18-660: Numerical Methods for Engineering Design and Optimization

Xin Li
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213

Electrical & Computer ENGINEERING

Overview

- Linear Equation Solver
  - Gaussian elimination
  - Condition number
  - Full/partial pivoting

Linear Equation

- Ordinary differential equation

$$\dot{x}(t) = A \cdot x(t) + B \cdot u(t) \quad x(0) = 0$$

Backward Euler

$$x(t_{n+1}) = (I - \Delta t \cdot A)^{-1} \cdot [x(t_n) + \Delta t \cdot B \cdot u(t_{n+1})] \quad x(t_0) = 0$$

- Partial differential equation

$$\rho \cdot C_p \cdot \frac{\partial T(x,y,z,t)}{\partial t} = \kappa \cdot \nabla^2 T(x,y,z,t) + f(x,y,z,t)$$

Finite Difference

$$\rho \cdot C_p \cdot \frac{\partial T_{i,j,k}}{\partial t} = f_{i,j,k} + \frac{\kappa \cdot [T_{i+1,j,k} - T_{i,j,k}]}{(\Delta x)^2} - \frac{\kappa \cdot [T_{i,j,k} - T_{i-1,j,k}]}{(\Delta x)^2} +$$
$$\frac{\kappa \cdot [T_{i,j+1,k} - T_{i,j,k}]}{(\Delta y)^2} - \frac{\kappa \cdot [T_{i,j,k} - T_{i,j-1,k}]}{(\Delta y)^2} + \frac{\kappa \cdot [T_{i,j,k+1} - T_{i,j,k}]}{(\Delta z)^2} - \frac{\kappa \cdot [T_{i,j,k} - T_{i,j,k-1}]}{(\Delta z)^2}$$

Linear Equation Solver

$$A \cdot X = B$$

- In theory, X is equal to $A^{-1}B$
- In practice, explicitly inverting a matrix is never a good idea
- A more efficient algorithm should be applied
  - E.g., use X = A\B in MATLAB

Gaussian Elimination

- Step 1: convert A to an upper triangular matrix

$$[A] \cdot [X] = [B] \quad \Rightarrow \quad [U] \cdot [X] = [Y]$$

- Step 2: solve for X via backward substitution

$$[U] \cdot [X] = [Y] \quad \Rightarrow \quad [X]$$

Gaussian Elimination

- A simple example

$$\underbrace{\begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{X} = \underbrace{\begin{bmatrix} 8 \\ -11 \\ -3 \end{bmatrix}}_{B}$$

Gaussian Elimination

- Step 1: convert A to an upper triangular matrix

$$\begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ -11 \\ -3 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 2 & 1 & -1 \\ 0 & 0.5 & 0.5 \\ 0 & 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \\ 5 \end{bmatrix}$$

Gaussian Elimination

- Step 1: convert A to an upper triangular matrix

$$\begin{bmatrix} 2 & 1 & -1 \\ 0 & 0.5 & 0.5 \\ 0 & 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \\ 5 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 2 & 1 & -1 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \\ 1 \end{bmatrix}$$

Gaussian Elimination

- Step 2: solve for X via backward substitution

$$\begin{bmatrix} 2 & 1 & -1 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \\ 1 \end{bmatrix}$$

$$x_3 = -1$$

$$0.5 \cdot x_2 - 0.5 = 1$$
$$x_2 = 3$$

$$2 \cdot x_1 + 3 + 1 = 8$$
$$x_1 = 2$$

Gaussian Elimination

- Gaussian elimination is much cheaper than calculating A^{-1}

$$\begin{bmatrix} & A & \end{bmatrix} \cdot \begin{bmatrix} X \end{bmatrix} = \begin{bmatrix} B \end{bmatrix}$$

Gaussian elimination: solve for X where B is an N x 1 vector

$$\begin{bmatrix} & A & \end{bmatrix} \cdot \begin{bmatrix} & A^{-1} & \end{bmatrix} = \begin{bmatrix} & I & \end{bmatrix}$$

Matrix inverse: solve for A^{-1} where I is an N x N identity matrix

The difference between Gaussian elimination and matrix inverse is significant for large matrix

Numerical Noise

- In theory, Gaussian elimination works well if A is nonsingular, i.e.,

$$A \cdot X = B \quad where \quad \det(A) \neq 0$$

  - A is singular if and only if det(A) = 0

- However, round-off errors in our numerical computation can bring about problems even if det(A) is not 0
  - Numerical noise can change the determinant value for Gaussian elimination

Numerical Noise

- A simple example

$$A = \begin{bmatrix} 100 & -100 \\ -100 & 100.01 \end{bmatrix} \qquad \det(A) = 100 \cdot 100.01 - 100 \cdot 100 = 1$$

  - If our machine only has 3 decimal digits of precision

$$A \approx \begin{bmatrix} 100 & -100 \\ -100 & 100 \end{bmatrix} \qquad \det(A) = 100 \cdot 100 - 100 \cdot 100 = 0$$

Condition Number

- The "singularity" of a linear equation can be quantitatively measured by its condition number

$$A \cdot X = B$$

- The condition number of A is defined as:

$$k(A) = \|A\| \cdot \|A^{-1}\|$$

  - ||•|| is the norm of a matrix

Condition Number

- We can get different condition number values when using different matrix norm definitions

1-norm $$\|A\|_1 = \max_{1 \le j \le N} \sum_{i=1}^{N} |a_{ij}|$$

F-norm $$\|A\|_F = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} |a_{ij}|^2}$$

Inf-norm $$\|A\|_\infty = \max_{1 \le i \le N} \sum_{j=1}^{N} |a_{ij}|$$

Condition Number

- Condition number is highly correlated to singularity
  - Use 1-norm as an example

1-norm $$\|A\|_1 = \max_{1\le j\le N} \sum_{i=1}^{N} |a_{ij}|$$

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Rightarrow A^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Rightarrow k(A) = 1 \cdot 1 = 1$$

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 10^{-5} \end{bmatrix} \Rightarrow A^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 10^{5} \end{bmatrix} \Rightarrow k(A) = 1 \cdot 10^5 = 10^5$$

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \Rightarrow A^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & \infty \end{bmatrix} \Rightarrow k(A) = 1 \cdot \infty = \infty$$

Condition Number

- For the equation AX = B, the solution error is bounded by:

$$\frac{\|\Delta X\|}{\|X\|} \le k(A)\cdot\left(\frac{\|\Delta A\|}{\|A\|}+\frac{\|\Delta B\|}{\|B\|}\right)$$

  - ΔA and ΔB: errors of A and B respectively
  - ΔX: errors of the solution X

- Large condition number yields large solution error
  - E.g., MATLAB will show a warning message if k(A) is more than 10^{16}~10^{17}

Simple Examples

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot X = B$$

$$k\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = 1$$

$$B = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Rightarrow X = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\Delta B = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \qquad \Delta X = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 1.1 \end{bmatrix} \Rightarrow X = \begin{bmatrix} 1 \\ 1.1 \end{bmatrix}$$

Simple Examples

$$\begin{bmatrix} 1 & 1 \\ 0.999 & 1 \end{bmatrix} \cdot X = B \qquad k\left(\begin{bmatrix} 1 & 1 \\ 0.999 & 1 \end{bmatrix}\right) = 4000$$

$$B = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Rightarrow X = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\Delta B = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \qquad \Delta X = \begin{bmatrix} -100 \\ 100 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 1.1 \end{bmatrix} \Rightarrow X = \begin{bmatrix} -100 \\ 101 \end{bmatrix}$$

Pivoting for Accuracy

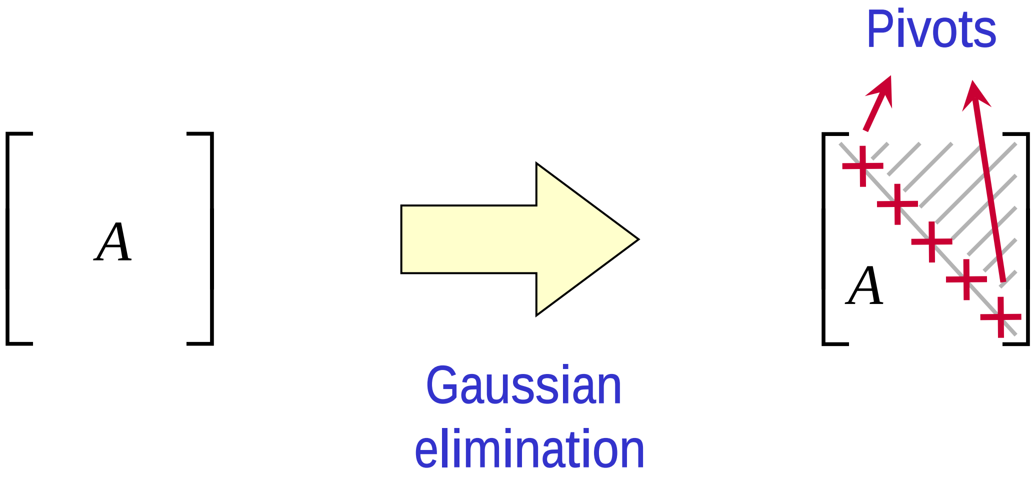
$$\frac{\|\Delta X\|}{\|X\|} \leq k(A) \cdot \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta B\|}{\|B\|} \right)$$

- This inequality only considers numerical errors in A and B
  - It assumes that no additional error is introduced when solving the equation (e.g., during Gaussian elimination)
- Gaussian elimination adds extra numerical errors
  - Every intermediate step is not perfect (due to rounding)

Pivoting for Accuracy

- When solving AX = B, we should minimize the additional numerical error introduced by the solver

- A general rule is to select large pivot values during Gaussian elimination



Pivoting for Accuracy

- Example: solve the following problem on a machine that has 3 decimal digits of precision

$$\begin{bmatrix} 1.00e-4 & 1.00 \\ 1.00 & 1.00 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.00 \\ 2.00 \end{bmatrix}$$

- If we directly apply Gaussian elimination w/o pivoting:

$$\begin{bmatrix} 1.00e-4 & 1.00 \\ 0 & -1.00e4 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.00 \\ -1.00e4 \end{bmatrix} \Rightarrow \begin{cases} x_1 = 0.00 \\ x_2 = 1.00 \end{cases}$$

Wrong Answer ! $x_1 + x_2 = 1.00$

Pivoting for Accuracy

- If we apply Gaussian elimination w/ pivoting:

$$\begin{bmatrix} 1.00e-4 & 1.00 \\ 1.00 & 1.00 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.00 \\ 2.00 \end{bmatrix}$$

Swap two rows to select large pivot

$$\begin{bmatrix} 1.00 & 1.00 \\ 1.00e-4 & 1.00 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2.00 \\ 1.00 \end{bmatrix}$$

Gaussian elimination

$$\begin{bmatrix} 1.00 & 1.00 \\ 0 & 1.00 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2.00 \\ 1.00 \end{bmatrix} \Rightarrow \begin{cases} x_1 = 1.00 \\ x_2 = 1.00 \end{cases}$$

Correct answer !

Pivoting helps to reach the correct answer in this example

Pivoting for Accuracy

- Various choices of pivoting (tradeoff between accuracy and runtime)
  - Full: Swap rows and columns to get largest magnitude on the diagonal
  - Partial: Swap to put largest magnitude from pivot row (or column) onto diagonal

Summary

- Linear equation solver
  - Gaussian elimination
  - Condition number
  - Full/partial pivoting