ELIMINATION METHODS

Numerical methods in chemical engineering

Ivo Roghair

Technische Universiteit Eindhoven
University of Technology

Where innovation starts

OVERVIEW

- We are going to write a program, which can solve a set of linear equations, using the method of Gaussian elimination
- We'll encounter some problems with Gaussian elimination
- Then LU decomposition will be introduced

Gaussian Elimination (Gauss-Jordan)

- **You've probably learned this method in high-school**
- **An example:**

$$Ax = b$$

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

<u>A=[1 1 1; 2 1 1; 1 2 0]</u>

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

<u>b= [1 1 1]</u>'

Let's do row operations!

- We use row operations to simplify the system. E.g. 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.

Eliminate element A_{21}

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

- `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`

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

Eliminate element A_{31}

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

- `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`

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

ELIMINATE A_{32}

- $A_{32} \rightarrow A_{32} - A_{22}d_{32}$
- $A_{33} \rightarrow A_{33} - A_{23}d_{32}$
- $b_3 \rightarrow b_3 - b_2d_{32}$

- `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)*d31`

We obtained a triangular matrix, solution can be obtained by back substitution!

$$\left[\begin{array}{ccc|c} A_{11} & A_{21} & A_{31} & b_1 \\ 0 & A_{22}' & A_{32}' & b_2' \\ \boxed{0} & \boxed{0} & \boxed{A_{33}''} & \boxed{b_3''} \end{array}\right]$$

Backsubstitution: start at i=N and work up to i=1

- `x(3)=b(3)/A(3,3)`
- `x(2)=(b(2)–A(2,3)*x(3))/A(2,2)`
- `x(1)=(b(1)-A(1,2)*x(2)-A(1,3)*x(3))/A(1,1)`

$$x_i = \frac{b_i}{U_{i,i}}$$

$$x_i = \frac{b_i - \sum_{j=i+1}^{N} U_{i,j} x_j}{U_{i,i}}$$

$$x_3 = b_3'' / A_{33}''$$

$$x_2 = \left(b_2' - A_{23}' x_3\right) / A_{22}'$$

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

WRITING THE PROGRAM

- **We will use "for loops" instead of typing out each command line.**
- **Handy to know:**
  - `A(1,:)` $=[A_{11},A_{12},A_{13}]$
  - `A(:,2)` $=[A_{12},A_{22},A_{32}]^T$
  - `A(1,2:end)` $=[A_{12},A_{13}]$
- **A row operation could look like:**
  - `A(i,:) = A(i,:) - 2*A(1,:)`

First part of the program (elimination)

```
function [x] = GaussianEliminate(A,b)
N = length(b);
for column=1:(N-1)
      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
```

Second part of the program (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=x';
return
```

$$x_i = \frac{b_i}{U_{i,i}}$$

$$x_i = \frac{b_i - \sum_{j=i+1}^{N} U_{i,j} x_j}{U_{i,i}}$$

Exercise: Gaussian Elimination

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

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

- Compare your solution with A\b

PARTIAL PIVOTING

- **Now try to run the algorithm with the following matrix (and any b):**

$$A = \begin{bmatrix} 0 & 2 & 1 \\ 3 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Swap rows to move largest element to the diagonal, values for d will always be less then one: reduction of accumulated numerical error!!

Does it work?

Division by zero! Yields NaN!

Solution: swap row 1 with row 2…

EXTENSION OF THE PROGRAM

- **Can you think of a procedure to do this?**
  - 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;
```

Making the program nicer by moving generic code to 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
```

Alternatives for this program

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

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 $2N^3$ (where N is the number of equations)
  - Exercise: verify this for our script
- LU decomposition takes $2N^3/3$ flops, 3 times less!
- Forward and backward substitution each take N^2 flops (both cases)

LU DECOMPOSITION

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

$$Ax_1 = b_1, Ax_2 = b_2, Ax_2 = b_2$$

$$A\begin{bmatrix} \vdots & \vdots & \vdots \\ x_1 & x_2 & x_3 \\ \vdots & \vdots & \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots \\ b_1 & b_2 & b_3 \\ \vdots & \vdots & \vdots \end{bmatrix}$$

FACTORIZING A

- **Let's factor our matrix A into two matrices, L and U, such that:**

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

$$A = LU$$

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

SUBSTITUTIONS

- $Ax = b$
- 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, i=N:-1:1)

- But how to get L and U?

LET'S DECOMPOSE!

- **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.**

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

NEXT STEP

- **When we eliminate the element A_{31}, we can multiply by a matrix that undoes this operation, so that the product remains equal to A.**

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

SO, FINISHED LU DECOMPOSITION

- **When we eliminate the element A_{32}, we can multiply by a matrix that undoes this row operation, so that the product remains equal to A.**

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

ROW SWAPPING DURING ELIMINATION

- **Suppose we have obtained the following stage in the elimination process:**
- **We need to exchange rows 2 and 3 …**

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

THE PERMUTATION MATRIX

- **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.**

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

RECIPE FOR LU DECOMPOSITION

1. Write down a permutation matrix
2. Write down the matrix to decompose
3. Promote the largest value in the column diagonal
4. Eliminate all elements below diagonal
5. Move on to the next column and move largest elements to diagonal
6. Eliminate elements below diagonal
7. Repeat 5 and 6
8. Write down L,U and P

WRITE DOWN A PERMUTATION MATRIX

- **Write down a permutation matrix P, initially the identity matrix:**

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

WRITE DOWN MATRIX TO DECOMPOSE

- **Write down the matrix you want to decompose, e.g.:**

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

PROMOTE LARGEST VALUE IN THE DIAGONAL

- **Starting with column 1, row swap to promote the largest value in the column to the diagonal. Do exactly the same row swap to matrix P:**

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

ELIMINATE ALL ELEMENTS BELOW DIAGONAL

- **Eliminate all the elements below the diagonal in column 1. Record the multiplier *d* used for elimination where you create the zero:**

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

Here we did: **row 3 = row 3 -0.5*row 1**

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

MOVE ON TO NEXT COLUMN

- **Move on to the next column. Swap rows to move the largest element to the diagonal (do the same row swap to P)**

$$\begin{bmatrix} 2 & 1 & 1 \\ \boxed{0.5} & 1.5 & -0.5 \\ \boxed{0} & 1 & 1 \end{bmatrix} \Rightarrow P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

ELEMINATE ELEMENTS BELOW DIAGONAL

- **Eliminate the elements below the diagonal:**

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

Row 3 = row 3 – 2/3*row 2

WRITE DOWN L,U AND P

- **Repeat 5 and 6 for all columns**
- **Write down L,U and P**

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

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

SUBSTITUTIONS

- $Ax = b \quad \Leftrightarrow \quad PAx = Pb \equiv d$
- $LUx = b \quad \Leftrightarrow \quad LUx = d$
- **Define a new vector $y = Ux$**
  - $Ly = b \quad \Leftrightarrow \quad Ly = d$
  - **Solve for y, use forward substitution, i=1:N**

$$x_1 = \frac{b_1}{L_{1,1}}$$

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} L_{i,j} x_j}{L_{i,i}}$$

- **Then solve**
  - $Ux = y$
  - **Solve for x, use backward substitu-tion, i=N:-1:1)**

$$x_i = \frac{b_i}{U_{i,i}}$$

$$x_i = \frac{b_i - \sum_{j=i+1}^{N} U_{i,j} x_j}{U_{i,i}}$$

- **So… No need to decompose the matrix again for different b!**

Now… How can we use this further?

```
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
```

Technische Universiteit Eindhoven University of Technology

SUMMARY

- Gaussian elimination can be slow (N^3)
- Back substitution is often faster (N^2)
- LU decomposition means that we don’t have to do Gaussian elimination every time (saves time and effort)
- MATLAB has build in routines for solving linear equations (\) and LU decomposition (LU).