Lecture 15: Reordering algorithms

COMP5930M Scientific Computation

Today

- Matrix permutations
- Overall solution process
- Renumbering
- Pivoting

Matrix permutation

- Any reordering process implies a permutation of the original equations (rows) or the unknowns (columns)
- In practice we do not reorder our system physically but use a permutation matrix **P** that stores the permutation
- **P** is, itself, a (very) sparse matrix. Each row and column has exactly one element equal to 1, the others 0

Permutation matrices

Example: P swaps 3rd and 4th row/column of a matrix **A**

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Permutation of the 3rd and 4th rows: **PA**
Permutation of the 3rd and 4th columns: **AP**
Permutation of the 3rd and 4th rows and columns: **PAP**

Formal permutation of the system

- We solve: $\mathbf{PAx} = \mathbf{Pb}$ where $\mathbf{P}$ is a permutation matrix
- If $\mathbf{P} = \mathbf{I}$ we have the original system
- We can swap rows i and j of the system
  by swapping rows i and j of $\mathbf{P}$
- Since $\mathbf{P}$ is a permutation matrix

$$\mathbf{P}^T = \mathbf{P}^{-1}$$

In practice

- When $\mathbf{A}$ is symmetric ($\mathbf{A} = \mathbf{A}^T$), we would like the permuted matrix to remain symmetric
- We can write

$$\mathbf{PAP}^T\mathbf{Px} \; = \; \mathbf{Pb}$$

since $\mathbf{P}^T\mathbf{P} \; = \; \mathbf{I}$

- We then solve

$$\begin{aligned} \mathbf{By} &= \mathbf{c} \\ \mathbf{x} &= \mathbf{Py} \end{aligned}$$

where $\mathbf{B} = \mathbf{PAP}^T$, $\mathbf{y} = \mathbf{Px}$, $\mathbf{c} = \mathbf{Pb}$

The overall solution process

- Before factorisation:
  - Renumber the system variables
- During factorisation:
  - Reorder the system rows: row-pivoting
- Solve the factorised system
- After solution:
  - Un-renumber the system variables

Renumbering

- **Problem:** Given a symmetric sparse matrix A, find a permutation P such that the factorisation $U^T U = PAP^T$ has the least amount of fill-in in the Cholesky-factor U
- This problem is NP-complete (Yannakakis 1981)
- Heuristic algorithms provide approximately optimal reorderings
- Typical heuristics algorithms:
  - Minimum degree 1
  - Nested dissection 2
  - Reverse Cuthill-McKee 3

A greedy approximate minimum degree (AMD) -algorithm

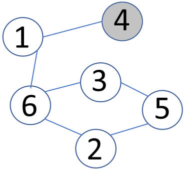
- Define the graph structure of the symmetric sparse matrix:
    - Nodes of the graph are equal to n the number of rows/columns
    - Edge between nodes i and j iff $a_{i,j} \neq 0$ and $i \neq j$ (no loops).
    - Matrix is symmetric so graph is undirected
- Define the degree of each node to be the number of connections it makes to other nodes
- Pick a starting node with minimal degree and renumber as 1
- For each renumbered node
    - Order the non-renumbered neighbours of that node by degree in ascending order
    - Renumber them in that sequence

Renumbering example

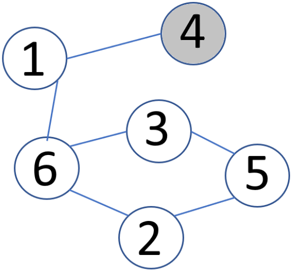
The sparse symmetric matrix

$$\mathbf{A} = \begin{bmatrix} 5 & & & 1 & & 1 \\ & 5 & & & 1 & 1 \\ & & 5 & & 1 & 1 \\ 1 & & & 5 & & \\ & 1 & 1 & & 5 & \\ 1 & 1 & 1 & & & 5 \end{bmatrix}$$

has the connectivity graph

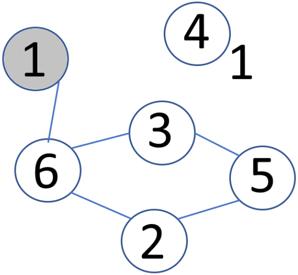


Renumbering example



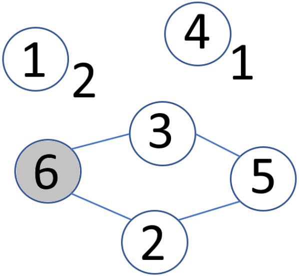
- Start from node 4, renumber it as 1.
- Only neighbour is node 1, so we pick it next.
- Eliminate node 4 from the connectivity graph.

Renumbering example



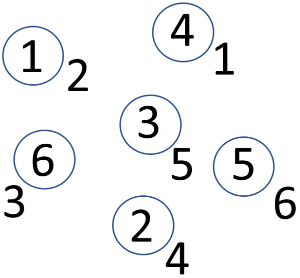
- Renumber node 1 as 2.
- Only neighbour is node 6, so we pick it next.
- Eliminate node 1 from the connectivity graph.

Renumbering example



- Neighbours of 6 are 2 and 3, both of which have $\deg = 2$.
- We can order them in any order, e.g. 2 becomes 4 and 3 becomes 5.

Renumbering example



- Neighbours of 6 are 2 and 3, both of which have deg = 2.
- We can order them in any order, e.g. 2 becomes 4 and 3 becomes 5.
- Finally node 5 becomes 6.

Renumbering example

Permutation matrix (by columns):

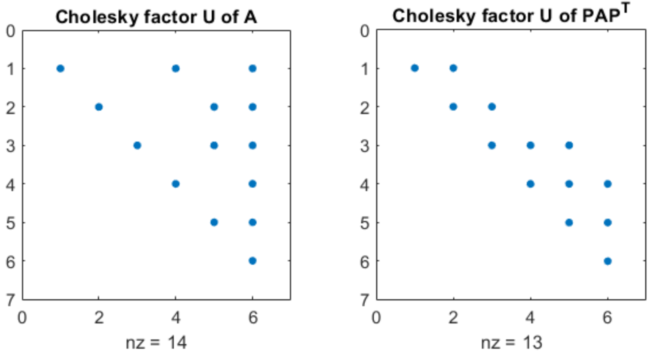
$$\mathbf{P} = \begin{bmatrix} & & & 1 & & \\ 1 & & & & & \\ & & & & & 1 \\ & 1 & & & & \\ & & 1 & & & \\ & & & & 1 & \end{bmatrix}$$

Symmetric permuted matrix:

$$\mathbf{PAP}^T = \begin{bmatrix} 5 & 1 & & & & \\ 1 & 5 & 1 & & & \\ & 1 & 5 & 1 & 1 & \\ & & 1 & 5 & & 1 \\ & & 1 & & 5 & 1 \\ & & & 1 & 1 & 5 \end{bmatrix}$$

Renumbering example

Cholesky factorisations $\mathbf{U}^T\mathbf{U} = \mathbf{A}$ and $\mathbf{U}^T\mathbf{U} = \mathbf{PAP}^T$:



The **U** factor has fewer nonzero elements (in fact, this is optimal)

Pivoting for Gaussian elimination

- Row-pivoting is a heuristic to minimise round-off error during factorisation
- Full-pivoting (simultaneous row and column) is precise but too complex for sparse matrices
- Both approaches seek a matrix entry of largest magnitude and then permute the matrix to make it the current pivot

Row-pivoting for dense $\mathbf{A}$ (partial pivoting)

- We solve the system in the form $\mathbf{P}\,\mathbf{A}\,\mathbf{x} \;=\; \mathbf{P}\,\mathbf{b}$
- At each row elimination step we modify $\mathbf{P}$ first if pivoting is required. For elimination of row i:
  - Find next pivot element $a_{j,i}$ for $j \geq i$ as either:
    (i) largest magnitude $\max_j |a_{j,i}|$, or
    (ii) least number of off-diagonal nonzero elements (AMD)
  - Construct permutation matrix $\mathbf{P}_i$ that swaps rows i and j

Row-pivoting for dense $\mathbf{A}$ (partial pivoting)

- We solve the system in the form $\mathbf{P}\,\mathbf{A}\,\mathbf{x} \;=\; \mathbf{P}\,\mathbf{b}$
- At each row elimination step we modify $\mathbf{P}$ first if pivoting is required. For elimination of row i:
  - Find next pivot element $a_{j,i}$ for $j \geq i$ as either:
    (i) largest magnitude $\max_j |a_{j,i}|$, or
    (ii) least number of off-diagonal nonzero elements (AMD)
  - Construct permutation matrix $\mathbf{P}_i$ that swaps rows i and j
  - Update $\mathbf{P} \to \mathbf{P}_i\mathbf{P}$

Row-pivoting for dense $\mathbf{A}$ (partial pivoting)

- We solve the system in the form $\mathbf{P}\ \mathbf{A}\ \mathbf{x}\ =\ \mathbf{P}\ \mathbf{b}$
- At each row elimination step we modify $\mathbf{P}$ first if pivoting is required. For elimination of row i:
  - Find next pivot element $a_{j,i}$ for $j \geq i$ as either:
    (i) largest magnitude $\max_j |a_{j,i}|$, or
    (ii) least number of off-diagonal nonzero elements (AMD)
  - Construct permutation matrix $\mathbf{P}_i$ that swaps rows i and j
  - Update $\mathbf{P} \to \mathbf{P}_i\mathbf{P}$
  - Apply row elimination to $\mathbf{P}\ \mathbf{A}$
  - Let $i \to i+1$ and iterate

Row-pivoting for dense **A** (partial pivoting)

- We solve the system in the form $\mathbf{P}\ \mathbf{A}\ \mathbf{x} \ = \ \mathbf{P}\ \mathbf{b}$
- At each row elimination step we modify **P** first if pivoting is required. For elimination of row i:
  - Find next pivot element $a_{j,i}$ for $j \geq i$ as either:
    (i) largest magnitude $\max_j |a_{j,i}|$, or
    (ii) least number of off-diagonal nonzero elements (AMD)
  - Construct permutation matrix $\mathbf{P}_i$ that swaps rows i and j
  - Update $\mathbf{P} \to \mathbf{P}_i\mathbf{P}$
  - Apply row elimination to $\mathbf{P}\ \mathbf{A}$
  - Let $i \to i + 1$ and iterate
- It can be shown that we end up with the factorisation:

$$\mathbf{L}^{-1}\left(\mathbf{P}_{n-1}\ldots\mathbf{P}_2\mathbf{P}_1\right)\mathbf{A} = \mathbf{U}$$

Row-pivoting for dense $\mathbf{A}$ (partial pivoting)

- We solve the system in the form $\mathbf{P}\ \mathbf{A}\ \mathbf{x}\ =\ \mathbf{P}\ \mathbf{b}$
- At each row elimination step we modify $\mathbf{P}$ first if pivoting is required. For elimination of row i:
  - Find next pivot element $a_{j,i}$ for $j \geq i$ as either:
    (i) largest magnitude $\max_j |a_{j,i}|$, or
    (ii) least number of off-diagonal nonzero elements (AMD)
  - Construct permutation matrix $\mathbf{P}_i$ that swaps rows i and j
  - Update $\mathbf{P} \to \mathbf{P}_i\mathbf{P}$
  - Apply row elimination to $\mathbf{P}\ \mathbf{A}$
  - Let $i \to i+1$ and iterate
- It can be shown that we end up with the factorisation:

$$\mathbf{L}^{-1}\left(\mathbf{P}_{n-1}\ldots\mathbf{P}_2\mathbf{P}_1\right)\mathbf{A} = \mathbf{U}$$

Row-pivoting for dense **A** (partial pivoting)

- We solve the system in the form $\mathbf{P}\ \mathbf{A}\ \mathbf{x} \ = \ \mathbf{P}\ \mathbf{b}$
- At each row elimination step we modify **P** first if pivoting is required. For elimination of row i:
  - Find next pivot element $a_{j,i}$ for $j \geq i$ as either:
    (i) largest magnitude $\max_j |a_{j,i}|$, or
    (ii) least number of off-diagonal nonzero elements (AMD)
  - Construct permutation matrix $\mathbf{P}_i$ that swaps rows i and j
  - Update $\mathbf{P} \to \mathbf{P}_i\mathbf{P}$
  - Apply row elimination to $\mathbf{P}\ \mathbf{A}$
  - Let $i \to i + 1$ and iterate
- It can be shown that we end up with the factorisation:

$$\mathbf{PA} = \mathbf{LU}$$

Row-pivoting for sparse **A**

- Row-pivoting implies swapping of rows which is non-trivial for sparse **A**
  - sparse column format ideal for elimination
  - sparse row format ideal for row-pivoting
- It can be achieved but the final algorithm is complex and not covered here
- It requires the sequence of pivoting operations to be stored and applied after factorisation is complete