# Lecture 2
# Gaussian Elimination (GE)

Jinn-Liang Liu

2017/4/18

*Basic Idea of Gaussian Elimination*

$$A = [a_{ij}]_{NxN}\,, \;\; i = \text{the } i^{th} \text{ row}, \;\; j = \text{the } j^{th} \text{ column}$$

$$\vec{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}_{Nx1}$$

We first merge $A$ and $\vec{b}$ as an augmented matrix and then perform elementary operations so that $A$ is transformed to an upper triangular matrix (all entries lying below the diagonal entries are zero):

$$\left[A \mid \vec{b}\right]_{NxN+1} \xrightarrow{\text{Elementary Operations}} \left[\begin{array}{ccccc|c} \times & \times & \cdots & \cdots & \times & \\ 0 & \times & \times & \cdots & \vdots & \\ \vdots & 0 & \ddots & \times & \vdots & \widetilde{b} \\ \vdots & \vdots & 0 & \times & \times & \\ 0 & \cdots & \cdots & 0 & \times & \end{array}\right] \tag{2.1}$$

*Elementary Operations*

(1) $cE_i \to E_i$: Multiply the $i$th row by a constant $c$.
(2) $(E_j + cE_i) \to E_j$: Add $cE_i$ to $E_j$.
(3) $E_i \longleftrightarrow E_j$: Exchange $E_i$ and $E_j$.

$$\begin{aligned} E_1 &: \; x_1 - x_2 + 2x_3 - x_4 = -8 \\ E_2 &: \; 2x_1 - 2x_2 + 3x_3 - 3x_4 = -20 \\ E_3 &: \; x_1 + x_2 + x_3 = -2 \\ E_4 &: \; -3x_1 - x_2 + x_3 + 3x_4 = 4 \end{aligned}$$

$$\left[A \mid \vec{b}\right] = \left[\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 2 & -2 & 3 & -3 & -20 \\ 1 & 1 & 1 & 0 & -2 \\ -3 & -1 & 1 & 3 & 4 \end{array}\right]$$

$$\xrightarrow[(3E_1+E_4)\to E_4]{\substack{(-2E_1+E_2)\to E_2\\(-E_1+E_3)\to E_3}} \left[\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & -4 & 7 & 0 & -20 \end{array}\right]$$

$$\xrightarrow[(5E_3+E_4)\to E_4]{\substack{E_2\longleftrightarrow E_3\\(2E_2+E_4)\to E_4}} \left[\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & -1 & -4 & -4 \\ 0 & 0 & 0 & -18 & -28 \end{array}\right]$$

We thus have the transformed system

$$\begin{aligned} x_1 - x_2 + 2x_3 - x_4 &= -8 \\ 2x_2 - x_3 + x_4 &= 6 \\ -x_3 - 4x_4 &= -4 \\ 18x_4 &= 28 \end{aligned}$$

$\Longrightarrow$ Backward substitution

$\Longrightarrow$ Solution: $x_4 = \dfrac{14}{9}$, $x_3 = \dfrac{-20}{9}$, $x_2 = \ldots$, $x_1 = \ldots$

***Algorithm GE: Gaussian Elimination*** Solve $A\vec{x} = \vec{b}$.

**Input:** $N$: Number of unknowns and equations; $a_{ij}$: Entries of $A$, $i, j = 1 \cdots N$; $b_i$: Entries of $\vec{b}$, $i = 1 \cdots N$.

**Output:** $x_i$: Entries of $\vec{x}$ (Solution) or Error Message.

**Step 1.** For $i = 1, \cdots, N-1$ do Step 2-4 (Elimination Process).

**Step 2.** Let $p$ be the smallest integer $i \le p \le N$ and $a_{pi} \neq 0$. If no integer $p$ can be found then OUTPUT ("Error: No Unique Solution Exists"), STOP.

**Step 3.** If $p \neq i$ then perform $(E_p \leftrightarrow E_i)$.

**Step 4.** For $k = i+1, \cdots, N$ do Step 5-6.

**Step 5.** If $a_{ki} = 0$ then go to Step 4, else set $m_{ki} = a_{ki}/a_{ii}$. $(N-i$ times$)$

**Step 6.** Perform $(E_k - m_{ki}E_i) \to E_k$. $((N-i+2)(N-i)$ times$)$

**Step 7.** If $a_{NN} = 0$ then OUTPUT ("Error: No Unique Solution Exists"), STOP.

**Step 8.** Set $x_N = \frac{b_N}{a_{NN}}$. (1 time)

**Step 9.** For $i = N-1, N-2, \cdots, 1$, set $x_i = \left( b_i - \sum\limits_{j=i+1}^{N} a_{ij}x_j \right) / a_{ii}$. $((N-i+1)$ times)

**Step 10.** OUTPUT $(x_1, \cdots, x_N)$; "Procedure completed successfully"), STOP.

***Complexity of the GE Algorithm***

$$\begin{aligned} & \text{Total number of } \times \text{ or } \div \text{ operations} \\ = & \ 1 + \sum_{i=1}^{N-1} [(N-i) + (N-i+2)(N-i) + (N-i+1)(N-1)] \\ = & \ \frac{N^3}{3} + N^2 - \frac{N}{3} = O(N^3) \end{aligned} \tag{2.2}$$

Operation $*$ or $\div$ is the most time consuming part of operations on a computer. We say that the computational complexity of the Gaussian elimination algorithm is $O(N^3)$, which means that the CPU time needed to solve $A\vec{x} = \vec{b}$ by GE is approximately proportional to $N^3$. You can think of $O(N^3) = cN^3$ as $N \to \infty$ where $c$ is a constant.

**Question 2.1.** If a computer solving $A\vec{x} = \vec{b}$ with $N = 100$ by GE spends 1 second, how much time will it spend for $N = 10000$?

**Project 2.1.** Consider the 1D Poisson Problem (1.1) (with $f(x) = 2$, $g_D = 0$, and $g_N = 0$) and implement the methods FDM and GE. Given a total number of nodes $N$, the mesh size $\Delta x = h = \frac{1}{N-1}$. The maximum error of an approximate solution $U(x)$ is defined as

$$\begin{aligned} E^u &= ||e(x)||_\infty = ||u(x) - U(x)||_\infty \\ &= \max_{1 \le i \le N} |e_i| = \max_{1 \le i \le N} |u_i - U_i| = O(h^\alpha). \end{aligned} \tag{2.3}$$

In general, $||e(x)||_\infty$ is expressed as $O(h^\alpha)$ where $\alpha$ is called the order of convergence of the numerical method (FDM here). With different $h$, we thus have

$$
\begin{aligned}
E_1^u &\propto (h_1)^\alpha \\
E_2^u &\propto (h_2)^\alpha \\
\frac{E_1^u}{E_2^u} &= (\frac{h_1}{h_2})^\alpha \\
\alpha &= \frac{\log(E_1^u) - \log(E_2^u)}{\log(h_1) - \log(h_2)}
\end{aligned} \tag{2.4}
$$

**Input:** $N$

**Output:**

| $N$ | $E^u$ | $\alpha$ |
|---|---|---|
| 5 | | |
| 9 | | |
| 17 | | |
| 33 | | |
| 65 | | |
| 129 | | |

.

**HW 2.1.** Consider 1D Poisson's equation (1.1a) with the Dirichlet boundary conditions $u(0) = \alpha$ and $u(1) = \beta$. This is the same problem (2.6) and (2.7) in LeVeque-FDM-2005.pdf. This problem is solved by using the central finite difference method to obtain an approximation solution $U(x)$. (A) Show that the local truncation error of the approximation solution is of $O(h^2)$. (B) Show that the method is stable. (C) Show that the convergence order of the method is $O(h^2)$. (See LeVeque-FDM-2005.pdf for the definitions of local truncation error, stability, consistence, and convergence and the proofs for these results.)

**HW 2.2.** Consider 1D Poisson's equation (1.1a) with the Dirichlet-Neumann boundary conditions $u'(0) = \sigma$ and $u(1) = \beta$ (See (2.33) in LeVeque-FDM-2005.pdf.) (A) Show that the local truncation error of our approximation (1.16) is $O(h^1)$. (B) Use the central approximation to $u'(0) = \sigma$ as given by (2.36) in LeVeque-FDM-2005.pdf. Show that the local truncation error is now $O(h^2)$.