UNIVERSITY OF MINNESOTA TWIN CITIES

Multilevel low-rank approximation preconditioners

Yousef Saad
Department of Computer Science and Engineering

University of Minnesota

SIAM CSE
Boston – March 1, 2013

First:

- Joint work with Ruipeng Li
- Work supported by NSF

Introduction

- Preconditioned Krylov subspace methods offer a good alternative to direct solution methods

- Especially for 3D problems

- Compromise between performance and robustness

.... But there are challenges:

- Highly indefinite systems [Helmholtz, Maxwell, ...]
- Highly ill-conditioned systems [structures,..]
- Problems with extremely irregular nonzero pattern
- Recent: impact of new architectures [many core, GPUs]

Introduction (cont.)

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

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
- Unknowns $x(i)$ of same level can be computed 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...

Or perhaps: Alternative preconditioners?

➤ What would be a good alternative?

Wish-list:

- A preconditioner requiring few 'irregular' computations
- Something that trades volume of computations for speed
- If possible something that is robust for indefinite case

➤ Good candidate:

- Multilevel Low-Rank (MLR) approximate inverse preconditioners

Related work:

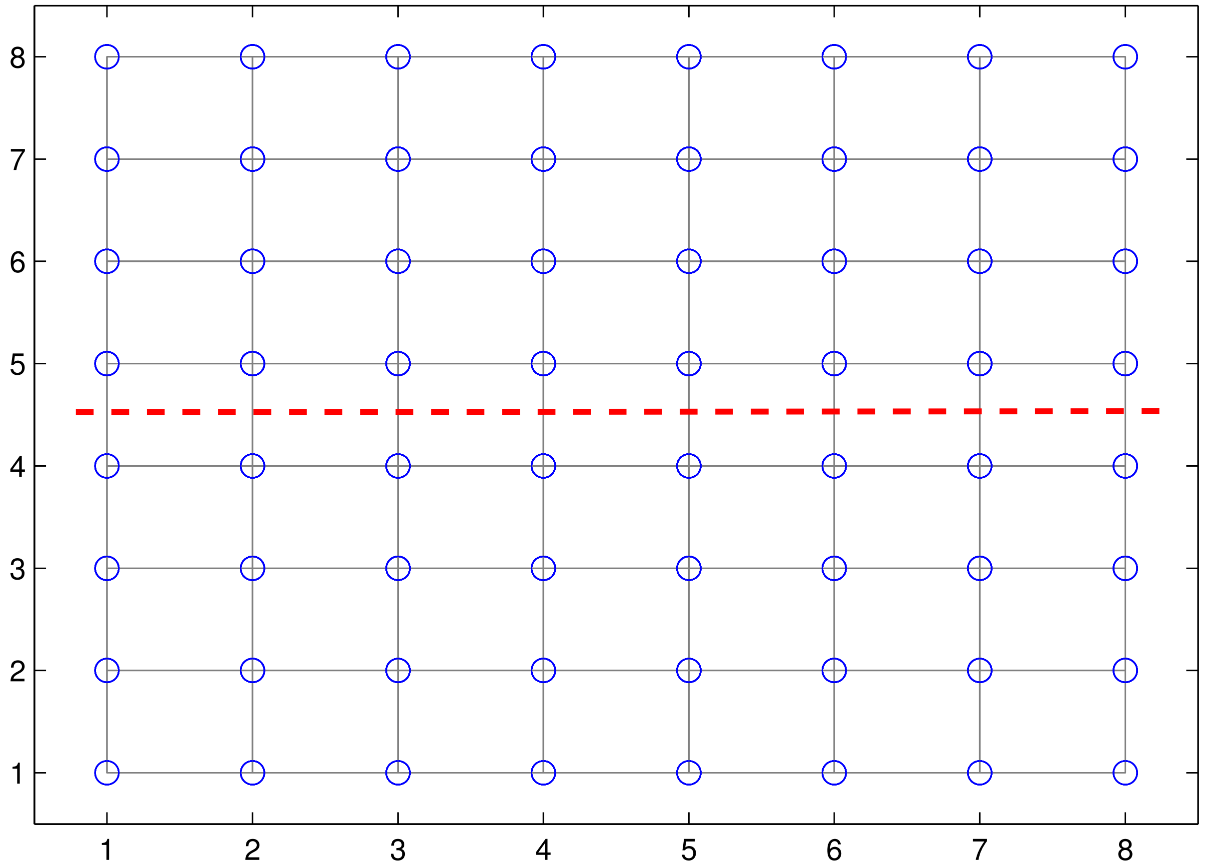
- Work on HSS matrices [e.g., JIANLIN XIA, SHIVKUMAR CHANDRASEKARAN, MING GU, AND XIAOYE S. LI, *Fast algorithms for hierarchically semiseparable matrices*, Numerical Linear Algebra with Applications, 17 (2010), pp. 953–976.]
- Work on H-matrices [Hackbusch, ...]
- Work on 'balanced incomplete factorizations'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

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

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

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

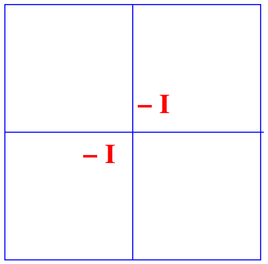
Corresponding splitting on FD mesh:



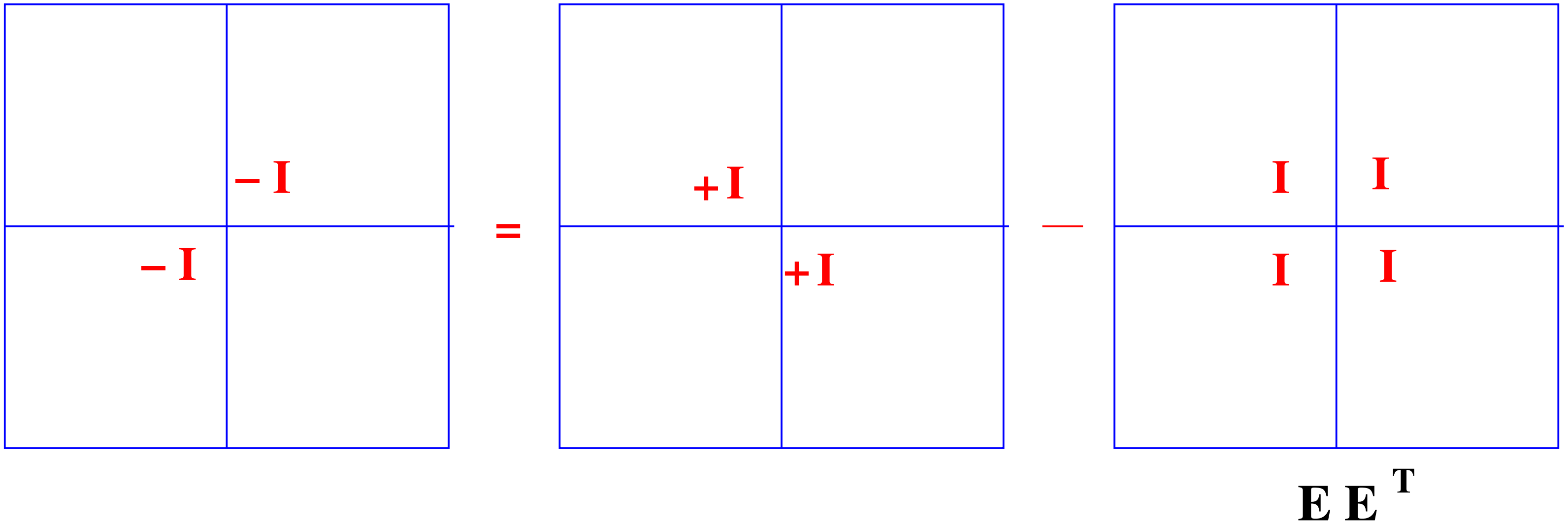
- $A_{11} \in \mathbb{R}^{m \times m}$, $A_{22} \in \mathbb{R}^{(n-m)\times(n-m)}$

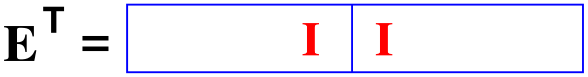
- In the simplest case second matrix is:

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



- Write 2nd matrix as:





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

- Shermann-Morrison formula:

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

$$A^{-1} \equiv B^{-1} + B^{-1} E X^{-1} E^T B^{-1}$$
$$X = I - E^T B^{-1} E$$

- Note: $E \in \mathbb{R}^{n \times n_x}$, $X \in \mathbb{R}^{n_x \times n_x}$

- n_x = number of points in separator [$O(n^{1/2})$ for 2-D mesh, $O(n^{2/3})$ for 3-D mesh]

- Use in a recursive framework

- Similar idea was used for computing the diagonal of the inverse [J. Tang YS '10]

Multilevel Low-Rank (MLR) algorithm

➤ 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:

$$M^{-1} = B^{-1} + U_k [V_k^T G_k^{-1} V_k] U_k^T$$

↖ Use recursivity (for B^{-1})

Note: From $A^{-1} = B^{-1}[I + EX^{-1}E^TB^{-1}]$ could define:

$$M_1^{-1} = B^{-1}[I + EG_k^{-1}V_kU_k^T].$$

[rationale: approximation made on 'one side only']

- It turns out M_1 and M are equal!
- We have:

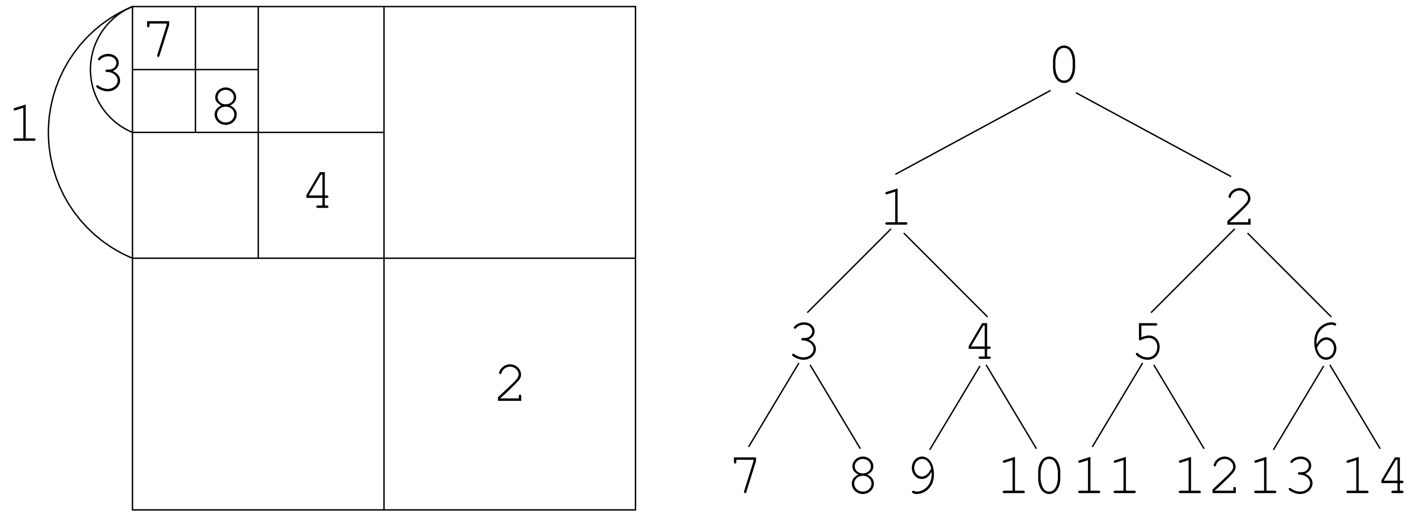
$$M^{-1} = B^{-1} + U_kH_kU_k^T, \quad \text{with} \quad H_k = V_k^TG_k^{-1}V_k.$$

- No need to store V_k: Only keep U_k and H_k ($k \times k$).
- We can show : $H_k = (I - U_k^TEV_k)^{-1}$

... and : H_k *is symmetric*

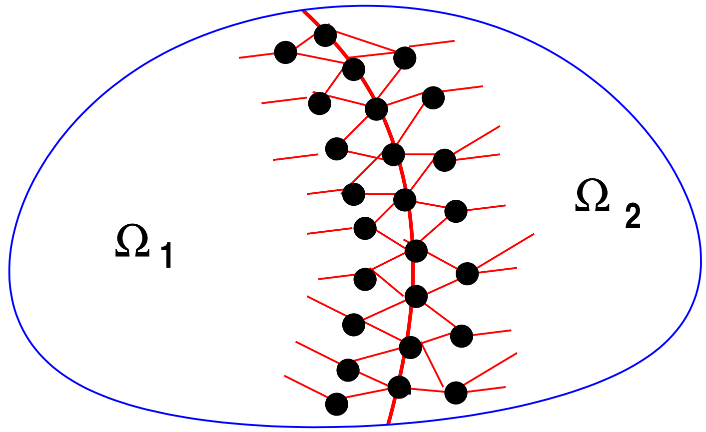
Recursive multilevel framework

- $A_i = B_i + E_i E_i^T$, $B_i \equiv \begin{pmatrix} B_{i_1} & \\ & B_{i_2} \end{pmatrix}$.
- 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.)

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

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

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

➤ α used for balancing

➤ Better results when using diagonals instead of αI

Theory: 2-level analysis for model problem

- Interested in eigenvalues γ_j of

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

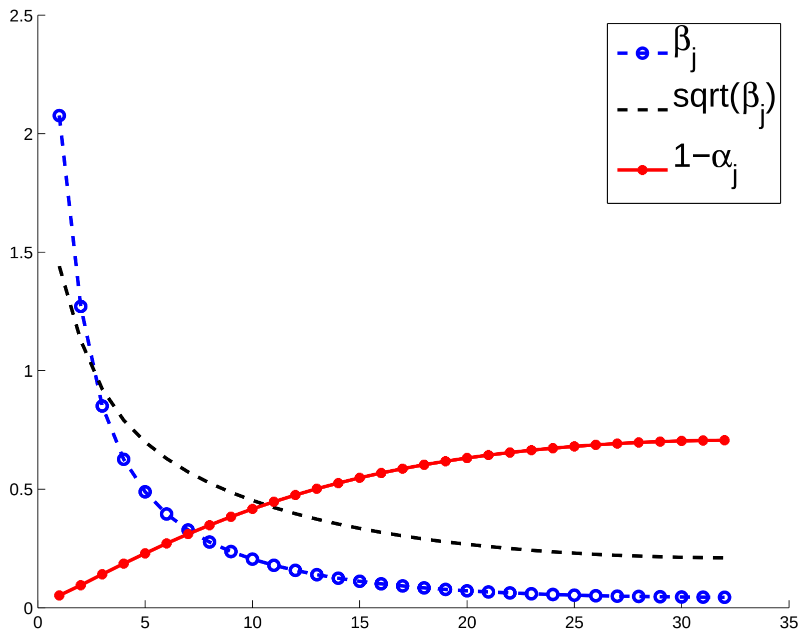
when A = Pure Laplacean .. They are:

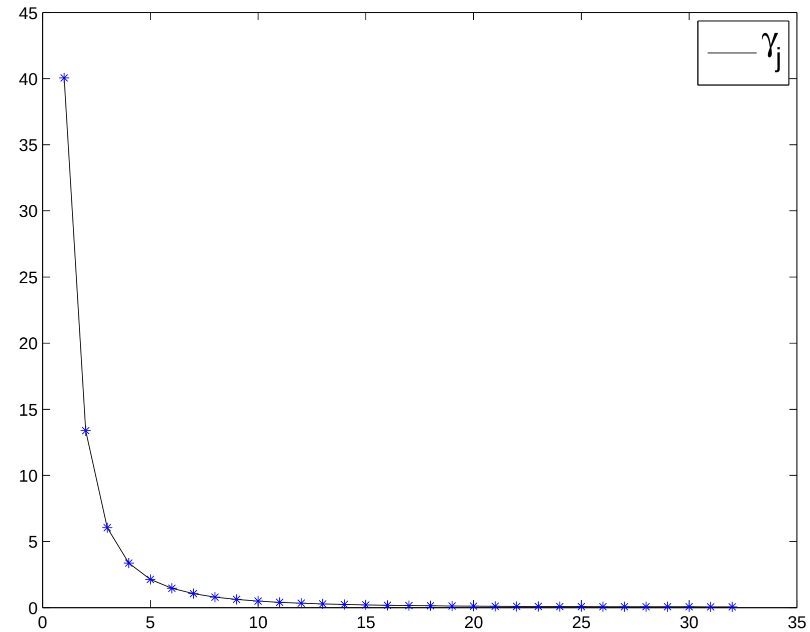
$$\gamma_j = \frac{\beta_j}{1 - \alpha_j}, \quad j = 1, \cdots, n_x \quad \text{with:}$$

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

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

➤ Decay of the γ_js 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.

EXPERIMENTS

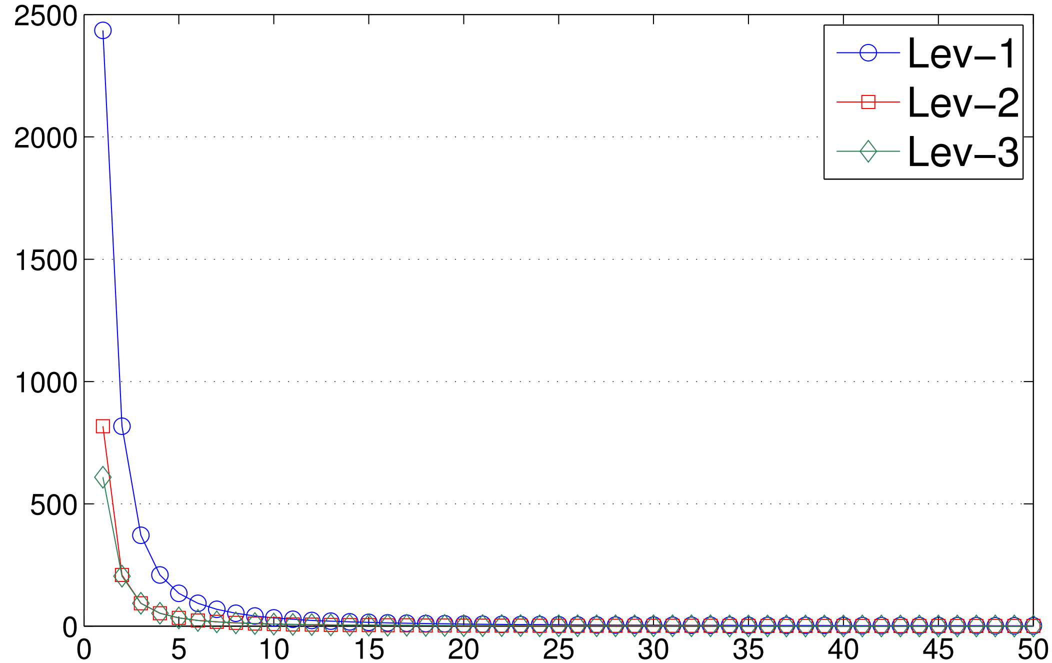
Experimental setting

- Hardware: Intel Xeon X5675 processor (12 MB Cache, 3.06 GHz, 6-core)
- C/C++; Intel Math Kernel Library (MKL,version 10.2)
- Stopping criteria:
  - $\| r_i \| \leq 10^{-8} \| r_0 \|$
  - Maximum number of iterations: 500

2-D/3-D model problems (theory)

$$-\Delta u - cu = -\left(x^2 + y^2 + c\right) e^{xy} \text{ in } (0,1)^2 ,$$
$$+ \text{ Dirichlet BC}$$

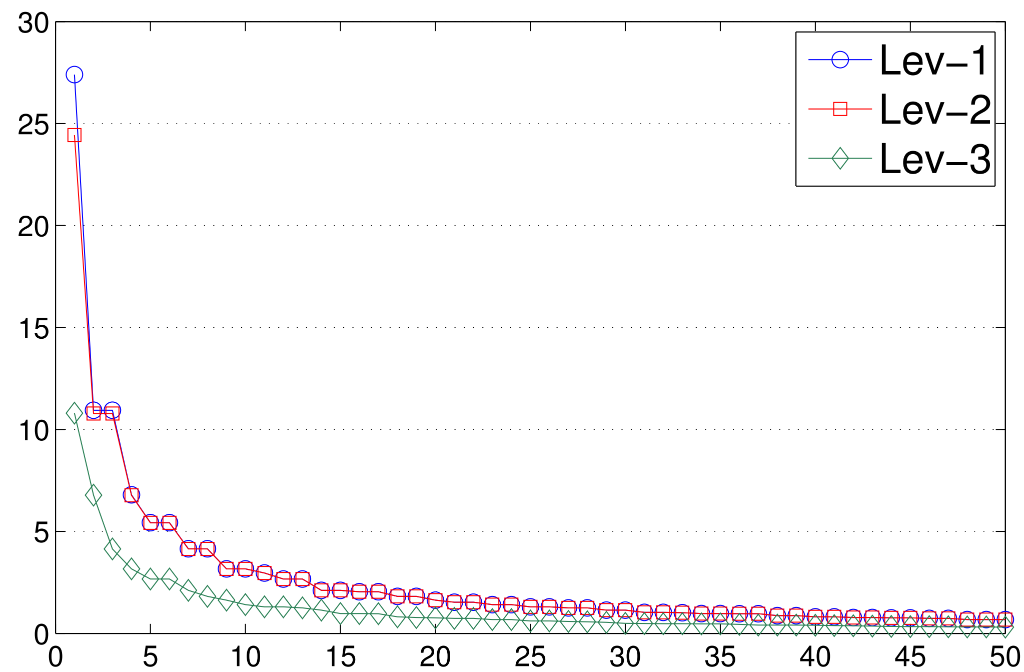
- FD discret.: $n_x = n_y = 256$
- Eigenvalues of $B_i^{-1} E_i X_i^{-1} E_i^T B_i^{-1}$
- $i = 0, 1, 3$
- Rapid decay.



3-D elliptic PDE

$$-\Delta u - cu = -6 - c\left(x^2 + y^2 + z^2\right) \text{ in } (0,1)^3,$$
$$+ \text{ Dirichlet BC}$$

- FD discret.: $n_x = n_y = 32$, $nz = 64$
- Eigenvalues of $B_i^{-1} E_i X_i^{-1} E_i^T B_i^{-1}$
- $i = 0, 1, 3$
- Rapid decay.



Tests: SPD cases

- SPD cases, pure Laplacean ($c = 0$ in previous equations)
- MLR + PCG compared to IC + PCG
- 2-D problems: #lev$= 5$, rank$= 2$
- 3-D problems: #lev$= 5, 7, 10$, rank$= 2$

Grid	N	ICT-CG				MLR-CG			
		fill	p-t	its	i-t	fill	p-t	its	i-t
$\mathbf{256^2}$	$\mathbf{65K}$	**3.1**	**0.08**	**69**	**0.19**	**3.2**	**0.45**	**84**	**0.12**
$\mathbf{512^2}$	$\mathbf{262K}$	**3.2**	**0.32**	**133**	**1.61**	**3.5**	**1.57**	**132**	**1.06**
$\mathbf{1024^2}$	$\mathbf{1,048K}$	**3.4**	**1.40**	**238**	**15.11**	**3.5**	**4.66**	**215**	**9.77**
$\mathbf{32^2.64}$	65K	2.9	0.14	33	0.10	3.0	0.46	43	0.08
$\mathbf{64^3}$	262K	3.0	0.66	47	0.71	3.1	3.03	69	0.63
$\mathbf{128^3}$	2,097K	3.0	6.59	89	13.47	3.2	24.61	108	10.27

- Set-up times for MLR preconditioners are higher
- Bear in mind the ultimate target architecture [SIMD...]

Symmetric indefinite cases

- $c > 0$ 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
- Difficulties in MLR: #lev cannot be large, [no convergence]
- inefficient factorization at the last level (memory, CPU time)

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	

General symmetric matrices - Test matrices

MATRIX	N	NNZ	SPD	DESCRIPTION
Andrews/Andrews	60,000	760,154	yes	computer graphics pb.
Williams/cant	62,451	4,007,383	yes	FEM cantilever
UTEP/Dubcova2	65,025	1,030,225	yes	2-D/3-D PDE pb.
Rothberg/cfd1	70,656	1,825,580	yes	CFD pb.
Schmid/thermal1	82,654	574,458	yes	thermal pb.
Rothberg/cfd2	123,440	3,085,406	yes	CFD pb.
Schmid/thermal2	1,228,045	8,580,313	yes	thermal pb.
Cote/vibrobox	12,328	301,700	no	vibroacoustic pb.
Cunningham/qa8fk	66,127	1,660,579	no	3-D acoustics pb.
Koutsovasilis/F2	71,505	5,294,285	no	structural pb.

Generalization of MLR via DD

- DD: `PartGraphRecursive` from METIS
- balancing with diagonals
- higher ranks used in two problems (`cant` and `vibrobox`)
- Show SPD cases first then non-SPD

MATRIX	ICT/ILDLT				MLR-CG/GMRES					
	fill	p-t	its	i-t	k	lev	fill	p-t	its	i-t
Andrews	2.6	0.44	32	0.16	2	6	2.3	1.38	27	0.08
cant	4.3	2.47	F	19.01	10	5	4.3	7.89	253	5.30
Dubcova2	1.4	0.14	42	0.21	4	4	1.5	0.60	47	0.09
cfd1	2.8	0.56	314	3.42	5	5	2.3	3.61	244	1.45
thermal1	3.1	0.15	108	0.51	2	5	3.2	0.69	109	0.33
cfd2	3.6	1.14	F	12.27	5	4	3.1	4.70	312	4.70
thermal2	5.3	4.11	148	20.45	5	5	5.4	15.15	178	14.96

MATRIX	ICT/ILDLT				MLR-CG/GMRES					
	fill	p-t	its	i-t	k	lev	fill	p-t	its	i-t
vibrobox	3.3	0.19	F	1.06	10	4	3.0	0.45	183	0.22
qa8fk	1.8	0.58	56	0.60	2	8	1.6	2.33	75	0.36
F2	2.3	1.37	F	13.94	5	5	2.5	4.17	371	7.29

Conclusion

- Promising approach –
- Many more avenues to explore:
  - Nonsymmetric case,
  - Implementation on GPUS,
  - Storage for 3D case
  - ...