Parallel Programming

Linear Algebra – Solving Linear Systems

Jan Schönherr
Technische Universität Berlin
School IV – Electrical Engineering and Computer Sciences
Communication and Operating Systems (KBS)
Einsteinufer 17, Sekr. EN6, 10587 Berlin

Linear systems (i)

- System of *m* linear equations
- With *n* variables x_j and constants a_{ij} and b_i

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m
\end{aligned}
$$

- How to solve this?

Linear systems (ii)

- Written as a matrix equation:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

- Or: $A\,x = b$

- For *m=n*:
  - System is solvable with exactly one solution, if det(*A*)≠0
  - That is, *A* can be transformed into a triangular matrix using Gaussian elimination rules and all diagonal entries are different from zero

Use cases

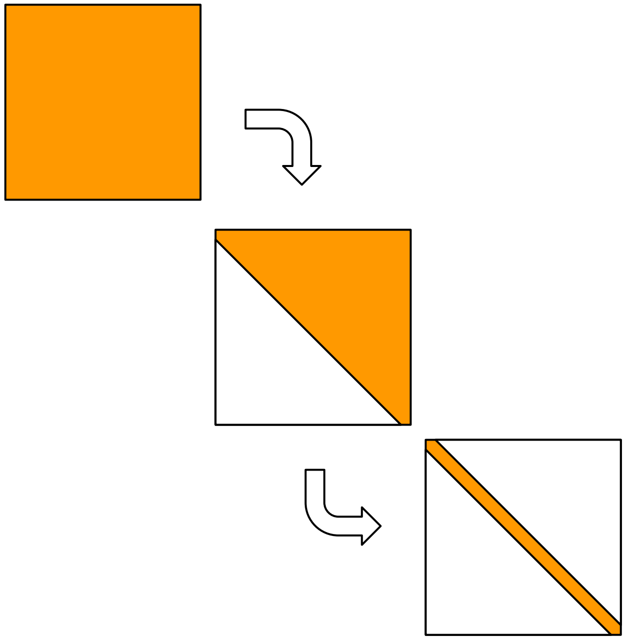
- Finite Element Method
  - Solve partial differential equations through discretization
  - For complex domains
    - E. g. general structural analysis, car crashes, architecture
- Finite Difference Method
  - Solve partial differential equations through discretization
  - For regular domains
    - E. g. heat transport, fluid dynamics
- Others
  - E. g. analysis of power grids, production planning, regression analysis

Solving linear systems

- Direct
  - The exact solution is calculated in a finite number of steps
  - E. g. Gaussian elimination, LU decomposition, Cholesky decomposition, QR decomposition, …
- Iterative
  - An approximation to the solution is improved with every iteration
  - E. g. Jacobi method, Gauss-Seidel method, conjugate gradient method, …
  - Convergence not always guaranteed
- (Some methods require the linear system to have a special structure)

Gaussian elimination/LU decomposition

- Two steps:
  - A) Forward elimination
    - Convert *A* into an upper triangular matrix *U*
  - B) Back substitution
    - Convert *U* into a diagonal matrix



- LU decomposition
  - Conversion factors are stored in a lower triangular matrix *L*
  - $A = LU \quad => \quad Ax = LUx = b = Ly$ (with $Ux=y$)
  - Better when the system must be solved for multiple *b*
- Problems
  - Numerical stability (solved by total/partial pivoting)
  - Fill-in in sparse matrices

Jacobi method

- Iterative solver for a linear system *Ax=b*
  - Diagonal elements of *A* must be different from zero
  - Convergence is guaranteed if *A* is strictly diagonally dominant
- Starting with an arbitrary $x^{(0)}$, this approximation is gradually improved by:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{i \neq j} a_{ij} x_j^{(k)} \right)$$

- Or: $$x^{(k+1)} = D^{-1}\left[b - (L+U)x^{(k)}\right]$$
- Where *A=D+L+U* (diagonal + lower + upper matrix)

Finite element method (FEM)

- Approximate solutions to partial differential equations through discretization
  - For complex domains
    - Higher resolution in interesting areas
  - E. g. general structural analysis, car crashes, architecture

- Questions
  - How does this actually work?
  - What has this to do with solving of linear systems?
  - Why is the linear system sparse?
  - How large are those systems, actually?

FEM – Steps

- 1. Have a (2D/3D) model of your object
- 2. Define materials and their physical properties
- 3. Mesh your model according to your needs
- 4. Define external influences
- 5. Convert the mesh into a (sparse) matrix K_S, the external influences into a (dense) vector f_S
- 6. Solve $f_S = K_S\, u_S$
- 7. Update mesh and external influences from u_S
- 8. Repeat steps 5 to 7, if necessary

FEM – Model to mesh

- The model is discretized into (many) (simple) *elements* with *finite* dimensions.
  - E. g. beams, triangles, tetrahedrons, bricks, …
- Elements adhere to some physical model
  - I. e. they are as good or bad as the model
- Elements have a certain degree of freedom
  - Depending on the flexibility of the physical model
  - I. e. a simple (2D) beam has a degree of 6 (2 nodes, each with displacement along two axis and a rotation)
  - Captured in a displacement vector u
  - This results in a corresponding force vector f
  - Relationship is described by a stiffness matrix K
  - $f = K u$

FEM – Mesh to matrix

- Elements are connected to other elements via their nodes, forming one large element: the system
- The system's stiffness matrix K_S is calculated by combining all element stiffness matrices
  - Each subblock of a element matrix contributes once to the corresponding subblock of the system matrix
  - If a system node is part of multiple elements, the subblocks "add up"
- This results in the relation for the complete system:
  - $f_S = K_S u_S$

FEM – Solving the system

- Boundary conditions and external influences are applied to the system $f_S = K_S\, u_S$
  - This sets some entries of f_S and u_S to fixed values

- Then, the system is actually solved

- The results may then be visualized in the model
- Or, in a simulation, the mesh and the external influences are updated and the next iteration starts