UNIVERSITY OF MINNESOTA TWIN CITIES

Multilevel Low-Rank preconditioners

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

SIAM-PP Feb 21,2014

Introduction, Motivation

- Primary goal for this work: Alternatives to ILU preconditioners
- First motivation: avoid irregular computations [Use: GPUs]
- Second: revisit approximate inverse preconditioners – but from the angle of Low-rank approximations.
- Third: exploit recent work on HSS matrices, H-matrices, ... [J. Xia et al 2010, Hackbusch, Engquist and Ying,...]

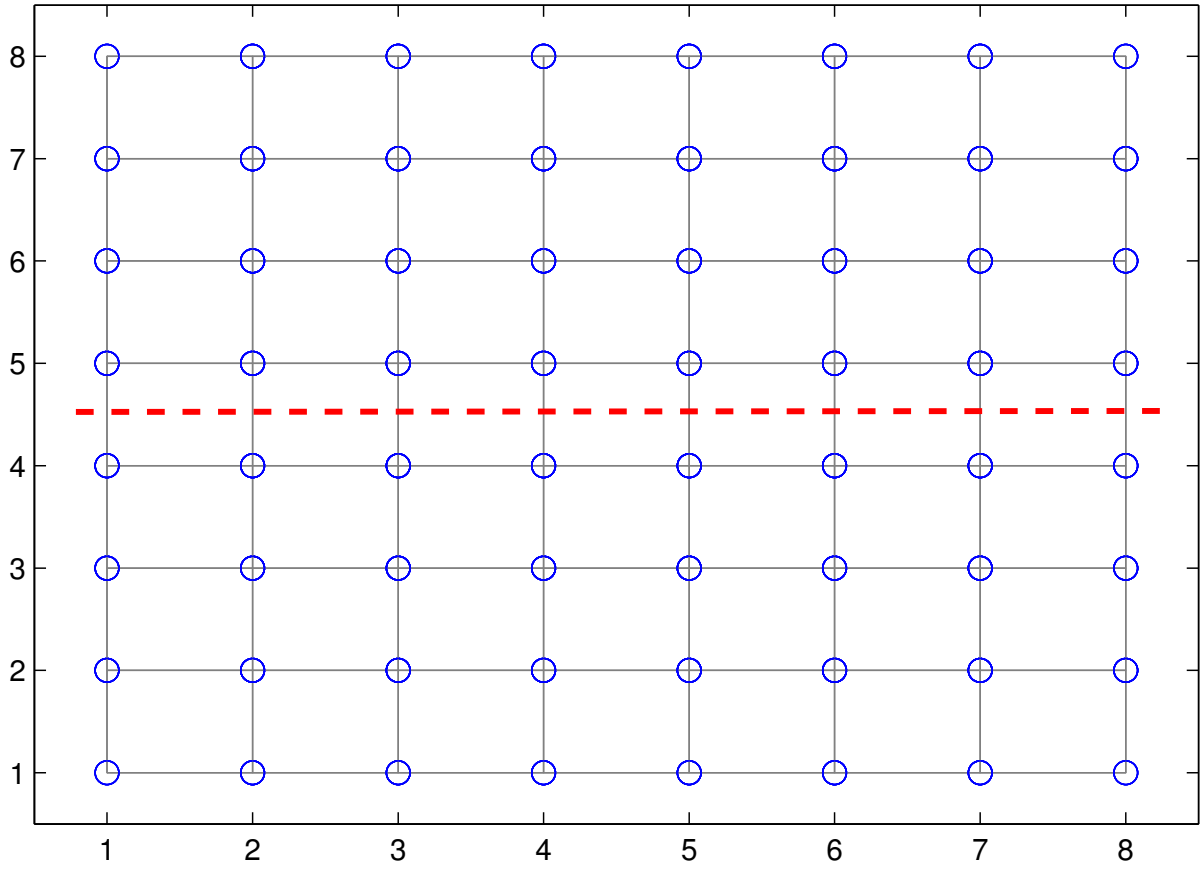
Low-rank Multilevel Approximations

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

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

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

Corresponding splitting on FD mesh:



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

- In the simplest case second matrix is:

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

- Write 2nd matrix as:

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

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

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

- Thus: $A = \underbrace{(A + EE^T)}_{B} - EE^T$
- Note: $E \in \mathbb{R}^{n \times n_x}$, $X \in \mathbb{R}^{n_x \times n_x}$
- n_x = | separator | = [$O(n^{1/2})$ in 2-D, $O(n^{2/3})$ in 3-D]

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

- Next step: use Sherman-Morrison formula:

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

Multilevel Low-Rank (MLR) algorithm

- Use in a recursive framework [apply recursively to B_1, B_2]
- Next step: low-rank approx. for $B^{-1}E$

$$B^{-1}E \approx U_k V_k^T, \qquad \begin{array}{l} U_k \in \mathbb{R}^{n \times k}, \\ V_k \in \mathbb{R}^{n_x \times k}, \end{array}$$

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

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

Preconditioner

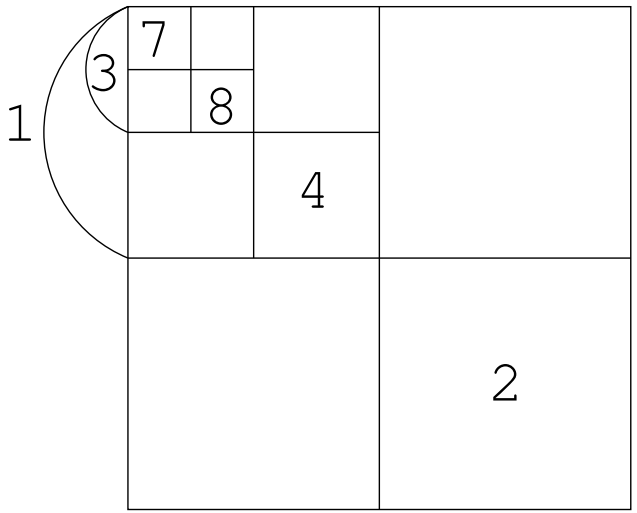
$$M^{-1} = B^{-1} + U_k H_k U_k^T$$

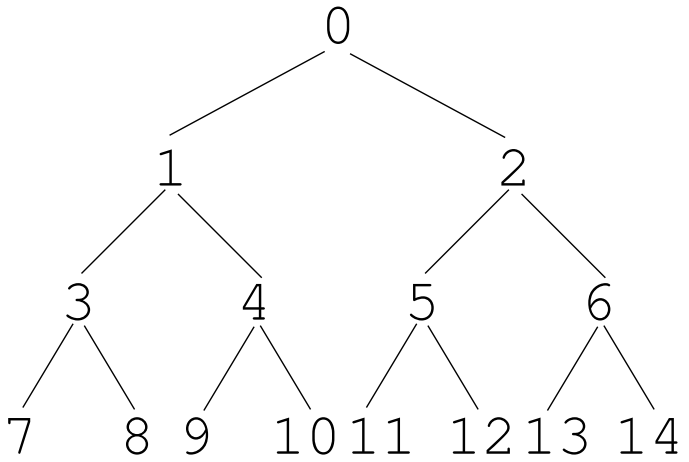
↖ Use recursivity

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

Recursive multilevel framework

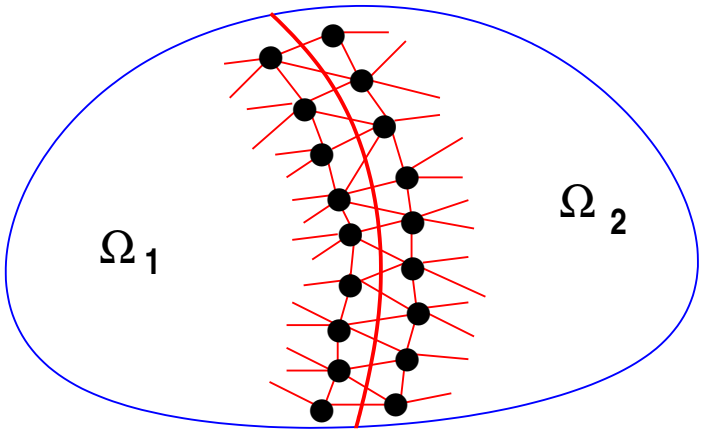
- $A_i = B_i + E_i E_i^T$, $B_i \equiv \begin{pmatrix} B_{i_1} & \\ & B_{i_2} \end{pmatrix}$.
- Next level, set $A_{i_1} \equiv B_{i_1}$ and $A_{i_2} \equiv B_{i_2}$
- Repeat on A_{i_1}, A_{i_2}
- Last level, factor A_i (IC, ILU)
- Binary tree structure:





Generalization: Domain Decomposition framework

Domain partitioned into 2 domains with an edge separator



➤ Matrix can be permuted to:

$$PAP^T = \left(\begin{array}{cc|cc} \hat{B}_1 & \hat{F}_1 & & \\ \hat{F}_1^T & C_1 & & -X \\ \hline & & \hat{B}_2 & \hat{F}_2 \\ & -X^T & \hat{F}_2^T & C_2 \end{array}\right)$$

➤ Interface nodes in each domain are listed last.

- Each matrix $\hat{B}_i$ is of size $n_i \times n_i$ (interior var.) and the matrix C_i is of size $m_i \times m_i$ (interface var.)

$$\text{Let:} \quad E_\alpha = \begin{pmatrix} 0 \\ \alpha I \\ 0 \\ \frac{X^T}{\alpha} \end{pmatrix} \quad \text{then we have:}$$

$$PAP^T = \begin{pmatrix} B_1 & \\ & B_2 \end{pmatrix} - EE^T \quad \text{with} \quad B_i = \begin{pmatrix} \hat{B}_i & \hat{F}_1 \\ \hat{F}_i^T & C_i + D_i \end{pmatrix}$$

$$\text{and} \quad \begin{cases} D_1 = \alpha^2 I \\ D_2 = \frac{1}{\alpha^2} X^T X \end{cases} .$$

- α used for balancing
- Better results when using diagonals instead of αI

Experiments

- Hardware: Intel Xeon X5675 processor
- C/C++; Intel Math Kernel Library (MKL,version 10.2)
- Model 2-D/3-D Problems discret. with 5pt/7pt FD

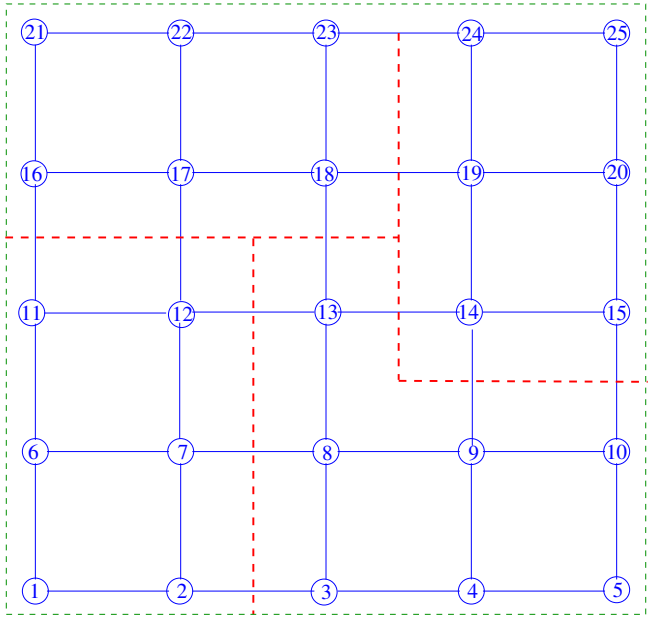
$$-\Delta u - cu = g \text{ in } \Omega \quad + \text{B.C.}$$

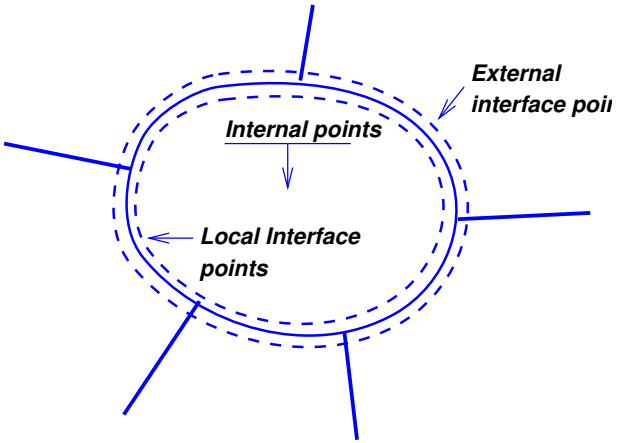
- $c > 0$ in $-\Delta u - cu$; i.e., $-\Delta$ shifted by $-sI$.
- 2D case: $s = 0.01$, 3D case: $s = 0.05$
- MLR + GMRES(40) compared to ILDLT + GMRES(40)
- 2-D problems: #lev$= 4$, rank$= 5, 7, 7$
- 3-D problems: #lev$= 5$, rank$= 5, 7, 7$
- ILDLT failed for most cases

Grid	ILDLT-GMRES				MLR-GMRES			
	fill	p-t	its	i-t	fill	p-t	its	i-t
256^2	6.5	0.16	F		6.0	0.39	84	0.30
512^2	8.4	1.25	F		8.2	2.24	246	6.03
1024^2	10.3	10.09	F		9.0	15.05	F	
$32^2 \times 64$	5.6	0.25	61	0.38	5.4	0.98	62	0.22
64^3	7.0	1.33	F		6.6	6.43	224	5.43
128^3	8.8	15.35	F		6.5	28.08	F	

Avoiding recursivity: 'standard' DD framework

- Work in progress
- Goal: avoid recursive version
- Consider a domain partition in p domains using an vertex- based partitioniong (edge-separation)
- Interface nodes in each domain are listed last.





$$\begin{pmatrix} \boldsymbol{B}_i & \boldsymbol{F}_i \\ \boldsymbol{E}_i^T & \boldsymbol{C}_i \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i \\ \boldsymbol{y}_i \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ \sum_{j \in \boldsymbol{N}_i} \boldsymbol{E}_{ij} \boldsymbol{y}_j \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i \\ \boldsymbol{g}_i \end{pmatrix}$$

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

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

- 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 : $$F \equiv \begin{pmatrix} \hat{F} \\ -I \end{pmatrix}; \quad E \equiv \begin{pmatrix} \hat{E} \\ -I \end{pmatrix}$$
- Then: $$\left[\begin{array}{c|c} B & \hat{F} \\ \hline \hat{E}^T & C \end{array}\right] = \left[\begin{array}{c|c} B + \hat{F}\hat{E}^T & 0 \\ \hline 0 & C + I \end{array}\right] - FE^T$$

Low-Rank Approximation DD preconditioners

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

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

$=$ block-diagonal

Sherman-Morrison $\rightarrow$

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

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 but will be more accurate

Let $G \approx G_k$
Preconditioner $\rightarrow$

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

Symmetric Positive Definite case

- Matrix A_0 is SPD.
- Let $H = E^T A_0^{-1} E$
- Eigenvalues of H are between 0 and 1.
- Next, approximate H as $H \approx U_k D_k U_k^T$; $G_k = I - U_k D_k U_k^T$. Then:

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

- Observation: $A^{-1} = M^{-1} + A_0^{-1} E[G^{-1} - G_k^{-1}]E^T A_0^{-1}$
- k largest eigenvalues of G matched – others set == 0

- Consequence: AM^{-1} has

- $n - s + k$ eigenvalues equal to 1, others between 0 and 1

- More general alternative. Let $\gamma = 1/(1-\theta)$ and

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

- k largest eigenvalues of G matched – others set == θ

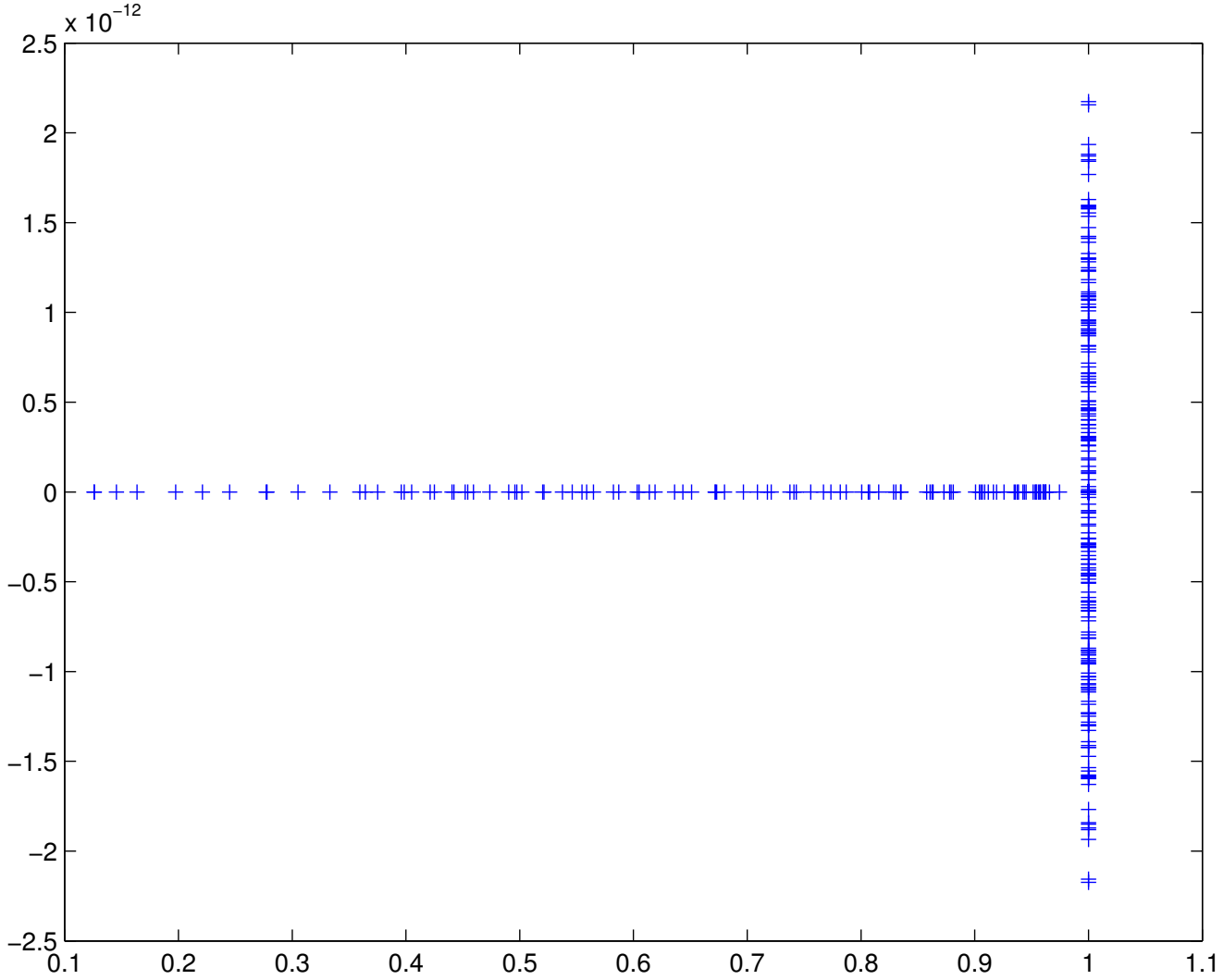
- $\theta = 0$ yields previous case

- When $\theta = \lambda_{k+1}$ we get

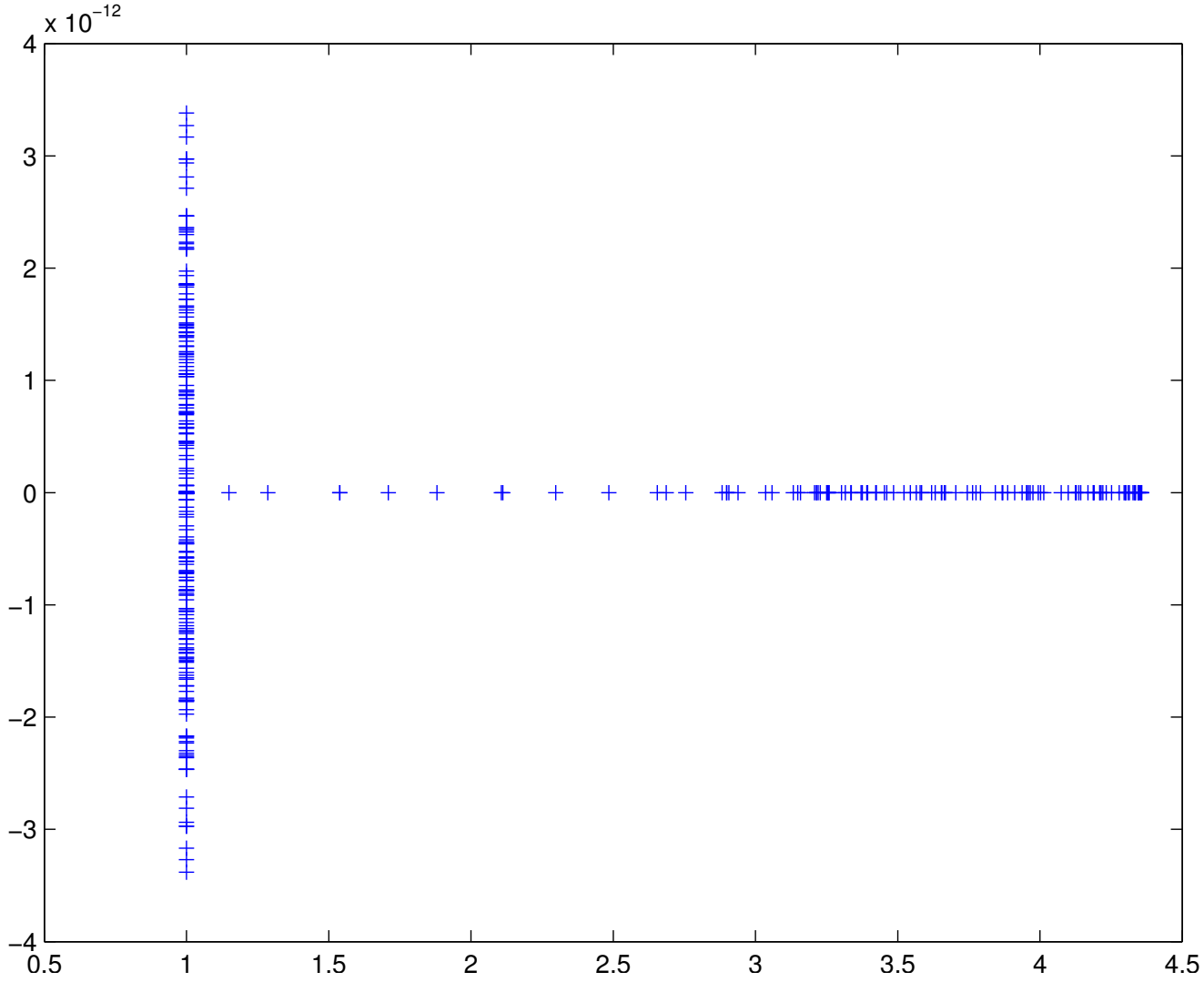
- $n - s + k$ eigenvalues equal to one, others ≥ 1

- Next: An example for a 900×900 Laplacean, 4 domains, $s = 119$

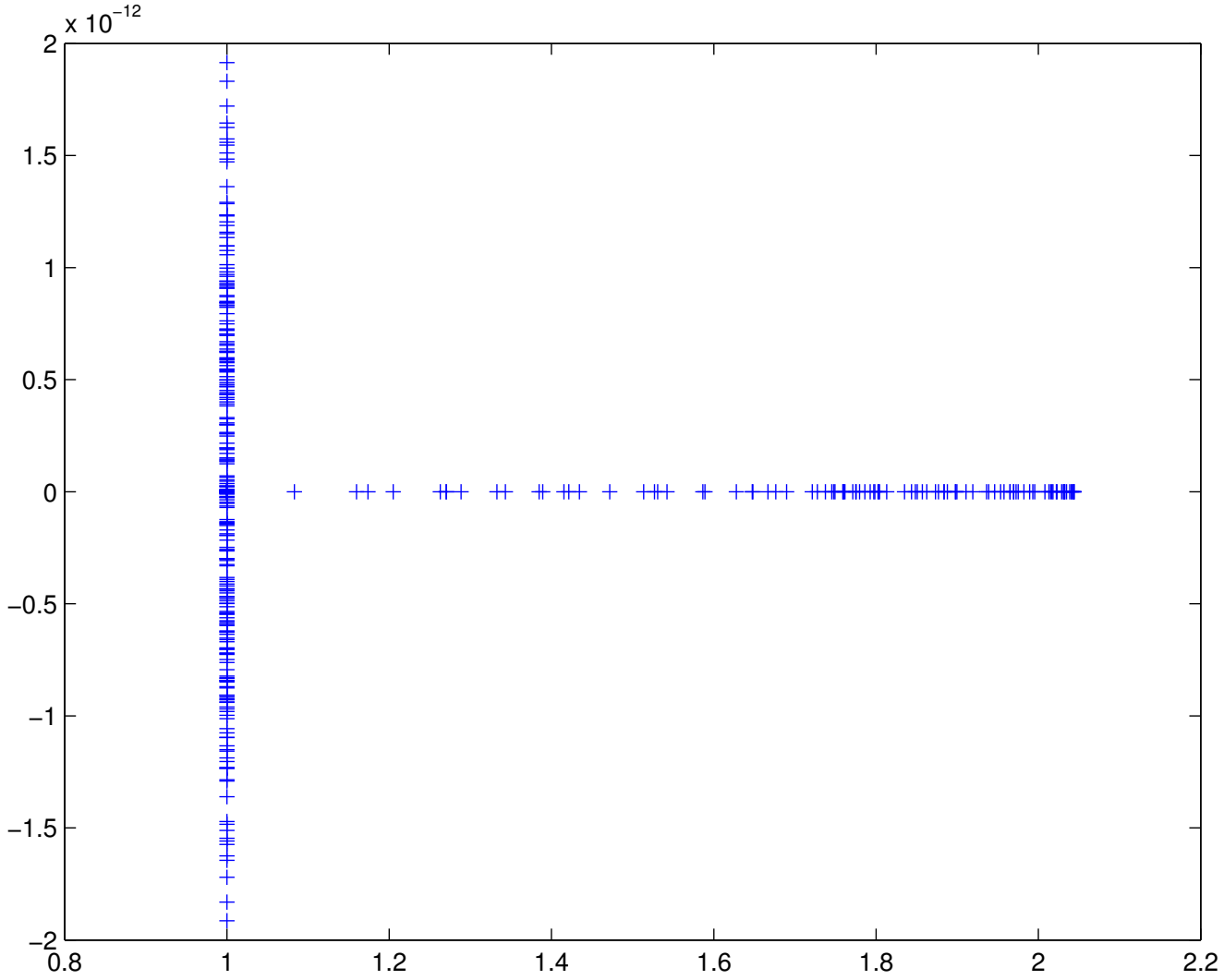
$k = 5$ Eigenvalues of AM^{-1} for the case $\theta = 0$



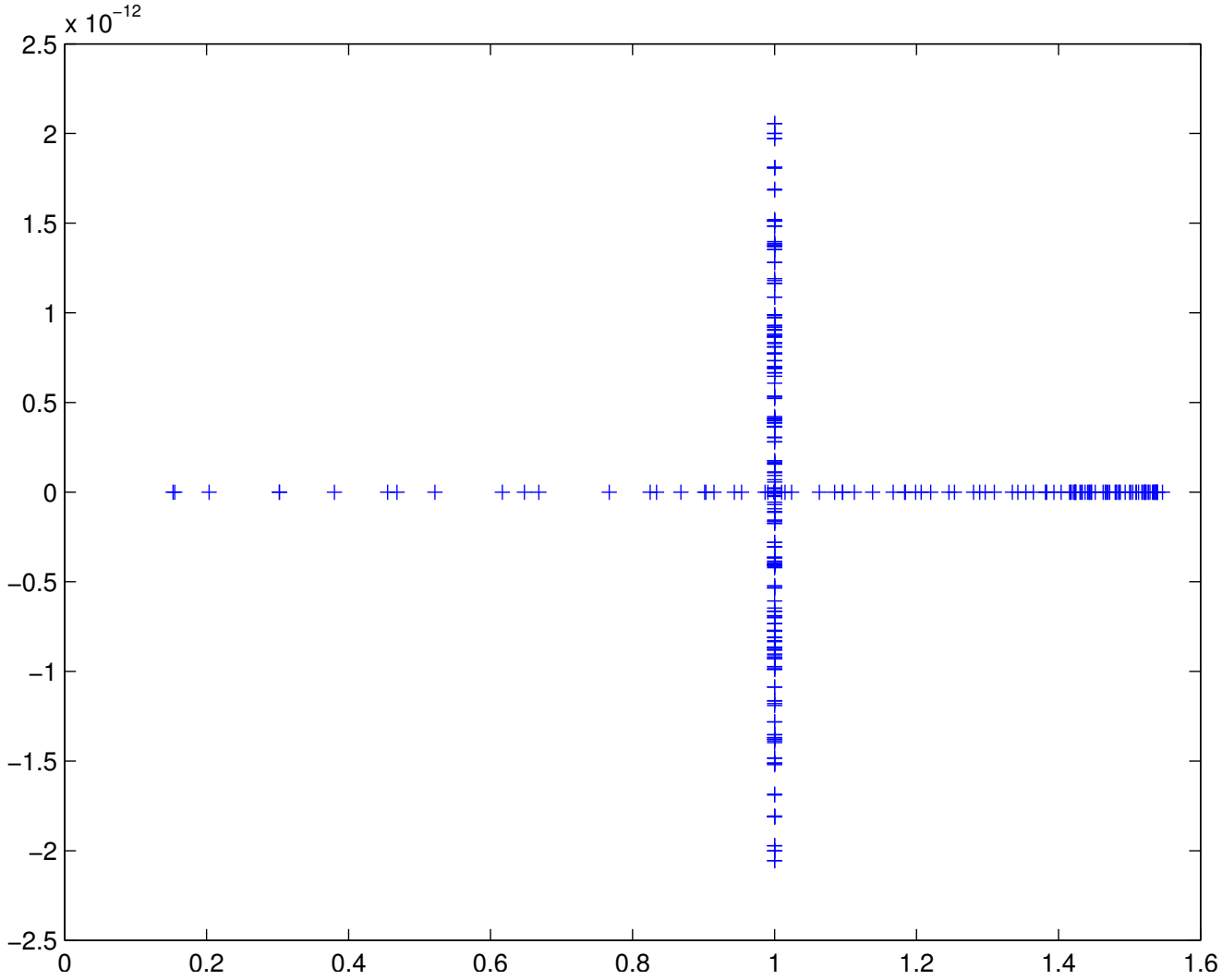
$k = 5$ Eigenvalues of AM^{-1} for the case $\theta = \lambda_{k+1}$



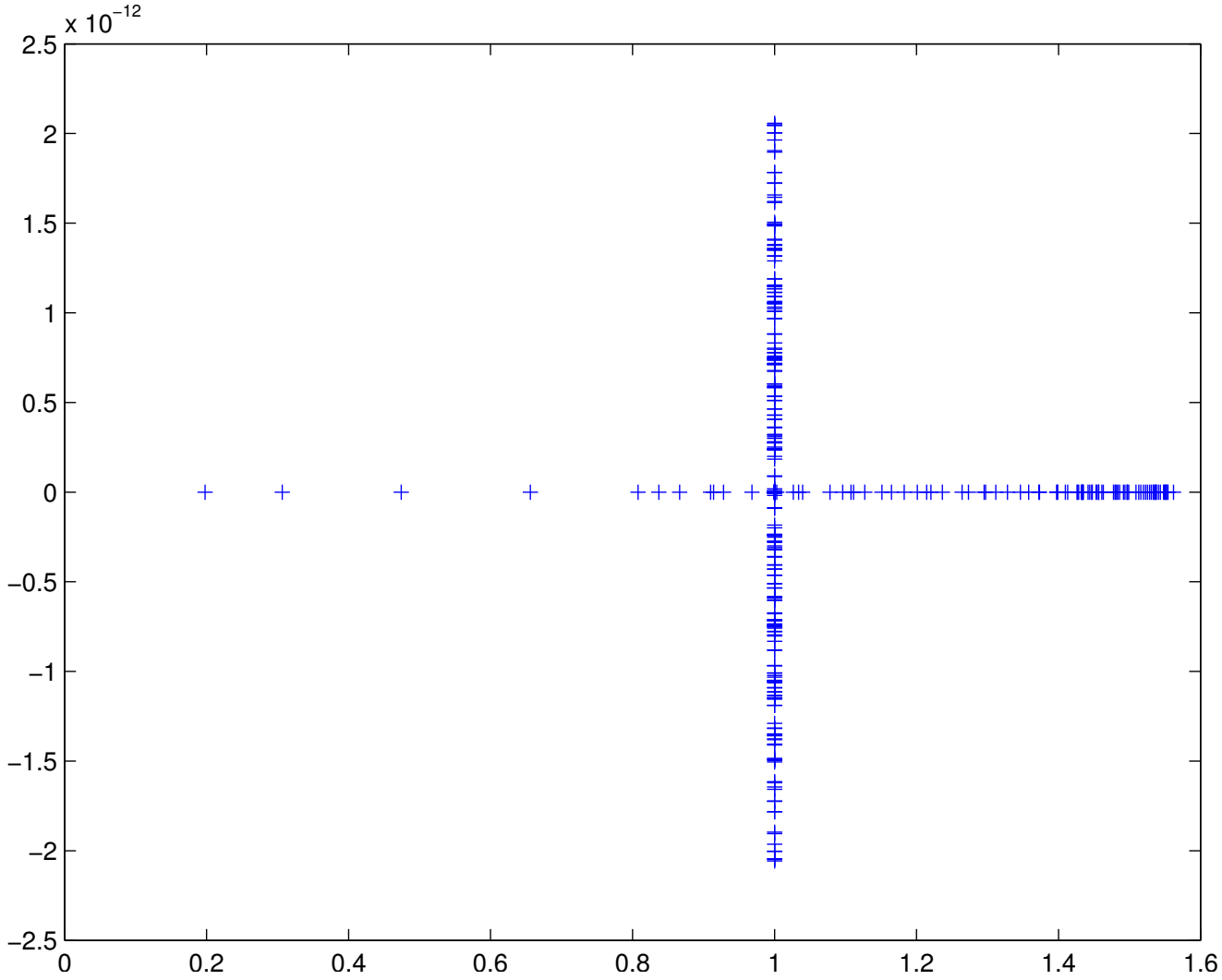
$k = 15$ Eigenvalues of AM^{-1} for the case $\theta = \lambda_{k+1}$



$k = 5$ Eigenvalues of AM^{-1} for the case $\theta = \lambda_{k+1}$ - computed from $3k$ steps of Lanczos



$k = 15$ Eigenvalues of AM^{-1} for the case $\theta = \lambda_{k+1}$ - computed from $3k$ steps of Lanczos



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

Other advantages:

- Easy to update
- Requires little memory [store U_k, D_k]