Chương 1

Giải hệ phương trình

Chúng ta có 2 loại hệ phương trình:

- Hệ phương trình tuyến tính
- Hệ phương trình phi tuyến

Chúng ta đã biết các phương pháp giải trực tiếp và gián tiếp hệ phương trình tuyến tính:

- Các phương pháp giải trực tiếp:
  - Phương pháp Cramer
  - Phương pháp thế
  - Phương pháp sử dụng ma trận nghịch đảo
  - Phương pháp khử Gauss, Gauss-Jordan
- Các phương pháp giải gián tiếp
  - Phương pháp lặp đơn
  - Phương pháp lặp Seidel
- Các phương pháp tìm trị riêng và véc tơ riêng của ma trận

1.1 Đặt bài toán và phương pháp giải

Hệ phương trình tuyến tính n phương trình, n ẩn x_1, x_2, ..., x_n là tập n phương trình E_1, E_2, ..., E_n dạng

$$\begin{cases} E_1 : a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ E_2 : a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots \\ E_n : a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

với các hệ số a_{jk} và b_j đã biết. Hệ được gọi là thuần nhất nếu các b_j bằng 0, trong trường hợp ngược lại được gọi là hệ không thuần nhất.

Dùng cách biểu diễn ma trận, ta có thể viết hệ (1.1) dưới dạng:

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{1.1}$$

Ở đây, ma trận hệ số $A = [a_{jk}]$ là ma trận vuông cấp n, x và b là các véc tơ cột:

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

Ma trận $\tilde{\boldsymbol{A}}$ sau được gọi là ma trận mở rộng (augmented matrix) của hệ (1.1):

$$\tilde{\boldsymbol{A}} = [\boldsymbol{A}, \boldsymbol{b}] = \left(\begin{array}{cccc:c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array}\right)$$

Nghiệm của (1.1) là bộ số x_1, x_2, ..., x_n thỏa mãn tất cả phương trình của hệ.

Véc tơ nghiệm của (1.1) là $\boldsymbol{x}$ mà các thành phần của nó lập thành một nghiệm của (1.1).

Hệ phương trình (1.1) có thể giải được bằng

- *phương pháp trực tiếp* (phương pháp khử Gauss, ...), hoặc
- *phương pháp gián tiếp* hay *phương pháp lặp*

1.2 Phương pháp khử Gauss

Chúng ta đã biết phương pháp sử dụng định thức để giải hệ phương trình (1.1), đó là *phương pháp Cramer*. Ở đây ta xét *phương pháp khử Gauss*, cũng là một phương pháp trực tiếp, để giải hệ phương trình tuyến tính.

1.2.1 Phương pháp khử Gauss

Phương pháp 1.1: Phương pháp khử Gauss

Phương pháp này khử liên tiếp các ẩn để từng bước đưa ma trận mở rộng $\tilde{\boldsymbol{A}}$ về dạng tam giác trên (upper triangular matrix, hay dạng bậc thang). Phương pháp gồm hai phần thực hiện lần lượt như sau:

1. *Quá trình thuận (forward elimination)*: Dùng các phép biến đổi hàng sơ cấp để đưa $\tilde{\boldsymbol{A}}$ về dạng tam giác trên:
   (a) Khử x_1 từ $E_{\geq 2}$ bằng cách:
   $$E_j \coloneqq E_j - \frac{a_{j1}}{a_{11}} E_1 \, \forall j \in [2, n] \tag{1.2}$$
   a_{11} gọi là *phần tử trục xoay (pivot)*, E_1 gọi là *phương trình chính (pivot equation)*.
   (b) Với $i = 2, \ldots, n-1$, khử x_i từ $E_{>i}$ bằng cách tương tự như trên, cuối cùng thu được dạng tam giác trên.
2. *Quá trình ngược (back substitution)*: Giải $\boldsymbol{x}$ từ cuối lên:
   (a) Giải x_n từ E_n, giải tiếp được x_{n-1} do đã biết x_n.
   (b) Tương tự giải được đến x_1, cuối cùng thu được nghiệm $\boldsymbol{x}$.

(*"back"* trong cụm *"back substitution"* nghĩa là các ẩn được giải từ cuối lên, do trước đó đã đưa được $\boldsymbol{A}$ về dạng tam giác trên. Tương tự, nếu $\boldsymbol{A}$ ở dạng tam giác dưới, việc thế từ trên xuống sẽ gọi là *"forward substitution"*. Dù hai dạng này tương đương, trong khử Gauss ta chỉ nói đến back substitution.)

Ta kí hiệu ma trận $\boldsymbol{A}$ khởi đầu là $\boldsymbol{A}^{(1)}$. Sau khi kết thúc bước khử x_i, ma trận $\boldsymbol{A}^{(i)}$ sẽ biến đổi thành $\boldsymbol{A}^{(i+1)}$. Bước cuối cùng là bước khử x_{n-1} (vì x_n không cần khử), và ma trận sau cùng là $\boldsymbol{A}^{(n)}$.

Xem xét kĩ hơn, ta thấy (1.2) có một chi tiết nhạy cảm là phép chia. Nếu $a_{11} = 0$, phương pháp không thể thực hiện được, cho dù thực tế hệ phương trình có thể có nghiệm. Một cách xử lí trường hợp này là *đổi vị trí hàng*, như trong hai ví dụ sau:

Ví dụ 1.1. *Hãy giải hệ phương trình:*

$$8x_2 + 2x_3 = -7 \tag{E_1}$$
$$3x_1 + 5x_2 + 2x_3 = 8 \tag{E_2}$$
$$6x_1 + 2x_2 + 8x_3 = 26 \tag{E_3}$$

Chúng ta xoay trục từ E_1, nhưng do E_1 không có ẩn x_1, trong khi đó hệ số của x_1 trong phương trình E_3 là lớn nhất. Vì vậy ta đổi chỗ E_1 và E_3 cho nhau.

Tới đây ta có ma trận mở rộng như sau:

$$\tilde{\boldsymbol{A}} = \left(\begin{array}{ccc:c} 6 & 2 & 8 & 26 \\ 3 & 5 & 2 & 8 \\ & 8 & 2 & -7 \end{array}\right)$$

Khử x_1 được:

$$\tilde{\boldsymbol{A}}^{(1)} = \left(\begin{array}{ccc:c} 6 & 2 & 8 & 26 \\ & 4 & -2 & -5 \\ & 8 & 2 & -7 \end{array}\right)$$

Khử x_2 được:

$$\tilde{\boldsymbol{A}}^{(2)} = \left(\begin{array}{ccc:c} 6 & 2 & 8 & 26 \\ & 4 & -2 & -5 \\ & & 6 & 3 \end{array}\right)$$

Vậy ta giải được $x_3 = 0{,}5$, $x_2 = -1$, $x_1 = 4$.

Ví dụ 1.2. *Hãy giải hệ phương trình:*

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

Ta có ma trận mở rộng như sau:

$$\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{A}}^{(1)} = \left(\begin{array}{cccc:c} 1 & -1 & 2 & -1 & -8 \\ 2 & -2 & 3 & -3 & -20 \\ 1 & 1 & 1 & 0 & -2 \\ 1 & -1 & 4 & 3 & 4 \end{array}\right)$$

Khử x_1 bằng chuỗi biến đổi

$$E_2 \coloneqq E_2 - 2E_1;\ E_3 \coloneqq E_3 - E_1;\ E_4 \coloneqq E_4 - E_1$$

cho kết quả:

$$\tilde{\boldsymbol{A}}^{(2)} = \left(\begin{array}{cccc:c} 1 & -1 & 2 & -1 & -8 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & 2 & 4 & 12 \end{array}\right)$$

Điểm quay $a_{22}^{(2)} = 0$, do đó cần phải đổi hàng. Ta chọn $a_{32}^{(2)} \neq 0$, do đó đổi chỗ E_2 và E_3:

$$\tilde{\boldsymbol{A}}^{(2)} = \left(\begin{array}{cccc:c} 1 & -1 & 2 & -1 & -8 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 0 & 2 & 4 & 12 \end{array}\right)$$

Do x_2 đã được khử khỏi E_3 và E_4, ta có:

$$\tilde{\boldsymbol{A}}^{(3)} = \tilde{\boldsymbol{A}}^{(2)} = \left(\begin{array}{cccc:c} 1 & -1 & 2 & -1 & -8 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 0 & 2 & 4 & 12 \end{array}\right)$$

Khử x_3 bằng

$$E_4 \coloneqq E_4 - (-2)E_3$$

cho kết quả:

$$\tilde{\boldsymbol{A}}^{(4)} = \left(\begin{array}{cccc:c} 1 & 1 & 2 & 1 & -8 \\ 0 & 2 & 1 & 1 & 6 \\ 0 & 0 & 1 & 1 & -4 \\ 0 & 0 & 0 & 2 & 4 \end{array}\right)$$

Vậy ta giải được $x_4 = 2$, $x_3 = 2$, $x_2 = 3$, $x_1 = -7$.

Phương pháp khử Gauss còn có một số biến thể:

- Phương pháp Gauss-Jordan: đưa $\boldsymbol{A}$ về dạng đường chéo thay vì dạng tam giác trên.
- Phương pháp Doolittle, phương pháp Crout, phương pháp Cholesky: đều dựa trên phân tích LU (LU factorization), sẽ giới thiệu ở phần sau.

1.2.2 Độ phức tạp

Ta phân tích độ phức tạp của phương pháp khử Gauss:

- Có tổng cộng $n - 1$ bước khử.
- Ở bước k, ta khử x_k trong các phương trình $E_{>k}$, tổng cộng là $n - k$ phương trình.
- Trong mỗi phương trình:
  - Có 1 phép chia: ví dụ $\dfrac{a_{i1}}{a_{11}}$ trong (1.2)
  - Có $n - k + 1$ phép nhân: ví dụ $\dfrac{a_{i1}}{a_{11}}E_1$ trong (1.2)
  - Có $n - k + 1$ phép trừ: ví dụ chính (1.2)

Do đó, tổng số phép tính của phương pháp này là:

$$\begin{aligned} C(n) &= \sum_{k=1}^{n-1}(n-k) + 2\sum_{k=1}^{n-1}(n-k)(n-k+1) \\ &= \sum_{s=1}^{n-1} s + 2\sum_{k=1}^{n-1} s(s+1) \,(\text{with } s = n-k) \\ &= \frac{1}{2}n(n-1) + \frac{2}{3}n(n^2-1) \\ &\approx \frac{2}{3}n^3 \end{aligned}$$

Ta nói rằng phương pháp khử Gauss có độ phức tạp $\mathcal{O}(n^3)$.

1.3 Phân tích LU và Ma trận nghịch đảo

Chúng ta tiếp tục thảo luận các phương pháp số giải hệ phương trình tuyến tính n phương trình, n ẩn. Trong phần này, ta xem xét ba phương pháp cải tiến từ phương pháp khử Gauss, cho phép tìm nghiệm nhanh chóng hơn, gồm phương pháp Doolittle, phương pháp Cholesky, và phương pháp Crout. Cả ba phương pháp đều dựa trên phân tích LU.

Điểm chung của các phương pháp này là đều cố gắng đưa ma trận hệ số về tích của các ma trận tam giác trên (Upper triangular matrix) và ma trận tam giác dưới (Lower triangular matrix). Hai dạng ma trận này rất hữu ích vì cho phép tìm $\boldsymbol{x}$ với độ phức tạp $\mathcal{O}(n^2)$; nếu xét theo khía cạnh này, phương pháp khử Gauss khác ở điểm là *trực tiếp* biến đổi về dạng tam giác. Tất nhiên, phần lớn tính toán lại chuyển về việc phân tích ra dạng ma trận đặc biệt này, và với một số hệ phương trình đặc biệt, độ phức tạp có thể thấp hơn $\mathcal{O}(n^3)$.

Ba phương pháp được trình bày trong phần này đều gắn chặt với một kiểu phân tích ra ma trận tam giác. Vì rất dễ để giải nghiệm từ dạng tam giác, nên tên phương pháp vừa chỉ phương pháp phân tích ra dạng tam giác tương ứng, vừa chỉ phương pháp giải hệ tuyến tính từ dạng tam giác đã phân tích.

1.3.1 Phân tích LU & phương pháp Doolittle

Định nghĩa 1.1: Phân tích LU (LU factorization)

Phân tích LU là việc phân tích ma trận vuông $\boldsymbol{A}$ thành tích hai ma trận

$$\boldsymbol{A} = \boldsymbol{LU}$$

trong đó $\boldsymbol{L}$ là ma trận tam giác dưới và $\boldsymbol{U}$ là ma trận tam giác trên.

Một số lưu ý về phân tích LU:

- Không phải mọi ma trận vuông đều có phân tích LU. Tuy nhiên, ta thừa nhận một kết quả quan trọng là mọi ma trận khả nghịch $\boldsymbol{A}_0$ đều có thể sắp xếp lại các hàng để thu được một ma trận $\boldsymbol{A}$ có phân tích LU.

- Có thể có nhiều cách phân tích LU.

Sau đây ta tìm hiểu phương pháp Doolittle để phân tích LU. Ta sẽ xem xét kết quả của phương pháp phân tích LU này với bài toán giải hệ phương trình tuyến tính trước, sau đó mới đi sâu vào công thức toán học của nó.

Phương pháp Doolittle: Dùng dạng LU để giải hệ tuyến tính

Định lí sau đây cũng chính là công thức cho phương pháp Doolittle.

Định lí 1.1

Nếu hệ tuyến tính $\boldsymbol{Ax} = \boldsymbol{b}$ có thể giải được bằng khử Gauss mà không cần đổi vị trí hàng, thì $\boldsymbol{A}$ có thể phân tích thành tích của ma trận tam giác trên $\boldsymbol{L}$ và tam giác dưới $\boldsymbol{U}$ (tức $\boldsymbol{A} = \boldsymbol{LU}$) với dạng sau:

$$\boldsymbol{U} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ & & \ddots & \vdots \\ & & & a_{nn}^{(n)} \end{pmatrix}, \text{ và } \boldsymbol{L} = \begin{pmatrix} 1 & & & \\ m_{21} & 1 & & \\ \vdots & \ddots & \ddots & \\ m_{n1} & \dots & m_{n,n-1} & 1 \end{pmatrix}$$

trong đó

- a_{ji}^{k} là hệ số của x_i trong phương trình E_j tại bước khử thứ k
- $m_{ji} = \dfrac{a_{ji}^{(i)}}{a_{ii}^{(i)}}$ là hệ số của E_j trong, ví dụ, công thức (1.2).

$\boldsymbol{U}$ cũng chính là ma trận $\boldsymbol{A}$ thu được sau khi khử Gauss.

Chú ý rằng $\boldsymbol{U}$ là $\boldsymbol{A}$ sau khi khử Gauss, tức ma trận $\tilde{\boldsymbol{A}}$ trong phần 1.2 nhưng bỏ đi cột cuối.

Cần nhắc lại rằng, đến đây, ta vẫn cần giả sử hệ có thể giải được bằng khử Gauss mà *không* cần đổi vị trí hàng. Mở rộng phương pháp Doolittle cho trường hợp cần đổi vị trí hàng không khó. Trước hết, ta cần tìm cách biểu diễn việc tráo đổi vị trí hàng.

Định nghĩa 1.2: Ma trận hoán vị (permutation matrix)

Ma trận hoán vị $\boldsymbol{P}$ là ma trận có được bằng cách sắp xếp lại các hàng của I tùy ý.

Nhân P vào bên trái A sẽ tráo các hàng của A theo đúng cách tráo các hàng của I để tạo ra P. Nói cách khác, P là tích các ma trận của *biến đổi sơ cấp tráo hàng (row swapping elementary operation)*.

Ta chọn $\boldsymbol{P}$ sao cho $\boldsymbol{PA}$ có thể giải bằng khử Gauss mà không cần tráo vị trí hàng. $\boldsymbol{P}$ được xây dựng đơn giản bằng cách áp dụng cách tráo hàng của Gauss cho ma trận $\boldsymbol{I}$.

Ví dụ 1.3. *Tìm ma trận hoán vị cho ma trận trong ví dụ 1.1, sao cho sau khi ma trận đó với ma trận gốc, nhận được một ma trận có thể dùng phương pháp khử Gauss.*

Trong ví dụ trên, hệ có thể dùng phương pháp khử Gauss sau khi đổi chỗ E_1

và E_3. Vậy một ma trận hoán vị phù hợp là:

$$\boldsymbol{P} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Sau khi tìm được $\boldsymbol{P}$, $\boldsymbol{PA}$ có thể áp dụng phương pháp khử Gauss mà không cần tráo vị trí hàng. Theo định lí 1.1, $\boldsymbol{PA}$ có phân tích LU:

$$\begin{aligned} & \boldsymbol{PA} = \boldsymbol{LU} \\ \iff & \boldsymbol{A} = \boldsymbol{P}^{-1}\boldsymbol{LU} = (\boldsymbol{P}^t\boldsymbol{L})\boldsymbol{U} \,(\text{do } \boldsymbol{P}^{-1} = \boldsymbol{P}^t) \end{aligned}$$

Chứng minh phương pháp Doolittle

Phần này chứng minh kĩ hơn về tính đúng đắn của phương pháp Doolittle, và có thể bỏ qua.

Chứng minh định lí 1.1. Ta xem xét công thức (1.2). Xét trường hợp $j = 2$:

$$E_2 \coloneqq E_2 - \frac{a_{21}}{a_{11}} E_1 = E_2 - m_{21} E_1$$

Sử dụng phép biến đổi sơ cấp cộng một hàng với α lần một hàng khác (row addition elementary operation), ta có ma trận biến đổi sau:

$$\boldsymbol{M}_2^{(1)} = \begin{pmatrix} 1 & & & \\ -m_{21} & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}$$

Tương tự, cùng trong bước khử đầu tiên (khử x_1) này, ta có dạng tổng quát hơn của $\boldsymbol{M}_j^{(1)}$:

$$\boldsymbol{M}_j^{(1)} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ -m_{j1} & & \ddots & \\ & & & 1 \end{pmatrix}$$

Sau khi kết thúc bước khử x_1, $\boldsymbol{A}$ được biến đổi thành

$$\boldsymbol{M}_2^{(1)}\boldsymbol{M}_3^{(1)} \dots \boldsymbol{M}_n^{(1)}\boldsymbol{A} = \boldsymbol{M}^{(1)}\boldsymbol{A} = \boldsymbol{M}^{(1)}\boldsymbol{A}^{(1)} = \boldsymbol{A}^{(2)}$$

và hơn nữa

$$\boldsymbol{A}^{(2)}\boldsymbol{x} = \boldsymbol{M}^{(1)}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}^{(1)}\boldsymbol{b} = \boldsymbol{b}^{(2)}$$

Dễ dàng chứng minh được $\boldsymbol{M}^{(1)}$ ở trên, gọi là ma trận biến đổi Gauss thứ nhất, có dạng sau:

$$\boldsymbol{M}^{(1)} = \begin{pmatrix} 1 & & & \\ -m_{21} & 1 & & \\ \vdots & & \ddots & \\ -m_{n1} & & & 1 \end{pmatrix}$$

Tương tự, ta chứng minh được $\boldsymbol{M}^{k}$ (ma trận biến đổi Gauss thứ k) có dạng sau:

$$\boldsymbol{M}^{(k)} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -m_{k+1,k} & \ddots & & \\ & & \vdots & & \ddots & \\ & & -m_{n,k} & & & 1 \end{pmatrix}$$

Cuối cùng, sau khi kết thúc bước khử x_n:

$$\boldsymbol{M}^{(1)}\boldsymbol{M}^{(2)}\dots\boldsymbol{M}^{(n)}\boldsymbol{A} = \boldsymbol{M}\boldsymbol{A} = \boldsymbol{A}^{(n)}$$

trong đó $\boldsymbol{A}^{(n)}$ chính là ma trận $\boldsymbol{A}$ sau khi khử Gauss:

$$\boldsymbol{A}^{(n)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ & & \ddots & \vdots \\ & & & a_{nn}^{(n)} \end{pmatrix}$$

Tới đây, chúng ta đã biến đổi được phương trình ban đầu (1.1) sang dạng sau:

$$\boldsymbol{A}^{(n)}\boldsymbol{x} = \boldsymbol{M}^{(n-1)}\boldsymbol{A}^{(n-1)}\boldsymbol{x} = \boldsymbol{M}^{(n-1)}\boldsymbol{b}^{(n-1)} = \boldsymbol{b}^{(n)} \tag{1.3}$$

trong đó $\boldsymbol{A}^{(n)}$ là một ma trận tam giác trên.

Giờ đây, nếu coi $\boldsymbol{A}^{(n)}$ là thành phần $\boldsymbol{U}$ cần tìm, ta cần nhân vào trước hai vế của (1.3) một thành phần $\boldsymbol{L}$ nào đó để đưa (1.3) về lại (1.1).

Thành phần $\boldsymbol{U}$ đã thấy chỉ đơn giản là $\boldsymbol{A}$ qua một chuỗi các biến đổi sơ cấp cộng một hàng với α lần một hàng khác. Do đó $\boldsymbol{L}$ chỉ cần là một ma trận có thể đảo ngược chuỗi biến đổi này.

Xét biến đổi $\boldsymbol{M}_2^{(1)}$, biến đổi

$$\boldsymbol{L}_2^{(1)} = \begin{pmatrix} 1 & & & \\ m_{21} & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}$$

sẽ đảo ngược được $\boldsymbol{M}_2^{(1)}$. Lý do cũng rất đơn giản:

- $\boldsymbol{M}_2^{(1)}$ lấy E_2 trừ đi m_{21} lần E_1, thì
- $\boldsymbol{L}_2^{(1)}$ lấy E_2 cộng với m_{21} lần E_1

Tương tự, ta dễ dàng thấy

$$\boldsymbol{L}^{(k)} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & m_{k+1,k} & \ddots & & \\ & & \vdots & & \ddots & \\ & & m_{n,k} & & & 1 \end{pmatrix}$$

sẽ đảo ngược được $\boldsymbol{M}^{(k)}$.

Nhân $\boldsymbol{L}^{(k)}$ vào trước hai vế của (1.3) theo thứ tự k tăng dần từ 1 đến $n-1$, ta có:

$$\begin{aligned} & \boldsymbol{L}^{(1)}\boldsymbol{L}^{(2)}\dots\boldsymbol{L}^{(n-1)}\boldsymbol{A}^{(n)}\boldsymbol{x} = \boldsymbol{L}^{(1)}\boldsymbol{L}^{(2)}\dots\boldsymbol{L}^{(n-1)}\boldsymbol{b}^{(n)} \\ \iff & \boldsymbol{L}^{(1)}\boldsymbol{L}^{(2)}\dots\boldsymbol{L}^{(n-1)}\boldsymbol{M}^{(n-1)}\dots\boldsymbol{M}^{(2)}\boldsymbol{M}^{(1)}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{L}^{(1)}\boldsymbol{L}^{(2)}\dots\boldsymbol{L}^{(n-1)} \\ & \qquad \boldsymbol{M}^{(n-1)}\dots\boldsymbol{M}^{(2)}\boldsymbol{M}^{(1)}\boldsymbol{b} \\ & \iff \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned}$$

Đến đây, ta nhận được phương trình (1.1) ban đầu. Vậy tích các $\boldsymbol{L}^{(k)}$ theo thứ tự trên là một giá trị $\boldsymbol{L}$ phù hợp. Không khó để chứng minh rằng:

$$\boldsymbol{L} = \boldsymbol{L}^{(1)}\boldsymbol{L}^{(2)}\dots\boldsymbol{L}^{(n-1)} = \begin{pmatrix} 1 & & & & & \\ m_{21} & \ddots & & & & \\ \vdots & \ddots & 1 & & & \\ \vdots & \vdots & m_{k+1,k} & \ddots & & \\ \vdots & \vdots & \vdots & \ddots & \ddots & \\ m_{n1} & \dots & m_{n,k} & \dots & m_{n,n-1} & 1 \end{pmatrix}$$

Vậy ta đã xây dựng được phân tích LU của $\boldsymbol{A}$, với $\boldsymbol{L}$ và $\boldsymbol{U}$ có dạng như yêu cầu.

đpcm.

1.3.2 Phân tích $\mathbf{LL^t}$ và phương pháp Cholesky

Thuật toán Cholesky phân tích một ma trận xác định dương ra dạng LL^{t}. Để tìm hiểu phân tích này, ta cần biết về ma trận xác định dương.

Định nghĩa 1.3: Ma trận xác định dương (positive definitive matrix)

Ma trận $\boldsymbol{A}$ $n \times n$ gọi là xác định dương nếu:

- $\boldsymbol{A}$ đối xứng, và
- $\boldsymbol{x}^t \boldsymbol{A}\boldsymbol{x} > 0 \forall \boldsymbol{x} \neq 0$

$\boldsymbol{L}$ có dạng như sau (viết rõ lại để tiện biểu diễn về sau):

$$\boldsymbol{L} = \begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & \dots & \dots & l_{nn} \end{pmatrix}$$

Ta thừa nhận định lí sau:

Định lí 1.2

Ma trận $\boldsymbol{A}$ là xác định dương khi và chỉ khi nó phân tích được ra dạng $\boldsymbol{L}\boldsymbol{L}^t$, trong đó $\boldsymbol{L}$ là ma trận tam giác dưới với đường chéo chính khác 0.

Do có số ẩn không lớn (với ma trận $n \times n$ cần tìm tổng cộng $\frac{n(n+1)}{2}$ ẩn), đồng thời ma trận tích có dạng phù hợp, nên phương pháp này có thể giải bằng tay các ẩn theo cách thế thông thường với n nhỏ. Nếu không tiện tính tay, ta có phương pháp chính xác sau:

Phương pháp 1.2: Phương pháp Cholesky

Phương pháp Cholesky phân tích ma trận xác định dương $\boldsymbol{A}$ thành dạng LL^{t}.
Lần lượt tính phần tử khác 0 ở cột 1, 2, ... Bước thứ i sẽ tính các phần tử thuộc cột i như sau:

- Tính l_{ii}:

$$l_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2 \right)^{0,5}$$

- Tính các phần tử còn lại, nếu có:

$$l_{ji} = \frac{1}{l_{ii}} \left(a_{ji} - \sum_{k=i+1}^{n} l_{jk} l_{ik} \right) \mid j \in [i+1, n]$$

1.3.3 Phân tích LU cho ma trận dải & thuật toán Crout

Ta quay trở lại với phân tích LU, tuy nhiên sử dụng thuật toán khác, nhanh vượt trội so với Doolittle, cho một loại ma trận đặc biệt: ma trận dải.

Định nghĩa 1.4: Ma trận dải (band matrix)

Ma trận $n \times n$ gọi là ma trận dải nếu có $1 < p,\ q < n$ sao cho $a_{ij} = 0$ khi $j - i \geq p$ hoặc $i - j \geq q$.

Nói cách khác, hai chỉ số p, q chỉ số đường chéo mà các phần tử trên đó không nhất thiết bằng 0:

- p đường chéo trên đường chéo chính, gồm cả đường chéo chính
- q đường chéo dưới đường chéo chính, gồm cả đường chéo chính

Ta lại xét tiếp một trường hợp đặc biệt: ma trận dải với $p = q = 2$, gọi là *ma trận ba đường chéo (tridiagonal matrix)*.

Nếu $\boldsymbol{A}$ trong (1.1) có dạng ba đường chéo:

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & & & \\ a_{21} & a_{22} & a_{23} & & \\ & a_{32} & a_{33} & \ddots & \\ & & \ddots & \ddots & a_{n-1,n} \\ & & & a_{n,n-1} & a_{nn} \end{pmatrix}$$

thì có thể phân tích $\boldsymbol{A}$ ra dạng LU như sau:

$$\boldsymbol{L} = \begin{pmatrix} l_{11} & & & & \\ l_{21} & l_{22} & & & \\ & l_{32} & l_{33} & & \\ & & \ddots & \ddots & \\ & & & l_{n,n-1} & l_{nn} \end{pmatrix}, \text{ và } \boldsymbol{U} = \begin{pmatrix} 1 & u_{12} & & & \\ & 1 & u_{23} & & \\ & & \ddots & \ddots & \\ & & & \ddots & u_{n-1,n} \\ & & & & 1 \end{pmatrix}$$

Do có số ẩn không lớn (với ma trận $n \times n$ cần tìm tổng cộng $3n - 2$ ẩn), đồng thời ma trận tích có dạng phù hợp, nên phương pháp này có thể giải bằng tay các ẩn theo cách thế thông thường với n nhỏ. Nếu không tiện tính tay, ta giới thiệu qua phương pháp sau:

Phương pháp 1.3: Phương pháp Crout cho ma trận ba đường chéo

Phương pháp Crout dùng để phân tích ma trận ra dạng LU. Dạng LU của phương pháp Crout có khác biệt so phương pháp Doolittle:

- Phương pháp Crout tạo ma trận $\boldsymbol{U}$ có đường chéo chính bằng 1.
- Phương pháp Doolittle tạo ma trận $\boldsymbol{L}$ có đường chéo chính bằng 1.

1.4 Các phương pháp lặp

Phương pháp khử Gauss và các biến thể của nó trong hai phần cuối cùng là các *phương pháp trực tiếp* để giải hệ phương trình tuyến tính; đây là những phương pháp đưa ra nghiệm sau một số tính toán được xác định trước. Ngược lại, trong trường hợp *giải gián tiếp* hoặc *phương pháp lặp (iterative method)* chúng ta bắt đầu từ một giá trị xấp xỉ nghiệm đúng và, nếu thành công, sẽ có được xấp xỉ tốt hơn và tốt hơn từ một quá trình tính toán lặp đi lặp lại. Trong các phương pháp này, số phép tính phụ thuộc vào độ chính xác cần thiết.

Chúng ta áp dụng các phương pháp lặp nếu tốc độ hội tụ đủ nhanh (nếu ma trận có các phần tử nằm trên đường chéo chính lớn hơn các phần tử nằm ngoài, như ta sẽ thấy), hoặc với *ma trận thưa (sparse matrix)*, tức ma trận có phần lớn các phần tử là 0.

1.4.1 Phương pháp lặp Gauss - Seidel

Đây là một phương pháp lặp quan trọng, được nghiên cứu và sử dụng nhiều. Trước khi đi vào công thức chi tiết, ta xem xét phương pháp qua ví dụ sau:

Ví dụ 1.4. *Xét hệ phương trình*

$$\begin{cases} x_1 - 0{,}25x_2 - 0{,}25x_3 = 50 \\ -0{,}25x_1 + x_2 - 0{,}25x_4 = 50 \\ -0{,}25x_1 + x_3 - 0{,}25x_4 = 25 \\ -0{,}25x_2 - 0{,}25x_3 + x_4 = 25 \end{cases}$$

Viết lại hệ theo cách sau:

$$\begin{cases} x_1 = 0{,}25x_2 + 0{,}25x_3 + 50 \\ x_2 = 0{,}25x_1 + 0{,}25x_4 + 50 \\ x_3 = 0{,}25x_1 + 0{,}25x_4 + 25 \\ x_4 = 0{,}25x_2 + 0{,}25x_3 + 25 \end{cases} \tag{1}$$

Ta bắt đầu bằng một xấp xỉ (cho dù có thể khác xa nghiệm cần tìm), ví dụ $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = x_4^{(0)} = 100$ *và tính* $\boldsymbol{x}^{(1)}$ *từ các giá trị này, theo công thức*

(1) *như sau:*

$$
\begin{aligned}
& & & \textit{Giá trị cũ (chưa có giá trị mới)} \downarrow & & \\
x_1^{(1)} &= & 0{,}25x_2^{(0)} + 0{,}25x_3^{(0)} & & & + 50 = 100 \\
x_2^{(1)} &= 0{,}25x_1^{(1)} & & & + 0{,}25x_4^{(0)} & + 50 = 100 \\
x_3^{(1)} &= 0{,}25x_1^{(1)} & & & + 0{,}25x_4^{(0)} & + 25 = 75 \\
x_4^{(1)} &= & 0{,}25x_2^{(1)} + 0{,}25x_3^{(1)} & & & + 25 = 68{,}75 \\
& \uparrow \textit{Giá trị mới} & & & &
\end{aligned}
$$

Trong tính toán trên, $\boldsymbol{x}^{(1)}$ *được tính theo công thức* (1)*, nhưng với các giá trị* x *mới nhất có được ở thời điểm tính toán. Tiếp tục tính toán, ta có bảng sau:*

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
1	89,062	88,281	63,281	62,891
2	87,891	87,695	62,695	62,598
3	87,598	87,549	62,549	62,524
4	87,524	87,512	62,512	62,506
5	87,506	87,503	62,503	62,502

Rõ ràng, nghiệm hội tụ rất nhanh.

Phương pháp Gauss-Seidel

Ta đã thấy Gauss-Seidel, cũng giống như phương pháp Newton hay phương pháp điểm bất động, là một phương pháp lặp (iterative method), tức tính một hàm lặp đi lặp lại, kết quả lần lặp trước là đầu vào của lần lặp sau. Trong phần này, ta sẽ đưa ra công thức lặp Gauss-Seidel theo một cách trực quan từ ví dụ 1.4.

Ví dụ 1.4 thực hiện được dễ dàng nhất khi hệ số của x_i trong phương trình thứ i là 1, tức đường chéo chính của $\boldsymbol{A}$ chỉ chứa 1. Để tiện trong việc đưa ra công thức, ta giả sử rằng $a_{jj} = 1 \, \forall j \in [1, n]$, do mọi hệ phương trình có thể đưa được về dạng này, thông qua các biến đổi hàng sơ cấp.

Ta tách $\boldsymbol{A}$ như sau:

$$\boldsymbol{A} = \boldsymbol{I} + \boldsymbol{L} + \boldsymbol{U}$$

trong đó $\boldsymbol{I}$ là ma trận đơn vị, $\boldsymbol{L}$ là ma trận tam giác dưới chặt (strictly), $\boldsymbol{U}$ là ma trận tam giác trên chặt.

Thay vào (1.1), ta có:

$$
\begin{aligned}
& \boldsymbol{A}\boldsymbol{x} = (\boldsymbol{I} + \boldsymbol{L} + \boldsymbol{U})\boldsymbol{x} = \boldsymbol{b} \\
\iff & \boldsymbol{x} = \boldsymbol{b} - \boldsymbol{L}\boldsymbol{x} - \boldsymbol{U}\boldsymbol{x}
\end{aligned}
$$

Theo ví dụ 1.4, ta thấy $\boldsymbol{U}$ chứa các giá trị x cũ, còn $\boldsymbol{L}$ chứa các giá trị x mới. Từ đó, ta có công thức lặp tổng quát:

$$\boldsymbol{x}^{(m+1)} = \boldsymbol{b} - \boldsymbol{L}\boldsymbol{x}^{(m+1)} - \boldsymbol{U}\boldsymbol{x}^{(m)} \tag{1.4}$$

trong đó $\boldsymbol{x}^{(m)}$ là véc tơ nghiệm xấp xỉ thứ m.

Phương pháp 1.4: Phương pháp Gauss-Seidel

Phương pháp này xấp xỉ nghiệm $\boldsymbol{x}$ của hệ $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ khi

- biết xấp xỉ bắt đầu $\boldsymbol{x}^{(0)}$, và
- $\boldsymbol{A}$ không chứa 0 trên đường chéo chính

Gọi các phương trình trong hệ là E_i. Ta đưa hết x_i sang riêng vế trái của E_i sao cho hệ số của x_i là 1. Nói cách khác, ta tính x_i qua các x_j, $j \neq i$. Phương pháp Gauss-Seidel thực hiện như sau:

1. Trong mỗi lần lặp, tính $\boldsymbol{x}$ như sau:
   - lần lượt tính x_i theo thứ tự i tăng dần như công thức đã tách vế trái ở trên, và
   - dùng giá trị x_j với mới nhất có thể khi tính x_i
2. Dừng lại khi đạt điều kiện dừng, nếu không lặp lại bước trên

Chú ý rằng, phương pháp trên chỉ cần điều kiện đường chéo chính của $\boldsymbol{A}$ không chứa 0. Điều kiện đường chéo chính chứa toàn 1 được bỏ qua vì trong bước đưa x_i sang vế trái của E_i, hệ số của x_i đã được chuyển thành 1.

Điều kiện hội tụ

Ta viết lại (1.4) như sau:

$$\begin{aligned} & \boldsymbol{x}^{(m+1)} = \boldsymbol{b} - \boldsymbol{L}\boldsymbol{x}^{(m+1)} - \boldsymbol{U}\boldsymbol{x}^{(m)} \\ \iff & (\boldsymbol{I} + \boldsymbol{L})\boldsymbol{x}^{(m+1)} = \boldsymbol{b} - \boldsymbol{U}\boldsymbol{x}^{(m)} \\ \iff & \boldsymbol{x}^{(m+1)} = \boldsymbol{C}\boldsymbol{x}^{(m)} + (\boldsymbol{I} + \boldsymbol{L})^{-1}\boldsymbol{b} \text{ với } \boldsymbol{C} = -(\boldsymbol{I} + \boldsymbol{L})^{-1}\boldsymbol{U} \end{aligned}$$

Chú ý rằng, nếu muốn tính $\boldsymbol{C}$ theo công thức trên, $\boldsymbol{A}$ phải ở dạng đường chéo chính chứa toàn 1.

Trước hết ta nhớ lại định nghĩa về *véc tơ riêng* và *giá trị riêng* của ma trận:

Định nghĩa 1.5: Véc tơ riêng (eigenvector) & giá trị riêng (eigenvalue)

Với mỗi ma trận $\boldsymbol{A}$, véc tơ $\boldsymbol{v} \neq \vec{0}$ và vô hướng λ được gọi lần lượt là véc tơ riêng và giá trị riêng ứng với véc tơ riêng đó nếu:

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v}$$

Ta thừa nhận một kết quả quan trọng để tính các giá trị riêng; sau khi đã có các giá trị riêng, ta có thể dễ dàng tìm các véc tơ riêng tương ứng:

$$\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = 0$$

Quay trở lại phương pháp Gauss-Seidel, ta thừa nhận phương pháp này hội tụ khi và chỉ khi tất cả trị riêng của $\boldsymbol{C}$ có giá trị tuyệt đối nhỏ hơn 1. Cụ thể hơn, ta xét *bán kính phổ* của $\boldsymbol{C}$.

Định nghĩa 1.6: Bán kính phổ (spectral radius)

Bán kính phổ của ma trận $\boldsymbol{C}$ là giá trị tuyệt đối lớn nhất của các giá trị riêng của $\boldsymbol{C}$.

Phương pháp Gauss-Seidel, với bán kính phổ r của $\boldsymbol{C}$, sẽ hội tụ:

- khi và chỉ khi $r < 1$
- nhanh hơn khi r nhỏ

Như vậy, ta đã biết điều kiện *cần và đủ* để phương pháp Gauss-Seidel hội tụ. Giờ ta sẽ xem xét một số điều kiện *đủ* cho sự hội tụ này.

Phương pháp Gauss-Seidel sẽ hội tụ khi

$$\|\boldsymbol{C}\| < 1$$

$\|\boldsymbol{C}\|$ là *chuẩn (norm)* của $\boldsymbol{C}$. Ta có một số chuẩn hay gặp như sau:

- Chuẩn Frobenius, tức căn bậc hai của tổng của bình phương mọi phần tử:

$$\|\boldsymbol{C}\| = \sqrt{\sum_{j=1}^{n}\sum_{k=1}^{n} c_{jk}^2}$$

- Chuẩn tổng cột, tức giá trị lớn nhất trong tổng các trị tuyệt đối của phần tử một cột:

$$\|\boldsymbol{C}\| = \max_k \sum_{j=1}^{n} |c_{jk}|$$

- Chuẩn tổng hàng, tức giá trị lớn nhất trong tổng các trị tuyệt đối của phần tử một hàng:

$$\|\boldsymbol{C}\| = \max_j \sum_{k=1}^{n} \left|c_{jk}\right|$$

Ba chuẩn này không tương đương với nhau. Hoàn toàn có thể có trường hợp một chuẩn thỏa mãn điều kiện trị tuyệt đối nhỏ hơn 1, đủ để kết luận phương pháp hội tụ, nhưng dùng chuẩn khác thì lại không thể kết luận về sự hội tụ.

1.4.2 Phương pháp Jacobi

Phương pháp Gauss-Seidel là một phương pháp *hiệu chỉnh liên tiếp (successive correction)* vì *trong* mỗi bước lặp, ta cập nhật giá trị một thành phần x_j mỗi khi thành phần đó có xấp xỉ mới. *Phương pháp Jacobi* sẽ giới thiệu dưới đây lại thuộc loại *hiệu chỉnh đồng thời*, tức khi tính toán chỉ dùng các xấp xỉ cũ, và cập nhật đồng thời các x_j ở cuối vòng lặp, sau khi đã tính xong x_n.

Khác biệt về việc sử dụng giá trị xấp xỉ mới nói ở trên là điểm khác biệt chính yếu của hai phương pháp này. Do đó, ta có thể viết công thức lặp tổng quát cho phương pháp Jacobi dựa trên (1.4) như sau:

$$\begin{aligned}
\boldsymbol{x}^{(m+1)} &= \boldsymbol{b} - \boldsymbol{L}\boldsymbol{x}^{(m)} - \boldsymbol{U}\boldsymbol{x}^{(m)} \\
&= \boldsymbol{b} - (\boldsymbol{L} + \boldsymbol{U})\boldsymbol{x}^{(m)} \\
&= \boldsymbol{b} - (\boldsymbol{A} - \boldsymbol{I})\boldsymbol{x}^{(m)} \\
&= \boldsymbol{b} + (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{x}^{(m)}
\end{aligned} \tag{1.5}$$

Tiếp tục chú ý, $\boldsymbol{A}$ ở đây vẫn phải ở dạng có đường chéo chính toàn 1.

Phương pháp Jacobi sẽ hội tụ với mọi $\boldsymbol{x}^{(0)}$ khi và chỉ khi bán kính phổ của $\boldsymbol{I} - \boldsymbol{A}$ nhỏ hơn 1.

Phương pháp Jacobi gần đây được chú ý nhiều vì cho phép tính toán song song. Bản chất cập nhật lập tức của phương pháp Gauss-Seidel khiến việc song song hóa rất khó khăn, còn việc cập nhật sau mỗi lần lặp khiến phương pháp Jacobi có thể được song song hóa vô cùng dễ dàng.