

## Lecture 14: Direct solvers for linear systems

COMP5930M Scientific Computation

# Today

Motivation

Direct solvers

Sparse linear systems

Reordering the linear system

## The Newton algorithm

- ▶ A (large) linear system of equations  $Ax = b$  must be solved at each Newton iteration
- ▶ For  $n$  equations the classical algorithms (Gaussian elimination etc.) have  $\mathcal{O}(n^3)$  expense

## Direct solvers

- ▶ We define a direct solution algorithm as one which produces a solution in a fixed number of operations at fixed expense for a given problem size
- ▶ Typical direct solvers operate on the matrix  $A$  and the right-hand side  $b$  and apply algebraic operations to reduce the problem (matrix decompositions)
- ▶ Our goal is introduce direct methods to sparse systems that achieve a computational cost that is less than  $\mathcal{O}(n^3)$

分解

## Standard algorithms

- ▶ Gaussian elimination
  - ▶ Often the first method covered in linear algebra
  - ▶ Forward elimination of  $A$  into upper triangular form, followed by back substitution
  - ▶ Special case: Thomas algorithm for tridiagonal systems
- ▶ LU factorisation
  - ▶ Considered a more practical approach
  - ▶ Factorisation into upper/lower triangular blocks, followed by upper and lower triangular solves
- ▶ They are fundamentally **the same algorithm** with a different sequence of operations

## Standard algorithms

- ▶ Gaussian elimination
  - ▶ Often the first method covered in linear algebra
  - ▶ Forward elimination of  $A$  into upper triangular form, followed by back substitution
  - ▶ Special case: Thomas algorithm for tridiagonal systems
- ▶ LU factorisation
  - ▶ Considered a more practical approach
  - ▶ Factorisation into upper/lower triangular blocks, followed by upper and lower triangular solves
- ▶ They are fundamentally **the same algorithm** with a different sequence of operations
- ▶ Popular implementation: LAPACK

## Standard algorithms

- ▶ **Gaussian elimination**
  - ▶ Often the first method covered in linear algebra
  - ▶ Forward elimination of  $A$  into upper triangular form, followed by back substitution
  - ▶ Special case: Thomas algorithm for tridiagonal systems
- ▶ LU factorisation
  - ▶ Considered a more practical approach
  - ▶ Factorisation into upper/lower triangular blocks, followed by upper and lower triangular solves
- ▶ They are fundamentally the same algorithm with a different sequence of operations
- ▶ Popular implementation: LAPACK

5

## LU factorisation

For any square matrix  $A \in \mathbb{R}^{n \times n}$ , we look for a decomposition:

$$A = LU,$$

where  $L$  is lower triangular and  $U$  is upper triangular  $n \times n$  matrix.

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} = \begin{pmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ & U_{22} & U_{23} \\ & & U_{33} \end{pmatrix}$$

Implementation: Doolittle algorithm



## LU factorisation

If such a decomposition can be found, we proceed in two steps:

1. Solve  $Lz = b$  to find  $z$ ;
2. Solve  $Ux = z$  to find  $x$ .

These sub-problems are solved efficiently with  $\mathcal{O}(n^2)$  cost using forward/backward substitution ( $\mathcal{O}(n)$  in Thomas algorithm)

However: Cost of LU factorisation is  $\mathcal{O}(n^3)$  for dense matrices.

## LU factorisation

If such a decomposition can be found, we proceed in two steps:

1. Solve  $Lz = b$  to find  $z$ ;
2. Solve  $Ux = z$  to find  $x$ .

These sub-problems are solved efficiently with  $\mathcal{O}(n^2)$  cost using forward/backward substitution ( $\mathcal{O}(n)$  in Thomas algorithm)

However: Cost of LU factorisation is  $\mathcal{O}(n^3)$  for dense matrices.

## Sparse matrix problems

- ▶ Assumption: The matrix  $A$  is sparse
- ▶ Both Gauss elimination and LU factorisation algorithms have been extended to sparse matrices
- ▶ Large-scale, widely-used implementations for both
  - ▶ Gauss elimination: UMFPACK ✓
  - ▶ LU factorisation: SuperLU ✓
  - ▶ Multifrontal parallel LU factorisation: MUMPS ✓
  - ▶ Many other variants

## Improving the basic algorithms

- ▶ To improve from  $\mathcal{O}(n^3)$ , need to exploit sparsity of  $A$

- ▶ Column-based sparse algorithm: storage

- ▶ Reordering

- ▶ Fill-reduction: efficiency
- ▶ Pivoting: numerical accuracy



## Why reorder the unknowns of the problem?

### ► Efficiency

- The sparse structure of a factorised matrix is determined by the sparse structure of the matrix itself
- We can reorder the matrix to minimise the size of the factorised matrix

### ► Accuracy

- The diagonal entries (or pivots) in the factorisation algorithms are critical to the accuracy
- Round-off error is minimised if the pivot magnitude is controlled through reordering

## An example with round-off problems

Given some small  $\epsilon$ , solve  $\mathbf{Ax} = \mathbf{b}$  to find  $\mathbf{x}$

$$\begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 + \epsilon \\ 2 \end{pmatrix}$$

Standard Gaussian elimination

$$\begin{pmatrix} \epsilon & 1 \\ 0 & 1 - \frac{1}{\epsilon} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 + \epsilon \\ 2 - \frac{1 + \epsilon}{\epsilon} \end{pmatrix}$$

Back-substitution

$$x_2 = \frac{2 - \frac{1 + \epsilon}{\epsilon}}{1 - \frac{1}{\epsilon}}, \quad x_1 = \frac{1 + \epsilon - x_2}{\epsilon}$$

# Round-off errors without reordering



As  $\epsilon \rightarrow 0$ , we observe catastrophic round-off errors:

| $\epsilon$ | $x_2$    | $x_1$             |
|------------|----------|-------------------|
| $10^{-7}$  | 1        | 1.000000000583867 |
| $10^{-8}$  | 1        | 0.999999993922529 |
| $10^{-9}$  | 1        | 0.999999860695766 |
| $\vdots$   | $\vdots$ | $\vdots$          |
| $10^{-14}$ | 1        | 0.999200722162641 |
| $10^{-15}$ | 1        | 0.888178419700125 |
| $10^{-16}$ | 1        | 2.220446049250313 |

## Reorder the equations

2

ie. reorder the rows before elimination (row pivoting)

$$\begin{pmatrix} 1 & 1 \\ \epsilon & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 + \epsilon \end{pmatrix}$$

Standard elimination

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 - \epsilon \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 - \epsilon \end{pmatrix} \quad ||$$

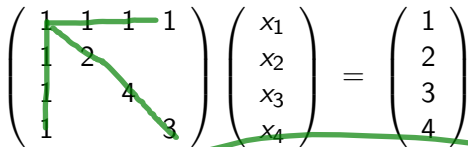
Back-substitution

$$x_2 = \frac{1 - \epsilon}{1 - \epsilon} = 1, \quad x_1 = 1$$



## An example with fill-in problems

Solve  $\mathbf{Ax} = \mathbf{b}$  to find  $\mathbf{x}$


$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & & \\ 1 & & 4 & \\ 1 & & & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

This kind of matrix is called an arrow matrix (very sparse).

## Standard algorithm

LU factorisation:

$$\mathbf{A} = \mathbf{LU} = \begin{pmatrix} 1 & & & \\ 1 & 1 & & \\ 1 & -1 & 1 & \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ & 1 & -1 & -1 \\ & & 2 & -2 \\ & & & -1 \end{pmatrix}$$

Solution:  $\mathbf{x} = (25, -11.5, -5.5, -7)^T$

LU factors of an arrow matrix are dense if no reordering is applied

# (1) Reorder the equations

ie. **reorder the rows**

Equations 1,2,3,4 → Equations 4,3,2,1

$$\begin{pmatrix} 1 & & & \\ 1 & 2 & & \\ 1 & 1 & 1 & \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

## (2) Reorder the variables

ie. **reorder the columns**

Variables  $x_1, x_2, x_3, x_4 \rightarrow$  variables  $x_4, x_3, x_2, x_1$

$$\begin{pmatrix} 3 & & & 1 \\ & 4 & & 1 \\ & & 2 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_4 \\ x_3 \\ x_2 \\ x_1 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

## Solve the reordered system

Standard LU factorisation:

$$\mathbf{LU} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} 3 & & 1 \\ & 4 & 1 \\ & & 2 & 1 \\ & & & -\frac{1}{12} \end{pmatrix}$$

Solution:  $(\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1)^T = (-7, -5.5, -11.5, 25)^T$

LU factors of reordered arrow matrix have sparsity pattern of  $A$