

Linear equation solvers

Direct methods (elimination methods)

Ivo Roghair, Martin van Sint Annaland

Chemical Process Intensification,
Eindhoven University of Technology

Today's outline

- ① Introduction
- ② Gauss elimination
- ③ Partial Pivoting
- ④ LU decomposition
- ⑤ Summary

Today's outline

- ① Introduction
- ② Gauss elimination
- ③ Partial Pivoting
- ④ LU decomposition
- ⑤ Summary

Overview

Goals

Today we are going to write a program, which can solve a set of linear equations

- The first method is called Gaussian elimination
- We will encounter some problems with Gaussian elimination
- Then LU decomposition will be introduced

Today's outline

- ① Introduction
- ② Gauss elimination
- ③ Partial Pivoting
- ④ LU decomposition
- ⑤ Summary

Define the linear system

Consider the system:

$$Ax = b$$

In general:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Desired solution:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \end{bmatrix}$$

Using row operations

- Use row operations to simplify the system. Eliminate element A_{21} by subtracting $A_{21}/A_{11} = d_{21}$ times row 1 from row 2.
- In this case, Row 1 is the pivot row, and A_{11} is the pivot element.

$$\left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ A_{21} & A_{22} & A_{23} & b_2 \\ A_{31} & A_{32} & A_{33} & b_3 \end{array} \right] \longrightarrow \left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ A_{31} & A_{32} & A_{33} & b_3 \end{array} \right]$$

Using row operations

Eliminate element A_{21} using $d_{21} = A_{21}/A_{11}$.

$$\left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ A_{21} & A_{22} & A_{23} & b_2 \\ A_{31} & A_{32} & A_{33} & b_3 \end{array} \right] \longrightarrow \left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ A_{31} & A_{32} & A_{33} & b_3 \end{array} \right]$$

Using row operations

Eliminate element A_{21} using $d_{21} = A_{21}/A_{11}$.

$$\left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ A_{21} & A_{22} & A_{23} & b_2 \\ A_{31} & A_{32} & A_{33} & b_3 \end{array} \right] \longrightarrow \left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ A_{31} & A_{32} & A_{33} & b_3 \end{array} \right]$$

- $d_{21} \rightarrow A_{21}/A_{11}$
- $A_{21} \rightarrow A_{21} - A_{11}d_{21}$
- $A_{22} \rightarrow A_{22} - A_{12}d_{21}$
- $A_{23} \rightarrow A_{23} - A_{13}d_{21}$
- $b_2 \rightarrow b_2 - b_1d_{21}$

Using row operations

Eliminate element A_{21} using $d_{21} = A_{21}/A_{11}$.

$$\left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ A_{21} & A_{22} & A_{23} & b_2 \\ A_{31} & A_{32} & A_{33} & b_3 \end{array} \right] \longrightarrow \left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ A_{31} & A_{32} & A_{33} & b_3 \end{array} \right]$$

- $d_{21} \rightarrow A_{21}/A_{11}$
- $A_{21} \rightarrow A_{21} - A_{11}d_{21}$
- $A_{22} \rightarrow A_{22} - A_{12}d_{21}$
- $A_{23} \rightarrow A_{23} - A_{13}d_{21}$
- $b_2 \rightarrow b_2 - b_1d_{21}$

```
d21 = A(2,1)/A(1,1);  
A(2,1) = A(2,1) - A(1,1)*d21;  
A(2,2) = A(2,2) - A(1,2)*d21;  
A(2,3) = A(2,3) - A(1,3)*d21;  
b(2) = b(2) - b(1)*d21;
```

Using row operations

Eliminate element A_{31} using $d_{31} = A_{31}/A_{11}$.

$$\left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ A_{31} & A_{32} & A_{33} & b_3 \end{array} \right] \longrightarrow \left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ 0 & A'_{32} & A'_{33} & b'_3 \end{array} \right]$$

Using row operations

Eliminate element A_{31} using $d_{31} = A_{31}/A_{11}$.

$$\left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ A_{31} & A_{32} & A_{33} & b_3 \end{array} \right] \longrightarrow \left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ 0 & A'_{32} & A'_{33} & b'_3 \end{array} \right]$$

- $d_{31} \rightarrow A_{31}/A_{11}$
- $A_{31} \rightarrow A_{31} - A_{11}d_{31}$
- $A_{32} \rightarrow A_{32} - A_{12}d_{31}$
- $A_{33} \rightarrow A_{33} - A_{13}d_{31}$
- $b_3 \rightarrow b_3 - b_1d_{31}$

```
d31 = A(3,1)/A(1,1);  
A(3,1) = A(3,1) - A(1,1)*d31;  
A(3,2) = A(3,2) - A(1,2)*d31;  
A(3,3) = A(3,3) - A(1,3)*d31;  
b(3) = b(3) - b(1)*d31;
```

Using row operations

Eliminate element A_{32} using $d_{32} = A_{32}/A'_{22}$. Note that now the second row has become the pivot row.

$$\left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ 0 & A_{32} & A_{33} & b_3 \end{array} \right] \longrightarrow \left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ 0 & 0 & A''_{33} & b''_3 \end{array} \right]$$

Using row operations

Eliminate element A_{32} using $d_{32} = A_{32}/A'_{22}$. Note that now the second row has become the pivot row.

$$\left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ 0 & A_{32} & A_{33} & b_3 \end{array} \right] \longrightarrow \left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ 0 & 0 & A''_{33} & b''_3 \end{array} \right]$$

- $d_{32} \rightarrow A_{32}/A'_{22}$
- $A_{32} \rightarrow A_{32} - A'_{22}d_{32}$
- $A_{33} \rightarrow A_{33} - A'_{23}d_{32}$
- $b_3 \rightarrow b_3 - b'_2d_{32}$

```
d32 = A(3,2)/A(2,2);  
A(3,2) = A(3,2) - A(2,2)*d32;  
A(3,3) = A(3,3) - A(2,3)*d32;  
b(3) = b(3) - b(2)*d32;
```

Using row operations

Eliminate element A_{32} using $d_{32} = A_{32}/A'_{22}$. Note that now the second row has become the pivot row.

$$\left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ 0 & A_{32} & A_{33} & b_3 \end{array} \right] \longrightarrow \left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & b_1 \\ 0 & A'_{22} & A'_{23} & b'_2 \\ 0 & 0 & A''_{33} & b''_3 \end{array} \right]$$

- $d_{32} \rightarrow A_{32}/A'_{22}$
- $A_{32} \rightarrow A_{32} - A'_{22}d_{32}$
- $A_{33} \rightarrow A_{33} - A'_{23}d_{32}$
- $b_3 \rightarrow b_3 - b'_2d_{32}$

```
d32 = A(3,2)/A(2,2);  
A(3,2) = A(3,1) - A(2,2)*d32;  
A(3,3) = A(3,2) - A(2,3)*d32;  
b(3) = b(3) - b(2)*d32;
```

The matrix is now a triangular matrix — the solution can be obtained by back-substitution.

Backsubstitution

The system now reads:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & 0 & A''_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b''_3 \end{bmatrix}$$

Backsubstitution

The system now reads:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & 0 & A''_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b''_3 \end{bmatrix}$$

Start at the last row N , and work upward until row 1.

$$x_3 = b''_3 / A''_{33}$$

$$x_2 = (b'_2 - A'_{23}x_3) / A'_{22}$$

$$x_1 = (b_1 - A_{12}x_2 - A_{13}x_3) / A_{11}$$

Backsubstitution

The system now reads:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & 0 & A''_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b''_3 \end{bmatrix}$$

Start at the last row N , and work upward until row 1.

$$x_3 = b''_3 / A''_{33}$$

$$x_2 = (b'_2 - A'_{23}x_3) / A'_{22}$$

$$x_1 = (b_1 - A_{12}x_2 - A_{13}x_3) / A_{11}$$

$$\begin{aligned} x(3) &= b(3) / A(3,3) \\ x(2) &= (b(2) - A(2,3)*x(3)) / \\ &\quad A(2,2) \\ x(1) &= (b(1) - A(1,2)*x(2) - \\ &\quad A(1,3)*x(3)) / A(1,1) \end{aligned}$$

In general:

$$x_N = \frac{b_N}{A_{NN}} \quad x_i = \frac{b_i - \sum_{j=i+1}^N A_{ij}x_j}{A_{ii}}$$

Writing the program

- Create a function that provides the framework: take matrix A and vector b as an input, and return the solution x as output:

```
function [x,A,b] = GaussianEliminate(A,b)
```

- We will use *for-loops* instead of typing out each command line.
- Useful Matlab shortcuts:
 - $A(1,:) = [A_{11}, A_{12}, A_{13}]$
 - $A(:,2) = [A_{12}, A_{22}, A_{32}]^T$
 - $A(1,2:\text{end}) = [A_{12}, A_{13}]$
- A row operation could look like:

```
A(i,:) = A(i,:) - d*A(1,:)
```

The program: elimination

```
function [x,A,b] = GaussianEliminate(A,b)

% Perform elimination to obtain an upper triangular
  matrix
N = length(b);
for column=1:(N-1) % Select pivot
    for row=(column+1):N % Loop over subsequent rows (
        below pivot)
        d=A(row,column)/A(column,column);
        A(row,:)=A(row,:)-d*A(column,:);
        b(row)= b(row)-d*b(column);
    end
end
```

The program: Backsubstitution

```
% Assign b to x
x=b;

% Perform backsubstitution
for row=N:-1:1
    x(row) = b(row);
    for i =(row+1):N
        x(row)=x(row)-A(row,i)*x(i);
    end
    x(row)=x(row)/A(row,row);
end
```

$$x_N = \frac{b_N}{A_{NN}} \quad x_i = \frac{b_i - \sum_{j=i+1}^N A_{ij}x_j}{A_{ii}}$$

Exercise: Gaussian Elimination

- The function we just made can be found on Canvas
- Use `help GaussianEliminate` to find out how it works
- Solve the following system of equations:

$$\begin{bmatrix} 9 & 9 & 5 & 2 \\ 6 & 7 & 1 & 3 \\ 6 & 4 & 3 & 5 \\ 2 & 6 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \\ 10 \\ 1 \end{bmatrix}$$

- Compare your solution with $A \setminus b$

Today's outline

- ① Introduction
- ② Gauss elimination
- ③ Partial Pivoting
- ④ LU decomposition
- ⑤ Summary

Partial pivoting

- Now try to run the algorithm with the following system:

$$\begin{bmatrix} 0 & 2 & 1 \\ 3 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 10 \end{bmatrix}$$

Partial pivoting

- Now try to run the algorithm with the following system:

$$\begin{bmatrix} 0 & 2 & 1 \\ 3 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 10 \end{bmatrix}$$

- It does not work! Division by zero, due to $A_{11} = 0$.
- Solution: Swap rows to move largest element to the diagonal.

Partial pivoting: implementing row swaps

- Find maximum element row below pivot in current column

```
[dummy,index] = max(abs(A(column:end,column)));  
Index = index+column-1;
```

Partial pivoting: implementing row swaps

- Find maximum element row below pivot in current column
- Store current row

```
[dummy,index] = max(abs(A(column:end,column)));  
Index = index+column-1;
```

```
temp = A(column,:);
```

Partial pivoting: implementing row swaps

- Find maximum element row below pivot in current column
- Store current row
- Swap pivot row and desired row in A

```
[dummy,index] = max(abs(A(column:end,column)));  
Index = index+column-1;
```

```
temp = A(column,:);
```

```
A(column,:) = A(index,:);  
A(index,:) = temp;
```

Partial pivoting: implementing row swaps

- Find maximum element row below pivot in current column
- Store current row
- Swap pivot row and desired row in A
- Do the same for b: store and swap

```
[dummy, index] = max(abs(A(column:end, column)));  
Index = index + column - 1;
```

```
temp = A(column, :);
```

```
A(column, :) = A(index, :);  
A(index, :) = temp;
```

```
temp = b(column);  
b(column) = b(index);  
b(index) = temp;
```

Improve the program by using re-usable functions

```
function [x] = GaussianEliminate(A,b)
% GaussianEliminate(A,b): solves x in Ax=b
N = length(b);
for c=1:(N-1)
    [dummy,index]=max(abs(A(c:end,c)));
    index=index+c-1;
    A = SWAP(A,c,index); % Created swap function
    b = SWAP(b,c,index);
    for row=(column+1):N
        d=A(row,column)/A(column,column);
        A(row,:)=A(row,:)-d*A(column,:);
        b(row)= b(row)-d*b(column);
    end
end
x = backwardSub(A,b); % Created BS function
return
```

This function is also provided (named GaussianEliminate_v2 and GaussianEliminate_v3 on Canvas).

Alternatives to this program

- MATLAB can compute the solution to $Ax=b$ with its own solvers (more efficient) $A \setminus b$
- Too many loops. Loops make MATLAB slow.
- There are fundamental problems with Gaussian elimination

Alternatives to this program

- MATLAB can compute the solution to $Ax=b$ with its own solvers (more efficient) $A \setminus b$
- Too many loops. Loops make MATLAB slow.
- There are fundamental problems with Gaussian elimination
 - You can add a counter to the algorithm to see how many subtraction and multiplication operations it performs for a given size of matrix A .
 - The number of operations to perform Gaussian elimination is $\mathcal{O}(2N^3)$ (where N is the number of equations)
 - Exercise: verify this for our script

Alternatives to this program

- MATLAB can compute the solution to $Ax=b$ with its own solvers (more efficient) $A \setminus b$
- Too many loops. Loops make MATLAB slow.
- There are fundamental problems with Gaussian elimination
 - You can add a counter to the algorithm to see how many subtraction and multiplication operations it performs for a given size of matrix A .
 - The number of operations to perform Gaussian elimination is $\mathcal{O}(2N^3)$ (where N is the number of equations)
 - Exercise: verify this for our script
 - LU decomposition takes $\mathcal{O}(2N^3/3)$ flops, 3 times less!
 - Forward and backward substitution each take $\mathcal{O}(N^2)$ flops (both cases)

Today's outline

- ① Introduction
- ② Gauss elimination
- ③ Partial Pivoting
- ④ LU decomposition
- ⑤ Summary

LU Decomposition

Suppose we want to solve the previous set of equations, but with several right hand sides:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} \vdots & \vdots & \vdots \\ x_1 & x_2 & x_3 \\ \vdots & \vdots & \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots \\ b_1 & b_2 & b_3 \\ \vdots & \vdots & \vdots \end{bmatrix}$$

LU Decomposition

Suppose we want to solve the previous set of equations, but with several right hand sides:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} \vdots & \vdots & \vdots \\ x_1 & x_2 & x_3 \\ \vdots & \vdots & \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots \\ b_1 & b_2 & b_3 \\ \vdots & \vdots & \vdots \end{bmatrix}$$

Factor the matrix A into two matrices, L and U , such that $A = LU$:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \times & 1 & 0 \\ \times & \times & 1 \end{bmatrix} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix}$$

Now we can solve for each right hand side, using only a forward followed by a backward substitution!

Substitutions

- Define a lower and upper matrix L and U so that $A = LU$
- Therefore $LUx = b$
- Define a new vector $y = Ux$ so that $Ly = b$
- Solve for y , use L and forward substitution
- Then we have y , solve for x , use $Ux = y$
- Solve for x , use U and backward substitution
- But how to get L and U ?

Decomposing the matrix (1)

When we eliminate the element A_{21} we can keep multiplying by a matrix that undoes this row operations, so that the product remains equal to A .

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ d_{21} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A_{22} - d_{21}A_{12} & A_{23} - d_{21}A_{13} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

Decomposing the matrix (2)

When we eliminate the element A_{31} we can keep multiplying by a matrix that undoes this row operations, so that the product remains equal to A .

$$A = \begin{bmatrix} 1 & 0 & 0 \\ d_{21} & 1 & 0 \\ d_{31} & 0 & 1 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A'_{22} = A_{22} - d_{21}A_{12} & A'_{23} = A_{23} - d_{21}A_{13} \\ 0 & A'_{32} = A_{32} - d_{31}A_{12} & A'_{33} = A_{33} - d_{31}A_{13} \end{bmatrix}$$

Decomposing the matrix (3)

When we eliminate the element A_{32} we can keep multiplying by a matrix that undoes this row operations, so that the product remains equal to A .

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ d_{21} & 1 & 0 \\ d_{31} & d_{32} & 1 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & A'_{32} & A''_{33} = A'_{33} - d_{32}A'_{23} \end{bmatrix}$$

Decomposing the matrix (3)

When we eliminate the element A_{32} we can keep multiplying by a matrix that undoes this row operations, so that the product remains equal to A .

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ d_{21} & 1 & 0 \\ d_{31} & d_{32} & 1 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & A'_{32} & A''_{33} = A'_{33} - d_{32}A'_{23} \end{bmatrix}$$

This finishes the LU decomposition!

Pivoting during decomposition

Suppose we have arrived at the situation below, where $A'_{32} > A'_{22}$:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ d_{21} & 1 & 0 \\ d_{31} & 0 & 1 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & A'_{32} & A'_{33} \end{bmatrix}$$

Pivoting during decomposition

Suppose we have arrived at the situation below, where $A'_{32} > A'_{22}$:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ d_{21} & 1 & 0 \\ d_{31} & 0 & 1 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & A'_{32} & A'_{33} \end{bmatrix}$$

Exchange rows 2 and 3 to get the largest value on the main diagonal. Use a permutation matrix to store the swapped rows:

Pivoting during decomposition

Suppose we have arrived at the situation below, where $A'_{32} > A'_{22}$:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ d_{21} & 1 & 0 \\ d_{31} & 0 & 1 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & A'_{32} & A'_{33} \end{bmatrix}$$

Exchange rows 2 and 3 to get the largest value on the main diagonal. Use a permutation matrix to store the swapped rows:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ d_{31} & 0 & 1 \\ d_{21} & 1 & 0 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A'_{32} & A'_{33} \\ 0 & A'_{22} & A'_{23} \end{bmatrix}$$

Pivoting during decomposition

Suppose we have arrived at the situation below, where $A'_{32} > A'_{22}$:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ d_{21} & 1 & 0 \\ d_{31} & 0 & 1 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & A'_{32} & A'_{33} \end{bmatrix}$$

Exchange rows 2 and 3 to get the largest value on the main diagonal. Use a permutation matrix to store the swapped rows:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ d_{31} & 0 & 1 \\ d_{21} & 1 & 0 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A'_{32} & A'_{33} \\ 0 & A'_{22} & A'_{23} \end{bmatrix}$$

Multiplying with a permutation matrix will swap the rows of a matrix. The permutation matrix is just an identity matrix, whose rows have been interchanged.

Recipe for LU decomposition

When decomposing matrix A into $A = LU$, it may be beneficial to swap rows to get the largest values on the diagonal of U (pivoting). A permutation matrix P is used to store row swapping such that:

$$PA = LU$$

- Write down a permutation matrix and the linear system
- Promote the largest value in the column diagonal
- Eliminate all elements below diagonal
- Move on to the next column and move largest elements to diagonal
- Eliminate elements below diagonal
- Repeat 5 and 6
- Write down L,U and P

LU decomposition example (1)

Write down a permutation matrix, which starts as the identity matrix, and the linear system:

$$PA = LU$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix}$$

LU decomposition example (1)

Write down a permutation matrix, which starts as the identity matrix, and the linear system:

$$PA = LU$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix}$$

Promote the largest value into the diagonal of column 1 — swap row 1 and 2:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix}$$

LU decomposition example (2)

Eliminate all **elements below the diagonal** — row 2 already contains a zero in column 1, row 3 = row 3 - 0.5 row 1. Record the **multiplier 0.5** in L :

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \mathbf{0.5} & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 \\ \mathbf{0} & 1 & 1 \\ \mathbf{0} & 1.5 & -0.5 \end{bmatrix}$$

LU decomposition example (2)

Eliminate all **elements below the diagonal** — row 2 already contains a zero in column 1, row 3 = row 3 - 0.5 row 1. Record the **multiplier 0.5** in L :

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \mathbf{0.5} & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 \\ \mathbf{0} & 1 & 1 \\ \mathbf{0} & 1.5 & -0.5 \end{bmatrix}$$

Elimination of column 1 is done. Step to the next column, and move the largest value below/on the diagonal to the diagonal (**swap rows 2 and 3**). Adjust P and **lower triangle of L** accordingly:

$$\begin{bmatrix} 0 & 1 & 0 \\ \mathbf{0 & 0 & 1} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 2 & 1 & 0 \\ 1 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \mathbf{0.5} & 1 & 0 \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 \\ \mathbf{0} & \mathbf{1.5} & \mathbf{-0.5} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

LU decomposition example (3)

Eliminate all elements below the diagonal —

row 3 = row 3 - $\frac{2}{3}$ row 2. Record the multiplier $\frac{2}{3}$ in L:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 2 & 1 & 0 \\ 1 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0 & \frac{2}{3} & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1.5 & -0.5 \\ 0 & 0 & \frac{4}{3} \end{bmatrix}$$

LU decomposition example (3)

Eliminate all elements below the diagonal —

row 3 = row 3 - $\frac{2}{3}$ row 2. Record the multiplier $\frac{2}{3}$ in L :

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 2 & 1 & 0 \\ 1 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0 & \frac{2}{3} & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1.5 & -0.5 \\ 0 & 0 & \frac{4}{3} \end{bmatrix}$$

We have obtained the matrices from $PA = LU$:

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0 & \frac{2}{3} & 1 \end{bmatrix} \quad U = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1.5 & -0.5 \\ 0 & 0 & \frac{4}{3} \end{bmatrix}$$

Proceed with solving for x .

Substitutions

$$\begin{aligned} Ax = b &\Rightarrow PAx = Pb \equiv d \\ PA = LU &\Rightarrow LUx = d \end{aligned}$$

- Define a new vector $y = Ux$
 - $Ly = b \Rightarrow Ly = d$
 - Solve for y , forward substitution:

$$\begin{aligned} y_1 &= \frac{d_1}{L_{11}} \\ y_i &= \frac{d_i - \sum_{j=1}^{i-1} L_{ij}y_j}{L_{ii}} \end{aligned}$$

- Then solve $Ux = y$:
 - Solve for x , backward substitution:

$$\begin{aligned} x_N &= \frac{y_N}{U_{NN}} \\ x_i &= \frac{y_i - \sum_{j=i+1}^{N-1} U_{ij}x_j}{U_{ii}} \end{aligned}$$

How to use the solver in Matlab

```
A = rand(5,5);           % Get random matrix
[L, U, P] = lu(A);        % Get L, U and P
b = rand(5,1);           % Random b vector
d = P*b;                  % Permute b vector
y = forwardSub(L,d);      % Can also do y=L\d
x = backwardSub(U,y);     % Can also do x=U\y
rnorm = norm(A*x - b);    % Residual

% Compare results to internal Matlab solver
x = A\b
```

How to use the solver in Matlab

```
A = rand(5,5);           % Get random matrix
[L, U, P] = lu(A);        % Get L, U and P
b = rand(5,1);           % Random b vector
d = P*b;                 % Permute b vector
y = forwardSub(L,d);      % Can also do y=L\d
x = backwardSub(U,y);     % Can also do x=U\y
rnorm = norm(A*x - b);   % Residual

% Compare results to internal Matlab solver
x = A\b
```

- Use this as a basis to create a function that takes A and b , and returns x .
- Use the function to check the performance for various matrix sizes and inspect the performance.

Today's outline

- ① Introduction
- ② Gauss elimination
- ③ Partial Pivoting
- ④ LU decomposition
- ⑤ Summary

Summary

- This lecture covered direct methods using elimination techniques.
- Gaussian elimination can be slow ($\mathcal{O}(N^3)$)
- Back substitution is often faster ($\mathcal{O}(N^2)$)
- LU decomposition means that we don't have to do Gaussian elimination every time (saves time and effort), but the matrix has to stay the same.
- Matlab has build in routines for solving linear equations (backslash operator `\`) and LU decomposition (`lu`).
- Advanced techniques such as (preconditioned) conjugate gradient or biconjugate gradient solvers are also available.
- Next part covers iterative approaches