UNIVERSITY OF MINNESOTA TWIN CITIES

# Low-rank correction preconditioning techniques

*Yousef Saad*

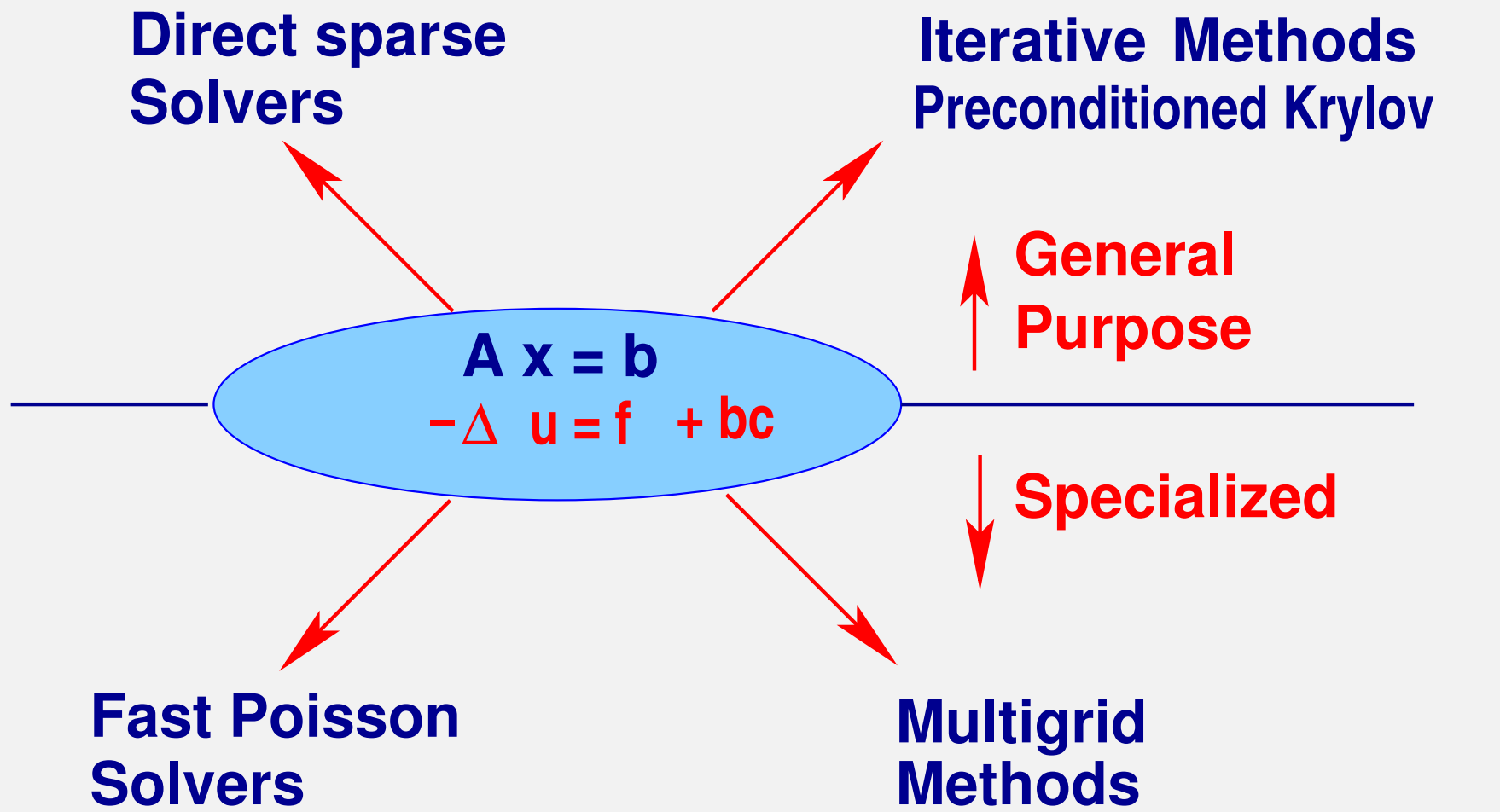
*Department of Computer Science and Engineering*

*University of Minnesota*

*SMAI 2017*

*Ronce Les Bains, France*
*June 6, 2017*

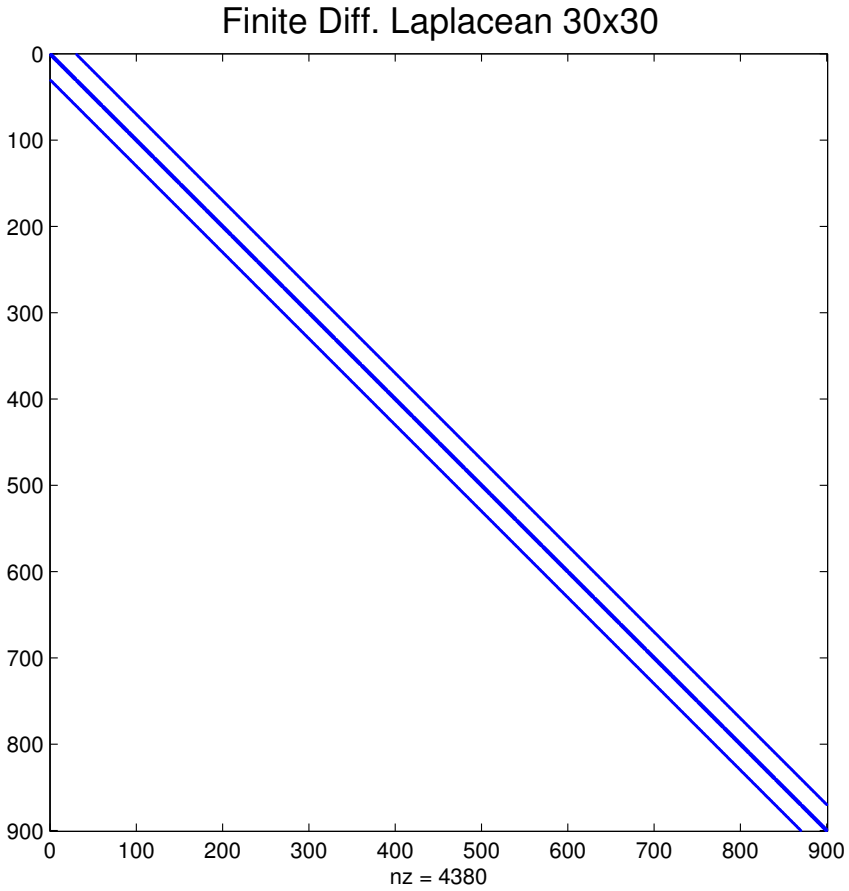
# *Introduction: Linear System Solvers*

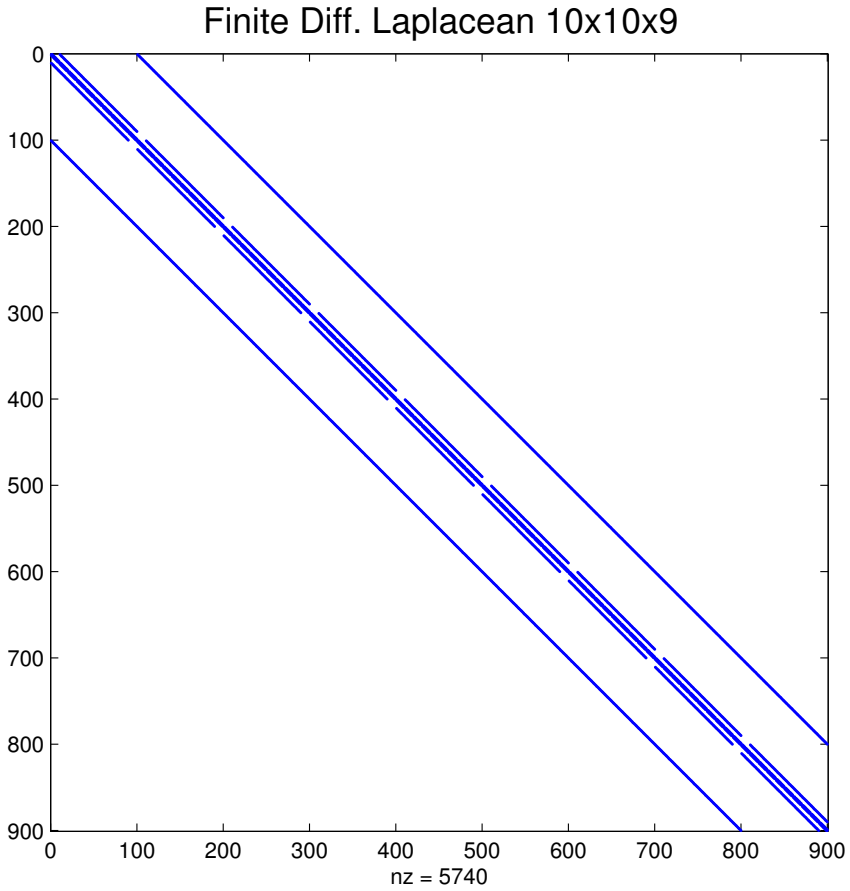


## *Long standing debate: direct vs. iterative*

- Starting in the 1970's: huge progress of sparse direct solvers
- Iterative methods – Initially 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 $350 \times 350$ grid (2D); Second: on a $50 \times 50 \times 49$ grid (3D)

# Pattern of a similar [much smaller] coefficient matrix



## *A few observations*

- Problems are getting harder for Sparse Direct methods
  (more 3-D models, much bigger problems,..)
- Problems are also getting difficult for iterative methods:
  More complex models - away from Poisson
- Researchers on both camps are learning each other's tricks to develop preconditioners.
- Much of recent work on solvers has focussed on:

(1) Parallel implementation – scalable performance

(2) Robustness, more general preconditioners

## *Background: Preconditioned iterative solvers*

To solve: $$Ax = b$$

- $A$ is a general sparse matrix
- Solve by: Preconditioned Krylov subspace methods

*Two ingredients:*

- A preconditioner: makes the system easier to solve by accelator, e.g. Incomplete LU factorizations;

$$Ax = b \quad \rightarrow \quad M^{-1}Ax = M^{-1}b$$

- An accelerator: Conjugate gradient, BiCG, GMRES, BICGSTAB,.. ['Krylov subspace methods']

# *Preconditioned CG (PCG); A: Symmetric Positive Definite*

## ALGORITHM : 1. *Preconditioned CG*

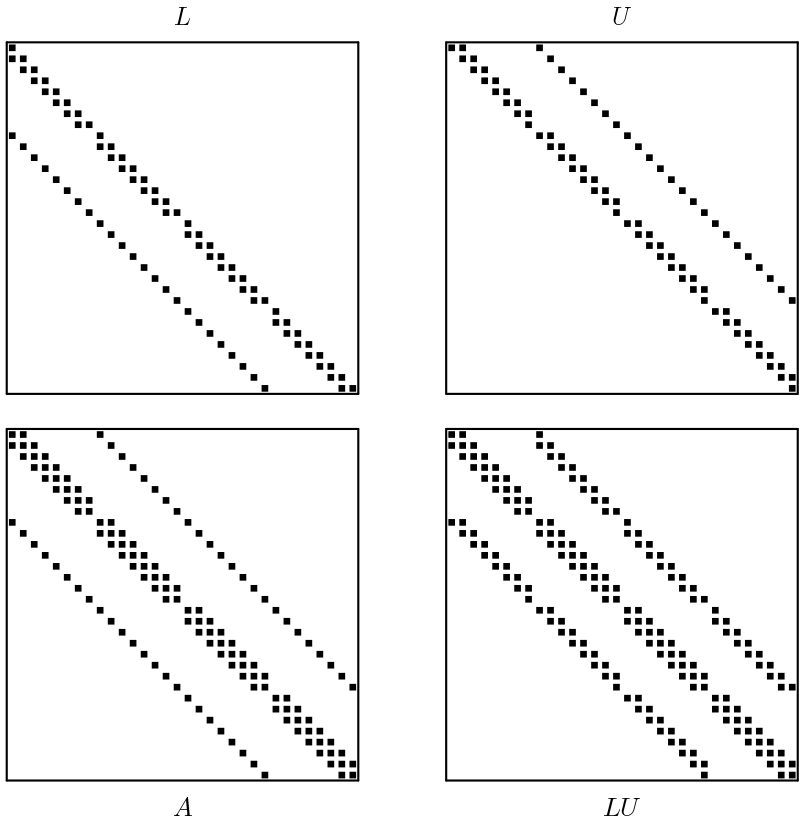
1. *Compute* $r_0 := b - Ax_0$, $z_0 = M^{-1}r_0$, *and* $p_0 := z_0$
2. *For* $j = 0, 1, \ldots$, *until convergence Do:*
3. $\alpha_j := (r_j, z_j)/(Ap_j, p_j)$
4. $x_{j+1} := x_j + \alpha_j p_j$
5. $r_{j+1} := r_j - \alpha_j \boxed{Ap_j}$
6. $z_{j+1} := \boxed{M^{-1} r_{j+1}}$
7. $\beta_j := (r_{j+1}, z_{j+1})/(r_j, z_j)$
8. $p_{j+1} := z_{j+1} + \beta_j p_j$
9. *EndDo*

- $M$ = preconditioning matrix, e.g., $M = LU$ from ILU

# *Background: Incomplete LU (ILU) preconditioners*

*ILU:* $A \approx LU$

Simplest Example: ILU(0) $\rightarrow$




*Common difficulties of ILUs:*

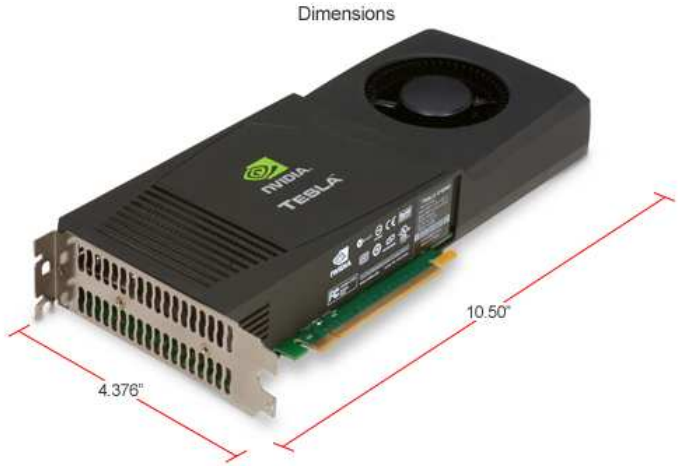
Often fail for indefinite problems

Not too good for highly parallel environments

## *Sparse matrix computations with GPUs* **

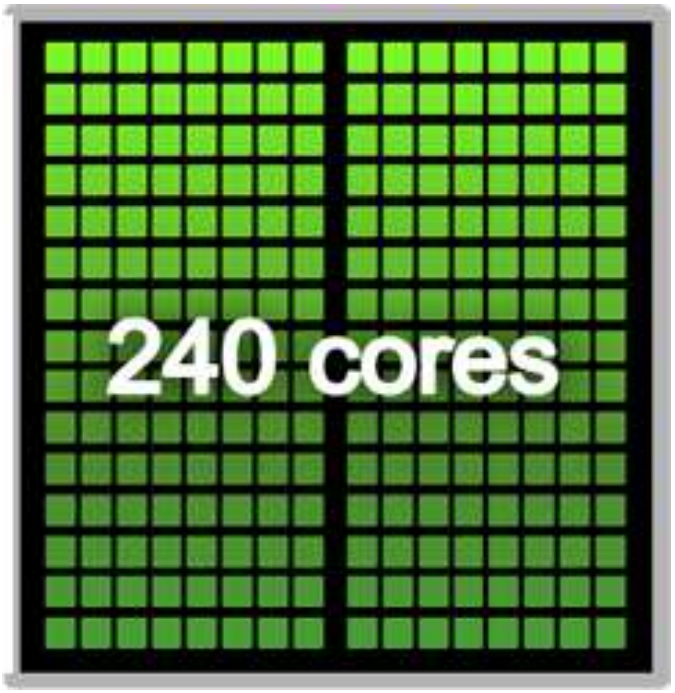
- GPUs Currently a very popular approach to: inexpensive supercomputing
- Can buy $\sim$ one Teraflop peak power for around a little more tham $1,000

*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 → GPU

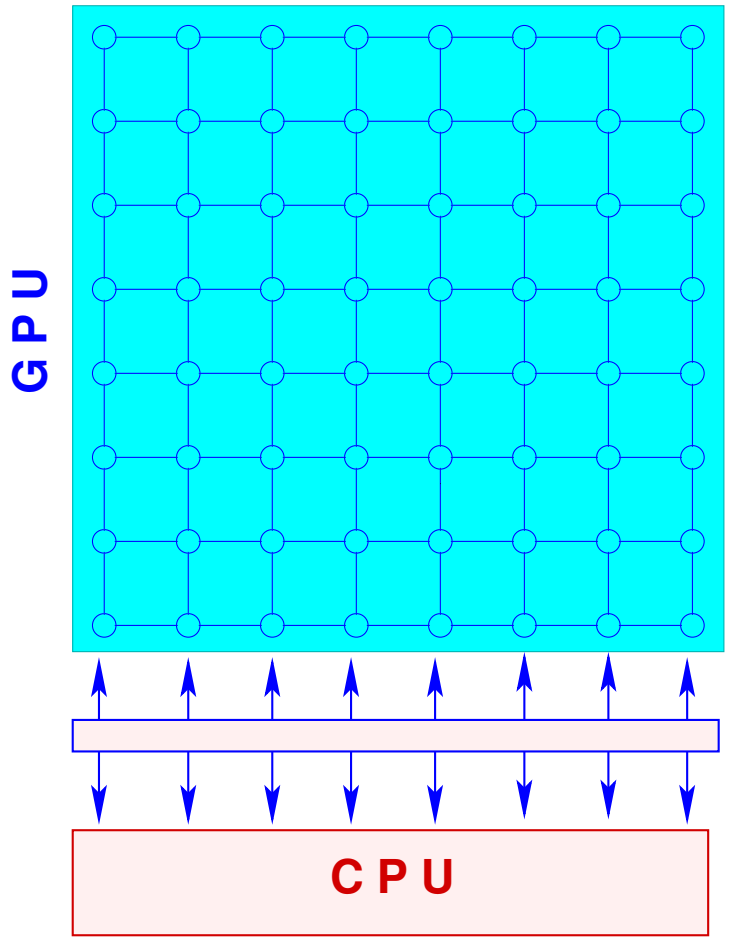
```
cudaMemcpy(...)
```

4. Execute GPU 'kernel':

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

5. Copy data GPU →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 | GPU-Lev | |
|---|---|---|---|---|
| | | Mflops | #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...

... prepare for the demise of the GPUs...

... or the demise of the ILUs ?

## *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

## *1. Low-rank Multilevel Approximations: divide & conquer*

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

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

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

➤ In the simplest case:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & \\ & A_{22} \end{pmatrix} + \begin{array}{|c|c|} \hline & -I \\ \hline -I & \\ \hline \end{array}$$

➤ Write 2nd matrix as:

$$\begin{array}{|c|c|} \hline & -I \\ \hline -I & \\ \hline \end{array} = \begin{array}{|c|c|} \hline +I & \\ \hline & +I \\ \hline \end{array} - \begin{array}{|c|c|} \hline I & I \\ \hline I & I \\ \hline \end{array}$$

$$\mathbf{E}\,\mathbf{E}^{\mathbf{T}}$$

$$\mathbf{E}^{\mathbf{T}} = \begin{array}{|c|c|} \hline I & I \\ \hline \end{array}$$

➤ 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: $B_1 := A_{11} + E_1 E_1^T, \quad B_2 := A_{22} + E_2 E_2^T.$

➤ 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 $B^{-1}E$

$$B^{-1}E \approx U_k V_k^T, \quad U_k \in \mathbb{R}^{n \times k}, \quad V_k \in \mathbb{R}^{n_x \times k},$$

- 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

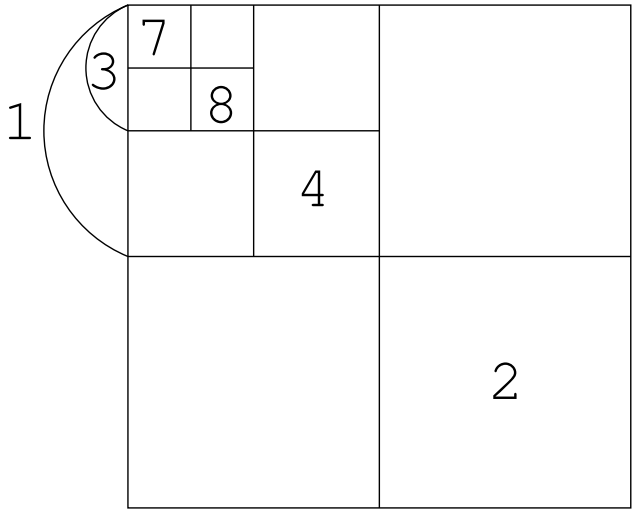
- We can show :

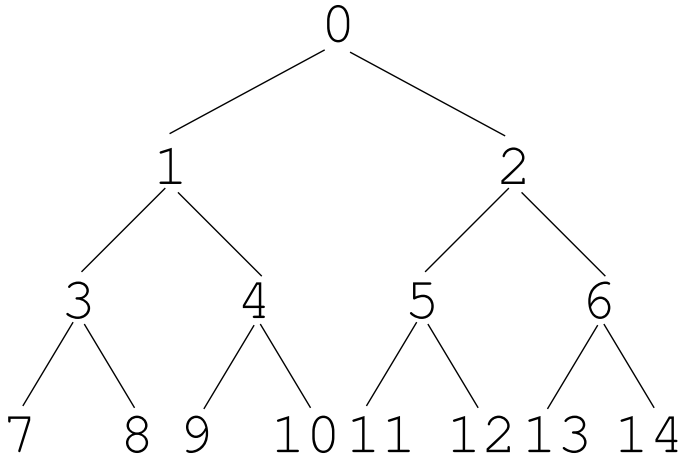
$$H_k = (I - U_k^T E V_k)^{-1}$$ and

$$H_k^T = H_k$$

# *Recursive multilevel framework*

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



## *Theory: 2-level analysis for model problem*

- Interested in eigenvalues $\gamma_j$ of

$$A^{-1} - B^{-1} = B^{-1} E X^{-1} E^T B^{-1}$$

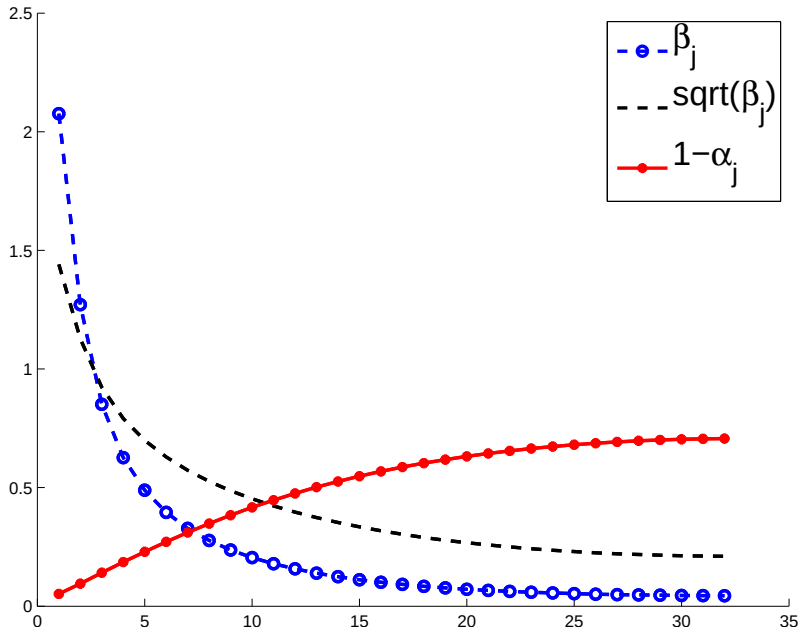
when $A$ = Pure Laplacean .. They are:

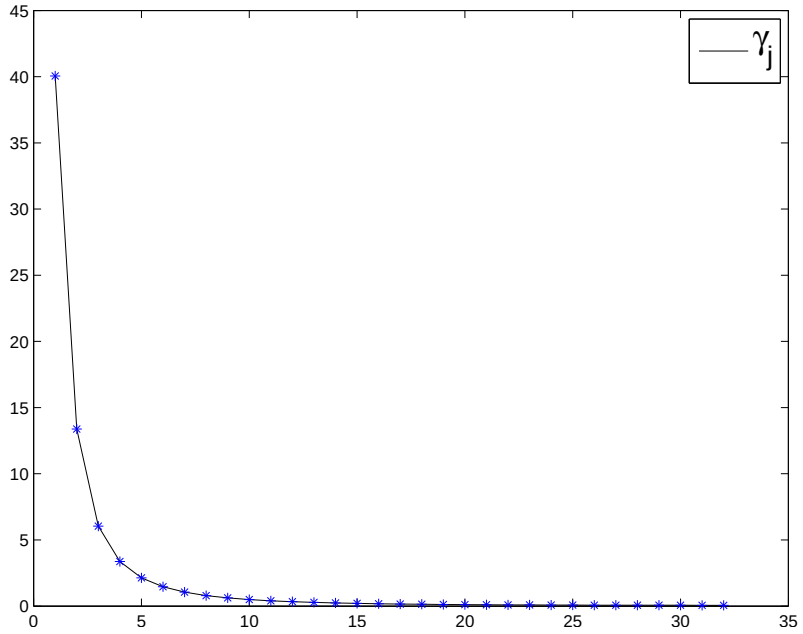
$$\gamma_j = \frac{\beta_j}{1-\alpha_j}, \quad j = 1, \cdots, n_x \quad \text{with:}$$

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

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

- Decay of the $\gamma_j$'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.

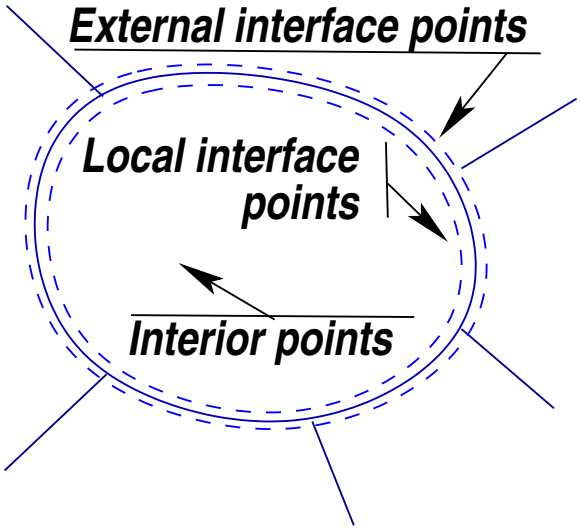
## 2. Avoiding recursivity: 'standard' DD framework

- Domain partitioned in $p$ sub-domains [edge-separation]. Interface nodes in each subdomain listed last.

➤ Global system can be permuted to the form →

➤ $u_i$'s internal variables

➤ $y$ interface variables

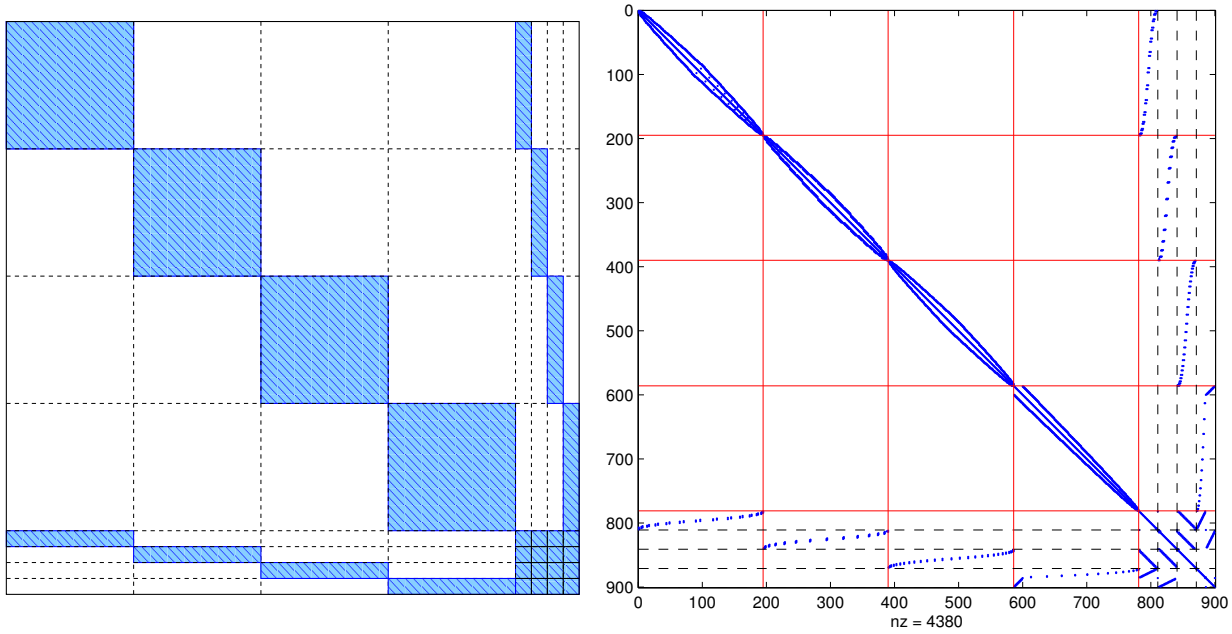



$$\begin{pmatrix} B_1 & & & \hat{F}_1 \\ & B_2 & & \hat{F}_2 \\ & & \ddots & \vdots \\ & & B_p & \hat{F}_p \\ \hat{E}_1^T & \hat{E}_2^T & \dots & C \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \\ y \end{pmatrix} = b$$

➤ $\hat{F}_i$ maps local interface points to interior points in domain $\Omega_i$

➤ $\hat{E}_i^T$ does the reverse operation

*Example:*




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

## *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:

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

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

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

## *Low-Rank Approximation DD preconditioners*

Sherman-Morrison $\longrightarrow$

$$A^{-1} = A_0^{-1} + A_0^{-1} F G^{-1} E^T A_0^{-1}$$
$$G \equiv I - E^T A_0^{-1} F$$

*Options:*

(a) Approximate $A_0^{-1}F, E^T A_0^{-1}, G^{-1}$ [as before]

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

➤ (b) requires 2 solves with $A_0$.

Let $G \approx G_k$

Preconditioner $\longrightarrow$

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

## *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: $0 \leq \lambda_j(H) < 1$ .

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

$$\begin{pmatrix} \lambda_1 & & & & & & \\ & \lambda_2 & & & & & \\ & & \ddots & & & & \\ & & & \lambda_k & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{pmatrix}$$

- Alternative: replace $\Lambda$ by

$$\begin{pmatrix} \lambda_1 & & & & & & \\ & \lambda_2 & & & & & \\ & & \ddots & & & & \\ & & & \lambda_k & & & \\ & & & & \theta & & \\ & & & & & \ddots & \\ & & & & & & \theta \end{pmatrix}$$

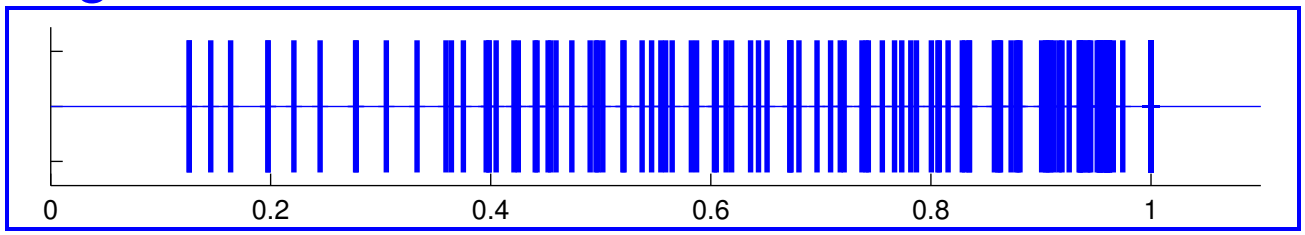
- Interesting case: $\theta = \lambda_{k+1}$
- Question: related approximation to $G^{-1}$?

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

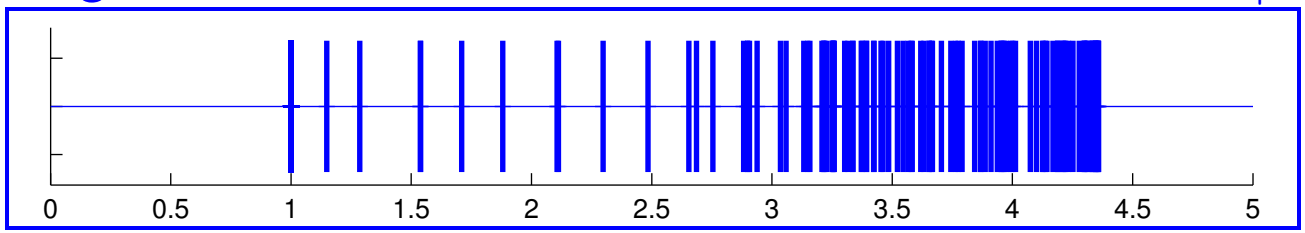
$$G_{k,\theta}^{-1} \equiv \gamma I + U_k[(I - D_k)^{-1} - \gamma I]U_k^T$$

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

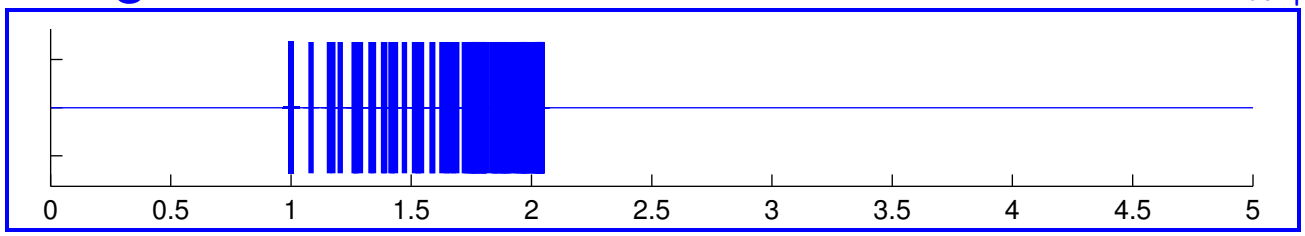
$k = 5$ Eigenvalues of $AM^{-1}$ for the case $\theta = 0$




$k = 5$ 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 $\theta$ is so that $\boxed{\lambda_{k+1} \leq \theta < 1}$. Then the eigenvalues $\eta_i$ of $AM^{-1}$ satisfy:

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

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

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

regardless of the mesh-size.

- Best upper bound for $\theta = \lambda_{k+1}$
- Set $\theta = \lambda_{k+1}$. Then $\kappa(AM^{-1}) \leq$ constant, if $k$ large enough so that $\lambda_{k+1} \leq$ constant.
- i.e., need to capture sufficient part of spectrum

## *Parallel implementations*

- Recall :

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

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

ALGORITHM : 2 *Preconditioning operation*

1. $z = A_0^{-1}x$ // $\hat{B}_i$-solves and $C_\alpha-$ solve
2. $y = E^T z$ // 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 $C_\alpha$-solve
- Can replace $C_\alpha^{-1}$ 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 |
|---|---|---|---|---|---|
| $256 \times 256$ | 2 | 8 | 29 | 2.30 | .343 |
| $512 \times 512$ | 8 | 16 | 57 | 2.62 | .747 |
| $1024 \times 1024$ | 32 | 32 | 96 | 3.30 | 1.32 |
| $2048 \times 2048$ | 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 |

## *3. 'Non-standard' DD framework: HID ordering*

➤ Issue: Schur complement can become large (3D Pbs)

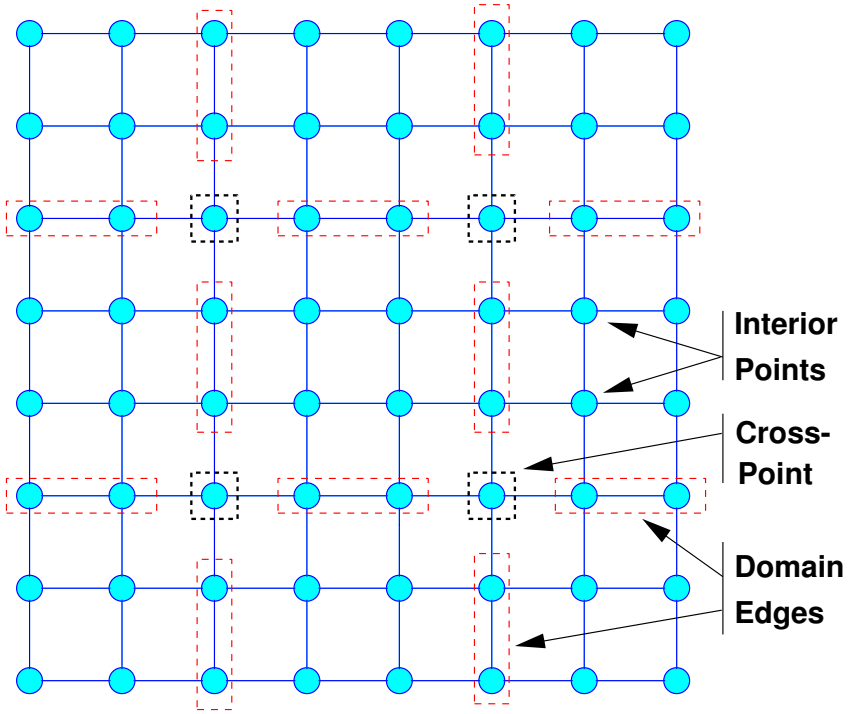
➤ Remedy: Use Hierarchical Interface Decomposition (HID) - Henon and YS'05

*Goal:* Define a method that descends into interface variables in a hierarchical way $\rightarrow$ need a hierarchy of 'interfaces'.

➤ Ideas of this type in the Domain Decomposition context (PDEs) by. Smith and Widlund (89) – ["Wirebasket" techniques]

# *The hierarchical decomposition of a graph*

➤ Partition $\mathcal{G}$ into $p$ subgraphs - with overlapping.




Example: $A$ originates from a 5-point FD discretization of a Laplacean on a 2-D domain.

➤ Three types of nodes: interior, interface, and cross-points.

➤ Extension expressed in terms of "connectors" of different levels.

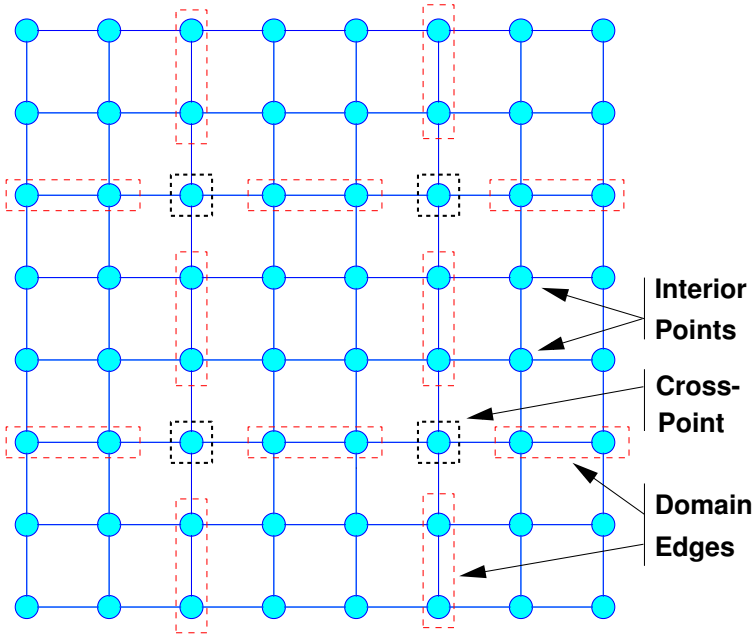
## *The hierarchical decomposition of a graph*

➤ Vertex node divided into 'connectors' (sets of vertices) grouped into $L$ levels

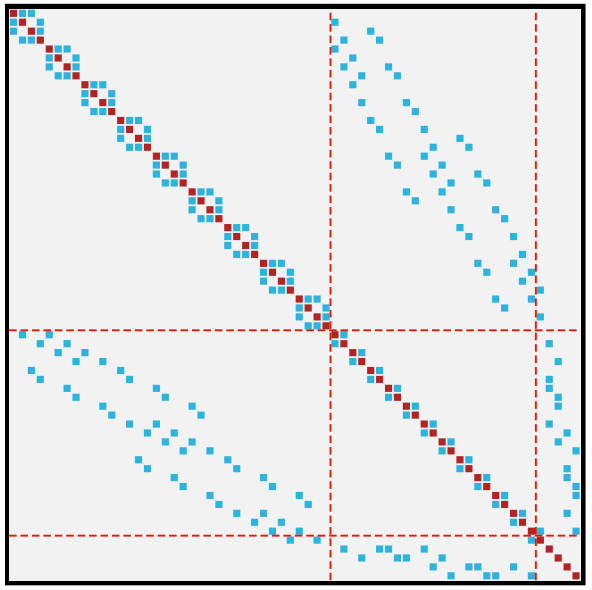
This set of connectors is a hierarchical decomposition of $\mathcal{G}$, if:

1. The connectors are disjoint and form a partition of the graph,
2. Connectors of the same level are not adjacent,
3. Connectors of a level $l > 1$ are separators for connectors at level $l - 1$.

# *The hierarchical decomposition of a graph - example*




Graph

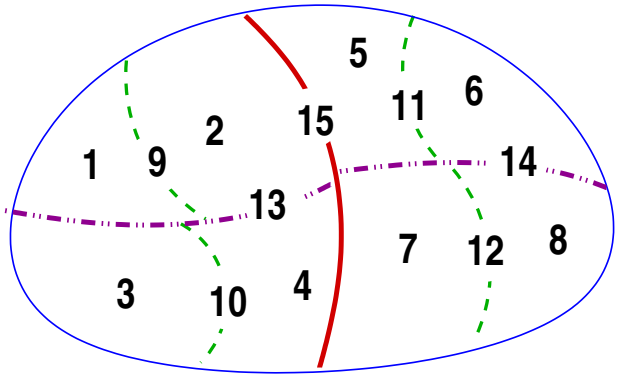


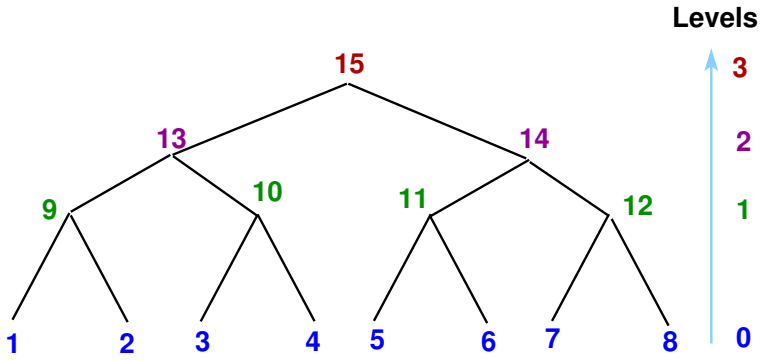
Matrix pattern

➤ $C^1$ = interior of each subdomain; $C^2$ = sets of edges; $C^3$ = none ; $C^4$ = cross-points

➤ Label by levels $\longrightarrow$ block-diagonal structure at each level

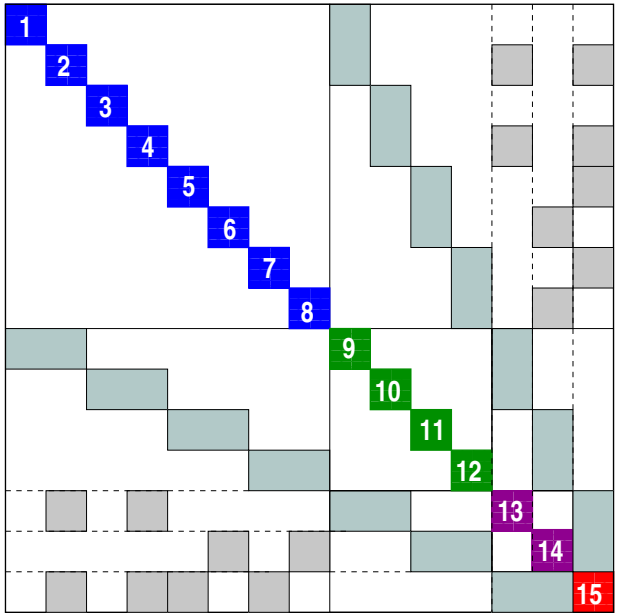
## Easy way to get an HID: Nested Dissection ordering




Up: 3-level partition of a 2-D domain. An HID tree with connector level information.

Right: Non-zero pattern of the re-ordered matrix.



## *Recursive preconditioner*

$$A_l = \begin{pmatrix} B_l & E_l \\ E_l^T & C_l \end{pmatrix} \quad \text{and} \quad C_l = A_{l+1} \quad \text{for} \quad l = 0 : L-1,$$

$A_0$ == reordered matrix of $A$ from the HID ordering

$A_l$ == matrix $C_{l-1}$ for $l = 1, 2, \cdots, L$

$A_L$ == submatrix associated with the top-level connector.

- Each leading block $B_l$ in $A_l$ has a block-diagonal structure

*Goal:* Explore multilevel strategies to approximate the factorization of $A_l$

➤ Recall factorization:

$$A_l = \begin{pmatrix} I & \\ E_l^T B_l^{-1} & I \end{pmatrix} \begin{pmatrix} B_l & \\ & S_l \end{pmatrix} \begin{pmatrix} I & B_l^{-1} E_l \\ & I \end{pmatrix}$$

$$S_l = C_l - E_l^T B_l^{-1} E_l$$

*Main Observation:* $S_l^{-1} - C_l^{-1}$ nearly small rank

➤ Rank bounded by number of cross-points (connectors at level $l$ that intersect with connectors of higher levels)..

*Idea:* Write

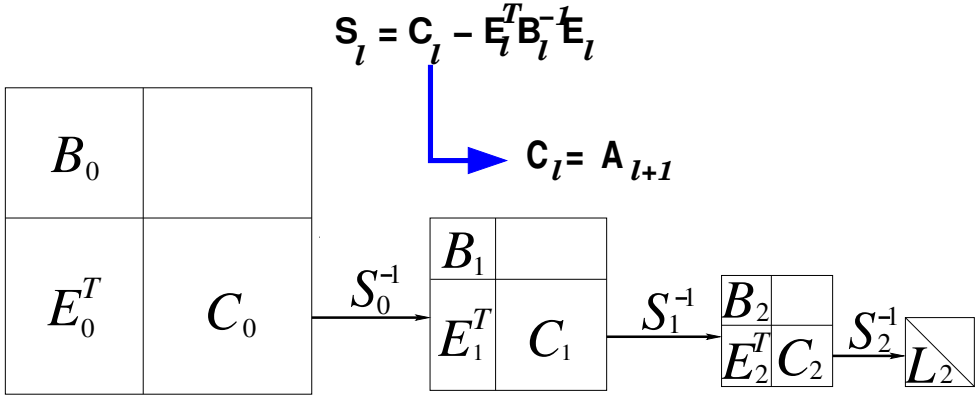
$$A_l^{-1} = \begin{pmatrix} I & -B_l^{-1}E_l \\ & I \end{pmatrix} \begin{pmatrix} B_l^{-1} & \\ & S_l^{-1} \end{pmatrix} \begin{pmatrix} I & \\ -E_l^T B_l^{-1} & I \end{pmatrix}.$$

- Approximate $S_l^{-1}$ as $S_l^{-1} \approx C_l^{-1} - W_l H_l W_l^T$
- Next: set $C_l = A_{l+1}$ $\rightarrow$ exploit recursivity
- Last level: use (incomplete) Cholesky.
- Next: illustration for 3 levels.

- At levels $l = 0, 1, 2$ express $A_l^{-1}$ as :

$$A_l^{-1} = \begin{pmatrix} I & -B_l^{-1}E_l \\ & I \end{pmatrix} \begin{pmatrix} B_l^{-1} & \\ & S_l^{-1} \end{pmatrix} \begin{pmatrix} I & \\ -E_l^T B_l^{-1} & I \end{pmatrix}.$$

- $S_l^{-1}$ needed $\rightarrow$ Approximate as $S_l^{-1} \approx C_l^{-1} + W_l H_l W_l^T$
- $C_l^{-1}$ needed $\rightarrow$ if $l == 2$ get $C_2 \approx L_2 L_2^T$,
  else set $A_{l+1} = C_l$ & go to next level



## *Computing the low-rank correction*

- Let $C = LL^T$ and define

$$\boxed{G = L^{-1}(C - S)L^{-T} = L^{-1}\left(E^T B^{-1} E\right) L}$$

We have $S = L(I - G)L^T \rightarrow$

$$\begin{aligned} S^{-1} - C^{-1} &= L^{-T}\left[(I - G)^{-1} - I\right] L^{-1} \\ &= L^{-T}\left[G(I - G)^{-1}\right] L^{-1}. \end{aligned}$$

- Use Lanczos algorithm to get a few of the largest eigenvalues of $G$ with associated eigenvectors:

$$[W_l, \Sigma_l] = \texttt{eigs}(C_l^{-1} E_l^T B_l^{-1} E_l, k) \rightarrow$$

$$\boxed{S_l^{-1} - C_l^{-1} \approx W_l H_l W_l^T, \quad \text{with} \quad H_l = \Sigma_l (I - \Sigma_l)^{-1}.}$$

- Need to solve with $C_l \rightarrow$ exploit recursivity

## *Test: Shifted Laplacean (Symm. indefinite)*

MSLR vs. ILDLT & RAS for symmetric indefinite systems (shifted Laplacean) (accelerator: GMRES)

| Grid | ILDLT-GMRES | | | | RAS-GMRES | | | | MSLR-GMRES | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | fill | p-t | its | i-t | fill | p-t | its | i-t | lev | rk | fill | p-t | its | i-t |
| $256^2$ | 8.18 | 0.18 | F | – | 7.56 | 0.26 | F | – | 4 | 64 | 6.58 | 0.29 | 20 | 0.07 |
| $512^2$ | 8.39 | 0.71 | F | – | 7.84 | 1.19 | F | – | 5 | 80 | 7.68 | 3.17 | 36 | 0.60 |
| $1024^2$ | 12.6 | 5.34 | F | – | 19.40 | 22.90 | F | – | 6 | 180 | 9.13 | 41.09 | 76 | 6.30 |
| $32^3$ | 5.89 | 0.11 | 21 | 0.11 | 5.78 | 0.09 | 40 | 0.06 | 7 | 32 | 5.60 | 0.25 | 17 | 0.04 |
| $64^3$ | 7.05 | 1.03 | F | – | 11.40 | 3.01 | F | – | 10 | 64 | 7.06 | 7.44 | 187 | 3.97 |
| $128^3$ | 9.35 | 12.20 | F | – | 10.2 | 31.20 | F | – | 13 | 64 | 8.07 | 80.20 | F | – |

## *Test: General sparse matrices*

| Matrix | order | nnz | SPD | Origin |
|---|---|---|---|---|
| cfd1 | $70,656$ | $1,825,580$ | yes | CFD Prb. |
| cfd2 | $123,440$ | $3,085,406$ | yes | CFD Prb. |
| Dubcova3 | $146,689$ | $3,636,643$ | yes | 2-D/3-D PDE Prb. |
| thermal1 | $82,654$ | $574,458$ | yes | thermal Prb. |
| thermal2 | $1,228,045$ | $8,580,313$ | yes | thermal Prb. |
| F2 | $71,505$ | $5,294,285$ | no | structural Prb. |
| Lin | $256,000$ | $1,766,400$ | no | structural Prb. |
| qa8fk | $66,127$ | $1,660,579$ | no | 3-D acoustics Prb. |
| vibrobox | $12,328$ | $301,700$ | no | vibroacoustic Prb. |

# MSLR vs. ICT / ILDLT, RAS for general symmetric linear systems [Accelerators: CG or GMRES(40)]

| Matrix | ICT/ILDLT | | | | RAS | | | | MSLR | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | fill | p-t | its | i-t | fill | p-t | its | i-t | lev | rk | fill | p-t | its | i-t |
| cfd1 | 5.81 | 3.40 | 298 | 11.7 | 5.61 | 1.83 | F | – | 7 | 64 | 5.00 | 1.42 | 85 | 0.96 |
| cfd2 | 4.47 | 3.20 | 271 | 13.60 | 4.52 | 2.65 | F | – | 8 | 50 | 4.47 | 1.53 | 155 | 2.58 |
| Dubcova3 | 1.19 | 0.47 | 45 | 0.93 | 1.18 | 1.17 | 54 | 0.45 | 5 | 64 | 1.17 | 0.39 | 20 | 0.18 |
| thermal1 | 4.57 | 0.20 | 52 | 0.50 | 4.55 | 0.38 | 235 | 0.77 | 6 | 50 | 4.50 | 0.18 | 38 | 0.14 |
| thermal2 | 5.98 | 4.21 | 90 | 17.80 | 5.87 | 8.11 | F | – | 8 | 64 | 6.02 | 3.29 | 87 | 6.02 |
| F2 | 2.82 | 3.43 | F | – | 2.90 | 3.46 | F | – | 6 | 64 | 2.49 | 1.24 | 79 | 1.51 |
| Lin | 4.61 | 0.53 | F | – | 5.36 | 4.63 | F | – | 10 | 22 | 4.55 | 0.88 | 185 | 2.96 |
| qa8fk | 1.90 | 0.26 | 17 | 0.20 | 4.56 | 1.16 | 25 | 0.20 | 7 | 32 | 1.8 | 0.44 | 21 | 0.14 |
| vibrobox | 4.13 | 0.43 | F | – | 4.35 | 0.36 | F | – | 5 | 16 | 3.86 | 0.18 | 57 | 0.16 |

## *Recent work: extension to nonsymmetric case*

- Use Arnoldi instead of Lanczos. Code implemented in C. Also: complex version
- Details skipped. Two examples shown.

First: set of SPD test matrices (SuiteSparse Matrix Collection)

| Matrix | Order | nnz | SPD | Origin |
|---|---|---|---|---|
| `cfd1` | 70,656 | 1,825,580 | yes | CFD pb. |
| `cfd2` | 123,440 | 3,085,406 | yes | CFD pb. |
| `Dubcova3` | 146,689 | 3,636,643 | yes | 2D/3D pb. |
| `Offshore` | 259,789, | 4,242,673 | yes | electromagnetics pb. |
| `2cubes` | 101,492 | 1,647,264 | yes | electromagnetics pb. |
| `boneS01` | 127,224 | 5,516,602 | yes | model reduction |

GMSLR and UMFPACK. OOM == Out Of Memory.

| Matrix | GMSLR | | | | | | UMFPACK | | |
|---|---|---|---|---|---|---|---|---|---|
| | fill | lev | rk | p-t | i-t | its | fact. time | solve time | fill |
| cfd1 | 2.22 | 9 | 16 | .306 | 1.81 | 8 | 10.11 | 0.21 | 41.16 |
| cfd2 | 2.26 | 10 | 16 | .572 | 2.46 | 5 | 32.25 | 0.41 | 48.55 |
| Dubcova3 | 1.48 | 10 | 32 | .487 | .327 | 3 | 0.69 | 0.1 | 3.95 |
| Offshore | 2.9 | 10 | 16 | .775 | .82 | 5 | OOM | OOM | OOM |
| 2cubes | 2.03 | 9 | 8 | .183 | .083 | 3 | 69.27 | 0.47 | 107.6 |
| boneS01 | 1.21 | 9 | 16 | .549 | 3.26 | 6 | 32.99 | 0.38 | 22.03 |

2nd: Set of nonsymmetric matrices (SuiteSparse Collection.)

| Matrix | Order | nnz | SPD | Origin |
|---|---|---|---|---|
| CoupCons | 416,800 | 22,322,336 | no | structural pb. |
| AtmosModd | 1,270,432 | 8,814,880 | no | atmospheric model |
| Transport | 1,602,111 | 23,500,731 | no | CFD pb. |

Comparison between GMSLR and ILUT preconditioners. Drop tolerance for ILUT == $10^{-3}$.

| Matrix | GMSLR | | | | | | ILUT | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | fill | nlev | rank | p-t | i-t | its | fill | p-t | i-t | its |
| CoupCons | 1.82 | 10 | 16 | 2.67 | 1.55 | 5 | 1.56 | 17.6 | 1.25 | 12 |
| AtmosModd | 7.69 | 9 | 8 | 3.07 | 10.0 | 12 | 11.2 | 5.01 | 12.0 | 44 |
| Transport | 2.88 | 12 | 16 | 6.43 | 32.5 | 12 | 4.66 | 11.73 | 43.5 | 119 |

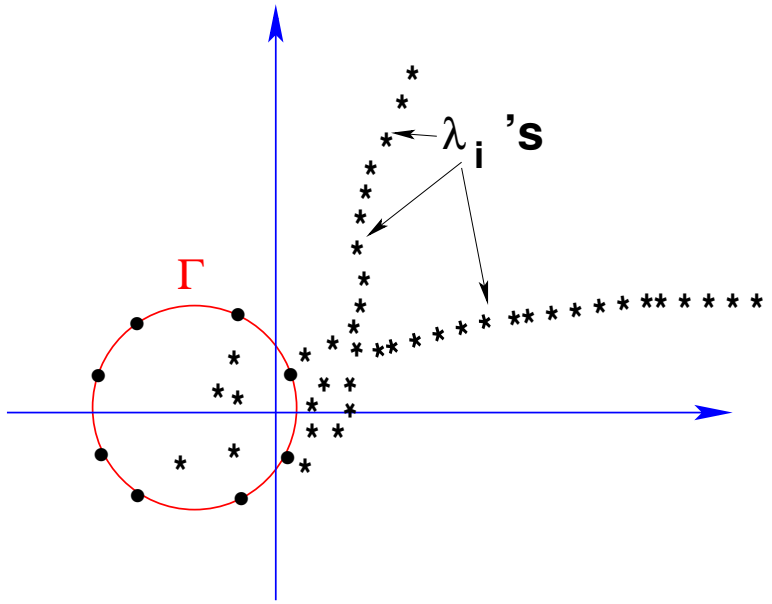
# 5. Cauchy integral approximation of $A^{-1}$

Define a preconditioner as follows:

*Step 1:* approximate $A^{-1}$ *outside* circle.

*Step 2:* Resolve the modes inside circle by a simple Krylov method.

Let $P$ = spectral projector associated with $\lambda_i$'s inside $\Gamma$




*Recall:*

$$Pf(A) = \frac{1}{2i\pi}\int_{\Gamma}(sI - A)^{-1} f(s)ds$$

➤ Computes $f(A)$ for spectrum inside $\Gamma$ when $f$ analytic.

*Idea:* Approximate $f(A)$ when $f(z) = 1/z$ *outside circle* by change of variables. Use numerical integration:

$$(I - P)A^{-1} \approx \frac{1}{2}\sum_{k=1}^{2p} \alpha_k (I - \sigma_k A)^{-1} \quad \rightarrow$$

$$A^{-1} = (I - P)A^{-1} + PA^{-1} \approx \sum_{k=1}^{2p} \xi_k U_k^{-1} L_k^{-1} + PA^{-1}$$

*Q:* How to compute term $PA^{-1}b$ for a given $b$ ?

- Can get approximate eigenspace associated with $P$ & solve in this subspace.
- OK – but may not be practical if subspace is large...
- Can just use a Krylov method to solve $APy = Pb$ ...
- Many details skipped

## *Test: 3D Helmholtz problem*

➤ Helmholtz equation of the following form on domain $(0,1)^3$

$$\left(-\Delta - \frac{\omega^2}{c(x)^2}\right) u = s$$

$\Delta \equiv$ Laplacian
$\omega \equiv$ angular frequency,
$c(x) \equiv$ seismic velocity field

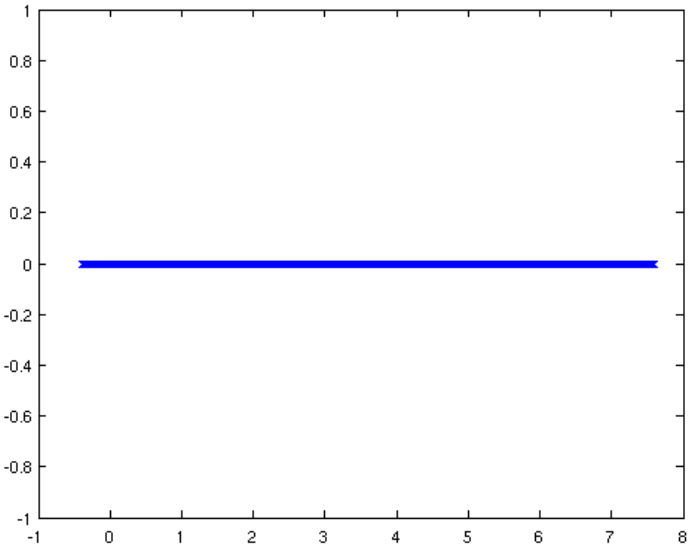
➤ $s(x,\omega) =$ forcing term - here generated by a Gaussian point source centered at $(1/2, 1/2, 1/2)$.

➤ $u(x,\omega) =$ time-harmonic wavefield solution

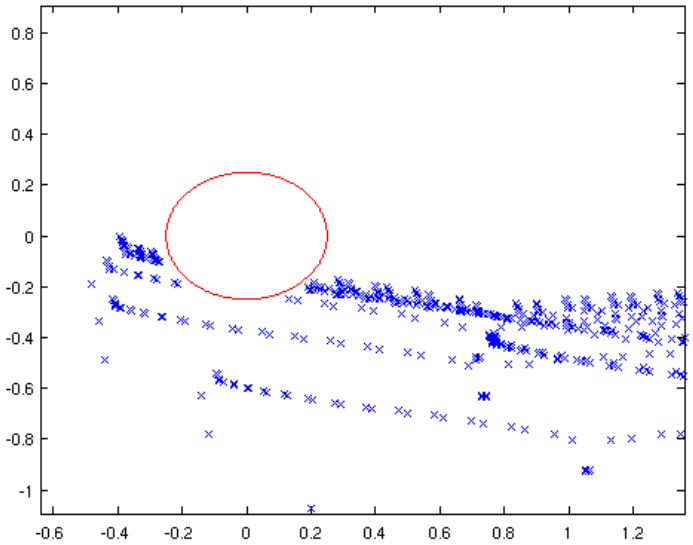
➤ Zero Dirichlet boundary condition applied to one side, PML conditions on others

➤ Discretization: $8$ points per wavelength when discretizing (7-point finite diff. on $N \times N \times N$ grids.

- PML has a significant impact on spectrum:




All Dirichlet




PML on 3 sides

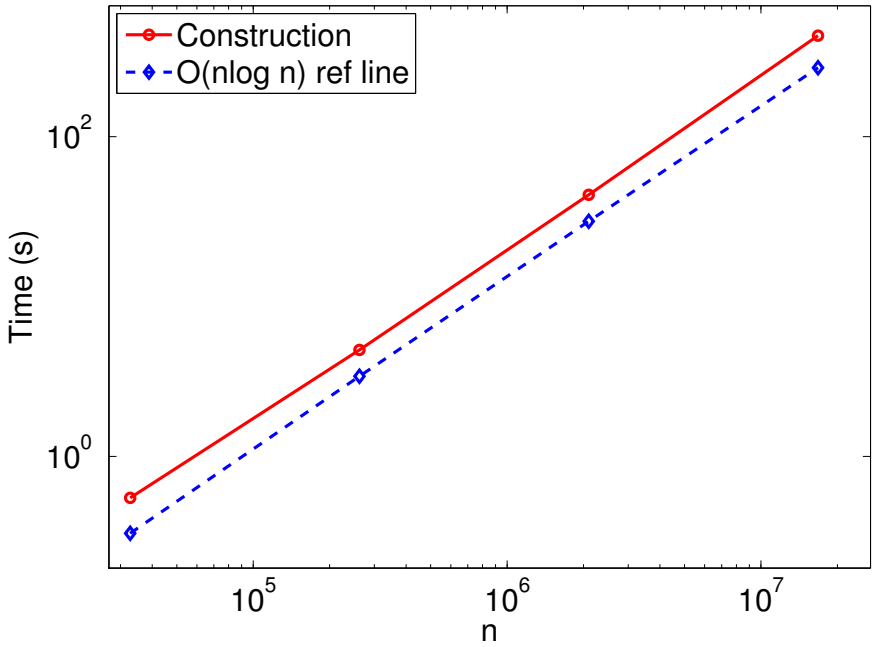
- Complex non-Hermitian matrices
- System solved to low accuracy (factor of $10^3$ res. reduction)

- Solve for $N = 2^5, 2^6, 2^7, 2^8$ on a standard workstation
- Matrices reordered by Nested Dissection ($lev$ levels)
- GMRES(35) used as inner solver
- Radius of circle $\equiv$ 30; 8 poles

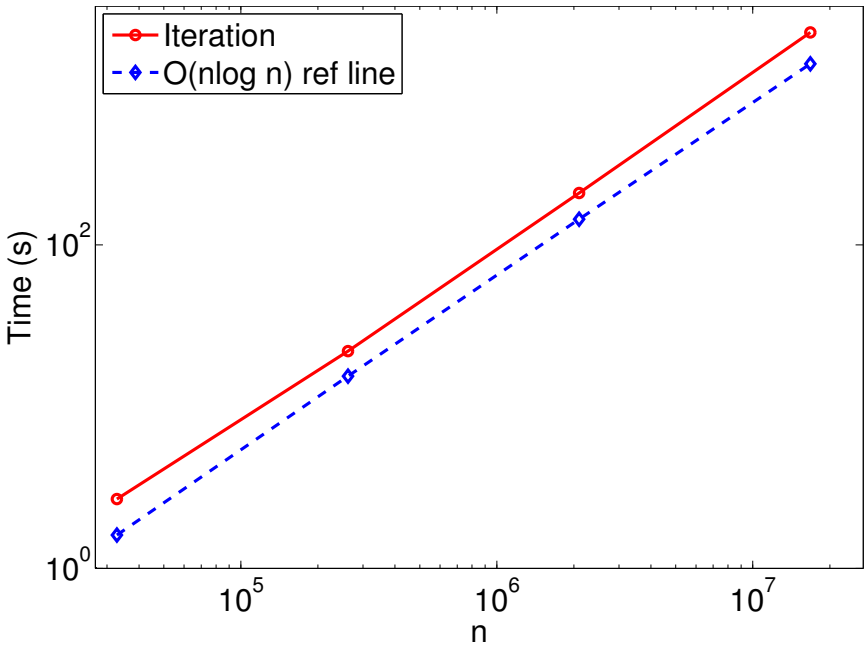
| $n = N^3$ | $\omega/(2\pi)$ | lev | fill | p-t | i-t | its |
|---|---|---|---|---|---|---|
| $32^3$ | 4 | 9 | 8.84 | 0.55 | 2.67 | 2 |
| $64^3$ | 8 | 12 | 10.89 | 4.63 | 22.04 | 2 |
| $128^3$ | 16 | 15 | 11.25 | 43.27 | 209.4 | 2 |
| $256^3$ | 32 | 18 | 11.67 | 428.77 | 2059.1 | 2 |

- p-t, i-t == preconditioning set-up time, iteration time; its = outer iterations.
- lev = # levels used in Nested Dissection.

# Time scaling




Construction




Iteration

## *Conclusion*

➤ New Mantra: Seek "rank-sparsity" or "spectral sparsity" instead of regular sparsity

➤ More work to do : (1) Good HID partitioniers; (2) Parallel implementations; (3) *Very* highly indefinite problems

*Advantages of Multilevel Low-Rank preconditioners:*

(1) Approximate inverses $\rightarrow$ less sensitive to indefiniteness;
(2) Exploit dense computations; (3) Easy to update.

---

➤ Visit `http://cs.umn.edu/~saad`
for this talk, papers, codes, ...

## *Conclusion – Embrace New Horizons*

➤ Many, many, interesting New matrix problems related to the new economy and new emerging scientific fields:

1. Information technologies [learning, data-mining, ...]
2. Computational Chemistry / materials science
3. Bio-informatics, computational biology, genomics, ..

*When one door closes, another opens; but we often look so long and so regretfully upon the closed door that we do not see the one which has opened for us.*

Alexander Graham Bell (1847-1922)