Grouping variables in Frontal Matrices to improve Low-Rank Approximations in a Multifrontal Solver

Clément Weisbecker
ENSEEIHT-IRIT, Université de Toulouse, France

Preconditioning 2011, May 16, 2011

Joint work with Patrick Amestoy, Cleve Ashcraft, Olivier Boiteau, Alfredo Buttari and Jean-Yves L’Excellent.

PhD started on October 1st, 2010 and financed by EDF.
Grouping variables in Frontal Matrices to improve Low-Rank Approximations in a Multifrontal Solver
Introduction

Grouping variables in Frontal Matrices to improve Low-Rank Approximations in a Multifrontal Solver

- Multifrontal solver:
  - direct solver for large linear systems
  - well known and studied
Introduction

Grouping variables in Frontal Matrices to improve
Low-Rank Approximations in a Multifrontal Solver

– Multifrontal solver:
  • direct solver for large linear systems
  • well known and studied

– Low-rank approximations:
  • already used in several areas for data compression
  • accuracy controlled by a numerical parameter
  • interesting algebraic features
Grouping variables in Frontal Matrices to improve Low-Rank Approximations in a Multifrontal Solver

- Multifrontal solver:
  - direct solver for large linear systems
  - well known and studied

- Low-rank approximations:
  - already used in several areas for data compression
  - accuracy controlled by a numerical parameter
  - interesting algebraic features

⇒ Try to combine these two notions to improve multifrontal solvers, in particular the MUMPS multifrontal solver
The multifrontal method

Idea
Idea

C. Weisbecker, Precond’ II, May 16, 2011
Idea

- Updated variables connected to $D_1$
- Updated variables connected to $D_2$
At each node, an incomplete factorization of the frontal matrix is performed:

\[
\begin{align*}
CB_1 & \oplus CB_2 \\
\text{assembly} & \quad \text{FS} \\
\text{facto.} & \quad CB
\end{align*}
\]
Low-rank theory
General idea

Outer-product form

Let $A \in \mathbb{R}^{m \times n}$ be a matrix of rank $k$. Let $U \in \mathbb{R}^{m \times k}$ and $V \in \mathbb{R}^{n \times k}$ be two matrices. The outer-product form of $A$ is:

$$A = UV^T$$

- Storage: $k(m + n)$ vs $mn$
- If $k < \frac{mn}{m + n}$ → low-rank form

C. Weisbecker, Precond’11, May 16, 2011
Advantages and Drawbacks

- **Drawbacks**
  - Each outer-product form requires the computation of a SVD or a QR decomposition
  - More sophisticated data to store and manipulate (Householder vectors)

- **Advantages**
  - Reduction of the quantity of information stored
  - Basic algebra operations can be done more efficiently
  - Accuracy of the approximation directly controlled by a numerical parameter
Implementation of low-rank methods within a multifrontal solver


L. Grasedyck, R. Kriemann and S. Le Borne, Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems, Computing and Visualization in Science.
"FSU" factorization of the front:

\[ FS = L_{11} \cdot L_{11}^T \] (Cholesky factorization)

2. Solve (TRSM):
\[ L_{21} = F_{21} \cdot L_{11}^T \]

3. Update (SYRK):
\[ SC = F_{22} - L_{21} \cdot L_{21}^T \]
Complete front processing (Cholesky)

"FSU" factorization of the front:

1. Factor: $FS = L_{11} \cdot L_{11}^{T}$ (Cholesky factorization)
**Complete front processing (Cholesky)**

- "FSU" factorization of the front:

\[
\begin{array}{|c|c|}
\hline
FS & F_1 \\
\hline
F_21 & F_22 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
L_{11} & \quad \\
\hline
L_{21} & SC \\
\hline
\end{array}
\]

1. **Factor:** \( FS = L_{11} \cdot L_{11}^T \) (Cholesky factorization)
2. **Solve (TRSM):** \( L_{21} = F_{21} \cdot L_{11}^{-T} \)
Complete front processing (Cholesky)

- "FSU" factorization of the front:

1. **Factor:** $FS = L_{11} \cdot L_{11}^T$ (Cholesky factorization)
2. **Solve** (TRSM): $L_{21} = F_{21} \cdot L_{11}^{-T}$
3. **Update** (SYRK): $SC = F_{22} - L_{21} \cdot L_{21}^T$
The UPDATE(SYRK) and SOLVE(TRSM) phases can be performed using low-rank operations!

- **LR-Update** \((L_{21} = U \cdot V^T)\): \(SC = F_{22} - U \cdot (V^T \cdot V) \cdot U^T\)
- **LR-Solve** \((F_{21} = U \cdot V^T)\): \(L_{21} = U \cdot (V^T \cdot L_{11}^{-T})\)
The UPDATE(SYRK) and SOLVE(TRSM) phases can be performed using low-rank operations!

- **LR-Update** \((L_{21} = U \cdot V^T)\): \(SC = F_{22} - U \cdot (V^T \cdot V) \cdot U^T\)
- **LR-Solve** \((F_{21} = U \cdot V^T)\): \(L_{21} = U \cdot (V^T \cdot L_{11}^{-T})\)

**Problem:**
The fronts of the multifrontal tree are FULL rank
The UPDATE(SYRK) and SOLVE(TRSM) phases can be performed using low-rank operations!

- **LR-Update** \( (L_{21} = U \cdot V^T) : SC = F_{22} - U \cdot (V^T \cdot V) \cdot U^T \)
- **LR-Solve** \( (F_{21} = U \cdot V^T) : L_{21} = U \cdot (V^T \cdot L_{11}^{-1}) \)

**Problem:**

The fronts of the multifrontal tree are FULL rank

---

The UPDATE(SYRK) and SOLVE(TRSM) phases can be performed using low-rank operations!

- **LR-Update** \((L_{21} = U \cdot V^T)\): \(SC = F_{22} - U \cdot (V^T \cdot V) \cdot U^T\)
- **LR-Solve** \((F_{21} = U \cdot V^T)\): \(L_{21} = U \cdot (V^T \cdot L_{11}^{-T})\)

**Problem:**
The fronts of the multifrontal tree are FULL rank

---

**Solution:**
Group variables to obtain low-rank subblocks

---

The low-rank method

"How to find low-rank subblocks?"
\[ \min \{ \text{diam}(\mathcal{I}), \text{diam}(\mathcal{J}) \} < \eta \cdot \text{dist}(\mathcal{I}, \mathcal{J}) \]  

(Bebendorf)

Variables of \(\mathcal{I}\) and \(\mathcal{J}\) well separated \(\Rightarrow\) \(L(\mathcal{I}, \mathcal{J})\) has a low numerical rank
Grouping variables

Objective: define groups of well separated variables

First way: geometric partitioning

- Geometric reordering: Geometric properties are taken into account
- Laplacian problem on square $500 \times 500$ domain

(density proportional to the rank)
Grouping variables

Objective: define groups of well separated variables

Second way: random partitioning

- Random reordering: geometry is not taken into account
- Laplacian problem on square $500 \times 500$ domain

(density proportional to the rank)
Grouping algorithm

⇒ VERY important to have a good grouping of the variables
Grouping algorithm

⇒ VERY important to have a good grouping of the variables

Implemented algorithm:

- Halo-based algorithm to catch the geometry
- Coupled with a third party partitioning tool
VERY important to have a good grouping of the variables

Implemented algorithm:

- Halo-based algorithm to catch the geometry
- Coupled with a third party partitioning tool

1. The separator
VERY important to have a good grouping of the variables

Implemented algorithm:

- Halo-based algorithm to catch the geometry
- Coupled with a third party partitioning tool

1. The separator
2. The halo
VERY important to have a good grouping of the variables

Implemented algorithm:

- Halo-based algorithm to catch the geometry
- Coupled with a third party partitioning tool

1. The separator
2. The halo
3. Extraction of the halo
Grouping algorithm

⇒ VERY important to have a good grouping of the variables

Implemented algorithm:

- Halo-based algorithm to catch the geometry
- Coupled with a third party partitioning tool

1. The separator
2. The halo
3. Extraction of the halo
4. Partition of the halo
⇒ VERY important to have a good grouping of the variables

Implemented algorithm:

- Halo-based algorithm to catch the geometry
- Coupled with a third party partitioning tool

1. The separator
2. The halo
3. Extraction of the halo
4. Partition of the halo
5. Partition of the separator
Remains to define how to use the low-rank operations within a front

⇒ 4 strategies to process a front :

- Strategy FSUD
- Strategy FSUD
- Strategy panel FSDU
- (Strategy FDSU)

Note : we do not use low-rank within the Schur complements yet
1. We consider all the fully summed variables
1. We consider all the fully summed variables
2. \textbf{Factor} the entire diagonal block
1. We consider all the fully summed variables
2. \textbf{Factor} the entire diagonal block
3. \textbf{Solve} operation on the off-diagonal block
1. We consider all the fully summed variables
2. **Factor** the entire diagonal block
3. **Solve** operation on the off-diagonal block
4. **Update** the Schur complement
1. We consider all the fully summed variables

2. **Factor** the entire diagonal block

3. **Solve** operation on the off-diagonal block

4. **Update** the Schur complement

5. **Demote** each block of grouped variables within the factor
1. We process the entire fully summed variables block
1. We process the entire fully summed variables block

2. \textbf{Factor} the entire diagonal block
1. We process the entire fully summed variables block
2. \textbf{Factor} the entire diagonal block
3. \textbf{Solve} operation on the off-diagonal block
1. We process the entire fully summed variables block
2. **Factor** the entire diagonal block
3. **Solve** operation on the off-diagonal block
4. **Demote** each block of grouped variables
1. We process the entire fully summed variables block

2. Factor the entire diagonal block

3. Solve operation on the off-diagonal block

4. Demote each block of grouped variables

5. LR-Update the Schur complement blockwise
1. We process the fully summed variables block \textit{panelwise}. 

C. Weisbecker, Precond’11, May 16, 2011
1. We process the fully summed variables block \textit{panelwise}

2. \textbf{Factor} the entire diagonal subblock
1. We process the fully summed variables block \textit{panelwise}

2. \textbf{Factor} the entire diagonal subblock

3. \textbf{Solve} operation on the off-diagonal subblocks
1. We process the fully summed variables block **panelwise**
2. **Factor** the entire diagonal subblock
3. **Solve** operation on the off-diagonal subblocks
4. **Demote** each off-diagonal subblock
1. We process the fully summed variables block \textit{panelwise}

2. \textbf{Factor} the entire diagonal subblock

3. \textbf{Solve} operation on the off-diagonal subblocks

4. \textbf{Demote} each off-diagonal subblock

5. \textbf{LR-Update} the trailing panels
## Set of matrices

<table>
<thead>
<tr>
<th>Matrix</th>
<th>N</th>
<th>NZ</th>
<th>Type</th>
<th>CSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>tpll01a_r6_t</td>
<td>66,053</td>
<td>295,947</td>
<td>thermic</td>
<td>6.4E-16</td>
</tr>
<tr>
<td>tpll01a_r7_t</td>
<td>263,173</td>
<td>1,181,707</td>
<td>thermic</td>
<td>8.6E-16</td>
</tr>
<tr>
<td>tpll01a_r8_t</td>
<td>1,050,629</td>
<td>4,722,699</td>
<td>thermic</td>
<td>1.3E-15</td>
</tr>
<tr>
<td>tpll01a_r6_m</td>
<td>132,106</td>
<td>1,117,719</td>
<td>mechanic</td>
<td>8.3E-16</td>
</tr>
<tr>
<td>tpll01a_r7_m</td>
<td>526,346</td>
<td>4,463,639</td>
<td>mechanic</td>
<td>1.8E-15</td>
</tr>
<tr>
<td>tpll01a_r8_m</td>
<td>2,101,258</td>
<td>17,840,151</td>
<td>mechanic</td>
<td>2.0E-15</td>
</tr>
</tbody>
</table>

- 2D problems
- thermo-mechanical simulations from Code_Aster (by EDF)
- different mesh refinements
- work still in progress on 3D problems

Componentwise Scaled Residual $CSR = \frac{\|b - A\bar{x}\|_i}{(\|b\| + |A| |\bar{x}|)_i}$.
Strategy FSUD: results ($\varepsilon = 10^{-14}$)

<table>
<thead>
<tr>
<th>Matrix</th>
<th># of fronts</th>
<th>$L$</th>
<th>MEMORY</th>
<th>OPS</th>
<th>CSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>tpll01a_r6_t</td>
<td>6</td>
<td>10.3</td>
<td>49.9</td>
<td>–</td>
<td>9.2E-16</td>
</tr>
<tr>
<td>tpll01a_r7_t</td>
<td>22</td>
<td>17.4</td>
<td>35.7</td>
<td>–</td>
<td>9.1E-15</td>
</tr>
<tr>
<td>tpll01a_r8_t</td>
<td>111</td>
<td>28.2</td>
<td>29.4</td>
<td>–</td>
<td>3.6E-15</td>
</tr>
<tr>
<td>tpll01a_r6_m</td>
<td>100</td>
<td>22.2</td>
<td>58.0</td>
<td>–</td>
<td>1.5E-16</td>
</tr>
<tr>
<td>tpll01a_r7_m</td>
<td>423</td>
<td>45.1</td>
<td>59.9</td>
<td>–</td>
<td>1.8E-15</td>
</tr>
<tr>
<td>tpll01a_r8_m</td>
<td>468</td>
<td>40.1</td>
<td>41.7</td>
<td>–</td>
<td>1.0E-13</td>
</tr>
</tbody>
</table>

Features

- Focus on memory compression of the factorization
- No flop reduction during the factorization
- No error propagation within the factorization of the front
- Can be done “off-line” (solution phase, OOC)
Strategy FSDU : results ($\varepsilon = 10^{-14}$)

<table>
<thead>
<tr>
<th>Matrix</th>
<th># of fronts</th>
<th>L</th>
<th>MEMORY</th>
<th>OPS</th>
<th>CSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>tpII01a_r6_t</td>
<td>6</td>
<td>10.3 %</td>
<td>49.9 %</td>
<td>78.8 %</td>
<td>9.3E-16</td>
</tr>
<tr>
<td>tpII01a_r7_t</td>
<td>22</td>
<td>17.4 %</td>
<td>35.7 %</td>
<td>56.7 %</td>
<td>1.8E-15</td>
</tr>
<tr>
<td>tpII01a_r8_t</td>
<td>111</td>
<td>28.2 %</td>
<td>29.4 %</td>
<td>45.9 %</td>
<td>3.7E-14</td>
</tr>
<tr>
<td>tpII01a_r6_m</td>
<td>100</td>
<td>22.2 %</td>
<td>58.0 %</td>
<td>75.5 %</td>
<td>9.7E-15</td>
</tr>
<tr>
<td>tpII01a_r7_m</td>
<td>423</td>
<td>45.1 %</td>
<td>59.9 %</td>
<td>65.0 %</td>
<td>5.3E-15</td>
</tr>
<tr>
<td>tpII01a_r8_m</td>
<td>468</td>
<td>40.1 %</td>
<td>41.7 %</td>
<td>50.3 %</td>
<td>2.6E-13</td>
</tr>
</tbody>
</table>

- Memory compression for the storage of the factor
- More efficient update of the Schur complement thanks to the LR-update operation
Strategy panel FSDU: results \( (\varepsilon = 10^{-14}) \)

<table>
<thead>
<tr>
<th>Matrix</th>
<th># of fronts</th>
<th>( L )</th>
<th>MEMORY</th>
<th>OPS</th>
<th>CSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>tpll01a_r6_t</td>
<td>6</td>
<td>10.3 %</td>
<td>49.9 %</td>
<td>61.7 %</td>
<td>9.6E-16</td>
</tr>
<tr>
<td>tpll01a_r7_t</td>
<td>22</td>
<td>17.4 %</td>
<td>35.7 %</td>
<td>39.9 %</td>
<td>2.0E-15</td>
</tr>
<tr>
<td>tpll01a_r8_t</td>
<td>111</td>
<td>28.2 %</td>
<td>29.4 %</td>
<td>28.2 %</td>
<td>1.2E-14</td>
</tr>
<tr>
<td>tpll01a_r6_m</td>
<td>100</td>
<td>22.2 %</td>
<td>58.0 %</td>
<td>65.0 %</td>
<td>5.8E-15</td>
</tr>
<tr>
<td>tpll01a_r7_m</td>
<td>423</td>
<td>45.1 %</td>
<td>59.9 %</td>
<td>52.1 %</td>
<td>1.7E-15</td>
</tr>
<tr>
<td>tpll01a_r8_m</td>
<td>468</td>
<td>40.1 %</td>
<td>41.7 %</td>
<td>34.5 %</td>
<td>2.3E-13</td>
</tr>
</tbody>
</table>

- Memory compression for the storage of the factor
- More efficient factorization due to \( LR\)-updates within the factor
- More efficient update of the Schur complement thanks to the \( LR\)-update operation
<table>
<thead>
<tr>
<th></th>
<th><strong>OPS (%)</strong></th>
<th></th>
<th><strong>ERROR</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FSUD</td>
<td>FSDU</td>
<td>panel</td>
<td>MUMPS</td>
</tr>
<tr>
<td><strong>tpll01a_r6_t</strong></td>
<td>–</td>
<td>78.8</td>
<td>61.7</td>
<td>6.4E-16</td>
</tr>
<tr>
<td><strong>tpll01a_r7_t</strong></td>
<td>–</td>
<td>56.7</td>
<td>39.9</td>
<td>8.6E-16</td>
</tr>
<tr>
<td><strong>tpll01a_r8_t</strong></td>
<td>–</td>
<td>45.9</td>
<td>28.2</td>
<td>1.3E-15</td>
</tr>
<tr>
<td><strong>tpll01a_r6_m</strong></td>
<td>–</td>
<td>75.5</td>
<td>65.0</td>
<td>8.3E-16</td>
</tr>
<tr>
<td><strong>tpll01a_r7_m</strong></td>
<td>–</td>
<td>65.0</td>
<td>52.1</td>
<td>1.8E-15</td>
</tr>
<tr>
<td><strong>tpll01a_r8_m</strong></td>
<td>–</td>
<td>50.3</td>
<td>34.5</td>
<td>2.0E-15</td>
</tr>
</tbody>
</table>

- not much error propagation from panel to panel
- Strategy panel FSDU is the most efficient
- need to process more fronts with the same efficiency
Influence of the block size

- Stability for a large enough block size
- Efficient block size depends on the front size
- Will ease parallelism adaptation

C. Weisbecker, Precond’11, May 16, 2011
3D target grouping (7-pts stencil, laplacian)

- hand-made separators (2 levels) and partitioning
- grouping results are close to this
- efficient on 2D separators
- problems with irregular separators
Error and accuracy

Local error on blocks: \[
\frac{\|B - B_k\|_F}{\|B\|_F} \leq \varepsilon
\]

Global error on solution: \(\sim \varepsilon\)

No propagation observed!
→ Strong link between structural and numerical aspects
→ Several gain spots
Summarizing

→ Strong link between structural and numerical aspects
→ Several gain spots

Two approaches
Summarizing

→ Strong link between structural and numerical aspects
→ Several gain spots

Two approaches

1: PSEUDO-EXACT

- $\varepsilon \sim 10^{-16}$
- little accuracy lost
- typically used to accurately solve linear systems
Summarizing

→ Strong link between structural and numerical aspects
→ Several gain spots

Two approaches

1: PSEUDO-EXACT
- $\varepsilon \sim 10^{-16}$
- little accuracy lost
- typically used to accurately solve linear systems

2: APPROXIMATED
- $\varepsilon \gg 10^{-16}$
- typically used to compute preconditioners
- can replace mixed precision iterative refinement
Conclusion

- Efficient method for 2D problems
- Work still in progress for 3D problems
- An important step: the partitioning of the separator

Further works

- Pivoting, OOC, parallelism ...
- Need to study the error propagation
- Theoretical work to have a better understanding of how the grouping should work
- 3D separator quality and partitioning
Thank you for your attention!

Any question?