Solving linear systems of equations

Simulation Methods in Acoustics

Agenda

- Problem definition
  - We want to solve $\mathbf{Ax} = \mathbf{b}$, so $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$
  - Main message: We do **NOT** compute $\mathbf{A}^{-1}$!
- Direct solution strategies
  - Unstructured methods (no special relations of elements a_{ij})
    1. Forward / backward substitution
    2. Gaussian elimination
    3. LU factorization
    4. Pivoting
  - Structured methods (make use of relations of a_{ij})
    - Symmetry – LDL^T
    - Symmetric, positive definite – Cholesky
    - Positive semidefinite
    - Banded systems – Band LU
    - Reducing bandwidth – Cuthill–McKee
- Iterative solution strategies (later)

Forward and backward substitution

- Solve $\mathbf{A}\mathbf{x} = \mathbf{b}$, with $\mathbf{A} = \mathbf{L}$, a lower triangular matrix. ($\mathbf{A}$ is $n \times n$, rank n, $a_{ij} = 0$ if $i < j$.)
- Example:

$$\begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \end{Bmatrix}$$

- Solution is easy in this case:
  1. $x_1 = b_1 / l_{11}$
  2. $x_2 = \left(b_2 - l_{21} x_1\right) / l_{22}$
- In general:

$$x_i = \left(b_i - \sum_{j=1}^{i-1} l_{ij} x_j \right) / l_{ii}$$

- We get the result in *forward* order: $x_1, x_2, \ldots, x_n$
- If $\mathbf{A} = \mathbf{U}$ upper triangular, similarly, but in *backward* order

The Gaussian elimination

- How do we solve $\mathbf{A}\mathbf{x} = \mathbf{b}$ by hand?
- Idea: transform $\mathbf{A}$ into upper triangular form $\mathbf{U}$ and use backward substitution
- Method: subtract equations from each other, such that zero coefficients are obtained
- Example:
  - Look at the simple system

$$\begin{aligned} 3x_1 + 5x_2 &= \quad 9 \\ 6x_1 + 7x_2 &= \quad 4 \end{aligned}$$

  - Subtract 2 times the first equation from the second:

$$\begin{aligned} 3x_1 + 5x_2 &= \quad 9 \\ - 3x_2 &= -14 \end{aligned}$$

  - Attain first $x_2 = 14/3$, and then $x_1 = -43/9$

The LU factorization I.

- LU factorization = Gauss elimination formalized
- Take the example $v_1 \neq 0$ and $\tau = v_2/v_1$, then

$$\begin{bmatrix} 1 & 0 \\ -\tau & 1 \end{bmatrix} \begin{Bmatrix} v_1 \\ v_2 \end{Bmatrix} = \begin{Bmatrix} v_1 \\ 0 \end{Bmatrix}$$

- More generally:

$$\boldsymbol{\tau}^{\mathrm{T}} = [0, \ldots, 0, \tau_{k+1}, \ldots, \tau_n], \quad \text{with} \quad \tau_i = \frac{v_i}{v_k} \quad (i > k)$$

- Define the *Gauss transformation* $\mathbf{M}_k = \mathbf{I} - \boldsymbol{\tau}\mathbf{e}_k^{\mathrm{T}}$, such that

$$\mathbf{M}_k \mathbf{v} = \begin{bmatrix} 1 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & & 1 & 0 & & 0 \\ 0 & & -\tau_{k+1} & 1 & & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & & -\tau_n & 0 & & 0 \end{bmatrix} \begin{Bmatrix} v_1 \\ \vdots \\ v_k \\ v_{k+1} \\ \vdots \\ v_n \end{Bmatrix} = \begin{Bmatrix} v_1 \\ \vdots \\ v_k \\ 0 \\ \vdots \\ 0 \end{Bmatrix}$$

The LU factorization II.

- It is (usually[1]) possible to find Gauss transformations $\mathbf{M}_{n-1} \cdots \mathbf{M}_2 \mathbf{M}_1 \mathbf{A} = \mathbf{U}$ is upper triangular.
- **Algorithm:** construction of the LU factorization

  Start with $\mathbf{A}^{(1)} := \mathbf{A}$, set $k := 1$
  1. Determine multipliers: $\tau_i^{(k)} := a_{ik}^{(k)} / a_{kk}^{(k)} \quad (i = k+1, \ldots, n)$
  2. Apply $\mathbf{M}_k = \mathbf{I} - \boldsymbol{\tau}^{(k)} \mathbf{e}_k^{\mathrm{T}}$ to get $\mathbf{A}^{(k+1)} = \mathbf{M}_k \mathbf{A}_k$
  3. While $k < n-1$, set $k := k+1$, repeat from step (1).
- Matrix entries $a_{kk}^{(k)}$ must not be zero. These are called *pivots*.
- Where is $\mathbf{L}$, then?
  - $\mathbf{M}_k^{-1} = \left(\mathbf{I} - \boldsymbol{\tau} \mathbf{e}_k^{\mathrm{T}}\right)^{-1} = \mathbf{I} + \boldsymbol{\tau} \mathbf{e}_k^{\mathrm{T}}$
  - $\mathbf{L}$ is the *unit lower triangular* matrix of multipliers:

$$\mathbf{L} = \mathbf{M}_{n-1}^{-1} \cdots \mathbf{M}_2^{-1} \mathbf{M}_1^{-1} = \cdots = \mathbf{I} + \sum_{k=1}^{n-1} \boldsymbol{\tau}^{(k)} \mathbf{e}_k^{\mathrm{T}}$$

- Finally: $\underbrace{\mathbf{M}_{n-1} \cdots \mathbf{M}_2 \mathbf{M}_1}_{\mathbf{L}^{-1}} \mathbf{A} = \mathbf{U} \qquad \rightarrow \qquad \mathbf{A} = \mathbf{LU}$

[1]We will see that usually means *always*, if $\mathbf{A}$ is full rank shortly.

The LU factorization III.

- Uniqueness:
  - **Theorem.** If $\mathbf{A} = \mathbf{LU}$ exsists, then it is unique.
  - **Proof.** Suppose $\mathbf{A} = \mathbf{L}_1\mathbf{U}_1 = \mathbf{L}_2\mathbf{U}_2$. Then, $\mathbf{L}_2^{-1}\mathbf{L}_1 = \mathbf{U}_2\mathbf{U}_1^{-1}$. As $\mathbf{L}_2^{-1}\mathbf{L}_1$ is unit lower triangular while $\mathbf{U}_2\mathbf{U}_1^{-1}$ is upper triangular, both most equal $\mathbf{I}$. Thus, $\mathbf{L}_1 = \mathbf{L}_2$ and $\mathbf{U}_1 = \mathbf{U}_2$. □
- Solving $\mathbf{Ax} = \mathbf{b}$ using LU factorization
  1. Compute $\mathbf{LU} = \mathbf{A}$ to get $\mathbf{LUx} = \mathbf{b}$
  2. Solve for $\mathbf{y}$ in the lower triangular system $\mathbf{LUx} = \mathbf{Ly} = \mathbf{b}$
  3. Solve for $\mathbf{x}$ in the upper triangular system $\mathbf{Ux} = \mathbf{y}$
- Think of formulas with $\mathbf{A}^{-1}$ as equation solving!
  - $s = \mathbf{c}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{b} \rightarrow$ solve $\mathbf{Ax} = \mathbf{b}$, then $s = \mathbf{c}^{\mathrm{T}}\mathbf{x}$
  - Multiple r.h.s.: $\mathbf{AX} = \mathbf{B} \rightarrow \mathbf{LUX} = \mathbf{B}$, solve for each r.h.s.
  - Constraint matrix: $\mathbf{X} = -\mathbf{A}_s^{-1}\mathbf{A}_m \rightarrow$ solve $\mathbf{A}_s\mathbf{X} = \mathbf{A}_m$
  - Computing the inverse: $\mathbf{AX} = \mathbf{I} \rightarrow$ also multi-r.h.s. problem

Pivoting I.

- Example:

$$\mathbf{A} = \begin{bmatrix} 0.0001 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 10000 & 1 \end{bmatrix} \begin{bmatrix} 0.0001 & 1 \\ 0 & -9999 \end{bmatrix} = \mathbf{LU}$$

- What is the problem here?
- Solving $\mathbf{Ax} = \mathbf{b}$ (i.e. $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$) propagates error from $\hat{\mathbf{b}}$ to $\hat{\mathbf{x}}$.
  - Error bounded by the condition number $\kappa(\mathbf{A}) = \|\mathbf{A}\| \|\mathbf{A}^{-1}\|$
  - If $\hat{\mathbf{b}} = \mathbf{b} + \epsilon$, then $\max\{\|\hat{\mathbf{x}} - \mathbf{x}\|\} \approx \kappa(\mathbf{A}) \|\epsilon\|$
  - Here, $\kappa(\mathbf{A}) \approx 2.7$, but $\kappa(\mathbf{L}) \approx \kappa(\mathbf{U}) \approx 10^8$
  - Even if $\mathbf{A}$ is well-conditioned, large error can arise
  - This is due to the small pivot ($a_{11}^{(1)} = 0.0001$)
- Strategy: interchange rows to avoid small pivots.

$$\mathbf{PA} = \begin{bmatrix} 1 & 1 \\ 0.0001 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0.0001 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0.9999 \end{bmatrix} = \mathbf{LU}$$

Pivoting II.

- Algorithm: LU factorization with *partial pivoting*. In each step of the LU algorithm, find a permutation matrix $\Pi^{(k)}$ that swaps $A_{kk}^{(k)}$ with the largest $\left|A_{jk}^{(k)}\right|$ $(j = k, k+1, \dots n)$.
- This computes $\mathbf{PA} = \mathbf{LU}$, where
  - $\mathbf{P}$ is an *interchange permutation matrix*
  - $\mathbf{L}$ is unit lower triangular with $|l_{ij}| < 1$
  - $\mathbf{U}$ is upper triangular
- Pivoting strategies (where to look for the largest element?)
  - Partial pivoting – swap rows
  - Complete pivoting – swap rows and columns ($\mathbf{PAQ} = \mathbf{LU}$)
  - Rook pivoting – swap rows or columns ($\mathbf{PAQ} = \mathbf{LU}$)

Symmetric systems – The LDL^T factorization

- **Theorem:** If $\mathbf{A}$ is symmetric (with nonsingular principal submatrices, i.e. nonzero pivots), then there exsists a unique factorization $\mathbf{A} = \mathbf{LDL}^{\mathrm{T}}$, where $\mathbf{D}$ is diagonal.
- **Proof:** $\mathbf{A}$ has a unique LU factorization $\mathbf{A} = \mathbf{LU}$. The matrix $\mathbf{L}^{-1}\mathbf{AL}^{-\mathrm{T}} = \mathbf{UL}^{-\mathrm{T}}$ is both symmetric and upper triangular, therefore it is diagonal. Thus, $\mathbf{D} = \mathbf{UL}^{-\mathrm{T}}$ and $\mathbf{A} = \mathbf{LDL}^{\mathrm{T}}$. □
- Solving $\mathbf{Ax} = \mathbf{b}$ ($\mathbf{LDL}^{\mathrm{T}}\mathbf{x} = \mathbf{b}$):
  1. Solve $\mathbf{Lz} = \mathbf{b}$ for $\mathbf{z}$
  2. Solve $\mathbf{Dy} = \mathbf{z}$ for $\mathbf{y}$ (this is very cheap)
  3. Solve $\mathbf{L}^{\mathrm{T}}\mathbf{x} = \mathbf{y}$ for $\mathbf{x}$

Symmetric positive definite system – Cholesky

- **Theorem:** If $\mathbf{A}$ is positive definite and symmetric, there exsists a unique factorization $\mathbf{A} = \mathbf{G}\mathbf{G}^{\mathrm{T}}$, such that $\mathbf{G}$ is lower triangular with positive diagonal entries.
  - **Theorem:** If $\mathbf{A}$ is positive definite and $\mathbf{X}$ is full rank, then $\mathbf{B} = \mathbf{X}^{\mathrm{T}}\mathbf{A}\mathbf{X}$ is also positive definite.
  - **Proof:** If $\mathbf{z}$ satisfies $0 \geq \mathbf{z}^{\mathrm{T}}\mathbf{B}\mathbf{z} = (\mathbf{X}\mathbf{z})^{\mathrm{T}}\mathbf{A}(\mathbf{X}\mathbf{z})$, then $\mathbf{X}\mathbf{z} = \mathbf{0}$. But since $\mathbf{X}$ is full rank, this implies that $\mathbf{z} = \mathbf{0}$.
- **Proof:** From the previous theorem $\mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-\mathrm{T}} = \mathbf{D}$ is positive definite. Thus, d_k in $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n)$ are positive and $\mathbf{G} = \mathbf{L}\,\mathrm{diag}(\sqrt{d_1}, \ldots, \sqrt{d_n})$. □

Pivoting and symmetry

- Reminder: pivoting is used for avoiding small dividers
- But: Pivoting *destroys* symmetry! If $\mathbf{A}$ is symmetric and $\mathbf{P}$ is an interchange permutation matrix, then $\mathbf{PA}$ is not symmetric.
- However, $\mathbf{PAP}^{\mathrm{T}}$ is symmetric. We can introduce *symmetric pivoting*, and formulate the factorization $\mathbf{PAP}^{\mathrm{T}} = \mathbf{LDL}^{\mathrm{T}}$.
- In case of symmetric pivoting, the factor a_{kk} is swapped with the maximal diagonal entry a_{jj} $(j = k+1, \ldots, n)$.
- Then, $\mathbf{Ax} = \mathbf{b}$ is solved as
  1. Solve $\mathbf{Lw} = \mathbf{Pb}$ for $\mathbf{w}$
  2. Solve $\mathbf{Dy} = \mathbf{w}$ for $\mathbf{y}$ (very cheap)
  3. Solve $\mathbf{L}^{\mathrm{T}}\mathbf{z} = \mathbf{y}$ for $\mathbf{z}$
  4. Compute $\mathbf{x} = \mathbf{P}^{\mathrm{T}}\mathbf{z}$ (only reordering)

Positive semidefinite systems

- LDL^T for a positive semidefinite matrix with rank r

$$\mathbf{PAP}^{\mathrm{T}} = \begin{bmatrix} \mathbf{L}_{11} \\ \mathbf{L}_{21} \end{bmatrix} \mathbf{D}_r \left[\mathbf{L}_{11}^{\mathrm{T}} | \mathbf{L}_{21}^{\mathrm{T}} \right]$$

  $\mathbf{D}_r = \mathrm{diag}\left(d_1, \ldots, d_r\right)$ has positive diagonal entries, $\mathbf{L}_{11}$ is unit lower triangular, and $\mathbf{L}_{21} \in \mathbb{R}^{(n-r)\times r}$

- The Cholesky decomposition is similar

$$\mathbf{PAP}^{\mathrm{T}} = \begin{bmatrix} \mathbf{G}_{11} \\ \mathbf{G}_{21} \end{bmatrix} \left[\mathbf{G}_{11}^{\mathrm{T}} | \mathbf{G}_{21}^{\mathrm{T}} \right]$$

Banded systems

- **Theorem:** If $\mathbf{A} = \mathbf{LU}$ has an upper bandwidth (ubw. in short) q and lower bandwidth (lbw.) p, then $\mathbf{U}$ has upper bandwidth q and $\mathbf{L}$ has lower bandwidth p.
- **Proof:** By induction. Write $\mathbf{A}$ as

$$\mathbf{A} = \begin{bmatrix} \alpha & \mathbf{w}^{\mathrm{T}} \\ \mathbf{v} & \mathbf{B} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \mathbf{v}/\alpha & \mathbf{I}_{n-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \mathbf{B} - \mathbf{v}\mathbf{w}^{\mathrm{T}}/\alpha \end{bmatrix} \begin{bmatrix} \alpha & \mathbf{w}^{\mathrm{T}} \\ 0 & \mathbf{I}_{n-1} \end{bmatrix}$$

Here, the matrix $\mathbf{B} - \mathbf{v}\mathbf{w}^{\mathrm{T}}/\alpha$ has ubw. q and lbw. p since only the first p components of $\mathbf{v}$ and the first q components of $\mathbf{w}$ are nonzero. Let $\mathbf{L}_1$ and $\mathbf{U}_1$ be the LU factorization of this matrix. Then,

$$\mathbf{L} = \begin{bmatrix} 1 & 0 \\ \mathbf{v}/\alpha & \mathbf{L}_1 \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} 1 & \mathbf{w}^{\mathrm{T}} \\ 0 & \mathbf{U}_1 \end{bmatrix}$$

Gives $\mathbf{A} = \mathbf{LU}$ with the above bandwidths. □

Reducing bandwidth

- Solving systems with small bandwidth (λ) is fast!
  Cost of solution is $O(n\lambda^2)$
- Bandwidth depends on the choice of the order of DOFs.
- **Algorithm:** Cuthill–McKee ordering for symmetric matrices

  Choose a peripheral vertex P (i.e. with the lowest vertex degree) and start with a result set $R := \{P\}$. Set $i := 1$.

  1. Construct the adjacency set $A_i := \mathrm{Adj}\{R_i\} \setminus R$
  2. Sort A_i in ascending vertex degree.
  3. Append A_i to the result set R.
  4. Set $i := i + 1$ and continue from step (1).

  Note: this is a breadth first search (BFS) algorithm with an extra ordering step. Finally, R is a re-indexing permutation.
- **Algorithm:** Reverse Cuthill–McKee. The same, but the result is reversed in the end. This gives less fill-in in practice.
- Example – the worst DOF ordering in 1D case ($n = 101$):

$$(1) \leftrightarrow (101) \leftrightarrow (2) \leftrightarrow (99) \leftrightarrow \cdots\cdots \leftrightarrow (50) \leftrightarrow (51)$$

Bandwidth reduction example

- Example: random points in 2D space, with near points connected by springs
- Bandwidth reduction of the resulting sparse stiffness matrix **K** using the permuation vector p computed by the Cuthill – McKee algorithm is shown in the figure (Left: original, right: after renumbering)