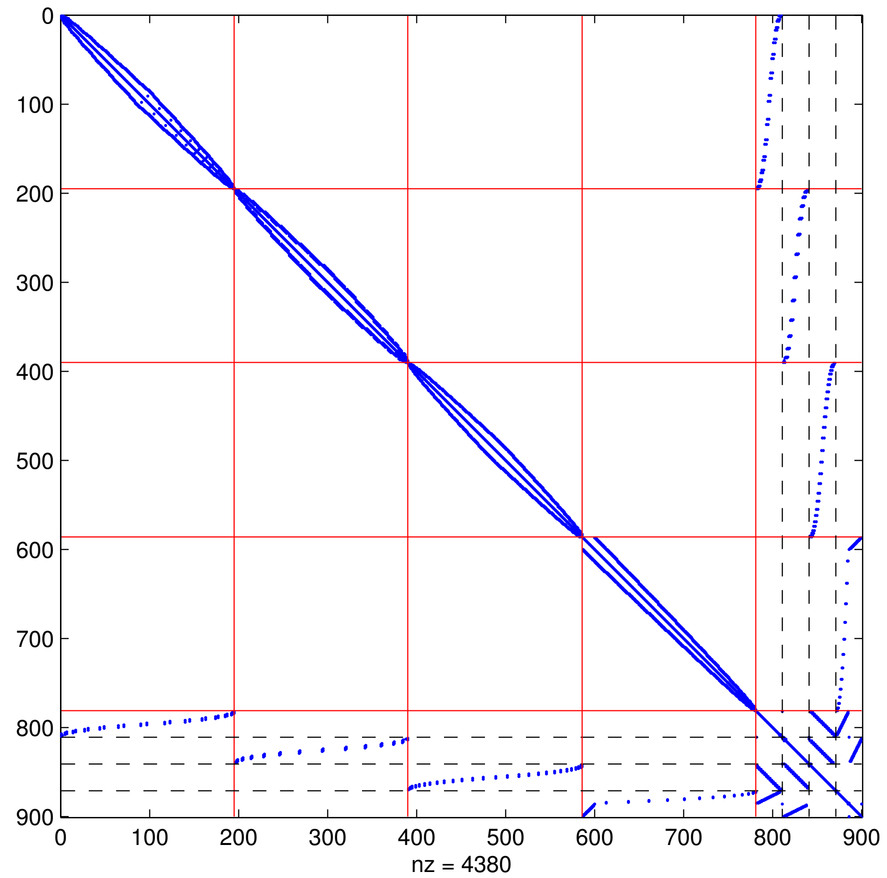
$$\boxed{\begin{pmatrix} B_1 & & & \dots & \hat{F}_1 \\ & B_2 & & \dots & \hat{F}_2 \\ \vdots & & \ddots & & \vdots \\ & & & B_p & \hat{F}_p \\ \hat{E}_1^T & \hat{E}_2^T & \dots & \hat{E}_p^T & C \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \\ y \end{pmatrix} = b}$$



- $\hat{F}_i$ maps local interface points to interior points in domain Ω_i
- $\hat{E}_i^T$ does the reverse operation

Example:





Splitting

- Split as: $$A \equiv \begin{pmatrix} B & \hat{F} \\ \hat{E}^T & C \end{pmatrix} = \begin{pmatrix} B & \\ & C \end{pmatrix} + \begin{pmatrix} & \hat{F} \\ \hat{E}^T & \end{pmatrix}$$
- Define: $F \equiv \begin{pmatrix} \alpha^{-1}\hat{F} \\ -\alpha I \end{pmatrix}$; $E \equiv \begin{pmatrix} \alpha^{-1}\hat{E} \\ -\alpha I \end{pmatrix}$ Then:

$$\left[\begin{array}{c|c} B & \hat{F} \\ \hline \hat{E}^T & C \end{array}\right] = \left[\begin{array}{c|c} B + \alpha^{-2}\hat{F}\hat{E}^T & 0 \\ \hline 0 & C + \alpha^2 I \end{array}\right] - FE^T.$$

- α is a parameter
- Property: $\hat{F}\hat{E}^T$ is 'local', i.e., no inter-domain couplings $\rightarrow$

$$A_0 \equiv \left[\begin{array}{c|c} B + \alpha^{-2}\hat{F}\hat{E}^T & 0 \\ \hline 0 & C + \alpha^2 I \end{array}\right]$$

$=$ block-diagonal

Low-Rank Approximation DD preconditioners

Sherman-Morrison $\rightarrow$

$$\boxed{\begin{aligned} \boldsymbol{A}^{-1} &= \boldsymbol{A}_0^{-1} + \boldsymbol{A}_0^{-1}\boldsymbol{F}\boldsymbol{G}^{-1}\boldsymbol{E}^T\boldsymbol{A}_0^{-1} \\ \boldsymbol{G} &\equiv \boldsymbol{I} - \boldsymbol{E}^T\boldsymbol{A}_0^{-1}\boldsymbol{F} \end{aligned}}$$

Options:

(a) Approximate $\boldsymbol{A}_0^{-1}\boldsymbol{F}, \boldsymbol{E}^T\boldsymbol{A}_0^{-1}, \boldsymbol{G}^{-1}$ [as before]

(b) Approximate only $\boldsymbol{G}^{-1}$ [new]

- (b) requires 2 solves with $\boldsymbol{A}_0$.

Let $\boldsymbol{G} \approx \boldsymbol{G}_k$

Preconditioner $\rightarrow$

$$\boldsymbol{M}^{-1} = \boldsymbol{A}_0^{-1} + \boldsymbol{A}_0^{-1}\boldsymbol{F}\boldsymbol{G}_k^{-1}\boldsymbol{E}^T\boldsymbol{A}_0^{-1}$$

Symmetric Positive Definite case

- Recap: Let $\boxed{G \equiv I - E^T A_0^{-1} E \equiv I - H}$. Then

$$A^{-1} = A_0^{-1} + A_0^{-1} E G^{-1} E^T A_0^{-1}$$

- Approximate G^{-1} by $G_k^{-1} \rightarrow$ preconditioner:

$$M^{-1} = A_0^{-1} + (A_0^{-1} E) G_k^{-1} (E^T A_0^{-1})$$

- Matrix A_0 is SPD
- Can show: $0 \le \lambda_j(H) < 1$.

- Now take rank-k approximation to H:

$$H \approx U_k D_k U_k^T \qquad G_k = I - U_k D_k U_k^T \quad \rightarrow$$

$$G_k^{-1} \equiv (I - U_k D_k U_k^T)^{-1} = I + U_k[(I - D_k)^{-1} - I]U_k^T$$

- Observation: $A^{-1} = M^{-1} + A_0^{-1} E[G^{-1} - G_k^{-1}]E^T A_0^{-1}$

- G_k: k largest eigenvalues of H matched – others set == 0

- Result: AM^{-1} has
  - $n - s + k$ eigenvalues == 1
  - All others between 0 and 1

Alternative: reset lowest eigenvalues to constant

- Let $\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ = exact (full) diagonalization of $\boldsymbol{H}$
- We replaced $\boldsymbol{\Lambda}$ by:

$$\begin{pmatrix} \boldsymbol{\lambda}_1 & & & & & & \\ & \boldsymbol{\lambda}_2 & & & & & \\ & & \ddots & & & & \\ & & & \boldsymbol{\lambda}_k & & & \\ & & & & \mathbf{0} & & \\ & & & & & \ddots & \\ & & & & & & \mathbf{0} \end{pmatrix}$$

- Alternative: replace $\boldsymbol{\Lambda}$ by

$$\begin{pmatrix} \boldsymbol{\lambda}_1 & & & & & & \\ & \boldsymbol{\lambda}_2 & & & & & \\ & & \ddots & & & & \\ & & & \boldsymbol{\lambda}_k & & & \\ & & & & \boldsymbol{\theta} & & \\ & & & & & \ddots & \\ & & & & & & \boldsymbol{\theta} \end{pmatrix}$$

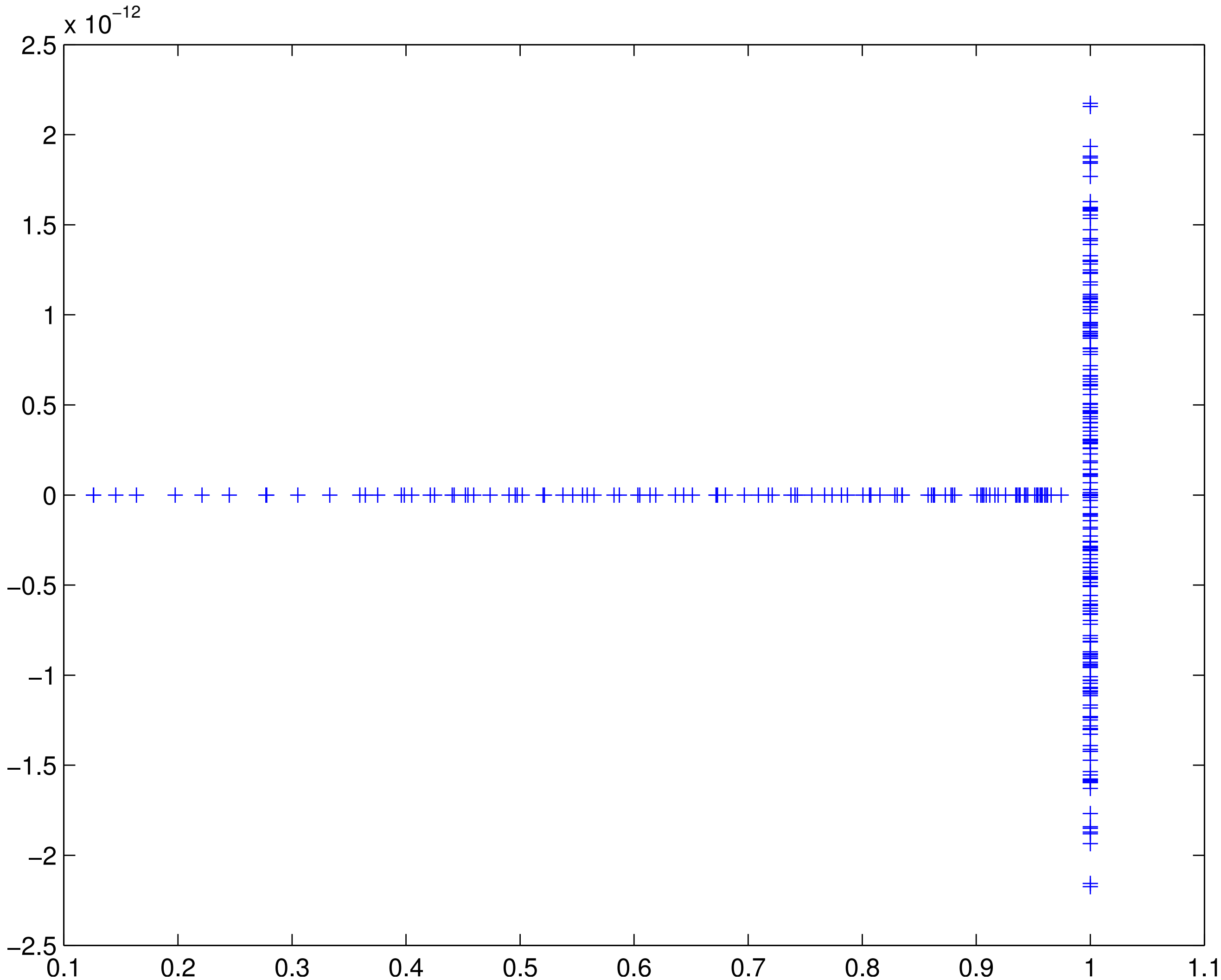
- Interesting case: $\boldsymbol{\theta} = \boldsymbol{\lambda}_{k+1}$
- Question: related approximation to $\boldsymbol{G}^{-1}$?

- Result: Let $\gamma = 1/(1-\theta)$. Then approx. to G^{-1} is:

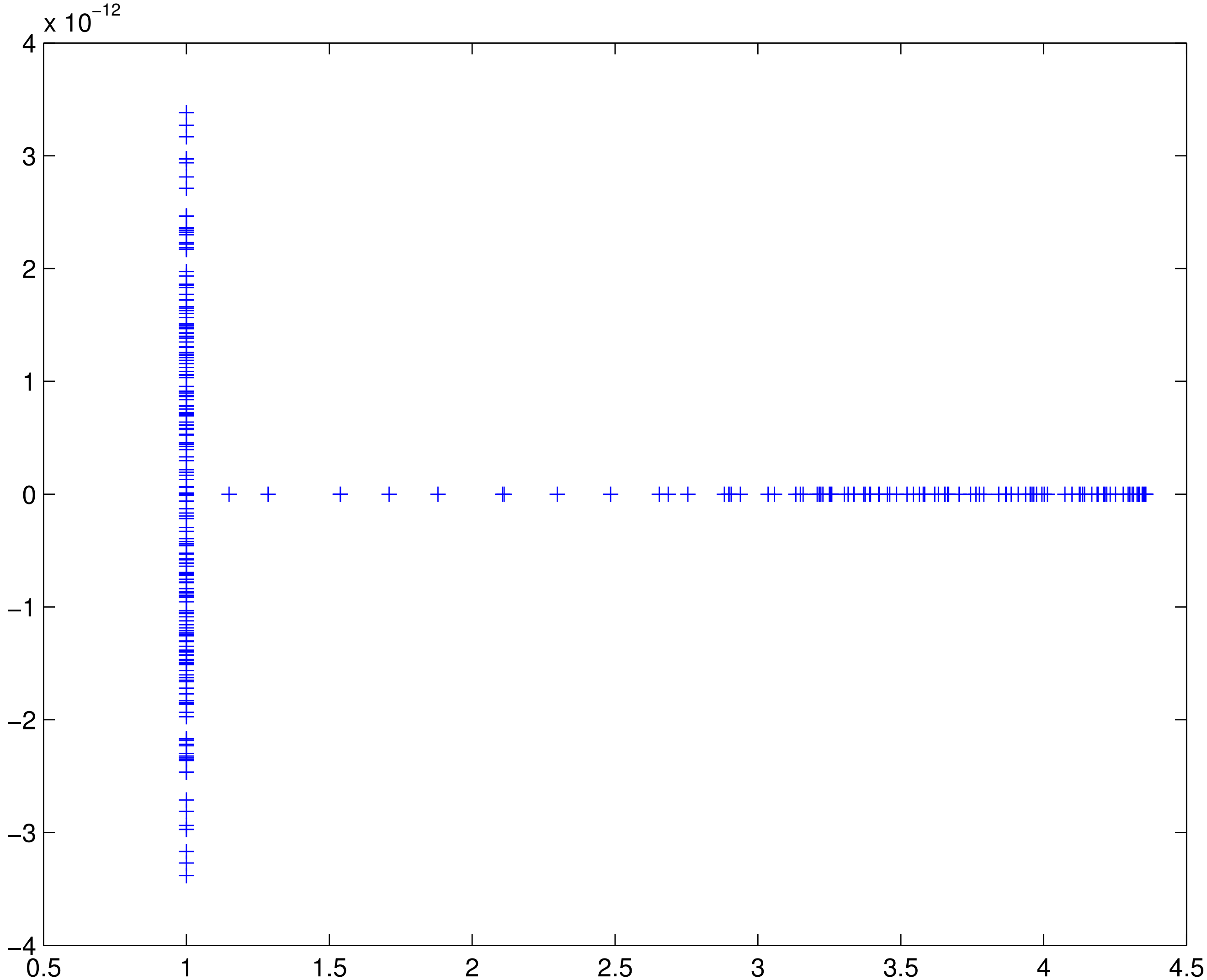
$$G_{k,\theta}^{-1} \equiv \gamma I + U_k[(I - D_k)^{-1} - \gamma I]U_k^T$$

- G_k: k largest eigenvalues of G matched – others set == θ
- $\theta = 0$ yields previous case
- When $\lambda_{k+1} \leq \theta < 1$ we get
- Result: AM^{-1} has
  - $n - s + k$ eigenvalues == 1
  - All others ≥ 1
- Next: An example for a 900×900 Laplacean, 4 domains, $s = 119$

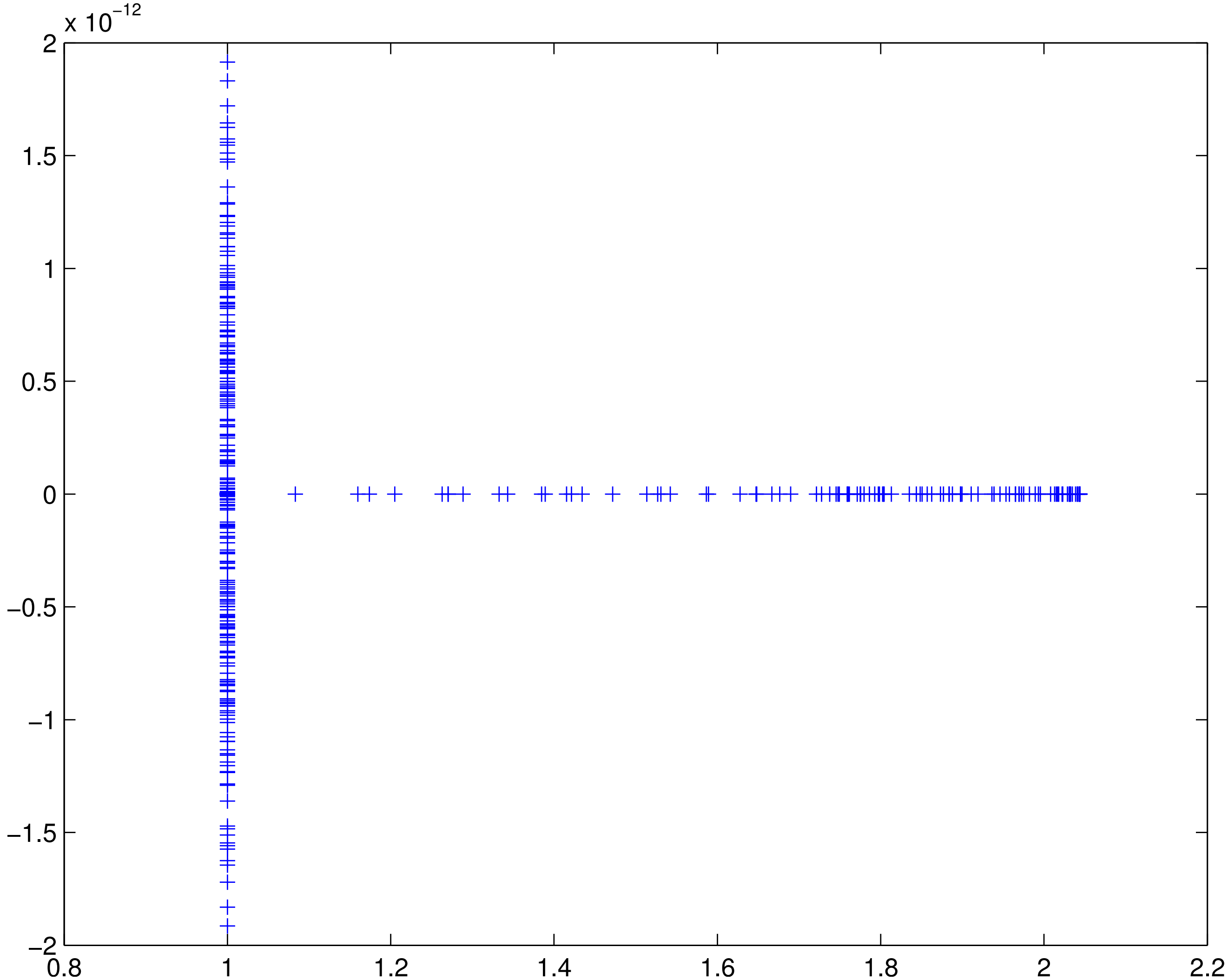
$k = 5$ Eigenvalues of AM^{-1} for the case $\theta = 0$



$k = 5$ Eigenvalues of AM^{-1} for the case $\theta = \lambda_{k+1}$



$k = 15$ Eigenvalues of AM^{-1} for the case $\theta = \lambda_{k+1}$



Proposition Assume θ is so that $\boxed{\lambda_{k+1} \le \theta < 1}$. Then the eigenvalues η_i of AM^{-1} satisfy:

$$1 \le \eta_i \le 1 + \frac{1}{1-\theta} \|A^{1/2} A_0^{-1} E\|_2^2 .$$

- Can Show: For the Laplacean (FD) and when $\alpha = 1$,

$$\|A^{1/2} A_0^{-1} E\|_2^2 = \|E^T A_0^{-1} A A_0^{-1} E\|_2 \le \frac{1}{4}$$

regardless of the mesh-size.

- Best upper bound for $\theta = \lambda_{k+1}$

- Set $\theta = \lambda_{k+1}$. Then $\kappa(AM^{-1}) \le$ constant, if k large enough so that $\lambda_{k+1} \le$ constant.

- i.e., need to capture sufficient part of spectrum

The symmetric indefinite case

- Appeal of this approach over ILU: approximate inverse $\rightarrow$ Not as sensitive to indefiniteness
- Part of the results shown still hold
- But $\lambda_i(H)$ can be > 1 now.
- Parameter α now plays a more important role

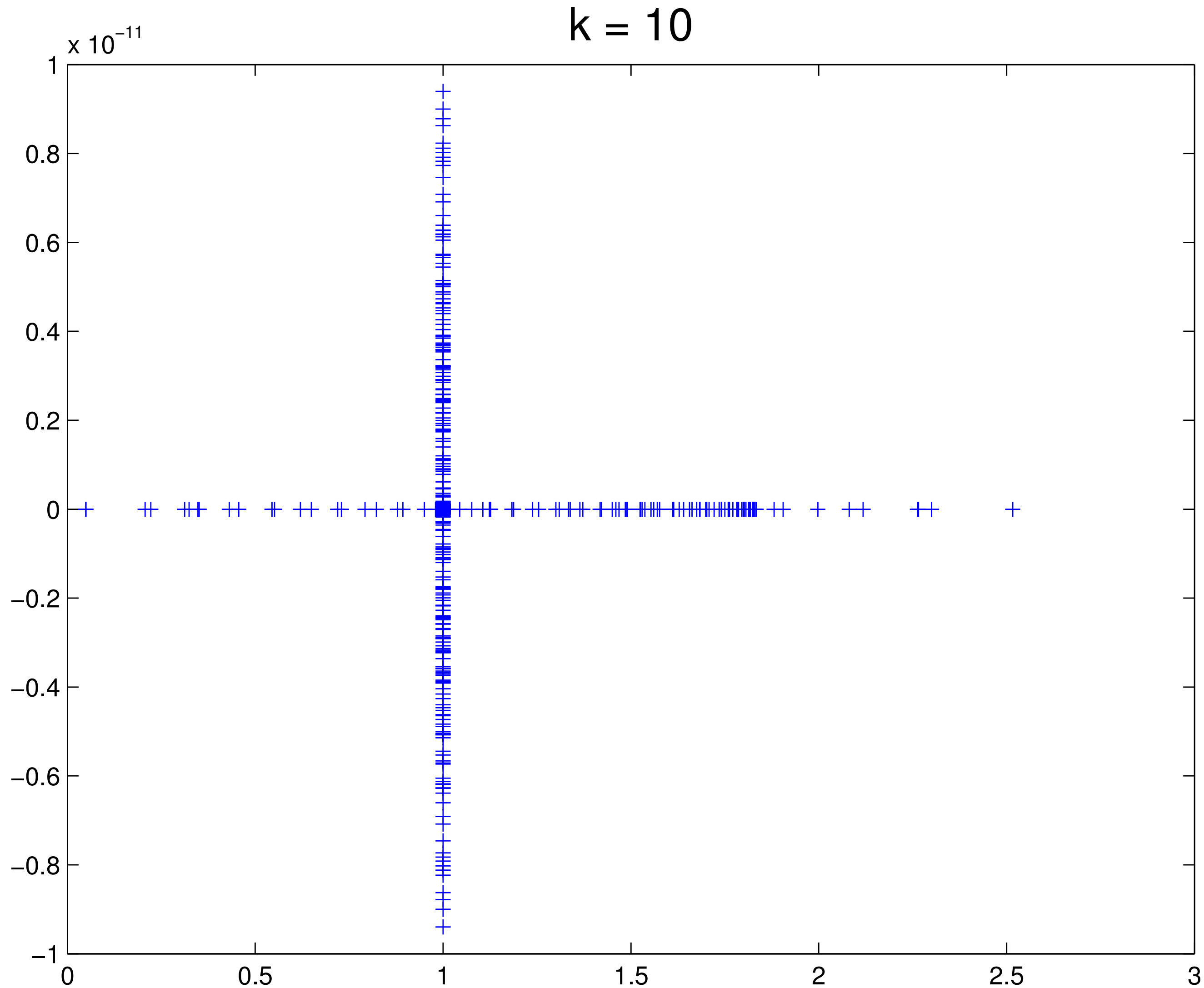
$\boxed{\textit{Example:}}$ Take Laplacean on a 30×30 FD grid.

- Subtract $0.4I$ – result: 26 negative eigenvalues

$$\lambda_{min} = -0.379477..., \qquad \lambda_{max} = 7.579477...$$

- Use $\alpha = 4.0$, $\theta = 0.9$;
- We do test for $k = 10$ and then $k = 5$

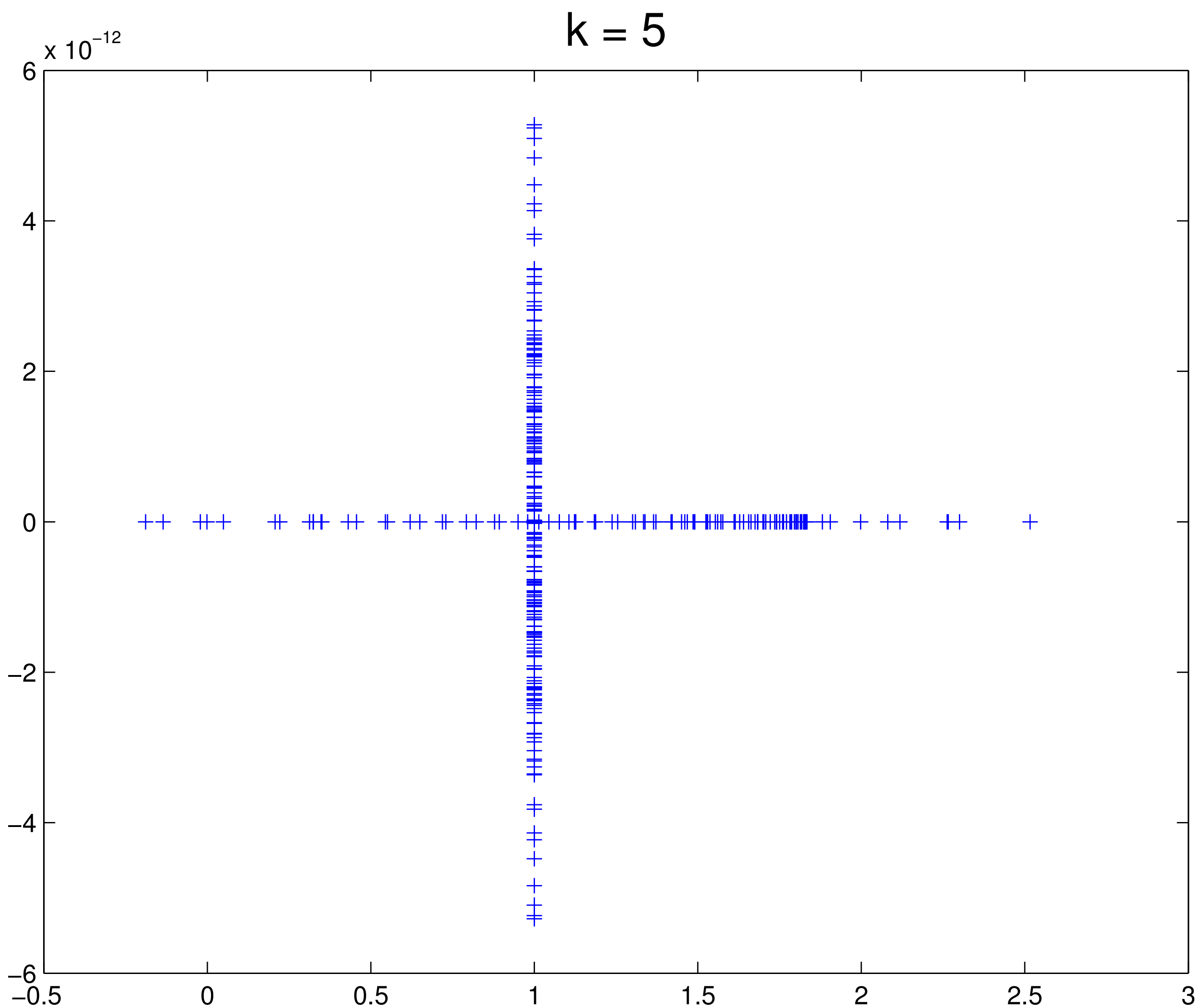
$k = 10$ Eigenvalues of AM^{-1} $[\theta = 0.90, \alpha = 4]$



➤ All eigen-values are > 0

$k = 5$ Eigenvalues of AM^{-1} $[\theta = 0.90, \alpha = 4]$

- 5 eigenvalues are < 0



Parallel implementations

➤ Recall :

$$\boxed{\begin{array}{c} M^{-1} = A_0^{-1}\left[I + EG_{k,\theta}^{-1}E^TA_0^{-1}\right] \\ G_{k,\theta}^{-1} = \gamma I + U_k[(I - D_k)^{-1} - \gamma I]U_k^T \end{array}}$$

➤ Steps involved in applying M^{-1} to a vector x :

ALGORITHM : 1 *Preconditioning operation*

1.	$z = A_0^{-1}x$	// $\hat{B}_i$-solves and $C_\alpha-$ solve
2.	$y = E^Tz$	// Interior points to interface (Loc.)
3.	$y_k = G_{k,\theta}^{-1}y$	// Use Low-Rank approx.
4.	$z_k = Ey_k$	// Interface to interior points (Loc.)
5.	$u = A_0^{-1}(x + z_k)$	// $\hat{B}_i$-solves and $C_\alpha-$ solve

A_0 Solves Note:

$$A_0 = \begin{pmatrix} \hat{B}_1 & & & & \\ & \hat{B}_2 & & & \\ & & \ddots & & \\ & & & \hat{B}_p & \\ & & & & C_\alpha \end{pmatrix}$$

- Recall $\hat{B}_i = B_i + \alpha^{-2} E_i E_i^T$
- A solve with A_0 amounts to all p $\hat{B}_i$-solves and a C_α-solve
- Can replace C_α^{-1} by a low degree polynomial [Chebyshev]
- Can use any solver for the $\hat{B}_i$'s

Parallel tests: Itasca (MSI)

- HP linux cluster- with Xeons 5560 (“Nehalem”) processors

2-D

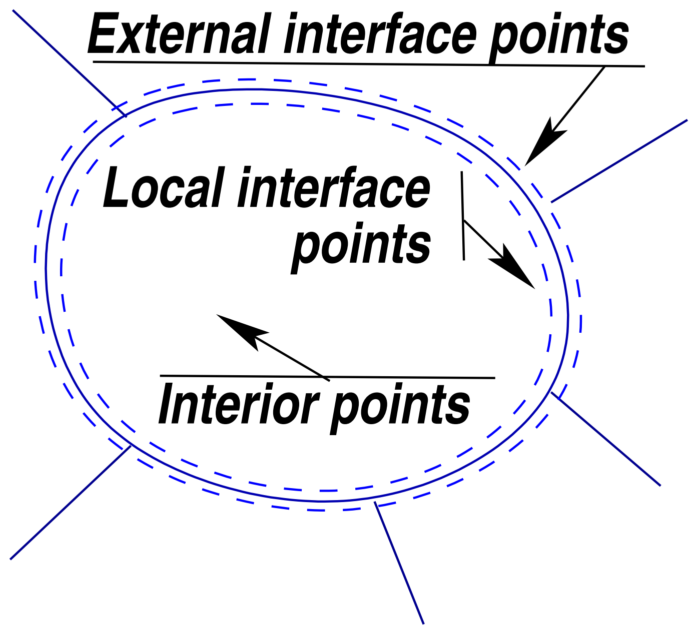
Mesh	Nproc	Rank	#its	Prec-t	Iter-t
256×256	2	8	29	2.30	.343
512×512	8	16	57	2.62	.747
1024×1024	32	32	96	3.30	1.32
2048×2048	128	64	154	4.84	2.38

3-D

Mesh	Nproc	Rank	#its	Prec-t	Iter-t
$32 \times 32 \times 32$	2	8	12	1.09	.0972
$64 \times 64 \times 64$	16	16	31	1.18	.381
$128 \times 128 \times 128$	128	32	62	2.42	.878

- Assume A is Symmetric Positive Definite (SPD)

Local view:

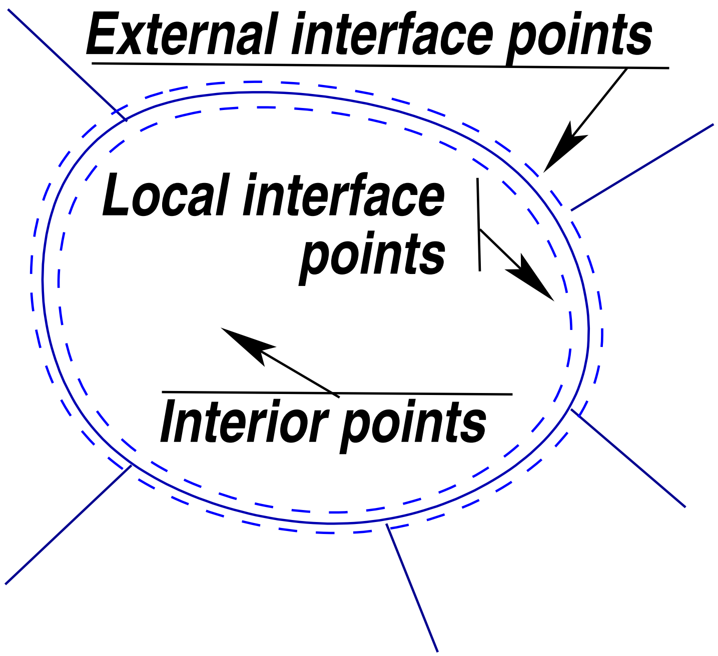


$$\begin{pmatrix} B_i & E_i \\ E_i^T & C_i \end{pmatrix} \begin{pmatrix} u_i \\ y_i \end{pmatrix} + \begin{pmatrix} 0 \\ \sum_{j \in N_i} E_{ij} y_j \end{pmatrix} = \begin{pmatrix} f_i \\ g_i \end{pmatrix}$$

The global system: Global view

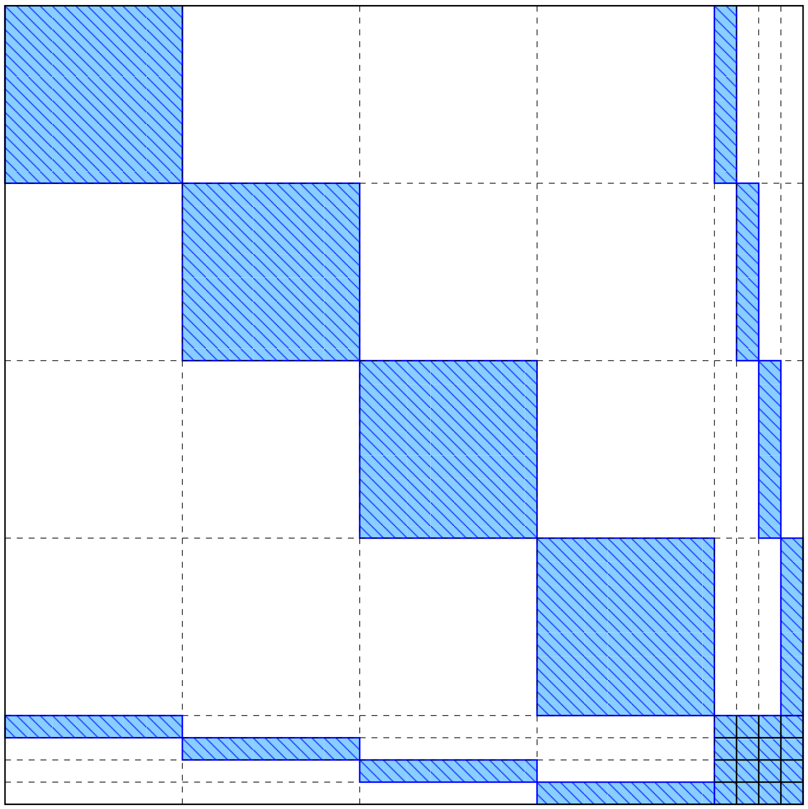
- Global system can be permuted to the form $\rightarrow$
- u_i's internal variables
- y interface variables

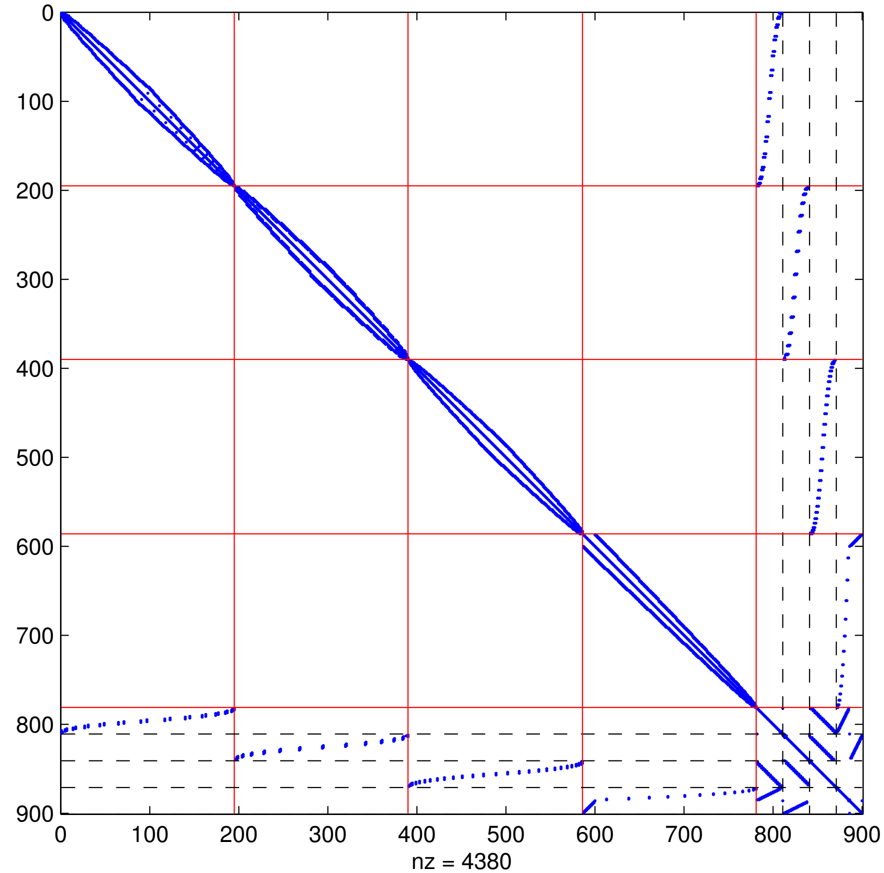
$$\begin{pmatrix} B_1 & & & \dots & E_1 \\ & B_2 & & \dots & E_2 \\ \vdots & & \ddots & & \vdots \\ & & & B_p & E_p \\ E_1^T & E_2^T & \dots & E_p^T & C \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \\ y \end{pmatrix} = b$$



- E_i maps local interface points to interior points in domain Ω_i
- E_i^T does the reverse operation

- Global matrix has the form $\begin{pmatrix} B & E \\ E^T & C \end{pmatrix}$





Schur Complement System

Background:

$$\begin{pmatrix} B & E \\ E^T & C \end{pmatrix} = \begin{pmatrix} I & \\ E^T B^{-1} & I \end{pmatrix} \begin{pmatrix} B & E \\ & S \end{pmatrix} \qquad S = C - E^T B^{-1} E$$

- $S \in \mathbb{R}^{s \times s}$ == 'Schur complement' matrix
- Solution obtained from two solves with B, one with S

Focus now: Solving Schur complement systems

- Assume C is SPD and let $C = LL^T$. Then:

$$S = L\left(I - L^{-1} E^T B^{-1} E L^{-T}\right) L^T \equiv L(I - H) L^T.$$

- Define: $H = L^{-1} E^T B^{-1} E L^{-T}$
- Note: $S^{-1} = L^{-T} (I - H)^{-1} L^{-1}$

Decay properties of $\mathrm{S}^{-1} - \mathrm{C}^{-1}$

- Spectral factorization of $H \in \mathbb{R}^{s \times s}$ $H = U \Lambda U^T$ where $U^T U = I$ and $\Lambda = \mathrm{diag}\,(\lambda_1, \ldots, \lambda_s)$
- Can show : $\lambda_j(H) \in [0, 1)$
- $S^{-1} = L^{-T}(I - H)^{-1} L^{-1}$. Can we write

$(I - H)^{-1} = I + L.R.C.\ ?$ [L.R.C : Low rank correction]

$$(I - H)^{-1} - I = L^T S^{-1} L - I = L^T (S^{-1} - C^{-1}) L \equiv X$$

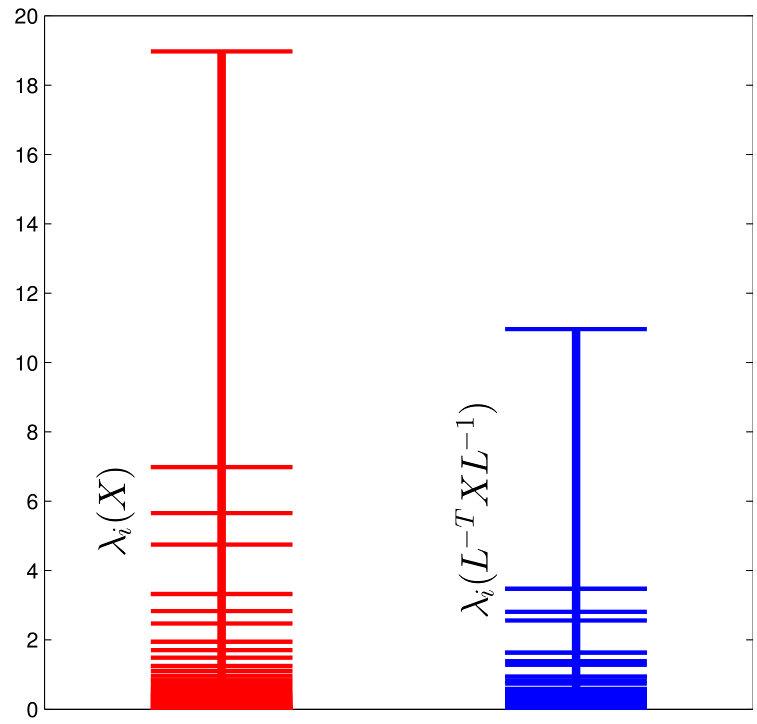
- Thus, $S^{-1} = C^{-1} + L^{-T} X L^{-1}$
- $\theta_k = \lambda(X)$, $\lambda_k = \lambda(H)$. Can show: $\boxed{\theta_k = \dfrac{\lambda_k}{1 - \lambda_k}}$
- $\dfrac{d\theta_k}{d\lambda_k} = \dfrac{1}{(1 - \lambda_k)^2} \Rightarrow \theta_k$ well separated when $\lambda_k \to 1$

Decay properties of $\mathrm{S}^{-1} - \mathrm{C}^{-1}$

- Example: 2-D Laplacian, $n_x = n_y = 32$, 4 subdomains
- Eigenvalues of X [the same as those of $\lambda(S^{-1}C - I)$], and $S^{-1} - C^{-1} = L^{-T} X L^{-1}$

5 eigenvectors:
82.5% of X, 85.1% of $L^{-T}XL^{-1}$

10 eigenvectors:
89.7% of X, 91.4% of $L^{-T}XL^{-1}$



Two-domain analysis for model problem

- $-\Delta$ on $\Omega = n_x \times (2n_y + 1)$ grid
- DD: $\Omega = (\Omega_1, \Omega_2, \Gamma)$, $\Omega_1, \Omega_2 : n_x \times n_y$, $\Gamma : n_x \times 1$
- $C, S \in \mathbb{R}^{n_x \times n_x}$.

Eigenvalues:

- γ_k of $S^{-1} - C^{-1}$,
- ζ_k of X

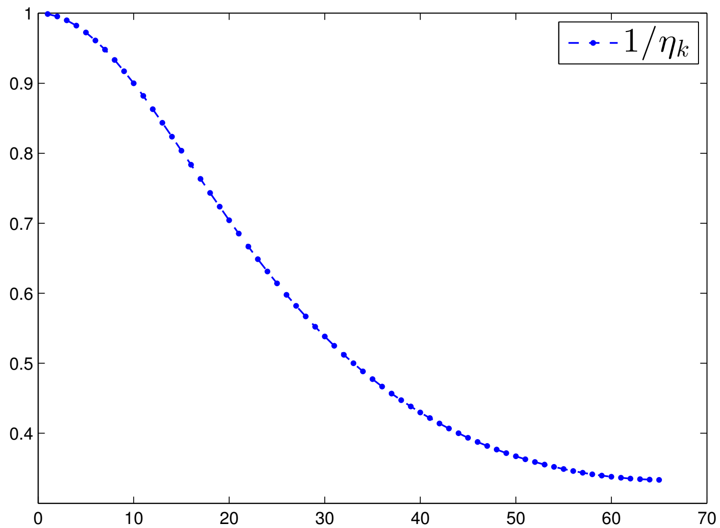
$(X \sim CS^{-1} - I)$

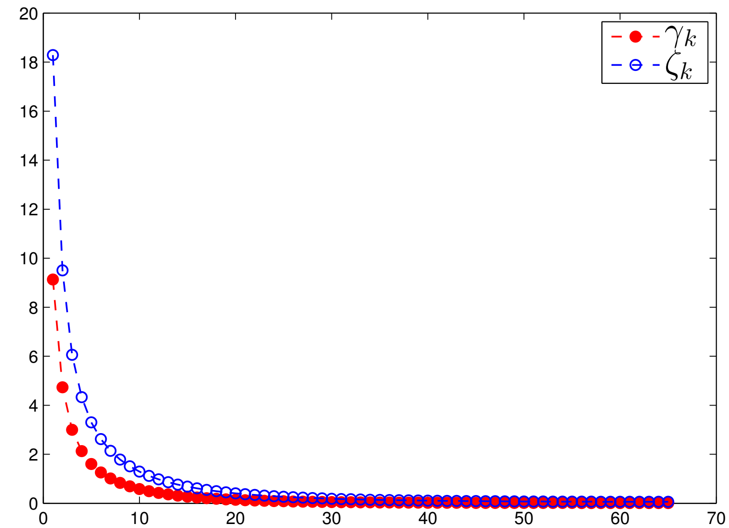
$$\gamma_k \approx \frac{1}{2}\left[\frac{1}{\sqrt{\eta_k^2 - 1}} - \frac{1}{\eta_k}\right]$$

$$\zeta_k = 2\eta_k\gamma_k \approx \frac{\eta_k}{\sqrt{\eta_k^2 - 1}} - 1$$

$$\eta_k = 1 + 2\sin^2\frac{k\pi}{2(n_x + 1)}$$

$n_x \times (2n_y + 1)$ grid, $n_x = 65$,$n_y = 32$, 2 subdomains





Eigenvalues γ_k of $S^{-1} - C^{-1}$ and ζ_k of $S^{-1}C - I$

Low-rank approximations to S^{-1}

- Preconditioner for A:

$$M = \begin{pmatrix} I & \\ E^T B^{-1} & I \end{pmatrix} \begin{pmatrix} B & E \\ & \tilde{S} \end{pmatrix}$$

- $(n-s)$ of $\lambda_i(AM^{-1}) = 1$, the other $s \to \lambda_i(S\tilde{S}^{-1})$
- Eigendecomposition $H = U\Lambda U^T$. Replace Λ with $\tilde{\Lambda}$
- Recall $S^{-1} = L^{-T}(I-H)^{-1}L^{-1}$, and rewrite

$$S^{-1} = L^{-T}U(I-\Lambda)^{-1}U^T L^{-1}$$
$$\tilde{S}^{-1} = L^{-T}U(I-\tilde{\Lambda})^{-1}U^T L^{-1}$$

- Can show: $\lambda(S\tilde{S}^{-1}) = \dfrac{1-\lambda_i}{1-\tilde{\lambda}_i}, \quad i = 1, \ldots, s$

- As before, let $\tilde{\Lambda}$ be

$$\tilde{\lambda}_i = \begin{cases} \lambda_i & i \leq k \\ \theta & i > k \end{cases}$$

- Then $\lambda(S\tilde{S}^{-1})$

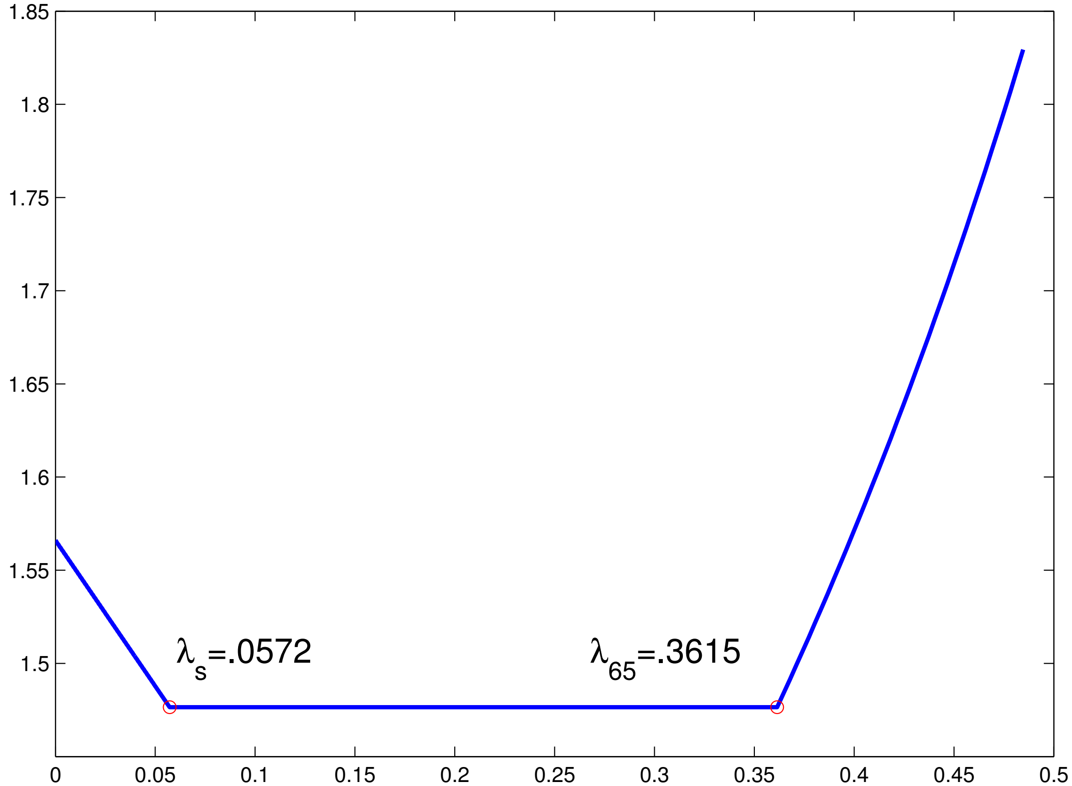
$$\begin{cases} 1 & i \leq k \\ (1-\lambda_i)/(1-\theta) & i > k \end{cases}$$

- The spectral condition number $\kappa(\theta)$ of $S\tilde{S}^{-1}$

$$\kappa(\theta) = \begin{cases} \dfrac{1-\theta}{1-\lambda_{k+1}} & \theta \in [0, \lambda_s) \\[2ex] \dfrac{1-\lambda_s}{1-\lambda_{k+1}} & \theta \in [\lambda_s, \lambda_{k+1}] \\[2ex] \dfrac{1-\lambda_s}{1-\theta} & \theta \in (\lambda_{k+1}, 1) \end{cases}$$

Minimum: $\boxed{\dfrac{1-\lambda_s}{1-\lambda_{k+1}}}$ for any $\theta \in [\lambda_s, \lambda_{k+1}]$

- How does $\boldsymbol{\kappa}(\boldsymbol{\theta})$ vary when $\boldsymbol{\theta}$ varies between $\mathbf{0}$ and $\mathbf{1}$?



$\boldsymbol{\kappa}(\boldsymbol{\theta})$ for 2-D Laplacian with $\boldsymbol{n_x = n_y = 256}$.
subdomains $== \mathbf{2}$; # eigenvectors used $== \mathbf{64}$.

Numerical Experiments

- Intel Xeon X5675 (12 MB Cache, 3.06 GHz, 6-core), Xeon X5560 (8 MB Cache, 2.8 GHz, 4-core) at MSI
- Written in C/C++, MKL; OpenMP parallelism
- Accelerators: CG, GMRES(40)
- Partitioning with METIS

Tests with SLR, SPD model problems

- Stopping: $\|r_i\|_2 \le 10^{-8}\|r_0\|_2$, max nits = 300
- Comparison with ILU and RAS (one-level overlap)

Grid	ICT-CG				RAS-GMRES				SLR-CG					
	fill	p-t	its	i-t	fill	p-t	its	i-t	nd	rk	fill	p-t	its	i-t
256^2	4.5	.07	51	.239	4.5	.09	129	.281	32	16	4.3	.09	67	.145
512^2	4.6	.30	97	1.93	4.8	.36	259	2.34	64	32	4.9	.65	103	1.01
1024^2	5.4	1.4	149	14.2	6.2	1.9	F	-	128	32	5.7	5.2	175	7.95
40^3	4.4	.13	25	.152	4.5	.15	36	.101	32	16	4.0	.18	31	.104
64^3	6.8	.98	32	1.24	6.2	.91	49	.622	64	32	6.3	1.5	38	.633
100^3	7.3	4.1	47	7.52	6.1	3.5	82	4.29	128	32	6.5	5.5	67	4.48

SLR, indefinite model problems

- $-\Delta$ shifted by $-sI$. 2D: $s = 0.01$, 3D: $s = 0.05$

Grid	ILDLT-GMRES				RAS-GMRES				SLR-GMRES					
	fill	p-t	its	i-t	fill	p-t	its	i-t	nd	rk	fill	p-t	its	i-t
256^2	8.2	.17	F	-	6.3	.13	F	-	8	32	6.4	.21	33	.125
512^2	8.4	.70	F	-	8.4	.72	F	-	16	64	7.6	2.1	93	1.50
1024^2	13	5.1	F	-	19	22	F	-	8	128	11	25	50	4.81
40^3	6.9	.25	54	.54	6.7	.25	99	.30	64	32	6.7	.49	23	.123
64^3	9.0	1.4	F	-	11.8	2.2	F	-	128	64	9.1	3.9	45	1.16
100^3	15	11	F	-	12	15	F	-	128	180	15	63	88	13.9

SLR, general matrices

Matrix	ICT/ILDLT				SLR					
	fill	p-t	its	i-t	nd	rk	fill	p-t	its	i-t
cant	4.7	3.8	150	9.34	32	90	4.9	5.5	82	1.92
cfd1	6.9	2.9	295	11.9	32	32	6.9	2.1	64	1.07
therm2	6.9	5.1	178	39.3	64	90	6.6	14	184	15.0
tmtsym	6.0	1.9	122	11.6	64	80	5.9	6.6	127	5.23
ecology	8.4	2.6	142	18.5	32	96	8.0	12	90	5.58
Lin	11	1.9	F	-	64	64	9.9	3.7	73	1.75
vibbox	6.0	.74	F	-	4	64	3.8	.43	226	.619
qa8fk	4.2	.79	22	.51	16	64	4.5	1.9	28	.309
F2	5.1	9.6	F	-	8	80	3.9	6.2	72	2.14
helm2d	14	14	F	-	16	128	11	12	63	2.63

Also tested: RAS – failed on all probems but one (qa8fk)

Conclusion

➤ Promising alternatives to ILUs can be found in new forms of approximate inverse techniques

➤ Seek "data-sparsity" instead of regular sparsity

➤ DD approch easier to implement, easier to understand than recursive approach

Advantages of Multilevel Low-Rank preconditioners:

➤ Approximate inverses $\rightarrow$ less sensitive to indefiniteness

➤ Exploit dense computations

➤ Easy to update