Comparison of Methods for Solving Sparse Linear Systems

About the Solving of Systems of Linear Equations

Willi Braun, Kaja Balzereit, Bernhard Bachmann

FH Bielefeld
University of Applied Sciences

University of Applied Sciences Bielefeld
Bielefeld, Germany

February 2, 2015
I love deadlines. I like the whooshing sound they make as they fly by.

Douglas Adams, *1952 †2001

- Discussion on this topic raised after the last OpenModelica Workshop.
- Status at that time:
  - Linear torn system solved with a non-linear solver.
  - All non torn systems were solved always just by lapack.
- Question: Why you don’t use sparse linear solvers?
Motivation

*I love deadlines. I like the whooshing sound they make as they fly by.*

*Douglas Adams, *1952 †2001*

- Discussion on this topic raised after the last OpenModelica Workshop.
- Status at that time:
  - Linear torn system solved with a non-linear solver.
  - All non torn systems were solved always just by lapack.
- Question: Why you don’t use sparse linear solvers?
Outline

1. Systems of Linear Equations
2. Different Solver
3. Benchmarks
Systems of Linear Equations

What are Algebraic Loops?

**Transformation steps for simulation**

\[ 0 = f(x(t), \dot{x}(t), y(t), u(t), p, t) \]

\[ \downarrow \]

\[ 0 = f(x(t), z(t), u(t), p, t), \quad z(t) = \left( \begin{array}{c} \dot{x}(t) \\ y(t) \end{array} \right) \]

\[ \downarrow \]

\[ z(t) = \left( \begin{array}{c} \dot{x}(t) \\ y(t) \end{array} \right) = g(x(t), u(t), p, t) \]

\[ \downarrow \]

\[ \dot{x}(t) = h(x(t), u(t), p, t) \]

\[ y(t) = k(x(t), u(t), p, t) \]

**Transformation example**

\[ f_2(z_2) = 0 \]

\[ f_4(z_1, z_2) = 0 \]

\[ f_3(z_2, z_3, z_5) = 0 \]

\[ f_5(z_1, z_3, z_5) = 0 \]

\[ f_1(z_3, z_4) = 0 \]

**Algebraic loop (SCC)**

\[ f_3(z_2, z_3, z_5) = 0 \]

\[ f_5(z_1, z_3, z_5) = 0 \]
Systems of Linear Equations

General form

\[ a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \]
\[ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \]
\[ \vdots \quad \vdots \quad = \quad \vdots \]
\[ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \]

in compact matrix form:

\[ Ax = b \]

Example

\[ 2x - 2y = 4 \]
\[ -2x + 4y = 0 \]

⇒ solution (4, 2)
Systems of Linear Equations

Solution set

A linear system may behave in any one of three possible ways:

- The system has no solution.
- The system has infinitely many solutions.
- The system has a single unique solution.
Systems of Linear Equations

Solution set
A linear system may behave in any one of three possible ways:

- The system has no solution.
- The system has infinitely many solutions.
- The system has a single unique solution.

No Solution

- System is inconsistent.
- System is overdetermined.

Example

\[
\begin{align*}
4x + 2y &= 4 \\
4x + 2y &= 2 \\
x + y &= 4 \\
x + 4y &= 2 \\
4x + 2y &= 0
\end{align*}
\]
Systems of Linear Equations

Solution set
A linear system may behave in any one of three possible ways:
- The system has no solution.
- The system has infinitely many solutions.
- The system has a single unique solution.

Infinitely many solutions
- System is known as an under-determined system.
- System as linear dependent equations.

Example
\[
\begin{align*}
  x + 2y + 2z &= 2 \\
  2x + 4y + 4z &= 4 \\
  4x + 2y + z &= 0
\end{align*}
\]
Solution set
A linear system may behave in any one of three possible ways:
- The system has no solution.
- The system has infinitely many solutions.
- The system has a single unique solution.

Single unique solution
- System is regular.
- If the \( \text{rg}(A) = \text{rg}(A, b) \).

Example
\[
\begin{align*}
2x - 2y &= 4 \\
-2x + 4y &= 0
\end{align*}
\]
Solving a linear system

- Elimination of variables
- Cramer’s rule
- Matrix solution (Inverse A)
- LU decomposition
- Iterative methods
- [...]
Systems of Linear Equations

- Elimination of variables
- Cramer’s rule
- Matrix solution (Inverse A)
- LU decomposition
- Iterative methods
- [..]

**LU decomposition**

Factorization is performed by replacing any row in $A$ by a linear combination of itself and any other row.

$$Ax = b = LUx$$

$$\iff$$

$$L(Ux) = b$$

Solution by substitution:

$$Ly = b$$

$$Ux = y$$
Systems of Linear Equations

Pivoting

Example

\[
A = LU
\]
\[
\begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix}
= \begin{bmatrix}
u_{11} & u_{12} \\
0 & u_{22}
\end{bmatrix}
\begin{bmatrix}
l_{11} & 0 \\
l_{21} & l_{22}
\end{bmatrix}
\]

- Without proper permutations of A, the factorization may fail.
- If \( a_{11} = 0 \), then \( l_{11} \) or \( u_{11} \) has to be zero, which implies either L or U is singular, since \( a_{11} = l_{11}u_{11} \).

Pivoting

- Factorization without pivoting is numerical unstable, or even not possible.
  - interchange rows (partial pivoting)
  - interchange rows and cols (full pivoting)
- Partial pivoting is easier than full pivoting, since one don’t need to track permutations.
Systems of Linear Equations

Pivoting

Example

\[ \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix} \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \]

Without proper permutations of A, the factorization may fail.

If \( a_{11} = 0 \), then \( l_{11} \) or \( u_{11} \) has to be zero, which implies either L or U is singular, since \( a_{11} = l_{11} u_{11} \).

Pivoting

- Factorization without pivoting is numerical unstable, or even not possible.
  - interchange rows (partial pivoting)
  - interchange rows and cols (full pivoting)
- Partial pivoting is easier than full pivoting, since one don’t need to track permutations.
Systems of Linear Equations
Under-determined Systems

Example

\[
\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
=
\begin{pmatrix}
b_1 \\
b_2 \\
b_3
\end{pmatrix}
\iff
\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
0 & 0 & \hat{a}_{23} \\
0 & 0 & \hat{a}_{33}
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
=
\begin{pmatrix}
b_1 \\
\hat{b}_2 \\
\hat{b}_3
\end{pmatrix}
\]

Repair a singular A Matrix

- Perform full pivoting, that moves the singular rows and cols to the end.
- If corresponding \(b_i\) entries are almost zero, we can choose \(x_i\) also as zero.
Systems of Linear Equations

Under-determined Systems

Example

\[
\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
=
\begin{pmatrix}
b_1 \\
b_2 \\
b_3
\end{pmatrix}
\]

\[
\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
0 & 0 & \hat{a}_{23} \\
0 & 0 & \hat{a}_{33}
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
=
\begin{pmatrix}
b_1 \\
\hat{b}_2 \\
\hat{b}_3
\end{pmatrix}
\]

\[
\begin{pmatrix}
a_{11} & a_{13} & a_{12} \\
0 & a_{33} & 0 \\
0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_3 \\
x_2
\end{pmatrix}
=
\begin{pmatrix}
b_1 \\
\hat{b}_3 \\
1e^{-16}
\end{pmatrix}
\]

Repair a singular A Matrix

- Perform full pivoting, that moves the singular rows and cols to the end.
- If corresponding \( b_i \) entries are almost zero, we can choose \( x_i \) also as zero.
Compressed Sparse Row (CSR) format
- Store non-zero values, corresponding column
- pointer into value array corresponding to first non-zero($nz$) in each row
- Storage required = $2nz + n$

Frontal methods
- Variant of Gauss elimination that automatically avoids operations involving zero terms.
- Based on symbolical analysis of the sparsity pattern.
- Based on proper permutation of the matrix.

Example:

$$J = \begin{bmatrix}
0 & 0 & 0 & 0 & * \\
0 & 0 & 0 & 0 & * \\
* & 0 & * & 0 & * \\
0 & * & 0 & 0 & 0 \\
0 & * & 0 & * & *
\end{bmatrix}$$

$$A_i = [0, 1, 2, 5, 6, 9]$$

$$A_p = [5, 5, 1, 3, 5, 2, 2, 4, 5]$$

$$A_x = [*, *, *, *, *, *, *, *]$$
Why tearing belong to this topic?
- Tearing reduce the size of systems.
- Reduction makes the system more dense.
- It is a kind of sparsity exploitation.

Example:

$F_1: z_1 + f(x)z_3 - 6 = 0$

$F_2: f(y)z_3 - z_2 + 3z_1 = 0$

$F_3: 3z_1 + 2z_2 + z_3 = 0$

$\Rightarrow z_3 = 42$

Notes:
Now we need only solve linear for $z_3$ symbolical jacobians are used to calculate $\frac{\partial F_3}{\partial z_3}$
Once $z_3$ has been found, the first two assignments return $z_1, z_2$. 

Systems of Linear Equations
Tearing
Systems of Linear Equations

Tearing

**Why tearing belong to this topic?**

- Tearing reduce the size of systems.
- Reduction makes the system more dense.
- It is a kind of sparsity exploitation.

**Example**

\[
F_1 : z_1 + f(x)z_3 - 6 = 0 \\
F_2 : f(y)z_3 - z_2 + 3z_1 = 0 \\
F_3 : 3z_1 + 2z_2 + z_3 = 0
\]
Why tearing belong to this topic?

- Tearing reduce the size of systems.
- Reduction makes the system more dense.
- It is a kind of sparsity exploitation.

Example

\[
\begin{align*}
F_1 & : z_1 + f(x)z_3 - 6 = 0 \\
F_2 & : f(y)z_3 - z_2 + 3z_1 = 0 \\
F_3 & : 3z_1 + 2z_2 + z_3 = 0
\end{align*}
\]

Select \( z_3 \) as tearing variable

\[
\begin{align*}
F_1 & : z_1 = -f(x)z_3 + 6 \\
F_2 & : z_2 = f(y)z_3 + 3z_1 \\
F_3 & : z_3 = z_1 + 2z_2 \\
\Leftrightarrow & \quad z_3 = 42
\end{align*}
\]
Why tearing belong to this topic?
- Tearing reduce the size of systems.
- Reduction makes the system more dense.
- It is a kind of sparsity exploitation.

Example

\[ F_1 : z_1 + f(x)z_3 - 6 = 0 \]
\[ F_2 : f(y)z_3 - z_2 + 3z_1 = 0 \]
\[ F_3 : 3z_1 + 2z_2 + z_3 = 0 \]

Select \( z_3 \) as tearing variable

\[ F_1 : z_1 = -f(x)z_3 + 6 \]
\[ F_2 : z_2 = f(y)z_3 + 3z_1 \]
\[ F_3 : z_3 = z_1 + 2z_2 \]
\[ \Leftrightarrow z_3 = 42 \]

Notes
- Now we need only solve linear for \( z_3 \)
- Symbolical jacobians are used to calculate \( \frac{\partial F_3}{\partial z_3} \)
- Once \( z_3 \) has been found, the first two assignments return \( z_1, z_2 \).
Different Solver
Overview in OpenModelica

Usage in OpenModelica
- `-ls <solver_name>`
- `-lv LOG_LS, LOG_LS_V`
- `-lv LOG_STATS_V`
  compiler flag to activate/de-activate linear tearing.

Usage in OpenModelica
- Solver Name
  - lapack
  - totalpivot
  - umfpack
  - lis
Different Solver
Linear solver in OpenModelica

Dense Solvers
- LAPACK
  - Standard linear solver package.
  - Well known library for solving linear systems.
  - What else to say about it?
- Total pivot
  - Self written linear solver for full control.
  - Abilities to solve under-determined systems.

Sparse Solvers
- UMFPACK
  - Sparse solver suite.
  - Direct sparse linear solver.
  - Well tested and heavily used (e.g., Matlab).
- Lis
  - Iterative linear solvers suite.
  - Iterative solvers work only for well-posed problems.
  - This library works better for a really big $N$. 
Different Solver
Linear solver in OpenModelica

Dense Solvers
- LAPACK
  - Standard linear solver package.
  - Well known library for solving linear systems.
  - What else to say about it?
- Total pivot
  - Self written linear solver for full control.
  - Abilities to solve under-determined systems.

Sparse Solvers
- UMFPACK
  - Sparse solver suite.
  - Direct sparse linear solver.
  - Well tested and heavily used (e.g. Matlab).
- Lis
  - Iterative linear solvers suite.
  - Iterative solvers work only for well-posed problems.
  - This libraries works better for a really big $N$. 
Different Solver
Linear solver in OpenModelica

Dense Solvers
- LAPACK
  - Standard linear solver package.
  - Well known library for solving linear systems.
  - What else to say about it?
- Total pivot
  - Self written linear solver for full control.
  - Abilities to solve under-determined systems.

Sparse Solvers
- UMFPACK
  - Sparse solver suite.
  - Direct sparse linear solver.
  - Well tested and heavily used (e.g. Matlab).
- Lis
  - Iterative linear solvers suite.
  - Iterative solvers work only for well-posed problems.
  - This libraries works better for a really big $N$. 
Different Solver

Linear solver in OpenModelica

Dense Solvers
- LAPACK
  - Standard linear solver package.
  - Well known library for solving linear systems.
  - What else to say about it?
- Total pivot
  - Self written linear solver for full control.
  - Abilities to solve under-determined systems.

Sparse Solvers
- UMFPACK
  - Sparse solver suite.
  - Direct sparse linear solver.
  - Well tested and heavily used (e.g. Matlab).
- Lis
  - Iterative linear solvers suite.
  - Iterative solvers work only for well-posed problems.
  - This libraries works better for a really big $N$. 
Benchmarks
Overview in OpenModelica

- Linear Equation System
- Tearing
- no Tearing
- LAPACK
- UMFFPACK
Benchmarks

NPendelum

- Break-even point around $N = 1500$ and sparsity 0.17.
- After Tearing the linear system is 100% dense.

NPendelum Statistics

<table>
<thead>
<tr>
<th>N</th>
<th>Size linear system</th>
<th>Tearing</th>
<th>noTearing</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>208</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>428</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>868</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>120</td>
<td>1308</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>160</td>
<td>1748</td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>240</td>
<td>2628</td>
<td></td>
</tr>
<tr>
<td>320</td>
<td>320</td>
<td>3508</td>
<td></td>
</tr>
</tbody>
</table>
Benchmarks
EngineVN Example

EngineV6 -> EngineV72
- Break-even point beyond $N = 4000$.
- After Tearing the linear system is still sparse.

<table>
<thead>
<tr>
<th>N</th>
<th>Size linear system</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tearing</td>
</tr>
<tr>
<td>6</td>
<td>31</td>
</tr>
<tr>
<td>12</td>
<td>61</td>
</tr>
<tr>
<td>18</td>
<td>91</td>
</tr>
<tr>
<td>24</td>
<td>121</td>
</tr>
<tr>
<td>30</td>
<td>151</td>
</tr>
<tr>
<td>36</td>
<td>181</td>
</tr>
<tr>
<td>48</td>
<td>241</td>
</tr>
<tr>
<td>60</td>
<td>301</td>
</tr>
<tr>
<td>72</td>
<td>361</td>
</tr>
</tbody>
</table>
Benchmarks
Impact on Modelica Libraries

**MSL**
- Not really given, since the largest linear system are around 400.

**other Libraries**
- ???
Who can tell me which Libraries should be tested?

Happy testing with your models!

OpenModelica achieved a lot improvements in the last year on the solutions of linear equation systems.

For MSL model Tearing with LAPACK superiors UMFPACK.

Further work:

- Combine UMFPACK with an integrator for bigger models.
Who can tell me which Libraries should be tested?

Happy testing with your models!

OpenModelica achieved a lot improvements in the last year on the solutions of linear equation systems.

For MSL model Tearing with LAPACK superiors UMFPACK.

Further work:
  ▶ Combine UMFPACK with an integrator for bigger models.
So long, and Thanks for All the infrastructure, help and the fish ;}
Who can tell me which Libraries should be tested?
Happy testing with your models!
OpenModelica achieved a lot improvements in the last year on the solutions of linear equation systems.
For MSL model Tearing with LAPACK superiors UMFPACK.

Further work:
  ▶ Combine UMFPACK with an integrator for bigger models.