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]
Main issue in using GPUs for sparse computations:

- Huge performance degradation due to ‘irregular sparsity’

<table>
<thead>
<tr>
<th>Matrix</th>
<th>N</th>
<th>NNZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEM/Cantilever</td>
<td>62,451</td>
<td>4,007,383</td>
</tr>
<tr>
<td>Boeing/pwtk</td>
<td>217,918</td>
<td>11,634,424</td>
</tr>
</tbody>
</table>

Performance of Mat-Vecs on NVIDIA Tesla C1060

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Single Precision</th>
<th>Double Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CSR</td>
<td>JAD</td>
</tr>
<tr>
<td>FEM/Cantilever</td>
<td>9.4</td>
<td>10.8</td>
</tr>
<tr>
<td>Boeing/pwtk</td>
<td>8.9</td>
<td>16.6</td>
</tr>
</tbody>
</table>
Sparse Forward/Backward Sweeps

- Next major ingredient of preconditioned Krylov subspace 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]

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

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

\[
\begin{pmatrix}
A_1 & D_2 & \\
D_2 & A_2 & D_3 \\
\vdots & \vdots & \vdots \\
D_\alpha & A_\alpha & D_{\alpha+1} \\
\hline
D_{\alpha+1} & A_{\alpha+1} & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
D_{n_y} & A_{n_y}
\end{pmatrix}
\]

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

\[
\begin{pmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{pmatrix}
= \begin{pmatrix}
A_{11} \\
A_{21}
\end{pmatrix}
+ \begin{pmatrix}
-1 & \\
-1 & -1
\end{pmatrix}
\]

Write 2nd matrix as:

\[
E^T = \begin{pmatrix}
I & I
\end{pmatrix}
\]

\[
EE^T
\]
Above splitting can be rewritten as

\[ A = \left( A + E E^T \right) - E E^T \]

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

Note: \( B_1 := A_{11} + E_1 E_1^T, \quad B_2 := A_{22} + E_2 E_2^T. \)
Shermann-Morrison formula:

\[ A^{-1} = B^{-1} + B^{-1} E \left( I - E^T B^{-1} E \right)^{-1} E^T B^{-1} \]

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

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 $B^{-1}E$

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

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

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

- Preconditioner:

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

  Use recursivity
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
• \( A_i = B_i + E_i E_i^T, \quad B_i \equiv \begin{pmatrix} B_{i1} \\ B_{i2} \end{pmatrix} \).

• Next level, set \( A_{i1} \equiv B_{i1} \) and \( A_{i2} \equiv B_{i2} \)

• Repeat on \( A_{i1}, A_{i2} \)

• Last level, factor \( A_i \) (IC, ILU)

• Binary tree structure:
Domain partitioned into 2 domains with an edge separator

Matrix can be permuted to:

$$P A P^T = \begin{pmatrix} \hat{B}_1 & \hat{F}_1 & \hat{F}_1^T & C_1 & -X \\ \hat{F}_1 & \hat{F}_1^T & \hat{B}_2 & \hat{F}_2 & \hat{F}_2^T & C_2 \end{pmatrix}$$

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:

$$E_\alpha = \begin{pmatrix} 0 \\ \alpha I \\ 0 \\ X^T \\ \frac{X}{\alpha} \end{pmatrix}$$

then we have:

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

and

$$\left\{ \begin{array}{l}
D_1 = \alpha^2 I \\
D_2 = \frac{1}{\alpha^2} X^T X
\end{array} \right.$$

$\alpha$ used for balancing

Better results when using diagonals instead of $\alpha I$
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:

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

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

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

\[-\Delta u - cu = -(x^2 + y^2 + c) e^{xy} \text{ in } (0, 1)^2 , \]

\[+ \text{ Dirichlet BC} \]

- **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.**
$-\Delta u - cu = -6 - c(x^2 + y^2 + z^2)$ in $(0, 1)^3$, 
+ Dirichlet BC

- FD discret.: $n_x = n_y = 32, n_z = 64$
- Eigenvalues of $B_i^{-1}E_iX_i^{-1}E_i^TB_i^{-1}$
- $i = 0, 1, 3$
- Rapid decay.
Tests: SPD cases

- SPD cases, pure Laplacean \((c = 0\) in previous equations)
- MLR + PCG compared to IC + PCG
- 2-D problems: \(\#\text{lev}= 5, \text{rank}= 2\)
- 3-D problems: \(\#\text{lev}= 5, 7, 10, \text{rank}= 2\)
<table>
<thead>
<tr>
<th>Grid</th>
<th>$N$</th>
<th>ICT-CG</th>
<th></th>
<th></th>
<th>MLR-CG</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>fill</td>
<td>p-t</td>
<td>its</td>
<td>i-t</td>
<td>fill</td>
<td>p-t</td>
</tr>
<tr>
<td>$256^2$</td>
<td>$65K$</td>
<td>3.1</td>
<td>0.08</td>
<td>69</td>
<td>0.19</td>
<td>3.2</td>
<td>0.45</td>
</tr>
<tr>
<td>$512^2$</td>
<td>$262K$</td>
<td>3.2</td>
<td>0.32</td>
<td>133</td>
<td>1.61</td>
<td>3.5</td>
<td>1.57</td>
</tr>
<tr>
<td>$1024^2$</td>
<td>$1,048K$</td>
<td>3.4</td>
<td>1.40</td>
<td>238</td>
<td>15.11</td>
<td>3.5</td>
<td>4.66</td>
</tr>
<tr>
<td>$32^2.64$</td>
<td>$65K$</td>
<td>2.9</td>
<td>0.14</td>
<td>33</td>
<td>0.10</td>
<td>3.0</td>
<td>0.46</td>
</tr>
<tr>
<td>$64^3$</td>
<td>$262K$</td>
<td>3.0</td>
<td>0.66</td>
<td>47</td>
<td>0.71</td>
<td>3.1</td>
<td>3.03</td>
</tr>
<tr>
<td>$128^3$</td>
<td>$2,097K$</td>
<td>3.0</td>
<td>6.59</td>
<td>89</td>
<td>13.47</td>
<td>3.2</td>
<td>24.61</td>
</tr>
</tbody>
</table>

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

SIAM-CSE – March 1, 2013
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)
<table>
<thead>
<tr>
<th>Grid</th>
<th>ILDLT-GMRES</th>
<th></th>
<th>MLR-GMRES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fill</td>
<td>p-t</td>
<td>its</td>
<td>i-t</td>
</tr>
<tr>
<td>256$^2$</td>
<td>6.5</td>
<td>0.16</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>512$^2$</td>
<td>8.4</td>
<td>1.25</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>1024$^2$</td>
<td>10.3</td>
<td>10.09</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>32$^2$ $\times$ 64</td>
<td>5.6</td>
<td>0.25</td>
<td>61</td>
<td>0.38</td>
</tr>
<tr>
<td>64$^3$</td>
<td>7.0</td>
<td>1.33</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>128$^3$</td>
<td>8.8</td>
<td>15.35</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>
### General symmetric matrices - Test matrices

<table>
<thead>
<tr>
<th>MATRIX</th>
<th>N</th>
<th>NNZ</th>
<th>SPD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrews/Andrews</td>
<td>60,000</td>
<td>760,154</td>
<td>yes</td>
<td>computer graphics pb.</td>
</tr>
<tr>
<td>Williams/cant</td>
<td>62,451</td>
<td>4,007,383</td>
<td>yes</td>
<td>FEM cantilever</td>
</tr>
<tr>
<td>UTEP/Dubcova2</td>
<td>65,025</td>
<td>1,030,225</td>
<td>yes</td>
<td>2-D/3-D PDE pb.</td>
</tr>
<tr>
<td>Rothberg/cfd1</td>
<td>70,656</td>
<td>1,825,580</td>
<td>yes</td>
<td>CFD pb.</td>
</tr>
<tr>
<td>Schmid/thermal1</td>
<td>82,654</td>
<td>574,458</td>
<td>yes</td>
<td>thermal pb.</td>
</tr>
<tr>
<td>Rothberg/cfd2</td>
<td>123,440</td>
<td>3,085,406</td>
<td>yes</td>
<td>CFD pb.</td>
</tr>
<tr>
<td>Schmid/thermal2</td>
<td>1,228,045</td>
<td>8,580,313</td>
<td>yes</td>
<td>thermal pb.</td>
</tr>
<tr>
<td>Cote/vibrobox</td>
<td>12,328</td>
<td>301,700</td>
<td>no</td>
<td>vibroacoustic pb.</td>
</tr>
<tr>
<td>Cunningham/qa8fk</td>
<td>66,127</td>
<td>1,660,579</td>
<td>no</td>
<td>3-D acoustics pb.</td>
</tr>
<tr>
<td>Koutsovasilis/F2</td>
<td>71,505</td>
<td>5,294,285</td>
<td>no</td>
<td>structural pb.</td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th>MATRIX</th>
<th>ICT/ILD/LT</th>
<th>MLR-CG/GMRES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fill  p-t</td>
<td>its  i-t</td>
</tr>
<tr>
<td>Andrews cant</td>
<td>2.6  0.44</td>
<td>32  0.16</td>
</tr>
<tr>
<td>Dubcova2</td>
<td>4.3 2.47</td>
<td>F  19.01</td>
</tr>
<tr>
<td>cfd1</td>
<td>1.4 0.14</td>
<td>42  0.21</td>
</tr>
<tr>
<td>thermal1</td>
<td>2.8 0.56</td>
<td>314  3.42</td>
</tr>
<tr>
<td>thermal2</td>
<td>3.1 0.15</td>
<td>108  0.51</td>
</tr>
<tr>
<td>cfd2</td>
<td>3.6 1.14</td>
<td>F  12.27</td>
</tr>
<tr>
<td></td>
<td>5.3 4.11</td>
<td>148  20.45</td>
</tr>
<tr>
<td>MATRIX</td>
<td>ICT/ILDLT</td>
<td>MLR-CG/GMRES</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td>fill p-t its i-t</td>
<td>k fill p-t its i-t</td>
</tr>
<tr>
<td>vibrobox</td>
<td>3.3 0.19 F 1.06</td>
<td>10 4 3.0 0.45 183 0.22</td>
</tr>
<tr>
<td>qa8fk</td>
<td>1.8 0.58 56 0.60</td>
<td>2 8 1.6 2.33 75 0.36</td>
</tr>
<tr>
<td>F2</td>
<td>2.3 1.37 F 13.94</td>
<td>5 5 2.5 4.17 371 7.29</td>
</tr>
</tbody>
</table>
Conclusion

- Promising approach –

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

SIAM-CSE – March 1, 2013