Software and Tools for Computational Engineering

RWTH AACHEN UNIVERSITY

# Sparse Direct Linear Solvers

Uwe Naumann

Informatik 12:
Software and Tools for Computational Engineering (STCE)

RWTH Aachen University

# Contents

# Outline

## Objective

- Introduction to direct $LR$- and $LL^T$-factorizations of sparse matrices.

## Learning Outcomes

- You will understand
  - need for prediction and minimization of fill-in
  - symbolic factorization methods
  - heuristics for elimination games
- You will be able to
  - build adjacency graphs of sparse matrices
  - apply heuristics

# Outline

The following statements are equivalent:

1. $A$ is regular.
2. A solution to $A \cdot \mathbf{x} = \mathbf{b}$ exists for any $\mathbf{b}$.
3. A solution to $A \cdot \mathbf{x} = \mathbf{b}$ is unique, if it exists.
4. $\forall \mathbf{x} : A \cdot \mathbf{x} = 0 \Rightarrow \mathbf{x} = \mathbf{0}$
5. the columns (rows) of $A$ are linearly independent
6. the inverse $A^{-1}$ of $A$ exists and $A^{-1} \cdot A = A \cdot A^{-1} = I_n$, where $I_n \in \boldsymbol{R}^{n \times n}$ denotes the identity in $\boldsymbol{R}^n$, i.e, $\forall \mathbf{v} \in \boldsymbol{R}^n : \; I_n \cdot \mathbf{v} = \mathbf{v}$.
7. $\det(A) \neq 0$ (nonzero determinant of $A$)

Lower / upper triangular system matrices yield linear systems the solution of which amounts to simple forward / backward substitution.

1. Lower triangular system by forward substitution, e.g.

$$\begin{pmatrix} 1 & 0 \\ -\frac{1}{3} & 1 \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \Rightarrow \quad \mathbf{y} = \begin{pmatrix} 1 \\ \frac{4}{3} \end{pmatrix}$$

2. Upper triangular system by backward substitution, e.g.

$$\begin{pmatrix} 3 & 1 \\ 0 & \frac{7}{3} \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{4}{3} \end{pmatrix} \quad \Rightarrow \quad \mathbf{y} = \begin{pmatrix} \frac{1}{7} \\ \frac{4}{7} \end{pmatrix}$$

Direct methods for the solution of linear systems aim to represent $A$ as a product of simpler (e.g, triangular) matrices, e.g,

$A = L \cdot R$ with lower triangular $L \in \boldsymbol{R}^{n \times n}$ and upper triangular $R \in \boldsymbol{R}^{n \times n}$

$$\Rightarrow \quad L \cdot \mathbf{v} = \mathbf{b} \; ; \quad R \cdot \mathbf{x} = \mathbf{v}$$

$A = L \cdot L^T$ with lower triangular $L \in \boldsymbol{R}^{n \times n}$

$$\Rightarrow \quad L \cdot \mathbf{v} = \mathbf{b} \; ; \quad L^T \cdot \mathbf{x} = \mathbf{v}$$

# Outline

From

$$\begin{pmatrix} \alpha & \mathbf{b}^T \\ \mathbf{a} & A_* \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \mathbf{a}_1 & L_* \end{pmatrix} \cdot \begin{pmatrix} \alpha & \mathbf{b}^T \\ 0 & R_* \end{pmatrix}$$

with $\alpha \in \boldsymbol{R}$, $\mathbf{a}, \mathbf{b} \in \boldsymbol{R}^{n-1}$ and lower / upper triangular matrices$L_*, R_* \in \boldsymbol{R}^{(n-1)\times(n-1)}$ follows

$$\mathbf{a} = \alpha \cdot \mathbf{a}_1 \ , \quad A_* = \mathbf{a}_1 \cdot \mathbf{b}^T + L_* \cdot R_*$$

and hence

$$\mathbf{a}_1 = \frac{\mathbf{a}}{\alpha} \quad L_* \cdot R_* = A_* - \mathbf{a}_1 \cdot \mathbf{b}^T \ .$$

Fill-in is induced by the rank-1 updates of the sparse $A_*$. Rows [columns] vanish identically in $a_1 \cdot b^T$ if the corresponding entries in $a_1$ [$b$] are equal to zero.

## Example

Let

$$A = \begin{pmatrix} a & b & \\ & c & \\ d & & e \end{pmatrix} \quad .$$

$LR$-factorization proceeds as follows.

$$\left(\begin{array}{c|cc|c} a & b & & \\ \hline & & c & \\ \frac{d}{a} & & -\frac{bd}{a} & e \end{array}\right) \rightarrow \begin{pmatrix} a & b & \\ & c & \\ \frac{d}{a} & -\frac{bd}{ac} & e \end{pmatrix} = L - I + R \quad .$$

Pivots are highlighted.

Rank-1 update with $(0, \frac{d}{a})^T \cdot (b, 0)$ causes fill-in in first step.

Note zero fill-in through switching first and third rows/columns.

Note triangularity through switching first and second rows/columns.

$A$ can be represented in RCS format as follows:

$$\mathbf{a} = (a, b, c, d, e)$$
$$\kappa = (0, 1, 1, 0, 2)$$
$$\rho = (0, 2, 3, 5)$$

Knowledge of fill-in generated during $LR$-factorization allows preallocation of

$$\mathbf{a} = (a, b, c, d, 0, e)$$
$$\kappa = (0, 1, 1, 0, 1, 2)$$
$$\rho = (0, 2, 3, 6)\ .$$

Expensive memory allocation and handling during $LR$-factorization can thus be avoided.

Symbolic methods (no floating-point arithmetic) are used to predict as well as minimize fill-in.

The fill-in during $LR$-factorization is minimized by permuting rows and columns of $A$ through multiplication of $A$ with permutation matrices. The latter

- are obtained from identity matrices by permuting columns
- are orthogonal, i.e. $P^{-1} = P^T$
- switch rows in $A$ when multiplied from left: $P \cdot A$
- switch columns in $A$ when multiplied from right: $A \cdot P$

Let $P_r$ and $P_c$ be obtained from $I$ through switching columns $i_r$ and $j_r$ and $i_c$ and $j_c$, respectively. Then, due to

$$
\begin{aligned}
& A \cdot \mathbf{x} = \mathbf{b} \quad \Rightarrow P_r \cdot A \cdot \mathbf{x} = P_r \cdot \mathbf{b} \\
\Rightarrow \quad & P_r \cdot A \cdot P_c \cdot P_c^{-1} \cdot \mathbf{x} = P_r \cdot \mathbf{b} \quad \Rightarrow P_r \cdot A \cdot P_c \cdot P_c^T \cdot \mathbf{x} = P_r \cdot \mathbf{b} \\
\Rightarrow \quad & P_r \cdot A \cdot P_c \cdot (\mathbf{x}^T \cdot P_c)^T = P_r \cdot \mathbf{b}
\end{aligned}
$$

switching columns in $A$ requires switching of corresponding entries in $\mathbf{x}$.

Given:

$$\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & 0 \\ a_{2,0} & 0 & a_{2,2} \end{pmatrix} \quad \Rightarrow \quad \text{2 fill-in } (l_{2,1},\ u_{1,2})$$

Wanted:

$$\begin{pmatrix} a_{2,2} & 0 & a_{2,0} \\ 0 & a_{1,1} & a_{1,0} \\ a_{0,2} & a_{0,1} & a_{0,0} \end{pmatrix} \quad \Rightarrow \quad \text{no fill-in}$$

Requires:

- $P_r$ to switch row 0 and 2
- $P_c = P_r$ to switch column 0 and 2

Find matrices $P_r$ and $P_c$ such that $LR$-factorization can be performed on $P_r \cdot A \cdot P_c$ with minimal fill-in.

The fill-in minimization problem is NP-complete [Rose,Tarjan,Gilbert].

Note that arbitrary permutations of rows and columns in $A$ can be expressed as a sequence of pair-wise row and column switches

$$P_r \cdot A \cdot P_c = P_{r_k} \cdot \ldots \cdot P_{r_1} \cdot A \cdot P_{c_1} \cdot \ldots \cdot P_{c_l}$$

Heuristics should be cheap; computational overhead potentially to be compensated by repeated solves on same static sparse data structure (e.g. in Newton's method).

We assume that all permutations of interest yield numerically stable versions of $A$. Pivoting for numerical stability is assumed to obsolete.

The adjacency graph of a matrix $A \in \boldsymbol{R}^{n \times n}$ is a directed graph

$$G_a(A) = (V, E), \quad V = \{0, \ldots, n-1\}, \quad E = \{(i,j) : a_{i,j} \neq 0 \wedge i \neq j\}$$

Diagonal entries of $A$ are assumed to be nonzero.

Examples:

$$\begin{pmatrix} * & * & \\ & * & \\ * & & * \end{pmatrix} \qquad \begin{pmatrix} * & * & * \\ * & * & \\ * & & * \end{pmatrix}$$

One $LR$ factorization step with pivot $a_{j,j}$, $0 \leq j < n$, can be represented as the elimination of vertex $j$ in $G_a(A)$.

Vertex $j$ is eliminated by connecting its predecessors with its successors (unless already connected) followed by the removal of $j$ from $G_a(A)$ together with all incident edges.

Newly generated edges in $G_a(A)$ correspond to fill-in generated in $A$.

Examples:

$$\begin{pmatrix} * & * & \\ & * & \\ * & & * \end{pmatrix} \qquad \begin{pmatrix} * & * & * \\ * & * & \\ * & & * \end{pmatrix}$$

Find a vertex elimination sequence that minimizes the number of newly generated edges.

During the elimination game we generate fill edges $(i, j)$ whenever there is a path from $i$ to $j$ through lower numbered vertices. Hence, equivalently, ...

... find an enumeration of the vertices in $G_a(A)$ such that the number of paths through lower numbered vertices is minimized.

Examples:

$$\begin{pmatrix} * & * & \\ & * & \\ * & & * \end{pmatrix} \qquad \begin{pmatrix} * & * & * \\ * & * & \\ * & & * \end{pmatrix}$$

The Markowitz heuristic eliminates $j$ if

$$|P_j| \cdot |S_j| \to \min \quad .$$

It can be applied statically to $G_a$ or dynamically to intermediate graphs constructed during the elimination game.

It needs to be combined with tie-breakers (e.g, [reverse] natural order).

Example:

$$\begin{pmatrix} * & & * & & & \\ & * & & & & \\ * & & * & & & * \\ & * & * & * & * & \\ & * & & * & * & \\ * & * & & & & * \end{pmatrix}$$

$\Rightarrow$ 3 x fill-in; dynamic Markowitz + reverse natural order $\Rightarrow$ no fill-in

# Outline

Let $A$ be symmetric positive definite. From

$$\begin{pmatrix} \alpha & \mathbf{a}^T \\ \mathbf{a} & A_* \end{pmatrix} = \begin{pmatrix} \rho & 0 \\ \mathbf{r} & R_*^T \end{pmatrix} \begin{pmatrix} \rho & \mathbf{r}^T \\ 0 & R_* \end{pmatrix}$$

follows

$$\alpha = \rho^2; \quad \mathbf{a}^T = \rho \cdot \mathbf{r}^T; \quad A_* = \mathbf{r} \cdot \mathbf{r}^T + R_*^T \cdot R_*$$

and hence

$$\rho = \sqrt{\alpha}$$
$$\mathbf{r} = \frac{\mathbf{a}}{\rho}$$
$$R_*^T \cdot R_* = A_* - \mathbf{r} \cdot \mathbf{r}^T \ .$$

Fill-in is induced by the rank-1 updates of the sparse $A_*$. Rows and columns vanish identically in $\mathbf{r} \cdot \mathbf{r}^T$ if the corresponding entries in $\mathbf{r}$ are equal to zero.

## Example

$$\begin{pmatrix} a & & \\ b & d & \\ c & 0 & e \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} \sqrt{a} & & \\ \frac{b}{\sqrt{a}} & d - \frac{b^2}{a} & \\ \frac{c}{\sqrt{a}} & -\frac{c \cdot b}{a} & e - \frac{c^2}{a} \end{pmatrix}$$

... in symmetric, fill-augmented RCS format:

$$a = (a, b, d, c, 0, e)\ , \quad \kappa = (0, 0, 1, 0, 1, 2)\ , \quad \rho = (0, 1, 3, 6)$$

$$a = \left(\sqrt{a}, \frac{b}{\sqrt{a}}, d - \frac{b^2}{a}, \frac{c}{\sqrt{a}}, -\frac{c \cdot b}{a}, e - \frac{c^2}{a}\right)$$
$$\kappa = (0, 0, 1, 0, 1, 2)$$
$$\rho = (0, 1, 3, 6)$$

The adjacency graph of a symmetric matrix $A \in \boldsymbol{R}^{n \times n}$ is an undirected graph

$$G_a(A) = (V, E), \quad V = \{0, \ldots, n-1\}, \quad E = \{(i,j) : a_{i,j} = a_{j,i} \neq 0 \wedge i \neq j\}$$

Diagonal entries of $A$ are assumed to be nonzero.

Examples:

$$\begin{pmatrix} * & * & * \\ * & * & \\ * & & * \end{pmatrix} \qquad \begin{pmatrix} * & * & * & \\ * & * & & * \\ * & & * & * \\ & * & * & * \end{pmatrix}$$

1. $G_0 = G_a(A)$
2. for $i = 1$ to $n$ do
   - 2.1 Make all neighbors of $i$ in $G_{i-1}$ mutually adjacent.
   - 2.2 Remove $i$ and all edges incident to $i$.
   - 2.3 The new graph is $G_i$.

The objective of the elimination game is to enumerate the vertices in $G_a(A)$ such that the fill-in is minimized.

Example:

$$\begin{pmatrix} * & * & * & \\ * & * & & * \\ * & & * & * \\ & * & * & * \end{pmatrix}$$

Software and Tools for Computational Engineering | RWTH AACHEN UNIVERSITY

Eliminate the vertex with the minimal number of neighbors next.

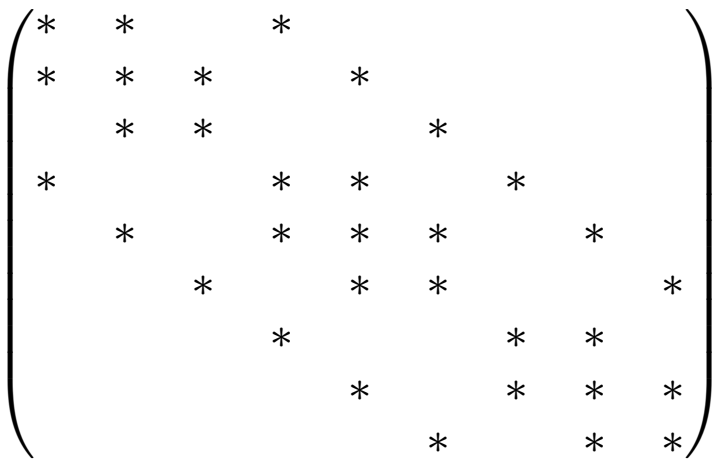
Example:

$$\begin{pmatrix}
* & * & & * & & & & & \\
* & * & * & & * & & & & \\
 & * & * & & & * & & & \\
* & & & * & * & & * & & \\
 & * & & * & * & * & & * & \\
 & & * & & * & * & & & * \\
 & & & * & & & * & * & \\
 & & & & * & & * & * & * \\
 & & & & & * & & * & *
\end{pmatrix}$$

$\Rightarrow$ 8 fill edges (within band); minimum degree ordering with natural order as tie-breaker yields 5 fill edges (within band)

- Eliminate vertices in (minimal) vertex separator $S$ ($V = V_1 \cup V_2 \cup S$, $V_1 \cap V_2 \cap S = \varnothing$) after those in $V_1$ and $V_2$ in order to avoid fill-in between vertices in $V_1$ and $V_2$.
- Use Minimum Degree heuristic inside of $S$
- Apply recursively to $V_1$ and $V_2$

Example:



$\Rightarrow$ 5 fill edges

- Perform breadth-first search to minimize band width (fill-in restricted to band)
- Apply Minimum Degree heuristic at each level. Use natural order as tie breaker.

Example:

$$\begin{pmatrix} * & * & & & * \\ * & * & * & & \\ & * & * & * & \\ & & * & * & * \\ * & & & * & * \end{pmatrix} \rightarrow \begin{pmatrix} * & * & * & & \\ * & * & & * & \\ * & & * & & * \\ & * & & * & * \\ & & * & * & * \end{pmatrix}$$

The order is reversed by the Reverse Cuthill-McKee heuristic, which often yields numerically more stable factorizations while keeping the same band width.

# Outline

## Summary

- Introduction to direct $LR$- and $LL^T$-factorizations of sparse matrices.
- Prediction and minimization of fill-in.
- Symbolic factorization methods.
- Heuristics for elimination games.

## Next Steps

- Practice construction of adjacency graphs of sparse matrices.
- Apply heuristics as part of elimination games.
- Continue the course to find out more ...