Sparse direct linear solvers
Woudschoten conference on
Parallel numerical linear algebra
6-7 Octobre 2010

Patrick Amestoy
INPT-IRIT (University of Toulouse)
http://amestoy.perso.enseeiht.fr/

in collaboration with members of MUMPS team
A. Buttari, A. Guermouche, J.Y. L’Excellent, B. Ucar, and F.H. Rouet
http://mumps.enseeiht.fr or
http://graal.ens-lyon.fr/MUMPS/
Outline

Context and motivations

(Pre)Processing sparse matrices for efficiency and accuracy
  Fill-in and reordering
  Numerical threshold pivoting
  Preprocessing unsymmetric matrices
  Preprocessing symmetric matrices

Approaches for parallel factorization
  Elimination trees
  Distributed memory sparse solvers
  Some parallel solvers
  Case study: comparison of MUMPS and SuperLU

Conclusion (Part I)
Sparse direct linear solvers (I)

Woudschoten conference 2010
Outline

Context and motivations
A selection of references

> **Books**

> **Articles**

> **Lecture Notes**
> 1. P. Amestoy and J.Y. L’Excellent, Lecture notes on Linear algebra and sparse direct methods, UNESCO (Tunis), Master lectures (ENS-Lyon and INPT-ENSEEIHT)
Motivations

- solution of linear systems of equations $\rightarrow$ key algorithmic kernel

\[\text{Continuous problem}\]
\[\downarrow\]
\[\text{Discretization}\]
\[\downarrow\]
\[\text{Solution of a linear system } Ax = b\]

- Main parameters:
  - Numerical properties of the linear system (symmetry, pos. definite, conditioning, ...)
  - Size and structure:
    - Large ($> 10^7 \times 10^7$), square/rectangular
    - Dense or sparse (structured / unstructured)
    - Target computer (sequential/parallel/multicore/Cell/GPU)
Exemple of sparse matrices

Matrix from CFD (Univ. Tel Aviv)

Chemical proc. Simulation Ihr01
Matrix factorizations

Solution of $Ax = b$

- **$A$ is unsymmetric:**
  - $A$ is factorized as: $A = LU$, where $L$ is a lower triangular matrix, and $U$ is an upper triangular matrix.
  - Forward-backward substitution: $Ly = b$ then $Ux = y$

- **$A$ is symmetric:**
  - $A = LDL^T$ or $LL^T$

- **$A$ is rectangular $m \times n$ with $m \geq n$ and $\min_x \|Ax - b\|_2$:**
  - $A = QR$ where $Q$ is orthogonal ($Q^{-1} = Q^T$) and $R$ is triangular.
  - Solve: $y = Q^Tb$ then $Rx = y$
Example in structural mechanics

BMW car body,
227,362 unknowns,
5,757,996 nonzeros,
MSC.Software

Size of factors: 51.1 million entries
Number of operations: $44.9 \times 10^9$
Solve $Ax = b$, $A$ sparse

Resolution with a 3 phase approach

- Analysis phase
  - preprocess the matrix
  - prepare factorization
- Factorization phase
  - symmetric positive definite $\rightarrow LL^T$
  - symmetric indefinite $\rightarrow LDL^T$
  - unsymmetric $\rightarrow LU$
- Solution phase exploiting factored matrices.
  - Postprocessing of the solution (iterative refinements and backward error analysis).
Sparse solver: only a black box?

Default (often automatic/adaptive) setting of the options is often available; however, a better knowledge of the options can help the user to further improve its solution.

- Preprocessing may influence:
  - Operation cost and/or computational time
  - Size of factors and/or memory needed
  - Reliability of our estimations
  - Numerical accuracy.

- Describe preprocessing options and functionalities that are most critical to both performance and accuracy.
Ax = b?

- Symmetric permutations to control increase in the size of the factors: \( Ax = b \rightarrow PAP^tPx = b \)
- Numerical pivoting to preserve accuracy.
- Unsymmetric matrices (\( A = LU \))
  - numerical equilibration (scaling rows/columns)
  - set large entries on the diagonal
  - modified problem: \( A'x' = b' \) with \( A' = P_nD_rPAQP^tD_c \)
- Symmetric matrices (\( A = LDL^t \)):
  Algorithms must also preserve symmetry (flops/memory divided by 2)
  - adapt equilibration and set large entries "on" diagonal while preserving symmetry
  - modified problem: \( A' = P_ND_sPQ^tAQP^tD_sP_N^t \)
- Preprocessing for parallelism (influence of task mapping on the performance)
Preprocessing - illustration

Original ($A = lhr01$)  Preprocessed matrix ($A'(lhr01)$)
Outline

(Pre)Processing sparse matrices for efficiency and accuracy
- Fill-in and reordering
- Numerical threshold pivoting
- Preprocessing unsymmetric matrices
- Preprocessing symmetric matrices
(Pre)Processing sparse matrices for efficiency and accuracy
Fill-in and reordering
Numerical threshold pivoting
Preprocessing unsymmetric matrices
Preprocessing symmetric matrices
Fill-in and reordering

Step $k$ of LU factorization ($a_{kk}$ pivot):

- For $i > k$ compute $l_{ik} = a_{ik} / a_{kk}$ ($= a_{ik}'$),
- For $i > k, j > k$

$$a_{ij}' = a_{ij} - \frac{a_{ik} \times a_{kj}}{a_{kk}} = a_{ij} - l_{ik} \times a_{kj}$$

- If $a_{ik} \neq 0$ and $a_{kj} \neq 0$ then $a_{ij}' \neq 0$
- If $a_{ij}$ was zero $\rightarrow$ non-zero $a_{ij}'$ must be stored: fill-in

Interest of permuting a matrix:

$$
\begin{pmatrix}
X & X & X & X & X
\end{pmatrix}
\quad
\begin{pmatrix}
X & 0 & 0 & 0 & 0 & X
\end{pmatrix}
\quad
\begin{pmatrix}
X & 0 & 0 & 0 & 0 & X
\end{pmatrix}
\quad
\begin{pmatrix}
X & X & 0 & 0 & 0 & X
\end{pmatrix}
\quad
\begin{pmatrix}
X & 0 & X & 0 & 0 & X
\end{pmatrix}
\quad
\begin{pmatrix}
X & 0 & X & 0 & 0 & X
\end{pmatrix}
\quad
\begin{pmatrix}
X & 0 & 0 & X & 0 & X
\end{pmatrix}
\quad
\begin{pmatrix}
X & 0 & 0 & 0 & X & X
\end{pmatrix}
\quad
\begin{pmatrix}
X & X & X & X & X & X
\end{pmatrix}
$$
Assumptions: \( A \) symmetric and pivots are chosen on the diagonal

Structure of \( A \) symmetric represented by the graph \( G = (V, E) \)

- Vertices are associated to columns: \( V = \{1, \ldots, n\} \)
- Edges \( E \) are defined by: \( (i, j) \in E \Leftrightarrow a_{ij} \neq 0 \)
- \( G \) undirected (symmetry of \( A \))
Symmetric matrices and graphs

Remarks:

- Number of nonzeros in column $j = |\text{adj}_G(j)|$
- Symmetric permutation $\equiv$ renumbering the graph

Symmetric matrix

Corresponding graph

\begin{figure}
\centering
\begin{tikzpicture}
\node at (0,0) {
\begin{tabular}{cccccc}
1 & 2 & 3 & 4 & 5 \\
1 & x & x & x & x \\
2 & x & x & x & x \\
3 & x & x & x & x \\
4 & x & x & x & x \\
5 & x & x & x & x \\
\end{tabular}
};
\node at (5,0) {
\begin{tikzpicture}
\node (1) at (0,0) [circle, draw] {1};
\node (2) at (1,-1) [circle, draw] {2};
\node (3) at (2,-1) [circle, draw] {3};
\node (4) at (1,-2) [circle, draw] {4};
\node (5) at (2,-2) [circle, draw] {5};
\path (1) edge (2);
\path (1) edge (3);
\path (2) edge (4);
\path (3) edge (5);
\end{tikzpicture}
};
\end{tikzpicture}
\end{figure}
The elimination graph model for symmetric matrices

Let $A$ be a symmetric positive definite matrix of order $n$.

The $LL^T$ factorization can be described by the equation:

$$A = A_0 = H_0 = \begin{pmatrix} d_1 & v_1^T \\ v_1 & H_1 \end{pmatrix}$$

$$= \begin{pmatrix} \sqrt{d_1} & 0 \\ \frac{v_1}{\sqrt{d_1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & H_1 \end{pmatrix} \begin{pmatrix} \sqrt{d_1} & 0 \\ \frac{v_1^T}{\sqrt{d_1}} & I_{n-1} \end{pmatrix}$$

$$= L_1 A_1 L_1^T$$, where

$$H_1 = H_1 - \frac{v_1 v_1^T}{d_1}$$

The basic step is applied on $H_1 H_2 \cdots$ to obtain:

$$A = (L_1 L_2 \cdots L_{n-1}) I_n \left( L_{n-1}^T \cdots L_2^T L_1^T \right) = LL^T$$
The basic step: $H_1 = \overline{H_1} - \frac{v_1v_1^T}{d_1}$

What is $v_1v_1^T$ in terms of structure?

$v_1$ is a column of $A$, hence the neighbors of the corresponding vertex.

$v_1v_1^T$ results in a dense sub-block in $H_1$.

If any of the nonzeros in dense submatrix are not in $A$, then we have fill-ins.
The elimination process in the graphs

\[
G_U(V, E) \leftarrow \text{undirected graph of } A
\]

for \( k = 1 : n - 1 \) do

\[
V \leftarrow V - \{k\} \quad \{\text{remove vertex } k\}
\]

\[
E \leftarrow E - \{(k, \ell) : \ell \in \text{adj}(k)\} \cup \{(x, y) : x \in \text{adj}(k) \text{ and } y \in \text{adj}(k)\}
\]

\[
G_k \leftarrow (V, E) \quad \{\text{for definition}\}
\]

end for

\( G_k \) are the so-called elimination graphs (Parter,'61).

\[
G_0 : \quad \begin{array}{ccc}
1 & 2 & 3 \\
\text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} \\
\end{array}
\]

\[
H_0 = \begin{bmatrix}
1 & \times & \times \\
\times & 2 & \times & \times \\
\times & \times & 3 & \times \\
\times & \times & \times & 4 \\
\times & \times & 5 & \times \\
\times & \times & \times & 6 \\
\end{bmatrix}
\]
A sequence of elimination graphs

\begin{align*}
G_0 : & \quad \begin{array}{c}
 1 & 2 & 3 \\
 4 & 6 & 5
\end{array} \\
H_0 = & \begin{bmatrix}
1 \times & \times & \times \\
\times & 2 \times & \times \\
\times & \times & 3 \times \\
\times & \times & \times 4 \\
\times & \times & \times 5 \times \\
\times & \times & \times 6
\end{bmatrix} \\
G_1 : & \quad \begin{array}{c}
 2 & 3 \\
 4 & 6 & 5
\end{array} \\
H_1 = & \begin{bmatrix}
2 \times & \times & + \\
\times & 3 \times & \times \\
\times & \times & 4 \times \\
\times & \times & \times 5 \times \\
+ & \times & \times 6
\end{bmatrix} \\
G_2 : & \quad \begin{array}{c}
 4 & 3 \\
 6 & 5
\end{array} \\
H_2 = & \begin{bmatrix}
3 + & \times & + \\
+ & 4 + & \times \\
+ & \times & 5 \times \\
+ & \times & \times 6
\end{bmatrix} \\
G_3 : & \quad \begin{array}{c}
 4 \\
 6 & 5
\end{array} \\
H_3 = & \begin{bmatrix}
4 + & + & \\
+ & 5 \times & + \\
+ & \times & 6
\end{bmatrix}
\end{align*}
Fill-in and reordering

“Before permutation"  

(A”(lhr01))

Permuted matrix  

(A′(lhr01))

Factored matrix  

(LU(A′))
Fill-in characterization

Let $A$ be a symmetric matrix ($G(A)$ its associated graph), $L$ the matrix of factors $A = LL^t$;

**Fill path theorem, Rose, Tarjan, Leuker, 76**

$l_{ij} \neq 0$ iff there is a path in $G(A)$ between $i$ and $j$ such that all nodes in the path have indices smaller than both $i$ and $j$. 
Fill-in characterization (proof intuition)

Let $A$ be a symmetric matrix ($G(A)$ its associated graph), $L$ the matrix of factors $A = LL^t$;

**Fill path theorem, Rose, Tarjan, Leuker, 76**

$l_{ij} \neq 0$ iff there is a path in $G(A)$ between $i$ and $j$ such that all nodes in the path have indices smaller than both $i$ and $j$. 
Fill-in characterization (proof intuition)

Let $A$ be a symmetric matrix ($G(A)$ its associated graph), $L$ the matrix of factors $A = LL^t$;

**Fill path theorem, Rose, Tarjan, Leuker, 76**

$l_{ij} \neq 0$ iff there is a path in $G(A)$ between $i$ and $j$ such that all nodes in the path have indices smaller than both $i$ and $j$. 
Fill-in characterization (proof intuition)

Let $A$ be a symmetric matrix ($G(A)$ its associated graph), $L$ the matrix of factors $A = LL^t$.

**Fill path theorem, Rose, Tarjan, Leuker, 76**

$l_{ij} \neq 0$ iff there is a path in $G(A)$ between $i$ and $j$ such that all nodes in the path have indices smaller than both $i$ and $j$. 

---

[Diagram showing fill-in process with nodes and paths labeled p1, p2, pk and i, j]
Three main classes of methods for minimizing fill-in during factorization

- Global approach: The matrix is permuted into a matrix with a given pattern
  - Fill-in is restricted to occur within that structure
  - Cuthill-McKee (block tridiagonal matrix)
  - Nested dissections (“block bordered” matrix)
    (Remark: interpretation using the fill-path theorem)

Graph partitioning

Permutated matrix
Fill-reducing heuristics

- Local heuristics: At each step of the factorization, selection of the pivot that is likely to minimize fill-in.
  - Method is characterized by the way pivots are selected.
  - Markowitz criterion (for a general matrix).
  - Minimum degree or Minimum fill-in (for symmetric matrices).

- Hybrid approaches: Once the matrix is permuted to block structure, local heuristics are used within the blocks.
Local heuristics to reduce fill-in during factorization

Let $G(A)$ be the graph associated to a matrix $A$ that we want to order using local heuristics.

Let $\text{Metric}$ such that $\text{Metric}(v_i) < \text{Metric}(v_j)$ implies $v_i$ is a better than $v_j$.

Generic algorithm
Loop until all nodes are selected
   Step1 : select current node $p$ (so called pivot) with minimum metric value,
   Step2 : update elimination graph,
   Step3 : update $\text{Metric}(v_j)$ for all non-selected nodes $v_j$.

*Step3 should only be applied to nodes for which the Metric value might have changed.*
Reordering unsymmetric matrices: Markowitz criterion

- At step $k$ of Gaussian elimination:

  \[ \begin{array}{c}
  r_i^k = \text{number of non-zeros in row } i \text{ of } A^k \\
  c_j^k = \text{number of non-zeros in column } j \text{ of } A^k \\
  a_{ij} \text{ must be large enough and should minimize } (r_i^k - 1) \times (c_j^k - 1) \quad \forall i, j > k
  \end{array} \]

- Minimum degree: Markowitz criterion for symmetric diagonally dominant matrices
Minimum fill based algorithm

- **Metric**($v_i$) is the amount of fill-in that $v_i$ would introduce if it were selected as a pivot.

- Illustration: $r$ has a degree $d = 4$ and a fill-in metric of $d \times (d - 1)/2 = 6$ whereas $s$ has degree $d = 5$ but a fill-in metric of $d \times (d - 1)/2 - 9 = 1$.
Minimum fill-in properties

- The situation typically occurs when \( \{i_1, i_2, i_3\} \) and \( \{i_2, i_3, i_4, i_5\} \) were adjacent to two already selected nodes (here \( e_2 \) and \( e_1 \)).

- The elimination of a node \( v_k \) affects the degree of nodes adjacent to \( v_k \). The fill-in metric of \( \text{Adj}(\text{Adj}(v_k)) \) is also affected.

- Illustration: selecting \( r \) affects the fill-in of \( i_1 \) (fill edge \( (j_3, j_4) \) should be deduced).
Impact of fill-reducing heuristics

<table>
<thead>
<tr>
<th></th>
<th>METIS</th>
<th>SCOTCH</th>
<th>PORD</th>
<th>AMF</th>
<th>AMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>gupta2</td>
<td>2757.8</td>
<td>4510.7</td>
<td>4993.3</td>
<td>2790.3</td>
<td>2663.9</td>
</tr>
<tr>
<td>ship_003</td>
<td>83828.2</td>
<td>92614.0</td>
<td>112519.6</td>
<td>96445.2</td>
<td>155725.5</td>
</tr>
<tr>
<td>twotone</td>
<td>29120.3</td>
<td>27764.7</td>
<td>37167.4</td>
<td>29847.5</td>
<td>29552.9</td>
</tr>
<tr>
<td>wang3</td>
<td>4313.1</td>
<td>5801.7</td>
<td>5009.9</td>
<td>6318.0</td>
<td>10492.2</td>
</tr>
<tr>
<td>xenon2</td>
<td>99273.1</td>
<td>112213.4</td>
<td>126349.7</td>
<td>237451.3</td>
<td>298363.5</td>
</tr>
</tbody>
</table>

- **METIS** (Karypis and Kumar) and **SCOTCH** (Pellegrini) are global strategies (recursive nested dissection based orderings).
- **PORD** (Schulze, Paderborn Univ.) recursive dissection based on a bottom up strategy to build the separator.
- **AMD** (Amestoy, Davis and Duff) is a local strategy based on Approximate Minimum Degree.
- **AMF** (Amestoy) is a local strategy based on Approx. Minimum Fill.
Impact of fill-reducing heuristics

### Time for factorization (seconds)

<table>
<thead>
<tr>
<th></th>
<th>1p</th>
<th>16p</th>
<th>32p</th>
<th>64p</th>
<th>128p</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>conesh1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>METIS</td>
<td>970</td>
<td>60</td>
<td>41</td>
<td>27</td>
<td>14</td>
</tr>
<tr>
<td>PORD</td>
<td>1264</td>
<td>104</td>
<td>67</td>
<td>41</td>
<td>26</td>
</tr>
<tr>
<td><strong>audi</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>METIS</td>
<td>2640</td>
<td>198</td>
<td>108</td>
<td>70</td>
<td>42</td>
</tr>
<tr>
<td>PORD</td>
<td>1599</td>
<td>186</td>
<td>146</td>
<td>83</td>
<td>54</td>
</tr>
</tbody>
</table>

### Matrices with quasi dense rows:
Impact on the analysis time (seconds) of gupta2 matrix

<table>
<thead>
<tr>
<th></th>
<th>AMD</th>
<th>METIS</th>
<th>QAMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>361</td>
<td>52</td>
<td>23</td>
</tr>
<tr>
<td>Total</td>
<td>379</td>
<td>76</td>
<td>59</td>
</tr>
</tbody>
</table>

- **QAMD** (Amestoy) Approximate Minimum Degree (local) strategy designed for matrices with *quasi dense rows*. 
(Pre)Processing sparse matrices for efficiency and accuracy

Fill-in and reordering

Numerical threshold pivoting

Preprocessing unsymmetric matrices

Preprocessing symmetric matrices
Numerical pivoting during LU factorization

Let \( A = \begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{1}{\epsilon} & 1 \end{bmatrix} \times \begin{bmatrix} \epsilon & 1 \\ 0 & 1 - \frac{1}{\epsilon} \end{bmatrix} \)

\( \kappa_2(A) = 1 + O(\epsilon). \)

If we solve:

\[
\begin{bmatrix}
\epsilon & 1 \\
1 & 1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
= \begin{bmatrix}
1 + \epsilon \\
2
\end{bmatrix}
\]

Exact solution: \( x^* = (1, 1). \)

<table>
<thead>
<tr>
<th>( \epsilon )</th>
<th>( \frac{|x^* - x|}{|x^*|} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 10^{-3} )</td>
<td>( 6 \times 10^{-6} )</td>
</tr>
<tr>
<td>( 10^{-9} )</td>
<td>( 9 \times 10^{-8} )</td>
</tr>
<tr>
<td>( 10^{-15} )</td>
<td>( 7 \times 10^{-2} )</td>
</tr>
</tbody>
</table>

Table: Relative error as a function of \( \epsilon \).
Even if $A$ well-conditioned then Gaussian elimination might introduce errors.

Explanation: pivot $\epsilon$ is too small (relative)

Solution: interchange rows 1 and 2 of $A$.

$$\begin{bmatrix} 1 & 1 \\ \epsilon & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 + \epsilon \end{bmatrix}$$

→ No more error.
Threshold pivoting for sparse matrices

- **Sparse LU factorization**
  - **Threshold u** : Set of eligible pivots = \( \{ r \mid |a_{rk}^{(k)}| \geq u \times \max_i |a_{ik}^{(k)}| \} \), where \( 0 < u \leq 1 \).
  - Among eligible pivots select one preserving sparsity.

- **Sparse LDL^T factorization**
  - Symmetric indefinite case : requires 2 by 2 pivots, e.g. \( \begin{pmatrix} \epsilon & x \\ x & \epsilon \end{pmatrix} \)
  - 2\times2 pivot \( P = \begin{pmatrix} a_{kk} & a_{kl} \\ a_{lk} & a_{ll} \end{pmatrix} : \)
    \[
    |P^{-1}| \begin{pmatrix} \max_i |a_{ki}| \\ \max_j |a_{lj}| \end{pmatrix} \leq \begin{pmatrix} 1/u \\ 1/u \end{pmatrix}
    \]

- **Static pivoting** : Add small perturbations to the matrix of factors to reduce the amount of numerical pivoting.
Threshold pivoting for sparse matrices

**Sparse LU factorization**
- **Threshold \( u \)**: Set of eligible pivots = \( \{ r \mid |a^{(k)}_{rk}| \geq u \times \max_i |a^{(k)}_{ik}| \} \), where \( 0 < u \leq 1 \).
- Among eligible pivots select one preserving sparsity.

**Sparse LDL^T factorization**
- Symmetric indefinite case: requires 2 by 2 pivots, e.g.\[
\begin{pmatrix}
\epsilon & x \\
x & \epsilon
\end{pmatrix}
\]
- 2 × 2 pivot \( P = \begin{pmatrix} a_{kk} & a_{kl} \\ a_{lk} & a_{ll} \end{pmatrix} \):\[
|P^{-1}| \begin{pmatrix} \max_i |a_{ki}| \\ \max_j |a_{lj}| \end{pmatrix} \ \& \ \begin{pmatrix} 1/u \\ 1/u \end{pmatrix}
\]

**Static pivoting**: Add small perturbations to the matrix of factors to reduce the amount of numerical pivoting.
Threshold pivoting for sparse matrices

- **Sparse LU factorization**
  - **Threshold** $u$ : Set of eligible pivots = 
    \[
    \{ r \mid |a_{rk}^{(k)}| \geq u \times \max_i |a_{ik}^{(k)}| \}, \text{ where } 0 < u \leq 1.
    \]
  - Among eligible pivots select one preserving sparsity.

- **Sparse LDL^T factorization**
  - Symmetric indefinite case : requires 2 by 2 pivots, e.g. 
    \[
    \begin{pmatrix}
    \epsilon & x \\
    x & \epsilon
    \end{pmatrix}
    \]
  - 2×2 pivot $P = \begin{pmatrix} a_{kk} & a_{kl} \\ a_{lk} & a_{ll} \end{pmatrix}$:
    \[
    |P^{-1}| \left( \begin{pmatrix} \max_i |a_{ki}| \\ \max_j |a_{lj}| \end{pmatrix} \right) \ll \left( \begin{pmatrix} 1/u \\ 1/u \end{pmatrix} \right)
    \]

- **Static pivoting** : Add small perturbations to the matrix of factors to reduce the amount of numerical pivoting.
(Pre)Processing sparse matrices for efficiency and accuracy

- Fill-in and reordering
- Numerical threshold pivoting
- Preprocessing unsymmetric matrices
- Preprocessing symmetric matrices
Preprocessing unsymmetric matrices - scaling

- **Objective**: Matrix equilibration to help threshold pivoting.
- **Row and column scaling**: \( B = D_r A D_c \) where \( D_r, D_c \) are diagonal matrices to respectively scale rows and columns of \( A \)
  - reduce the amount of numerical problems

Let \( A = \begin{bmatrix} 1 & 2 \\ 10^{16} & 10^{16} \end{bmatrix} \) → Let \( B = D_r A = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \)
  - better detect real problems.

Let \( A = \begin{bmatrix} 1 & 10^{16} \\ 1 & 1 \end{bmatrix} \) → Let \( B = D_r A = \begin{bmatrix} 10^{-16} & 1 \\ 1 & 1 \end{bmatrix} \)
  - Influence quality of **fill-in estimations and accuracy**.
  - Should be activated when the number of uneliminated variables is large.
Objective: Set large entries on the diagonal

- Unsymmetric permutation and scaling
- Preprocessed matrix $B = D_1 A Q D_2$
  is such that $|b_{ii}| = 1$ and $|b_{ij}| \leq 1$

Original ($A = lh01$)  

Permuted ($A' = AQ$)
Combine maximum transversal and fill-in reduction

- Consider the **LU** factorization \( A = LU \) of an unsymmetric matrix.
- Compute the column permutation \( Q \) leading to a maximum numerical transversal of \( A \). \( AQ \) has large (in some sense) numerical entries on the diagonal.
- Find best ordering of \( AQ \) preserving the diagonal entries. Equivalent to finding symmetric permutation \( P \) such that the factorization of \( PAQP^T \) has reduced fill-in.
Preprocessing - Maximum weighted matching

- Influence of maximum weighted matching (Duff and Koster (99,01) on the performance

| Matrix    | Symmetry | \(|LU|\) (10^6) | Flops (10^9) | Backwd Error |
|-----------|----------|----------------|--------------|--------------|
| twotone   | OFF      | 28             | 235          | 1221         |
| ON        |          | 43             | 22           | 29           |
| fidapm11  | OFF      | 100            | 16           | 10           |
| ON        |          | 46             | 28           | 29           |

- On very unsymmetric matrices: reduce flops, factor size and memory used.

- In general improve accuracy, and reduce number of iterative refinements.

- Improve reliability of memory estimates.
Preprocessing - Maximum weighted matching

- **Influence of maximum weighted matching** (Duff and Koster (99,01) on the performance

| Matrix      | Symmetry | $|LU|$ $(10^6)$ | Flops $(10^9)$ | Backwd Error |
|-------------|----------|--------------|---------------|--------------|
| twotone OFF | 28       | 235          | 1221          | 10$^{-6}$    |
| twotone ON  | 43       | 22           | 29            | 10$^{-12}$   |
| fidapm11 OFF| 100      | 16           | 10            | 10$^{-10}$   |
| fidapm11 ON | 46       | 28           | 29            | 10$^{-11}$   |

- On very unsymmetric matrices: reduce flops, factor size and memory used.
- In general **improve accuracy**, and reduce number of iterative refinements.
- **Improve reliability** of memory estimates.
Preprocessing - Maximum weighted matching

- **Influence of maximum weighted matching** (Duff and Koster (99,01) on the performance

| Matrix    | Symmetry | $|LU|$ (10$^6$) | Flops (10$^9$) | Backwd Error |
|-----------|----------|---------------|----------------|---------------|
| twotone   | OFF      | 28            | 235            | 1221          | $10^{-6}$    |
|           | ON       | 43            | 22             | 29            | $10^{-12}$   |
| fidapm11  | OFF      | 100           | 16             | 10            | $10^{-10}$   |
|           | ON       | 46            | 28             | 29            | $10^{-11}$   |

- On very unsymmetric matrices: *reduce flops, factor size and memory used.*

- In general *improve accuracy*, and reduce number of iterative refinements.

- *Improve reliability* of memory estimates.
(Pre)Processing sparse matrices for efficiency and accuracy
  Fill-in and reordering
  Numerical threshold pivoting
  Preprocessing unsymmetric matrices
  Preprocessing symmetric matrices
Preprocessing symmetric matrices (Duff and Pralet (2004, 2005))

- **Symmetric scaling**: Adapt MC64 (Duff and Koster, 2001)
  
  unsymmetric scaling:
  
  let \( D = \sqrt{D_r D_c} \), then \( B = DAD \) is a symmetrically scaled matrix which satisfies
  
  \[
  \forall i, |b_{i\sigma(i)}| = \|b_{\sigma(i)}\|_\infty = \|b_i\|_\infty = 1
  \]
  
  where \( \sigma \) is the permutation from the unsym. transv. algo.

- Influence of scaling on augmented matrices \( K = \begin{pmatrix} H & A \\ A^T & 0 \end{pmatrix} \)

<table>
<thead>
<tr>
<th>Scaling</th>
<th>Total time (seconds)</th>
<th>Nb of entries in factors (millions)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>cont-300</td>
<td>45</td>
<td>5</td>
</tr>
<tr>
<td>cvxqp3</td>
<td>1816</td>
<td>28</td>
</tr>
<tr>
<td>stokes128</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
Preprocessing - Compressed ordering

- Perform an unsymmetric weighted matching

Matched entry
Preprocessing - Compressed ordering

- Perform an unsymmetric weighted matching
- Select matched entries
Preprocessing - Compressed ordering

- Perform an unsymmetric weighted matching
- Select matched entries
- Symmetrically permute matrix to set large entries near diagonal

\[
\begin{bmatrix}
\times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times
\end{bmatrix}
\]

Selected entries
\[\text{Permute } B = Q^T A Q\]
Preprocessing - Compressed ordering

- Perform an unsymmetric weighted matching
- Select matched entries
- Symmetrically permute matrix to set large entries near diagonal
- Compression: $2 \times 2$ diagonal blocks become supervariables.
Preprocessing - Compressed ordering

- Perform an unsymmetric weighted matching
- Select matched entries
- Symmetrically permute matrix to set large entries near diagonal
- Compression: $2 \times 2$ diagonal blocks become supervariables.

Influence of using a compressed graph (with scaling)

<table>
<thead>
<tr>
<th>Compression :</th>
<th>Total time (seconds)</th>
<th>Nb of entries in factors in Millions (estimated)</th>
<th>Nb of entries in factors in Millions (effective)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>cont-300</td>
<td>5</td>
<td>4</td>
<td>12.3</td>
</tr>
<tr>
<td>cvxqp3</td>
<td>28</td>
<td>11</td>
<td>3.9</td>
</tr>
<tr>
<td>stokes128</td>
<td>1</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>
Preprocessing - Constrained ordering

- Part of matrix sparsity is lost during graph compression
- **Constrained ordering**: *only pivot dependency within $2 \times 2$ blocks need be respected.*

Ex: $k \rightarrow j$ indicates that if $k$ is selected before $j$ then $j$ must be eliminated together with $k$.

*if $j$ is selected first then no more constraint on $k$.***
Preprocessing – Constrained ordering

- **Constrained ordering**: *only pivot dependency within $2 \times 2$ blocks need be respected.*

### Influence of using a constrained ordering (with scaling)

<table>
<thead>
<tr>
<th>Constrained</th>
<th>Total time (seconds)</th>
<th>Nb of entries in factors in Millions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>cvxqp3</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>stokes128</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Outline

Approaches for parallel factorization
- Elimination trees
- Distributed memory sparse solvers
- Some parallel solvers
- Case study: comparison of MUMPS and SuperLU
Approaches for parallel factorization

Elimination trees

Distributed memory sparse solvers

Some parallel solvers

Case study: comparison of MUMPS and SuperLU
Elimination DAG and unsymmetric matrices

L factors of unsymmetric matrix

Directed graph G(L)

Elimination dag (L)

Elimination dags: transitive reduction of the G(L)

- Because of unsymmetry the transitive reduction is not a tree
- What makes L be the factors of an unsymmetric matrix?
Elimination DAG and unsymmetric matrices

Elimination dags : transitive reduction of the G(L)
- Because of unsymmetry the transitive reduction is not a tree
- What makes L be the factors of an unsymmetric matrix?
Elimination tree and symmetric matrices

Elimination trees and symmetric matrices

Directed graph $G(L)$

Elimination dag (L)

$L$ factors of symmetric matrix

Elimination dags: transitive reduction of the $G(L)$

- Because of unsymmetry the transitive reduction is not a tree
- What makes $L$ be the factors of an unsymmetric matrix?
Elimination tree

To summarize (for symmetric structured matrices):

- The elimination tree expresses dependencies between the various steps of the factorization.
- It also exhibits parallelism arising from the sparse structure of the matrix.

Building the elimination tree

- Permute matrix (to reduce fill-in) $PAP^T$.
- Build filled matrix $A_F = L + L^T$ where $PAP^T = LL^T$.
- Transitive reduction of associated filled graph.

→ Each column corresponds to a node of the graph. Each node $k$ of the tree corresponds to the factorization of a frontal matrix whose row structure is that of column $k$ of $A_F$. 
Approaches for parallel factorization

- Elimination trees
- Distributed memory sparse solvers
- Some parallel solvers
- Case study: comparison of MUMPS and SuperLU
Distributed memory sparse solvers
Computational strategies for parallel direct solvers

- The parallel algorithm is characterized by:
  - Computational graph dependency
  - Communication graph

- Three classical approaches
  1. “Fan-in”
  2. “Fan-out”
  3. “Multifrontal”
Preamble: left and right looking approaches for Cholesky factorization

- $cmo\text{d}(j, k)$: Modification of column $j$ by column $k$, $k < j$,
- $cd\text{i}v(j)$: Division of column $j$ by the pivot

**Left-looking approach**

```plaintext
for $j = 1$ to $n$ do
    for $k \in \text{Struct}(\text{row } L_{j,1:j-1})$ do
        $cmo\text{d}(j, k)$
        $cd\text{i}v(j)$
```

**Right-looking approach**

```plaintext
for $k = 1$ to $n$ do
    $cd\text{i}v(k)$
    for $j \in \text{Struct}(\text{col } L_{k+1:n,k})$ do
        $cmo\text{d}(j, k)$
```
Illustration of Left and right looking

Left–looking

- used for modification

Right–looking

- modified
Fan-in variant (similar to left looking)

Algorithm: (Cholesky)
For $j = 1$ to $n$ do
  $\text{cdiv}(j)$
Endfor

For $k$ in $\text{Struct}(L_{j,*})$ do
  $\text{cmod}(j,k)$
Endfor

if $\text{map}(1) = \text{map}(2) = \text{map}(3) = p$ and $\text{map}(4) \neq p$ (only) one message sent by $p$ to update column 4 $\rightarrow$ exploits data locality in the tree.
Fan-in variant

\[ \forall i \in \text{children} \quad \text{map}(i) = P0 \quad \text{and} \quad \text{map}(\text{father}) \neq P0 \quad (\text{only}) \quad \text{one message sent by } P0 \rightarrow \text{exploits data locality in the tree.} \]
Fan-in variant

if $\forall i \in \text{children} \, \text{map}(i) = P0$ and $\text{map(father)} \neq P0$ (only) one message sent by $P0 \rightarrow$ exploits data locality in the tree.
if $\forall i \in \text{children} \ \text{map}(i) = P_0$ and $\text{map}($father$) \neq P_0$ (only) one message sent by $P_0 \rightarrow$ exploits data locality in the tree.
Fan-in variant

\[ \forall i \in \text{children} \; \text{map}(i) = P0 \text{ and } \text{map(father)} \neq P0 \text{ (only) one message sent by } P0 \rightarrow \text{exploits data locality in the tree.} \]
Fan-in variant

\[ \forall i \in \text{children \ map}(i) = P_0 \text{ and } \text{map(father)} \neq P_0 \text{ (only) one message sent by } P_0 \rightarrow \text{exploits data locality in the tree.} \]
if $\forall i \in \text{children} \ \text{map}(i) = P0$ and $\text{map(father)} \neq P0$ (only) one message sent by $P0 \rightarrow$ exploits data locality in the tree.
if $\forall i \in \text{children} \ \text{map}(i) = P0$ and $\text{map}(\text{father}) \neq P0$ (only) one message sent by $P0 \rightarrow$ exploits data locality in the tree.
Fan-out variant (similar to right-looking)

Algorithm: (Cholesky)

For k=1 to n do
  cmod(j,k)
  cdv(k)
  For j in Struct(L*,k) do
    cmod(j,k)
  Endfor
Endfor

if $\text{map}(2) = \text{map}(3) = p$ and $\text{map}(4) \neq p$ then 2 messages (for column 2 and 3) are sent by $p$ to update column 4.
Fan-out variant

if $\forall i \in \text{children} \; \text{map}(i) = P0$ and $\text{map}(\text{father}) \neq P0$ then $n$ messages (where $n$ is the number of children) are sent by $P0$ to update the processor in charge of the father

if $\forall i \in \text{children} \; \text{map}(i) = P0$ and $\text{map}(\text{father}) \neq P0$ then $n$ messages (where $n$ is the number of children) are sent by $P0$ to update the processor in charge of the father
Fan-out variant

\[ \forall i \in \text{children} \; \text{map}(i) = P0 \; \text{and} \; \text{map}(\text{father}) \neq P0 \; \text{then} \; n \; \text{messages (where} \; n \; \text{is the number of children) are sent by} \; P0 \; \text{to update the processor in charge of the father} \]
Fan-out variant

\[
\text{if } \forall i \in \text{children map}(i) = P0 \text{ and map(father)} \neq P0 \text{ then } n \text{ messages (where } n \text{ is the number of children) are sent by } P0 \text{ to update the processor in charge of the father}
\]
Fan-out variant

if $\forall i \in \text{children map}(i) = P0$ and $\text{map}(\text{father}) \neq P0$ then $n$ messages (where $n$ is the number of children) are sent by $P0$ to update the processor in charge of the father.
Fan-out variant

\[
\forall i \in \text{children} \; \text{map}(i) = P0 \; \text{and} \; \text{map}(\text{father}) \neq P0 \; \text{then} \; n \; \text{messages} \; (\text{where} \; n \; \text{is the number of children}) \; \text{are sent by} \; P0 \; \text{to update the processor in charge of the father}
\]
Fan-out variant

**Properties of fan-out**:

- Historically the first implemented.
- Incurs greater interprocessor communications than fan-in (or multifrontal) approach both in terms of
  - total number of messages
  - total volume
- Does not exploit data locality in the mapping of nodes in the tree
- Improved algorithm (local aggregation):
  - send aggregated update columns instead of individual factor columns for columns mapped on a single processor.
  - Improve exploitation of data locality
  - But memory increase to store aggregates can be critical (as in fan-in).
Multifrontal variant

Algorithm:
For k=1 to n do
    Build full frontal matrix with all indices in Struct(L[*,k])
    Partial factorisation
    Send Contribution Block to Father
Endfor

"Multifrontal Method"
Multifrontal variant

**Figure**: Communication schemes for the three approaches.
Multifrontal variant

**Figure:** Communication schemes for the three approaches.
Multifrontal variant

Figure: Communication schemes for the three approaches.
Multifrontal variant

Figure: Communication schemes for the three approaches.
Multifrontal variant

Figure: Communication schemes for the three approaches.
Approaches for parallel factorization

- Elimination trees
- Distributed memory sparse solvers

Some parallel solvers

Case study: comparison of MUMPS and SuperLU
Some parallel solvers
## Distributed-memory sparse direct codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Technique</th>
<th>Scope</th>
<th>Availability (<a href="http://www">www</a>.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSCPACK</td>
<td>Multifr./Fan-in</td>
<td>SPD</td>
<td>cse.psu.edu/~raghavan/Dscpack</td>
</tr>
<tr>
<td>MUMPS</td>
<td>Multifrontal</td>
<td>SYM/UNS</td>
<td>MUMPS Bordeaux-Lyon-Toulouse</td>
</tr>
<tr>
<td>PaStiX</td>
<td>Fan-in</td>
<td>SPD</td>
<td>labri.fr/perso/ramet/pastix</td>
</tr>
<tr>
<td>PSPASES</td>
<td>Multifrontal</td>
<td>SPD</td>
<td>cs.umn.edu/~mjoshi/pspases</td>
</tr>
<tr>
<td>SPOOLES</td>
<td>Fan-in</td>
<td>SYM/UNS</td>
<td>netlib.org/linalg/spooles</td>
</tr>
<tr>
<td>SuperLU</td>
<td>Fan-out</td>
<td>UNS</td>
<td>nersc.gov/~xiaoye/SuperLU</td>
</tr>
<tr>
<td>S+</td>
<td>Fan-out†</td>
<td>UNS</td>
<td>cs.ucsb.edu/research/S+</td>
</tr>
<tr>
<td>WSMP †</td>
<td>Multifrontal</td>
<td>SYM</td>
<td>IBM product</td>
</tr>
</tbody>
</table>

† Only object code is available.
## Distributed-memory sparse direct codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Technique</th>
<th>Scope</th>
<th>Availability (<a href="http://www">www</a>.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSCPACK</td>
<td>Multifr. / Fan-in</td>
<td>SPD</td>
<td>cse.psu.edu/~raghavan/Dscpack</td>
</tr>
<tr>
<td>MUMPS</td>
<td>Multifrontal</td>
<td>SYM/UNS</td>
<td>MUMPS Bordeaux-Lyon-Toulouse</td>
</tr>
<tr>
<td>PaStiX</td>
<td>Fan-in</td>
<td>SPD</td>
<td>labri.fr/perso/ramet/pastix</td>
</tr>
<tr>
<td>PSPASES</td>
<td>Multifrontal</td>
<td>SPD</td>
<td>cs.umn.edu/~mjoshi/pspases</td>
</tr>
<tr>
<td>SPOOLES</td>
<td>Fan-in</td>
<td>SYM/UNS</td>
<td>netlib.org/linalg/spoole</td>
</tr>
<tr>
<td>SuperLU</td>
<td>Fan-out</td>
<td>UNS</td>
<td>nersc.gov/~xiaoye/SuperLU</td>
</tr>
<tr>
<td>S+</td>
<td>Fan-out†</td>
<td>UNS</td>
<td>cs.ucsb.edu/research/S+</td>
</tr>
<tr>
<td>WSMP †</td>
<td>Multifrontal</td>
<td>SYM</td>
<td>IBM product</td>
</tr>
</tbody>
</table>

**Case study:** Comparison of MUMPS and SuperLU
Approaches for parallel factorization
  Elimination trees
  Distributed memory sparse solvers
  Some parallel solvers
  Case study: comparison of MUMPS and SuperLU
1. Analysis and Preprocessing
   - Preprocessing (max. transversal, scaling)
   - Fill-in reduction on $A + A^T$
   - Partial static mapping (elimination tree) with dynamic scheduling during factorization.

2. Factorization
   - Multifrontal (elimination tree of $A + A^T$)\
     $\text{Struct}(L) = \text{Struct}(U)$
   - Partial threshold pivoting
   - Node and tree level asynchronous parallelism
     - Partitioning (1D Front - 2D Root)
     - Dynamic distributed scheduling

3. Solution step and iterative refinement
SuperLU (Gaussian elimination with static pivoting)
X.S. Li and J.W. Demmel

1. **Analysis and Preprocessing**
   - Preprocessing (Max. transversal, scaling)
   - Fill-in reduction on $A + A^T$
   - Static mapping on a 2D grid of processes

2. **Factorization**
   - Fan-out based on elimination DAGs (preserves unsymmetry)
   - Static pivoting
     - If $|a_{ii}| < \sqrt{\varepsilon} \|A\|$ set $a_{ii}$ to $\sqrt{\varepsilon} \|A\|$
   - 2D irregular block cyclic partitioning (based on supernode structure)
   - Pipelining / BLAS3 based factorization

3. **Solution step and iterative refinement**
Traces of execution (bbmat, 8 proc. CRAY T3E)
Influence of maximum weighted matching $\text{MC64}$ on flops ($10^9$) for factorization (AMD ordering)

<table>
<thead>
<tr>
<th>Matrix</th>
<th>MC64</th>
<th>StrSym</th>
<th>MUMPS</th>
<th>SuperLU</th>
</tr>
</thead>
<tbody>
<tr>
<td>lhr71c</td>
<td>No</td>
<td>0</td>
<td>1431.0(*)</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>21</td>
<td>1.4</td>
<td>0.5</td>
</tr>
<tr>
<td>twotone</td>
<td>No</td>
<td>28</td>
<td>1221.1</td>
<td>159.0</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>43</td>
<td>29.3</td>
<td>8.0</td>
</tr>
<tr>
<td>fidapm11</td>
<td>No</td>
<td>100</td>
<td>9.7</td>
<td>8.9</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>29</td>
<td>28.5</td>
<td>22.0</td>
</tr>
</tbody>
</table>

(*) Estimated during analysis,
- Not enough memory to run the factorization.
Backward error analysis: \( Berr = \max_i \frac{|r_i|}{(|A|\cdot |x| + |b|)_i} \)

One step of iterative refinement generally leads to \( Berr \approx \varepsilon \)
Cost (1 step of iterative refinement) \( \approx \) Cost \( (LUx = b - Ax) \)
Communication issues

Average Vol. (64 procs)

Average Message Size (64 procs)
Time Ratios of the numerical phases

\[ \frac{\text{Time}(\text{SuperLU})}{\text{Time}(\text{MUMPS})} \]

### Factorization

![Factorization Graph](image)

### Solve

![Solve Graph](image)
Summary

- **Sparsity and Total memory**
  - SuperLU preserves better sparsity
  - SuperLU ($\approx 20\%$) less memory on 64 Procs (Asymmetry - Fan-out/Multifrontal)

- **Communication**
  - Global volume is comparable
  - MUMPS: much smaller (/10) nb of messages

- **Factorization / Solve time**
  - MUMPS is faster on nprocs $\leq 64$
  - SuperLU is more scalable

- **Accuracy**
  - MUMPS provides a better initial solution
  - SuperLU: one step of iter. refin. often enough
Conclusion (Part I)
Sparse solver: only a black box?

Default (often automatic/adaptive) setting of the options is often available; However, a better knowledge of the options can help the user to further improve its solution.

- Preprocessing options are critical to both performance and accuracy.
- Preprocessing may influence:
  - Operation cost and/or computational time
  - Size of factors and/or memory needed
  - Reliability of our estimations
  - Numerical accuracy.

- Therefore, not a real black box . . .
- Even if in general more a black box than most iterative solvers . . .
Sparse solver: only a black box?

Default (often automatic/adaptive) setting of the options is often available; However, a better knowledge of the options can help the user to further improve its solution.

- Preprocessing options are critical to both performance and accuracy.
- Preprocessing may influence:
  - Operation cost and/or computational time
  - Size of factors and/or memory needed
  - Reliability of our estimations
  - Numerical accuracy.
- Therefore, not a real black box . . .
- Even if in general more a black box than most iterative solvers . . .
Sparse solver: only a black box?

Default (often automatic/adaptive) setting of the options is often available; however, a better knowledge of the options can help the user to further improve its solution.

- Preprocessing options are critical to both performance and accuracy.
- Preprocessing may influence:
  - Operation cost and/or computational time
  - Size of factors and/or memory needed
  - Reliability of our estimations
  - Numerical accuracy.

- Therefore, not a real black box...
- Even if in general more a black box than most iterative solvers...
Direct solver: also kernels for iterative solvers?

**Direct**
- Very general/robust
  - Numerical accuracy
  - Irregular/unstructured problems
- Factorization of A
  - May be costly (memory/flops)
  - Factors can be reused for multiple/successive right-hand sides

**Iterative**
- Efficiency depends on:
  - Convergence preconditioning
  - Numerical prop./struct. of A
- Rely on efficient Mat-Vect product
  - Memory effective
  - Successive right-hand sides is problematic

**Hybrid approaches**
(Domain Decomposition, Schur, Block Cimmino)

*often strongly rely on both iterative and direct technologies*
Appendix
Unsymmetric test problems

| Name            | Order  | nnz    | $nnz(L|U)$ $\times 10^6$ | Ops $\times 10^9$ | Origin               |
|-----------------|--------|--------|--------------------------|------------------|----------------------|
| conv3d64        | 836550 | 12548250 | 2693.9                  | 23880            | CEA/CESTA            |
| fidapm11        | 22294  | 623554  | 11.3                     | 4.2              | Matrix market        |
| lhr01           | 1477   | 18427   | 0.1                      | 0.007            | UF collection        |
| qimonda07       | 8613291| 66900289 | 556.4                    | 45.7             | QIMONDA AG           |
| twotone         | 120750 | 1206265 | 25.0                     | 29.1             | UF collection        |
| ultrasound80    | 531441 | 33076161| 981.4                    | 3915             | Sosonkina            |
| wang3           | 26064  | 177168  | 7.9                      | 4.3              | Harwell-Boeing       |
| xenon2          | 157464 | 3866688 | 97.5                     | 103.1            | UF collection        |

*Ops* and *nnz($L|U$)* when provided obtained with METIS and default MUMPS input parameters.

UF Collection: University of Florida sparse matrix collection.

Harwell-Boeing: Harwell-Boeing collection.

PARASOL: Parasol collection
### Symmetric test problems

<table>
<thead>
<tr>
<th>Name</th>
<th>Order</th>
<th>( \text{nnz} ) ( \times 10^6 )</th>
<th>( \text{nnz}(L) ) ( \times 10^9 )</th>
<th>( \text{Ops} ) ( \times 10^9 )</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>audikw_1</td>
<td>943695</td>
<td>39297771</td>
<td>1368.6</td>
<td>5682</td>
<td>PARASOL</td>
</tr>
<tr>
<td>brgm</td>
<td>3699643</td>
<td>155640019</td>
<td>4483.4</td>
<td>26520</td>
<td>BRGM</td>
</tr>
<tr>
<td>coneshl2</td>
<td>837967</td>
<td>22328697</td>
<td>239.1</td>
<td>211.2</td>
<td>Samtech S.A.</td>
</tr>
<tr>
<td>coneshl1</td>
<td>1262212</td>
<td>43007782</td>
<td>790.8</td>
<td>1640</td>
<td>Samtech S.A.</td>
</tr>
<tr>
<td>cont-300</td>
<td>180895</td>
<td>562496</td>
<td>12.6</td>
<td>2.6</td>
<td>Maros &amp; Meszanos</td>
</tr>
<tr>
<td>cvxqp3</td>
<td>17500</td>
<td>69981</td>
<td>6.3</td>
<td>4.3</td>
<td>CUTEr</td>
</tr>
<tr>
<td>gupta2</td>
<td>62064</td>
<td>4248386</td>
<td>8.6</td>
<td>2.8</td>
<td>A. Gupta, IBM</td>
</tr>
<tr>
<td>ship_003</td>
<td>121728</td>
<td>4103881</td>
<td>61.8</td>
<td>80.8</td>
<td>PARASOL</td>
</tr>
<tr>
<td>stokes128</td>
<td>49666</td>
<td>295938</td>
<td>3.9</td>
<td>0.4</td>
<td>Arioli</td>
</tr>
<tr>
<td>thread</td>
<td>29736</td>
<td>2249892</td>
<td>24.5</td>
<td>35.1</td>
<td>PARASOL</td>
</tr>
</tbody>
</table>