Computational physics

Một vài kỹ thuật lặp giải hệ phương trình tuyến tính

Iterative Techniques to Solve Linear Systems

1

Phương pháp lặp / Iterative Techniques

- Phần này sẽ xét hai phương pháp lặp "kinh điển" – phương pháp lặp Jacobi và phương pháp lặp Gauss – Seidel [The Jacobi and the Gauss-Seidel iterative methods]
- Hai phương pháp này hiệu quả trong việc giải những hệ phương trình tuyến tính loại "lớn" [ví dụ từ vài trăm phương trình, như sẽ gặp khi giải số phương trình đạo hàm riêng].

2

Phương pháp lặp / Iterative Technique

- Kỹ thuật lặp để giải hệ phương trình tuyến tính $n \times n$ [iterative techniques to solve the $n \times n$ linear system]

$$A\boldsymbol{x} = \boldsymbol{b}$$

- $A, \boldsymbol{x}, \boldsymbol{b}$: các ma trận [matrices].

3

the $n \times n$ linear system

$$A\boldsymbol{x} = \boldsymbol{b}$$

$$\Leftrightarrow \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

$$\Leftrightarrow \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

4

PP lặp/ Iterative Technique

- Starts with an initial approximation $\boldsymbol{x}^{(0)}$ to the solution $\boldsymbol{x}$ / Chọn ("đại") 1 bộ nghiệm ban đầu $\boldsymbol{x}^{(0)}$ xấp xỉ nghiệm $\boldsymbol{x}$
- Generates a sequence of vectors / Từ hệ tuyến tính, tạo "chuỗi nghiệm" [nghiệm lần thứ nhất $\boldsymbol{x}^{(1)}$, lần hai $\boldsymbol{x}^{(2)}$, ... lần thứ k $\boldsymbol{x}^{(k)}$]

$$\{\boldsymbol{x}^{(k)}\}$$

that converges to / hội tụ về nghiệm (được chờ đợi)

$$\boldsymbol{x}$$

5

Jacobi iterative method

- The Jacobi iterative method is obtained by solving the i-th equation in $A\boldsymbol{x} = \boldsymbol{b}$ for x_i to obtain (provided $a_{ii} \neq 0$)

$$x_i = \frac{1}{a_{ii}}\left[\sum_{j=1, j\neq i}^{n} -a_{ij}x_j + b_i\right], i = 1, 2, \dots, n$$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \Rightarrow x_i = \frac{1}{a_{ii}}\left[\sum_{j=1, j\neq i}^{n} -a_{ij}x_j + b_i\right]$$

6

Jacobi iterative method

- For each $k \geq 1$, generate the components $x_i^{(k)}$ of $\boldsymbol{x}^{(k)}$ from the components of $\boldsymbol{x}^{(k-1)}$ by

$$x_i^{(k)} = \frac{1}{a_{ii}} \left[\sum_{\substack{j=1 \\ j \neq i}}^{n} -a_{ij} x_j^{(k-1)} + b_i \right], i = 1, 2, \ldots, n$$

until

$$\left\| \vec{x}^{(k)} - \vec{x}^{(k-1)} \right\| < \varepsilon \text{ or } \frac{\left\| \vec{x}^{(k)} - \vec{x}^{(k-1)} \right\|}{\left\| \vec{x}^{(k)} \right\|} < \varepsilon.$$

- Chú ý:

$$\left\| \vec{x}^{(k)} \right\| \stackrel{\text{def}}{=} \max_{1 \leq i \leq n} |x_i|$$

7

Jacobi iterative method

- For each $k \geq 1$, generate the components $x_i^{(k)}$ ["new"] of $\boldsymbol{x}^{(k)}$ from the components of $\boldsymbol{x}^{(k-1)}$ ["old" / known/ đã biết/ đã có] by

$$x_i^{(k)} = \frac{1}{a_{ii}} \left[\sum_{\substack{j=1 \\ j \neq i}}^{n} -a_{ij} x_j^{(k-1)} + b_i \right], i = 1, 2, \ldots, n$$

$$x_i^{(\text{new})} = \frac{1}{a_{ii}} \left[\sum_{\substack{j=1 \\ j \neq i}}^{n} -a_{ij} x_j^{(\text{old})} + b_i \right], i = 1, 2, \ldots, n$$

8

Jacobi iterative method - Example

The linear system:

$$
\begin{aligned}
E_1&: \ 10x_1 - \ \ x_2 + \ \ 2x_3 \qquad\quad = 6\\
E_2&: \ -x_1 + 11x_2 - \ \ 2x_3 + 3x_4 = 25\\
E_3&: \ \ 2x_1 - \ \ x_2 + 10x_3 - \ \ x_4 = -11\\
E_4&: \qquad\quad 3x_2 - \ \ x_3 + \ 8x_4 = 15
\end{aligned}
$$

9

Jacobi iterative method - Example

- Use Jacobi's iterative technique to find approximations $\boldsymbol{x}^{(k)}$ to $\boldsymbol{x}$ starting with $\boldsymbol{x}^{(0)} = \ (0, 0, 0, 0)$ until

$$\frac{\left\|\vec{x}^{(k)} - \vec{x}^{(k-1)}\right\|}{\|\vec{x}^{(k)}\|} < 10^{-3}$$

10

Jacobi iterative method - Example

- We first solve equation E_i for x_i, for each $i = 1, 2, 3, 4$, to obtain

$$x_1 = \frac{1}{10}x_2 - \frac{1}{5}x_3 + \frac{3}{5}$$
$$x_2 = \frac{1}{11}x_1 + \frac{1}{11}x_3 - \frac{3}{11}x_4 + \frac{25}{11}$$
$$x_3 = -\frac{1}{5}x_1 + \frac{1}{10}x_2 + \frac{1}{10}x_4 - \frac{11}{10}$$
$$x_4 = -\frac{3}{8}x_2 + \frac{1}{8}x_3 + \frac{15}{8}$$

11

Jacobi iterative method - Example

- From the initial "guess" (approximation) $\boldsymbol{x}^{(0)} = (0, 0, 0, 0)$ we have $\boldsymbol{x}^{(1)}$ given by

$$x_1^{(1)} = \frac{1}{10}x_2^{(0)} - \frac{1}{5}x_3^{(0)} + \frac{3}{5} = 0.6000$$
$$x_2^{(1)} = \frac{1}{11}x_1^{(0)} + \frac{1}{11}x_3^{(0)} - \frac{3}{11}x_4^{(0)} + \frac{25}{11} = 2.2727$$
$$x_3^{(1)} = -\frac{1}{5}x_1^{(0)} + \frac{1}{10}x_2^{(0)} + \frac{1}{10}x_4^{(0)} - \frac{11}{10} = -1.1000$$
$$x_4^{(1)} = -\frac{3}{8}x_2^{(0)} + \frac{1}{8}x_3^{(0)} + \frac{15}{8} = 1.8750$$

13

Jacobi iterative method - Example

- Additional iterates, $\boldsymbol{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)})$, are generated in a similar manner. ...

k	0	1	2	…	9	10
$x_1^{(k)}$	0.0000	0.6000	1.0473		0.9991	1.0001
$x_2^{(k)}$	0.0000	2.2727	1.7159		2.0004	1.9998
$x_3^{(k)}$	0.0000	−1.1000	−0.8052		−1.0004	−0.9998
$x_4^{(k)}$	0.0000	1.8750	0.8852		1.0006	0.9998

- The process was stopped after 10 iterations because

$$\frac{\left\|\boldsymbol{x}^{10} - \boldsymbol{x}^{9}\right\|}{\left\|\boldsymbol{x}^{10}\right\|} < 10^{-3}$$

- In fact, $\left\|\boldsymbol{x}^{10} - \boldsymbol{x}\right\| < 10^{-3}$

15

Jacobi iterative method - Example

The linear system:

$$\begin{aligned}
E_1&: \ 10x_1 - \ \ x_2 + \ \ 2x_3 \qquad\quad = 6\\
E_2&: \ -x_1 + 11x_2 - \ \ 2x_3 + 3x_4 = 25\\
E_3&: \ \ 2x_1 - \ \ x_2 + 10x_3 - \ \ x_4 = -11\\
E_4&: \qquad\quad 3x_2 - \ \ x_3 + \ 8x_4 = 15
\end{aligned}$$

has the unique solution $\boldsymbol{x} = (1, 2, -1, 1)$

16

Jacobi Iterative Algorithm

- To solve $A\boldsymbol{x} = \boldsymbol{b}$ given an initial approximation $\boldsymbol{x}^{(\mathbf{0})}$:
- INPUT:
  - the number of equations and unknowns n;
  - a_{ij} , $1 \leq i, j \leq n$ of the matrix A;
  - b_i , $1 \leq i \leq n$ of $\boldsymbol{b}$;
  - $\boldsymbol{XO} = \boldsymbol{x}^{(\mathbf{0})}$;
  - TOL ε;
  - Maximum number of iterations N.
- OUTPUT:
  - the approximate solution $x_1, \ldots, x_n$ or a message that the number of iterations was exceeded.

17

Jacobi Iterative Algorithm

- Step 1: Set $k = 1$ // k: k-th iteration [lần lặp thứ k]
- Step 2: While $(k \leq N)$ do Step 3 -6
  - Step 3: For $i = 1, n$

$$\text{Set} \quad x_i = \frac{1}{a_{ii}} \left[-\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} XO_j + b_i \right]$$

  - Step 4: If $\|x - XO\| < \varepsilon$ then OUTPUT $x_1, \ldots, x_n$
    STOP
  - Step 5: Set $k = k + 1$
  - Step 6: For $i = 1, n$ set $XO_i = x_i$
- Step 7: OUTPUT "Maximum number of iterations exceeded!"
  STOP (The procedure was unfortunately unsuccessful ☹)

18

Gauss-Seidel method

- Jacobi's method

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[-\sum_{j=1}^{n} a_{ij}x_j^{(k-1)} + b_i\right], i = 1,2 \dots, n$$

- Gauss-Seidel method

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[-\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} + b_i\right]$$

19

Gauss-Seidel method - example

The linear system $A\boldsymbol{x} = \boldsymbol{b}$ given by

$$\begin{aligned}
E_1&: \ 10x_1 - x_2 + 2x_3 = 6\\
E_2&: \ -x_1 + 11x_2 - 2x_3 + 3x_4 = 25\\
E_3&: \ 2x_1 - x_2 + 10x_3 - x_4 = -11\\
E_4&: \ 3x_2 - x_3 + 8x_4 = 15
\end{aligned}$$

has the unique solution $\boldsymbol{x} = (1, 2, -1, 1)$

20

Gauss-Seidel method - example

- For the Gauss-Seidel technique we write the system, for each $k = 1, 2, \ldots$ as

$$x_1^{(k)} = \frac{1}{10}x_2^{(k-1)} - \frac{1}{5}x_3^{(k-1)} + \frac{3}{5} = 0.6000$$

$$x_2^{(k)} = \frac{1}{11}x_1^{(k)} + \frac{1}{11}x_3^{(k-1)} - \frac{3}{11}x_4^{(k-1)} + \frac{25}{11} = 2.2727$$

$$x_3^{(k)} = -\frac{1}{5}x_1^{(k)} + \frac{1}{10}x_2^{(k)} + \frac{1}{10}x_4^{(k-1)} - \frac{11}{10} = -1.1000$$

$$x_4^{(k)} = -\frac{3}{8}x_2^{(k)} + \frac{1}{8}x_3^{(k)} + \frac{15}{8} = 1.8750$$

21

Gauss-Seidel method - example

- When $x^{(0)} = (0, 0, 0, 0)$, we have $x^{(1)} = (0.6000, 2.3272, -0.9873, 0.8789)$.
- Subsequent iterations give the values in the following table:

k	0	1	2	3	4	5
$x_1^{(k)}$	0.0000	0.6000	1.030	1.0065	1.0009	1.0001
$x_2^{(k)}$	0.0000	2.3272	2.037	2.0036	2.0003	2.0000
$x_3^{(k)}$	0.0000	−0.9873	−1.014	−1.0025	−1.0003	−1.0000
$x_4^{(k)}$	0.0000	0.8789	0.9844	0.9983	0.9999	1.0000

22

Gauss-Seidel Iterative Algorithm

- To solve $A\boldsymbol{x} = \boldsymbol{b}$ given an initial approximation $\boldsymbol{x}^{(\mathbf{0})}$:
- INPUT:
  - the number of equations and unknowns n;
  - a_{ij} , $1 \leq i, j \leq n$ of the matrix A;
  - b_i , $1 \leq i \leq n$ of $\boldsymbol{b}$;
  - $\boldsymbol{XO} = \boldsymbol{x}^{(\mathbf{0})}$;
  - TOL ε ;
  - Maximum number of iterations N.
- OUTPUT:
  - the approximate solution $x_1, \ldots, x_n$ or
  - a message that the number of iterations was exceeded

23

Gauss-Seidel Iterative Algorithm

- Step 1: Set $k = 1$ // k: k-th iteration [lần lặp thứ k]
- Step 2: While $(k \leq N)$ do Step 3 -6
  - Step 3: For $i = 1, n$

$$\text{Set } x_i = \frac{1}{a_{ii}}\left[-\sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}XO_j + b_i\right]$$

  - Step 4: If $\|x - XO\| < \varepsilon$ then OUTPUT $x_1, \ldots, x_n$
    STOP
  - Step 5: Set $k = k + 1$
  - Step 6: For $i = 1, n$ set $XO_i = x_i$
- Step 7: OUTPUT "Maximum number of iterations exceeded!"
  STOP (The procedure was unfortunately unsuccessful ☹)

24

Nhận xét

- Trong một số trường hợp, phương pháp Gauss – Seidel hội tụ nhanh hơn phương pháp Jacobi.
- Nhưng không có kết luận chung rằng phương pháp Gauss – Seidel tốt hơn phương pháp Jacobi !

25

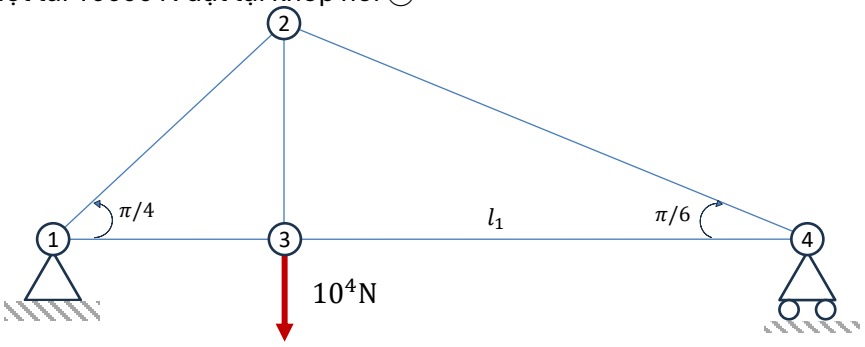
Thực hành

1. Hãy viết code để giải hệ phương trình tuyến tính đã được chọn làm ví dụ trong slides trước bằng i) phương pháp lặp Jacobi, và ii) PP Gausse-Seidel.
2. Xem bài toán giàn kèo ở slides sau và thực hành.

26

Một bài toán về cấu trúc giàn

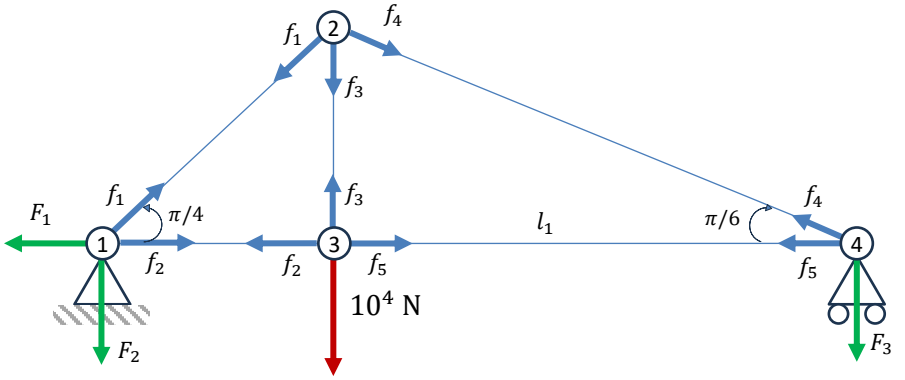
- Giàn kèo [truss] là 1 cấu trúc gồm các thành phần tạo thành một – nhiều đơn vị tam giác→ giàn nhẹ có khả năng chịu tải trọng lớn. Ở thiết kế cầu, các thành phần riêng lẻ của giàn được kết nối với các khớp chốt [pin joints] có thể xoay cho phép lực được truyền từ thành phần này sang thành phần khác của giàn.
- Hình dưới mô tả 1 giàn được giữ **tĩnh/cố định** tại điểm cuối [endpoint] ①; giàn được phép di chuyển theo chiều ngang ở điểm cuối ④, và có các khớp chốt tại ①, ②, ③, và ④.
- Một tải 10000 N đặt tại khớp nối ③



27

Bài toán về cấu trúc giàn

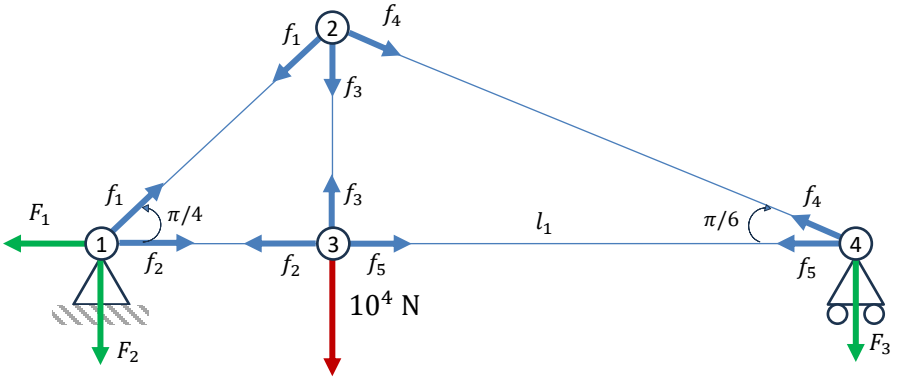
- Tải đặt ở khớp chốt ③ → các lực f_1, f_2, f_3, f_4, f_5 ở các khớp nối, như được phác trong hình. Khi lực dương → lực căng, âm → lực nén tác động lên các thành phần của giàn. Khớp đỡ tĩnh ③ có thể có cả thành phần lực ngang F_1 và dọc F_2. Ở đầu đỡ có thể di chuyển thì chỉ có phần lực dọc F_3.
- Nếu hệ cân bằng tĩnh, tại mỗi khớp, chốt tổng các vector lực $= \mathbf{0}$. Chiếu lên phương dọc, ngang → Hệ phương trình tuyến tính ở slide sau.



28

Bài toán về cấu trúc giàn

- ① : Ngang: $-F_1 + \frac{\sqrt{2}}{2} f_1 + f_2 = 0$; Dọc: $\frac{\sqrt{2}}{2} f_1 - F_2 = 0$
- ② : Ngang: $-\frac{\sqrt{2}}{2} f_1 + \frac{\sqrt{3}}{2} f_4 = 0$; Dọc: $-\frac{\sqrt{2}}{2} f_1 - f_3 - \frac{1}{2} f_4 = 0$
- ③ : Ngang: $-f_2 + f_5 = 0$; Dọc: $f_3 - 10000 = 0$
- ④ : Ngang: $-\frac{\sqrt{3}}{2} f_4 - f_5 = 0$; Dọc: $\frac{1}{2} f_4 - F_3 = 0$



29

Giải bài toán về cấu trúc giàn kèo

Hãy đưa hệ 8 phương trình ở slide trước về dạng ma trận $A\boldsymbol{x} = \boldsymbol{b}$, trong đó $\boldsymbol{x}$ và $\boldsymbol{b}$ là 2 ma trận cột như sau:

$$\boldsymbol{x} = \begin{pmatrix} F_1 \\ F_2 \\ F_3 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 10000 \\ 0 \\ 0 \end{pmatrix}$$

- Xác định ma trận A. Với các phần tử của ma trận A này, hãy giải số hệ trên, độ chính xác $\varepsilon = 10^{-2}$, $\boldsymbol{x}^{(0)} = \mathbf{1}$, bằng i) PP Jacobi, và ii) bằng PP Gauss – Seidel.

30