Preconditioning techniques for highly indefinite linear systems

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

MSI, July 9th, 2008
**Introduction: Linear System Solvers**

- **Direct sparse Solvers**
- **Iterative Methods**
  - Preconditioned Krylov
  - General Purpose
  - Specialized
  - Fast Poisson Solvers
  - Multigrid Methods
Much of recent work on solvers has focused on:

1. Parallel implementation – scalable performance
2. Improving Robustness, developing more general preconditioners
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 Cause:
  more complex models - away from Poisson

- Researchers in iterative methods are borrowing techniques from
  direct methods: → preconditioners

- The inverse is also happening: Direct methods are being adapted
  for use as preconditioners
An overview of recent progress on ILU

- More rigorous dropping strategies [Bollhöfer 2002]
- Vaidya preconditioners – for problems in structures [very successful in industry]
- Support theory for preconditioners
- Use of different forms of LU factorizations [ILUC, N. Li, YS, Chow]
- Most significant: Nonsymmetric permutations
NONSYMMETRIC REORDERINGS
**Enhancing robustness: One-sided permutations**

- **Very useful** techniques for matrices with extremely poor structure.
- Not as helpful in other cases.

*Previous work:*

- Benzi, Haws, Tuma ’99 [compare various permutation algorithms in context of ILU]
- Duff, Koster, ’99 [propose various permutation algorithms. Also discuss preconditioners]
- Duff ’81 [Propose max. transversal algorithms. Basis of many other methods. Also Hopcroft & Karp ’73, Duff ’88]
Transversals - bipartite matching: Find (maximal) set of ordered pairs \((i, j)\) s.t. \(a_{ij} \neq 0\) and \(i\) and \(j\) each appear only once (one diagonal element per row/column). Basis of many algorithms.
Criterion: Find a (column) permutation $\pi$ such that

$$\prod_{i=1}^{n} |a_{i,\pi(i)}| = \max$$

Olchowsky and Neumaier '96 translate this into

$$\min_{\pi} \sum_{i=1}^{n} c_{i,\pi(i)} \quad \text{with} \quad c_{ij} = \begin{cases} \log \left[ \frac{||a_{i,.}||_{\infty}}{|a_{ij}|} \right] & \text{if } a_{ij} \neq 0 \\ +\infty & \text{else} \end{cases}$$

- Dual problem is solved -
- Algorithms utilize depth-first-search to find max transversals.
- Many variants. Best known code: Duff & Koster’s MC64
NONSYMMETRIC REORDERINGS: MULTILEVEL FRAMEWORK
Independent set orderings permute a matrix into the form

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix}$$

where $B$ is a diagonal matrix.

- Unknowns associated with the $B$ block form an independent set (IS).
- IS is maximal if it cannot be augmented by other nodes.
- Finding a maximal independent set is inexpensive.
Main observation: Reduced system obtained by eliminating the unknowns associated with the IS, is still sparse since its coefficient matrix is the Schur complement

\[ S = C - EB^{-1}F \]

- Idea: apply IS set reduction recursively.
- When reduced system small enough solve by any method
- ILUM: ILU factorization based on this strategy. YS ’92-94.

- See work by [Botta-Wubbs ’96, ’97, YS’94, ’96, Leuze ’89,..]
**Main goal:** generalize independent sets to improve robustness

**Main idea:** use “cliques”, or “aggregates”. No coupling between the aggregates.

- Label nodes of independent sets first
Typical shape of reordered matrix:

\[ P A P^T = \begin{pmatrix} B & F \\ E & C \end{pmatrix} \]

Block factorize:

\[ \begin{pmatrix} B & F \\ E & C \end{pmatrix} = \begin{pmatrix} L & 0 \\ EU^{-1} & I \end{pmatrix} \begin{pmatrix} U & L^{-1}F \\ 0 & S \end{pmatrix} \]

\[ S = C - EB^{-1}F = \text{Schur complement + dropping to reduce fill} \]

Next step: treat the Schur complement recursively
Level $l$ Factorization:

$$\left( \begin{array}{cc} B_l & F_l \\ E_l & C_l \end{array} \right) \approx \left( \begin{array}{cc} L_l & 0 \\ E_l U_l^{-1} & I \end{array} \right) \left( \begin{array}{cc} I & 0 \\ 0 & A_{l+1} \end{array} \right) \left( \begin{array}{cc} U_l & L_l^{-1} F_l \\ 0 & I \end{array} \right)$$

- L-solve $\sim$ restriction; U-solve $\sim$ prolongation.
- Perform above block factorization recursively on $A_{l+1}$
- Blocks in $B_l$ treated as sparse. Can be large or small.
- Algorithm is fully recursive
- Stability criterion in block independent sets algorithm
Simple strategy: Level traversal until there are enough points to form a block. Reverse ordering. Start new block from non-visited node. Continue until all points are visited. Add criterion for rejecting “not sufficiently diagonally dominant rows.”
Block size of 20
Two-sided permutations with diag. dominance

**Idea:** ARMS + exploit nonsymmetric permutations

- No particular structure or assumptions for $B$ block
- Permute rows * and * columns of $A$. Use two permutations $P$ (rows) and $Q$ (columns) to transform $A$ into

$$PAQ^T = \begin{pmatrix} B & F \\ E & C \end{pmatrix}$$

$P, Q$ is a pair of permutations (rows, columns) selected so that the $B$ block has the ‘most diagonally dominant’ rows (after nonsym perm) and few nonzero elements (to reduce fill-in).
At the $l$-th level reorder matrix as shown above and then carry out the block factorization ‘approximately’

$$P_l A_l Q_l^T = \begin{pmatrix} B_l & F_l \\ E_l & C_l \end{pmatrix} \approx \begin{pmatrix} L_l & 0 \\ E_l U_l^{-1} & I \end{pmatrix} \times \begin{pmatrix} U_l & L_l^{-1} F_l \\ 0 & A_{l+1} \end{pmatrix},$$

where

$$B_l \approx L_l U_l$$

$$A_{l+1} \approx C_l - (E_l U_l^{-1})(L_l^{-1} F_l).$$

As before the matrices $E_l U_l^{-1}, L_l^{-1} F_l$ or their approximations

$$G_l \approx E_l U_l^{-1}, \quad W_l \approx L_l^{-1} F_l$$

need not be saved.
**Interpretation in terms of complete pivoting**

- **Rationale:** Critical to have an accurate and well-conditioned $B$ block [Bollhöfer, Bollhöfer-YS’04]

- Case when $B$ is of dimension 1 → a form of complete pivoting ILU. Procedure $\sim$ block complete pivoting ILU

- **Matching sets:** define $B$ block. $\mathcal{M}$ is a set of $n_M$ pairs $(p_i, q_i)$ where $n_M \leq n$ with $1 \leq p_i, q_i \leq n$ for $i = 1, \ldots, n_M$ and

  $$p_i \neq p_j, \text{ for } i \neq j \quad q_i \neq q_j, \text{ for } i \neq j$$

- When $n_M = n$ → (full) permutation pair $(P, Q)$. A partial matching set can be easily completed into a full pair $(P, Q)$ by a greedy approach.
Matching - preselection

Algorithm to find permutation consists of 3 phases.

1. **Preselection:** to filter out poor rows (dd. criterion) and sort the selected rows.
2. **Matching:** scan candidate entries in order given by preselection and accept them into the $M$ set, or reject them.
3. **Complete the matching set:** into a complete pair of permutations (greedy algorithm)

- Let $j(i) = \arg\max_j |a_{ij}|$.
- Use the ratio $\gamma_i = \frac{|a_{i,j(i)}|}{\|a_{i,:}\|_1}$ as a measure of diag. domin. of row $i$. 
Matching: Greedy algorithm

- Simple algorithm: scan pairs \((i_k, j_k)\) in the given order.

- If \(i_k\) and \(j_k\) not already assigned, assign them to \(M\).

Matrix after preselection

Matrix after Matching perm.
MATLAB DEMO
‘REAL’ TESTS
**Numerical illustration**

<table>
<thead>
<tr>
<th>Matrix</th>
<th>order</th>
<th>nonzeros</th>
<th>Application (Origin)</th>
</tr>
</thead>
<tbody>
<tr>
<td>barrier2-9</td>
<td>115,625</td>
<td>3,897,557</td>
<td>Device simul. (Schenk)</td>
</tr>
<tr>
<td>matrix_9</td>
<td>103,430</td>
<td>2,121,550</td>
<td>Device simul. (Schenk)</td>
</tr>
<tr>
<td>mat-n_3*</td>
<td>125,329</td>
<td>2,678,750</td>
<td>Device simul. (Schenk)</td>
</tr>
<tr>
<td>ohne2</td>
<td>181,343</td>
<td>11,063,545</td>
<td>Device simul. (Schenk)</td>
</tr>
<tr>
<td>para-4</td>
<td>153,226</td>
<td>5,326,228</td>
<td>Device simul. (Schenk)</td>
</tr>
<tr>
<td>cir2a</td>
<td>482,969</td>
<td>3,912,413</td>
<td>circuit simul.</td>
</tr>
<tr>
<td>scircuit</td>
<td>170998</td>
<td>958936</td>
<td>circuit simul. (Hamm)</td>
</tr>
<tr>
<td>circuit_4</td>
<td>80209</td>
<td>307604</td>
<td>Circuit simul. (Bomhof)</td>
</tr>
<tr>
<td>wang3.rua</td>
<td>26064</td>
<td>177168</td>
<td>Device simul. (Wang)</td>
</tr>
<tr>
<td>wang4.rua</td>
<td>26068</td>
<td>177196</td>
<td>Device simul. (Wang)</td>
</tr>
</tbody>
</table>
## Parameters

<table>
<thead>
<tr>
<th>( nlev_{\text{max}} )</th>
<th>( tol_{DD} )</th>
<th>Drop tolerance</th>
<th>( Fill_{\text{max}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.1</td>
<td>LU-B 0.01 GW 0.01 S 0.01 LU-S 1.e-05</td>
<td>LU-B 3 GW 3 S 3 LU-S 20</td>
</tr>
<tr>
<td>Matrix</td>
<td>Fill Factor</td>
<td>Set-up Time</td>
<td>GMRES(60) Its.</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>barr2-9</td>
<td>0.62</td>
<td>4.01e+00</td>
<td>113</td>
</tr>
<tr>
<td>mat-n_3</td>
<td>0.89</td>
<td>7.53e+00</td>
<td>40</td>
</tr>
<tr>
<td>matrix_9</td>
<td>1.77</td>
<td>5.53e+00</td>
<td>160</td>
</tr>
<tr>
<td>ohne2</td>
<td>0.62</td>
<td>4.34e+01</td>
<td>99</td>
</tr>
<tr>
<td>para-4</td>
<td>0.62</td>
<td>5.70e+00</td>
<td>49</td>
</tr>
<tr>
<td>wang3</td>
<td>2.33</td>
<td>8.90e-01</td>
<td>45</td>
</tr>
<tr>
<td>wang4</td>
<td>1.86</td>
<td>5.10e-01</td>
<td>31</td>
</tr>
<tr>
<td>scircuit</td>
<td>0.90</td>
<td>1.86e+00</td>
<td>Fail</td>
</tr>
<tr>
<td>circuit_4</td>
<td>0.75</td>
<td>1.60e+00</td>
<td>199</td>
</tr>
<tr>
<td>circ2a</td>
<td>0.76</td>
<td>2.19e+02</td>
<td>18</td>
</tr>
</tbody>
</table>

Results for the 10 systems - ARMS-ddPQ + GMRES(60) & GMRES(100)
<table>
<thead>
<tr>
<th>Fill Factor</th>
<th>Set-up Time</th>
<th>GMRES(60)</th>
<th>GMRES(100)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Its.</td>
<td>Time</td>
</tr>
<tr>
<td>Same param’s</td>
<td>0.89</td>
<td>1.81</td>
<td>400</td>
</tr>
<tr>
<td>Droptol = .001</td>
<td>1.00</td>
<td>1.89</td>
<td>98</td>
</tr>
</tbody>
</table>

Solution of the system `scircuit` – no scaling + two different sets of parameters.
Application to the Helmholtz equation


- Problem is set in the open domain $\Omega_e$ of $\mathbb{R}^d$

\[
\begin{align*}
\Delta u + k^2 u &= f \quad \text{in} \quad \Omega \\
u &= -u_{inc} \quad \text{on} \quad \Gamma \\
\text{or} \quad \frac{\partial u}{\partial n} &= -\frac{\partial u_{inc}}{\partial n} \quad \text{on} \quad \Gamma \\
\lim_{r \to \infty} r^{(d-1)/2} \left( \frac{\partial u}{\partial \vec{n}} - iku \right) &= 0 \quad \text{Sommerfeld condition}
\end{align*}
\]

where: $u$ the wave diffracted by $\Gamma$, $f$ = source function = zero outside domain
Issue: non-reflective boundary conditions when making the domain finite.

Artificial boundary $\Gamma_{art}$ added – Need non-absorbing BCs.

For high frequencies, linear systems become very ‘indefinite’ – [eigenvalues on both sides of the imaginary axis]

Not very good for iterative methods
Application to the Helmholtz equation

**Problem 1:**

\[ \begin{cases} 
\Delta u + k^2 u = 0 \text{ in } \Omega_e \\
\frac{\partial u}{\partial n} +iku = g \text{ in } \Gamma_{art}
\end{cases} \]

- Domain: \( \Omega = (0, 1) \times (0, 1) \)

- Function \( g \) selected so that exact solution is \( u(x, y) = \exp[ik \cos(\theta) x + k \sin(\theta) y] \).

- Structured meshes used for the discretization
**Problem 2.** Soft obstacle \( \equiv \text{disk of radius } r_0 = 0.5 \text{m} \). Incident plane wave with a wavelength \( \lambda \); propagates along the \( x \)-axis. 2nd order Bayliss-Turkel boundary conditions used on \( \Gamma_{art} \), located at a distance \( 2r_0 \) from the obstacle. Discretization uses isoparametric elements with 4 nodes. Analytic solution is known.
**Impact of the dropping strategy in ILUT**

**Pb 1. Convergence of ILUT-GMRES for different values of \( l_{fil} \)**

![Graph showing convergence of ILUT-GMRES for different values of \( l_{fil} \)](image_url)
Using a preconditioner from a lower wavenumber

- Good strategy for high frequencies. Test with Problem 2 –
Solution found – (Thanks: R. Kechroud)

Figure 8: Lignes de contour (solution analytique)
Several papers promoted the use of complex shifts [or very similar approaches] for Helmholtz


Illustration with an experiment: finite difference discretization of $-\Delta$ on a $25 \times 20$ grid.

Add a negative shift of $-1$ to resulting matrix.

Do an ILU factorization of $A$ and plot eigs of $L^{-1}AU^{-1}$.

Used LUINC from matlab - no-pivoting and threshold $= 0.1$. 
Terrible spectrum:
Now plot eigs of $L^{-1}AU^{-1}$ where $L, U$ are inc. LU factors of $B = A + 0.25 \times i$

Much better! Observed by many

[PDE viewpoint]

Idea: Add complex shifts in ILUT. Goal: to reinforce diagonal dominance
**Explanation**

**Question:**

What if we do an exact factorization $[\text{droptol} = 0]$?

- $\Lambda(L^{-1}AU^{-1}) = \Lambda[(A + \alpha iI)^{-1}A]$  
- $\Lambda = \{\frac{\lambda_j}{\lambda_j + i\alpha}\}$  
- Located on a circle – with a cluster at one.  
- Figure shows situation on the same example
Recent comparisons

[with : Daniel Osei-Kuffuor]

- Setting: Problem 2. Mesh size fixed to $1/h = 160$. Problem size $n = 28,980$, Number of nonzeros $nnz = 260,280$

- For each preconditioner $lfil = 5 \times nnz/n$

- Wavenumber varied [until convergence fails]
ILUT with $\text{droptol} = 0.02$

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\frac{\lambda}{h}$</th>
<th>No. iters</th>
<th>Setup Time (s)</th>
<th>Iter. Time (s)</th>
<th>Fill Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2\pi$</td>
<td>160</td>
<td>191</td>
<td>0.1</td>
<td>6.03</td>
<td>1.35</td>
</tr>
<tr>
<td>$4\pi$</td>
<td>80</td>
<td>214</td>
<td>0.1</td>
<td>6.86</td>
<td>1.37</td>
</tr>
<tr>
<td>$8\pi$</td>
<td>40</td>
<td>317</td>
<td>0.11</td>
<td>9.67</td>
<td>1.42</td>
</tr>
<tr>
<td>$16\pi$</td>
<td>20</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
</tbody>
</table>
ILUT – with complex shifts – with $droptol = 0.02$

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\frac{\lambda}{h}$</th>
<th>No. iters</th>
<th>Setup Time (s)</th>
<th>Iter. Time (s)</th>
<th>Fill Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2\pi$</td>
<td>160</td>
<td>191</td>
<td>0.1</td>
<td>5.34</td>
<td>1.35</td>
</tr>
<tr>
<td>$4\pi$</td>
<td>80</td>
<td>211</td>
<td>0.1</td>
<td>5.90</td>
<td>1.36</td>
</tr>
<tr>
<td>$8\pi$</td>
<td>40</td>
<td>280</td>
<td>0.11</td>
<td>7.89</td>
<td>1.41</td>
</tr>
<tr>
<td>$16\pi$</td>
<td>20</td>
<td>273</td>
<td>0.11</td>
<td>7.90</td>
<td>1.60</td>
</tr>
<tr>
<td>$32\pi$</td>
<td>10</td>
<td>163</td>
<td>0.18</td>
<td>5.41</td>
<td>2.5</td>
</tr>
<tr>
<td>$64\pi$</td>
<td>5</td>
<td>107</td>
<td>0.33</td>
<td>4.25</td>
<td>3.84</td>
</tr>
<tr>
<td>$k$</td>
<td>$\frac{\lambda}{h}$</td>
<td>No. iters</td>
<td>Setup Time (s)</td>
<td>Iter. Time (s)</td>
<td>Fill Factor</td>
</tr>
<tr>
<td>------</td>
<td>---------------------</td>
<td>-----------</td>
<td>---------------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>$2\pi$</td>
<td>160</td>
<td>180</td>
<td>0.68</td>
<td>9.20</td>
<td>2.07</td>
</tr>
<tr>
<td>$4\pi$</td>
<td>80</td>
<td>224</td>
<td>0.71</td>
<td>11.5</td>
<td>2.09</td>
</tr>
<tr>
<td>$8\pi$</td>
<td>40</td>
<td>261</td>
<td>0.54</td>
<td>11.8</td>
<td>2.17</td>
</tr>
<tr>
<td>$16\pi$</td>
<td>20</td>
<td>127</td>
<td>0.58</td>
<td>5.71</td>
<td>2.39</td>
</tr>
<tr>
<td>$32\pi$</td>
<td>10</td>
<td>187</td>
<td>0.69</td>
<td>8.61</td>
<td>3.15</td>
</tr>
<tr>
<td>$64\pi$</td>
<td>5</td>
<td>231</td>
<td>0.39</td>
<td>8.89</td>
<td>3.50</td>
</tr>
</tbody>
</table>
Distributed Sparse Systems: Simple illustration

- Naive partitioning of equations -
- Does not work well in practice (performance)
Best idea is to use the adjacency graph of $A$:

Vertices = $\{1, 2, \cdots, n\}$;
Edges: $i \rightarrow j$ iff $a_{ij} \neq 0$

Graph partitioning problem:

- Want a partition of the vertices of the graph so that

  (1) partitions have $\sim$ the same sizes

  (2) interfaces are small in size
General Partitioning of a sparse linear system

$S_1 = \{1, 2, 6, 7, 11, 12\}$: This means equations and unknowns 1, 2, 3, 6, 7, 11, 12 are assigned to Domain 1.

$S_2 = \{3, 4, 5, 8, 9, 10, 13\}$

$S_3 = \{16, 17, 18, 21, 22, 23\}$

$S_4 = \{14, 15, 19, 20, 24, 25\}$

Partitioners: Metis, Chaco, Scotch, ...

More recent: Zoltan, H-Metis, PaToH
Standard dual objective: “minimize” communication + “balance” partition sizes

Recent trend: use of hypergraphs [PaToh, Hmetis,...]
A distributed sparse system

In each domain [Local interface variables ordered last]:

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

- \(u_i\) : Internal variables; \(y_i\) : Interface variables
Order all interior variables first

\[
\begin{pmatrix}
B_1 \\
B_2 \\
\vdots \\
\vdots \\
B_p
\end{pmatrix}
\begin{pmatrix}
F_1 \\
F_2 \\
\vdots \\
\vdots \\
F_p
\end{pmatrix}
= 
\begin{pmatrix}
u_1 \\
u_2 \\
\vdots \\
u_p
\end{pmatrix}
= 
\begin{pmatrix}
f_1 \\
f_2 \\
\vdots \\
g_p
\end{pmatrix}
\]

← Interior variables → Interface variables
Parallel implementation

- Preliminary work – with Zhongze Li
- Ideally would use hypergraph partitioning [in the plans]
- We used only a local version of ddPQ
- Schur complement version not yet available
- In words: Construct the local matrix, extend it with overlapping data and use ddPQ ordering on it.
- Can be used with Standard Schwarz procedures – or with restrictive version [RAS]
Restricted Additive Schwarz Preconditioner (RAS)

Domain 1

Domain 2

Domain 1 local matrix

Extended Domain 1

Domain 2

Domain 1 local matrix

Univ. Lyon-1, 03/25/08
RAS + ddPQ uses arms-ddPQ on extended matrix - for each domain.

ddPQ Improves robustness enormously in spite of simple (local) implementation.

Test with problem from MHD problem.
Example: a system from a MHD simulation

- Source of problem: Coupling of Maxwell equations with Navier-Stokes.

- Matrices come from solution of Maxwell’s equation:

\[
\frac{\partial B}{\partial t} - \nabla \times (u \times B) - \frac{1}{Re_m} \nabla \times (\nabla \times B) + \nabla q = 0
\]

\[
\nabla \cdot B = 0 ,
\]

- See [Ben-Salah, Soulaimani, Habashi, Fortin, IJNMF 1999]

- Cylindrical domain, tetrahedra used.

- Not an easy problem for iterative methods.
<table>
<thead>
<tr>
<th>np</th>
<th>its</th>
<th>$t_{set}$</th>
<th>$t_{it}$</th>
<th>np</th>
<th>its</th>
<th>$t_{set}$</th>
<th>$t_{it}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>107</td>
<td>236.58</td>
<td>320.74</td>
<td>1</td>
<td>60</td>
<td>204.06</td>
<td>187.05</td>
</tr>
<tr>
<td>2</td>
<td>118</td>
<td>136.28</td>
<td>232.78</td>
<td>2</td>
<td>104</td>
<td>108.45</td>
<td>162.34</td>
</tr>
<tr>
<td>4</td>
<td>354</td>
<td>72.66</td>
<td>326.03</td>
<td>4</td>
<td>109</td>
<td>60.24</td>
<td>86.25</td>
</tr>
<tr>
<td>8</td>
<td>2640</td>
<td>40.06</td>
<td>1303.16</td>
<td>8</td>
<td>119</td>
<td>41.56</td>
<td>52.11</td>
</tr>
<tr>
<td>16</td>
<td>3994</td>
<td>21.87</td>
<td>1029.88</td>
<td>16</td>
<td>418</td>
<td>22.84</td>
<td>97.88</td>
</tr>
<tr>
<td>32</td>
<td>&gt;10,000</td>
<td>–</td>
<td>–</td>
<td>32</td>
<td>537</td>
<td>12.34</td>
<td>65.77</td>
</tr>
</tbody>
</table>

- Simple Schwarz (RAS) : very poor performance
- severe deterioration of performance with higher $np$
Conclusion

- ARMS+DDpq works well as a “general-purpose” solver.
- Far from being a 100% robust iterative solver ...
- Recent work on generalizing nonsymmetric permutations to symmetric matrices [Duff-Pralet, 2006].
- As a general rule: ILU-based preconditioners are not meant to replace tailored preconditioners – but they can be used as general purpose tools as parts of other techniques.
A x = b
-\Delta u = f + bc

What is missing from this picture?
1. Intermediate methods which lie in between general purpose and specialized – exploit some information from origin of the problem.

2. Considerations related to parallelism. Development of ‘robust’ solvers remains limited to serial algorithms in general.

Problem: parallel implementations of iterative methods are less effective than their serial counterparts.
Software:

- ARMS-C [C-code] - available from ITSOL package.
  
  http://www.cs.umn.edu/~saad/software

- More comprehensive package: ILUPACK – developed mainly by Matthias Bollhoefer and his team
  
  http://www.tu-berlin.de/ilupack/.