Comparison of Tearing Algorithms

Volker Waurich

Linköping, 04/02/2013
Outline

1. the concept of tearing
2. classification of tearing algorithms
3. some tearing methods
4. evaluation
the concept of tearing

tearing:

- symbolic method for large, sparse systems
- for linear, non-linear and mixed systems
- using graph theory
- speed up
- higher robustness
the concept of tearing

pre-work/preparation
- equation system
- algebraic representation
- partitioning/precedence ordering (BLTF)

tearing
- tearing heuristic and output assignment

- numerical computation
the concept of tearing

pre-work/preparation

\[
\begin{align*}
  f_1 &= f(x_3) \\
  f_2 &= f(x_1x_2, x_6) \\
  f_3 &= f(x_1, x_3, x_6) \\
  f_4 &= f(x_2, x_5, x_7) \\
  f_5 &= f(x_1x_4, x_5) \\
  f_6 &= f(x_4, x_5) \\
  f_7 &= f(x_1, x_2, x_3)
\end{align*}
\]

structure-incidence matrix

block lower triangular form

equation system  algebraic representation  partitioning
- implicit equations  - incidence matrix or adjacency matrix  - tearing of each block (algebraic loops)

the concept of tearing

basic principle of tearing

- tearing of algebraic loops
- assuming variables to be known (tearing variables)
- solve remaining equations
- iterate tearing variables with residual equations (Newton iteration)

→ the aim is to choose the tearing set with the least number of variables
→ only heuristic methods exist to choose variables in polynomial time (proven to be NP-hard)
→ solvability has to be considered
the concept of tearing

basic principle of tearing

1. bipartite graph
2. algebraic loop
3. tear the loop
4. solving with x1 → f3 for x3 → f2 for x2 (output assignment)
5. Newton iteration \( x_{1\text{new}} = x_{1\text{old}} - \frac{f_1(x_{1\text{old}})}{f'_1(x_{1\text{old}})} \)
tearing = output assignment + variable selection

previously matched
- output assignment is done before selection (Steward)
- works on digraph
- e.g. Steward, Ollero-Amselem

simultaneously matched
- output assignment is done during selection (Tarjan/Cellier)
- works on bipartite graph
- e.g. Cellier, Carpanzano
Celliers algorithm

- Bipartite graph
- Partially causalized
- Tearing selection
- Residual equation
Carpanzanos algorithm

- simultaneously matched
- similar to Cellier
- considers solvability of equations during tearing selection

→ see omcTearing (omc default) selection weights considering rearranging effort
Stewards algorithm

Structure-incidence matrix

<table>
<thead>
<tr>
<th>x_1</th>
<th>x_2</th>
<th>x_3</th>
<th>x_4</th>
<th>x_5</th>
<th>x_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>f_2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>f_3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>f_4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f_5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>f_6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Adjacency matrix

<table>
<thead>
<tr>
<th>k_1</th>
<th>k_2</th>
<th>k_3</th>
<th>k_4</th>
<th>k_5</th>
<th>k_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>k_2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>k_3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>k_4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>k_5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>k_6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Loop finding

l_1: k_1-k_5
l_2: k_2-k_3
l_3: k_4-k_5-k_3
l_4: k_6-k_2-k_3-k_4-k_5-k_1

Find tearing set

<table>
<thead>
<tr>
<th>k_1</th>
<th>k_2</th>
<th>k_3</th>
<th>k_4</th>
<th>k_5</th>
<th>k_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_1</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>l_2</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>l_3</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>l_4</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>
Ollero-Amselems algorithm

- previously matched
- works with contraction of nodes in the digraph
- if: contraction causes self-loops
  then: tearing variable found and removed from graph
comparison

previously matched

- works on matched-system-graph
- higher computational effort
- re-transformation from matched system

simultaneously matched

- works on incidence matrix
- output-assignment, precedence-ordering and tearing at once
  → this concept will be pursued
size of tearing sets

- Literature examples
- Pendulum
- EngineV6

- Cellier + partial Tarjan
- Ollero-Amselem + Steward
- Carpanzano + partial Tarjan
- Steward + Steward

- Linear (Cellier + partial Tarjan)
- Linear (Ollero-Amselem + Steward)
- Linear (Carpanzano + partial Tarjan)
- Linear (Steward + Steward)
conclusion

- simultaneously matched tearing method is more effective (smaller tearing set)
- less administrative overhead for the simultaneously matched method
- previous matching is not unique and may effect tearing selection
- solvability has to be considered during selection
- finish implementation
- manual selection via annotation
- choice of residual equation
- improve tearing algorithm
»Wissen schafft Brücken.«

thank you for your attention