TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

BÁO CÁO HỌC PHẦN GIẢI TÍCH SỐ

ĐỀ TÀI

SỬ DỤNG PHƯƠNG PHÁP GAUSS GIẢI HỆ PHƯƠNG TRÌNH TUYẾN TÍNH

GV hướng dẫn: HÀ THỊ NGỌC YẾN

Nhóm sinh viên thực hiện:

Họ và tên	***MSSV***	***Mã lớp***
Hoàng Tú Linh	**20185463**	**116439**
Phạm Tùng Huy	**20185456**	**116439**
Nguyễn Thị Thu Hoài	**20185358**	**116439**
Phan Anh Chiến	**20185330**	**116439**
Phùng Mạnh Sang	**20185473**	**116439**

Hà Nội, tháng 5 năm 2020

Mục lục

Mở đầu

Ngày nay hầu hết các công việc tính toán phức tạp trong khoa học kỹ thuật đều do máy tính đảm nhiệm. Một trong những tính toán cơ bản và hay gặp là giải hệ phương trình tuyến tính. Hệ phương trình tuyến tính không chỉ đơn thuần xuất hiện trong sách vở, bài học mà nó xuất hiện và giải quyết nhiều vấn đề thuộc nhiều lĩnh vực khoa học như vật lý, hóa học, sinh học, kinh tế, trắc địa,... và cả trong cuộc sống thường ngày của chúng ta.

Để giải hệ phương trình tuyến tính có nhiều phương pháp và phương pháp khử Gauss là một phương pháp giải hệ phương trình tuyến tính trực tiếp và hẳn là đã rất quen thuộc với chúng ta. Nó là một phương pháp tốt, thích hợp để cài đặt trên máy tính. Với các hệ phương trình tuyến tính có số phương trình nhỏ, con người có thể tự tính toán sử dụng phương pháp Gauss và cho ra kết quả. Nhưng với những hệ phương trình tuyến tính có số phương trình lớn hoặc rất lớn rõ ràng con người cần đến sự trợ giúp của máy tính. Tuy nhiên, do nhu cầu của thực tiễn, nhiều bài toán của thực tế hoặc của chính toán học (sai phân hóa và giải số phương trình vi phân,...), phương pháp khử Gauss nói tiêng, phương pháp giải hệ phương trình địa số tuyến tính nói chung, vẫn được quan tâm nghiên cứu, đặc biệt vào những năm gần đây với việc sử dụng những thành tựu mới của công nghệ thông tin (máy tính tốc độ cao, máy tính song song,...). Tử đó chúng em làm bài báo cáo để tìm hiểu kỹ hơn về phương pháp khử Gauss và từ đó đưa ra những đánh giá cũng như các giải pháp cải thiện, khắc phục những hạn chế của phương pháp khử Gauss.

1 Kiến Thức Đại Số Tuyến Tính

1.1 Hệ phương trình tuyến tính

Hệ m phương trình tuyến tính n ẩn là một hệ phương trình có dạng:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \qquad \vdots \qquad \ddots \qquad \vdots \qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases} \tag{1.1}$$

Trong đó, các hệ số a_{ij}, b_i thuộc trường $\mathbb{K}$. Trong phạm vi bài báo cáo, ta coi $\mathbb{K} \equiv \mathbb{R}$.

Ta có có các định nghĩa sau:

a) Ma trận A là *ma trận của các hệ số*, được lập thành từ các hệ số a_{ij} của hệ phương trình:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

b) Ma trận $\overline{A}$ là *ma trận mở rộng của hệ* khi ta ghép thêm số cột hạng tự do.

$$\overline{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{bmatrix}$$

c) Ma trận B là *ma trận hệ số tự do của hệ*, thu được từ cột hệ số tự do.

$$B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Như vậy, nếu gọi X là ma trận cột biểu diễn các ẩn của hệ phương trình, thì hệ phương trình ta có thể viết dưới dạng:

$$AX = B \tag{1.2}$$

Từ nay, ta sử dụng dạng trên để biểu diễn hệ phương trình là chủ yếu.

1.2 Nghiệm của hệ phương trình tuyến tính

Mỗi nghiệm của hệ phương trình (1.2) là một bộ sắp thứ tự $(c_1, c_2, \ldots, c_n)$ thỏa mãn tất cả các phương trình của hệ (1.2) được thỏa mãn khi thay các ẩn x_i tương ứng bởi c_i.

Một hệ phương trình có thể có duy nhất nghiệm, vô số nghiệm hoặc không có nghiệm.

1.3 Biến đổi tương đương

Hai hệ phương trình được gọi là *tương đương* nếu hai hệ phương trình đó có cùng không gian nghiệm.

Các phép biến đổi sơ cấp của hệ phương trình là các phép biến đổi:

a) Nhân một phương trình nào đó của hệ với một số khác 0.

b) Cộng vào một phương trình nào đó của hệ với một phương trình khác đã nhân với một số khác 0.

c) Đổi chỗ hai phương trình của hệ. Từ đây, ta có thể rút ra một số nhận xét sau:

- Các phép biến đổi sơ cấp của hệ phương trình tương đương ứng với các phép biển đổi trên hàng của ma trận mở rộng của hệ.
- Khi thực hiện các phép biển đổi sơ cấp, ta thu được một hệ phương trình mới tương đương với hệ phương trình ban đầu.

1.4 Các kết quả quan trọng

a) **Hệ phương trình Cramer - Định lý Cramer**

Một *hệ phương trình Cramer* là một hệ phương trình mà số ẩn bằng số phương trình và định thức của ma trân hệ số khác không.

Định lý Cramer: Hệ phương trình Cramer luôn có duy nhất nghiệm.

Chứng minh:

Xét một hệ phương trình Cramer:

$$AX = B$$

Do đây là hệ phương trình Cramer, nên ta có:

$$det(A) \neq 0$$

Như vậy tồn tại ma trận nghịch đảo của A là A^{-1}. Nhân cả hai vế hệ phương trình ban đầu với A^{-1} ta thu được.

$$X = A^{-1}B$$

Như vậy, hệ phương trình đã cho có nghiệm duy nhất là $X = A^{-1}B$.

b) Định lý Kronecker – Capelli và các hệ quả

Trong định lý này, ta sẽ nêu ra điều kiện để một hệ phương trình tổng quát có nghiệm, cũng như các hệ quả xung quanh.

Định lý Cramer: Hệ phương trình tuyến tính:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \quad \vdots \quad \ddots \quad \vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases} \tag{1.1}$$

Hệ phương trình có nghiệm khi và chỉ khi hạng của ma trận hệ số bằng hạng của ma trận mở rộng của nó.

Chứng minh:

- Điều kiện cần:

  Giả sử hệ phương trình (1.1) có nghiệm $(c_1, c_2, \ldots, c_n)$. Khi đó, ta có đẳng thức đối với các ma trận cột:

$$c_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} + c_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + c_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Do đó, cột cuối cùng của ma trận mở rộng $\overline{A}$ là tổ hợp tuyến tính của các cột còn lại. Như vậy, khi ta trừ các cột còn lại cho cột cuối cùng của ma trận $\overline{A}$ ta thu được ma trận:

$$A^0 = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & 0 \\ a_{21} & a_{22} & \cdots & a_{2n} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & 0 \end{bmatrix}$$

Ma trận A^0 có cùng hạng với ma trận $\overline{A}$ (do nó thu được từ ma trân $\overline{A}$ thông qua những phép biến đổi cơ bản ở cột). Hiển nhiên ma trận A^0 cùng hạng với ma trận A. Như vậy ma trận $\overline{A}$ cùng hạng với ma trận A (đpcm).

- Điều kiện đủ:

  Giả sử $r(A) = r(\overline{A}) = r$. Khi đó tồn tại một định thức con M khác không cấp r của A. Không mất tính tổng quát, ta giả sử định thức con này nằm ở góc phía trên bên trái của A. Với chỉ số j cố định, $(1 < j \leq n)$ ta xét định thức:

$$\Delta_{ij} = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1r} & a_{1j} \\ a_{21} & a_{22} & \cdots & a_{2r} & a_{2j} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{r1} & a_{r2} & \cdots & a_{rr} & a_{rj} \\ a_{i1} & a_{i2} & \cdots & a_{ir} & a_{ij} \end{vmatrix}, \qquad i = 1, 2, \ldots, m$$

  Nếu như một trong hai i, j không lớn hơn r thì $\Delta_{ij} = 0$ vì trong định thức có hai dòng (hay cột) bằng nhau. Nếu i và j cùng lớn hơn r thì $\Delta_{ij} = 0$ vì nó là định thức vây quanh định thức M.

  Khai triển Δ_{ij} theo dòng cuối cùng ta được:

$$\Delta_{ij} = a_{i1}\alpha_1 + a_{i2}\alpha_2 + \ldots + a_{ir}\alpha_r + a_{ij}\alpha_j \tag{1.3}$$

  Trong đó, $\alpha_1, \alpha_2, \ldots, \alpha_r, \alpha_j$ là các phần tử bù đại số của các phần tử ở dòng cuối trong Δ_{ij}. Ta thấy các số $\alpha_1, \alpha_2, \ldots, \alpha_r$ không phụ thuộc vào i, còn $\alpha_j = M$ không phụ thuộc vào cả i lẫn j. Lưu ý rằng $\Delta_{ij} = 0$ và $M \neq 0$, từ đẳng thức (1.3) ta có:

$$a_{ij} = \beta_1\alpha_{i1} + \beta_2\alpha_{i2} + \ldots + \beta_r\alpha_{ir}, \qquad i = 1, 2, \ldots, m$$

Trong đó:

$$\beta_t = -\frac{\alpha_t}{M}(t = 1, 2, \ldots, r)$$

Điều này chứng tỏ cột thứ j của ma trận A là tổ hợp tuyến tính của cột đầu tiên của A. Điều này cũng đúng cho ma trận $\overline{A}$, nghĩ là:

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = \beta_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} + \beta_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + \beta_r \begin{bmatrix} a_{1r} \\ a_{2r} \\ \vdots \\ a_{mr} \end{bmatrix}$$

Hay $(\beta_1, \beta_2, \ldots, \beta_r, 0, 0, \ldots, 0)$ là một nghiệm của hệ đã cho (đpcm)

Từ đây, ta rút ra hai hệ quả quan trọng sau:

Hệ quả 1: Nếu hệ phương trình (1.1) với n ẩn có $r_A = r_{\overline{A}} = n$ thì hệ có nghiệm duy nhất.

Chứng minh: Giả sử hệ phương trình (1.1) có $r_A = r_{\overline{A}} = n$. Khi đó tồn tại định thức con cơ sở của M của cả $\overline{A}$ và A. Bây giờ chúng ta sẽ chứng minh rằng mỗi dòng của $\overline{A}$ khác với n dòng đầu tiên là tổ hợp tuyến tính của n này. Thật vậy, trong phép chứng minh điều kiện đủ của ***Định lý Kronecker – Capelli*** thì không chỉ mỗi cột mà cả mỗi dòng ngoài định thức cơ sở M đều là tổ hợp tuyến tính của n dòng đầu tiên. Khi đó hệ phương trình đã cho tương đương với hệ phương trình:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \quad\vdots \qquad\quad \vdots \qquad \ddots \qquad \vdots \qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

Do $M \neq 0$ nên hệ này là một hệ Cramer, vì vậy hệ có nghiệm duy nhất theo định lý Cramer

Hệ quả 2: Nếu hệ phương trình (1.1) có $r_A = r_{\overline{A}} < n$ thì hệ có vô số nghiệm phụ thuộc vào $n - r_A$ tham số.

Chứng minh: Giả sử $r_A = r_{\overline{A}} < n$. Ký hiệu M là định thức con cơ sở của A và $\overline{A}$. Giả thiết rằng định thức này nằm ở góc phía trên bên trái của A. Như vậy:

$$M = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1r} \\ a_{21} & a_{22} & \cdots & a_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r1} & a_{r2} & \cdots & a_{rr} \end{vmatrix}$$

Do mỗi dòng thứ i của $\overline{A}$, $i > r$ là tổ hợp tuyến tính của r dòng đầu tiên nên hệ đã cho tương đương với:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1r}x_r = b_1 - a_{1(r+1)}x_{r+1} - \cdots - a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2r}x_r = b_2 - a_{2(r+1)}x_{r+1} - \cdots - a_{2n}x_n \\ \vdots \qquad \vdots \qquad \ddots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \ddots \qquad \vdots \\ a_{r1}x_1 + a_{r2}x_2 + \cdots + a_{rr}x_r = b_r - a_{r(r+1)}x_{r+1} - \cdots - a_{rn}x_n \end{cases} \tag{1.4}$$

Đây là một hệ Cramer gồm r phương trình r ẩn $x_1, x_2, \ldots, x_r$. Với mỗi bộ giá trị $(c_{r+1}, \ldots, c_n)$ của các ẩn $(x_{r+1}, \ldots, x_n)$ ở vế phải ta thu được một nghiệm của hệ (1.4) là $(c_1, c_2, \ldots, c_r)$. Khi đó:

$$(c_1, c_2, \ldots, c_r, c_{r+1}, \ldots, c_n)$$

Là nghiệm của hệ phương trình (1.1). Bởi vậy hệ có vô số nghiệm phụ thuộc vào $n-r_A$ tham số.

Nhận xét: Những hệ quả của định lý Kronecker Capelli rất qua trọng, đó là cơ sở giúp chúng ta thực hiện phương pháp khử Gauss khi cài đặt phương pháp giải quyết những trường hợp phương trình vô nghiệm hoặc có vô số nghiệm.

2 Phương Pháp Guass

2.1 Lịch sử phương pháp

Phương pháp Gauss hay còn gọi là phép khử Gauss, phương pháp giảm cột liên tiếp là một thuật toán trong đại số tuyến tính nhằm giải quyết phương trình tuyến tính.

Phương pháp Guass được xuất hiện ở Chương 8 *"Rectoangular Arrays"* của Cuốn *"The Nine Chapters on the Mathematical Art"*. Nó được sử dụng minh họa cho 18 bài toán, với 2 đến 5 phương trình. Tài liệu tham khảo đầu tiên cho cuốn sách này được sử dụng từ năm

179 sau công nguyên, nhưng một phần của nó đã được viết rất sớm, xấp xỉ năm 150 trước công nguyên. Nó được bình luận với LiuHui vào thế kỉ thứ .

Phương pháp này ở Châu Âu xuất phát từ chú thích của Isaac Newton. Năm 1670 ông viết rằng toàn bộ sách đại số học mà ông đã biết đều thiếu phần giải hệ phương trình, sau đó Newton đã cung cấp. Đại học Cambridge cuối cùng cũng xuất bản các ghi chú với tên *"Arithmetica Universalis"* vào năm 1707 sau khi Newton rời cuộc sống học thuật. Các ghi chú đã được sử dụng rộng rãi, điều này đã biến phép khử Gauss thành một bài học tiêu chuẩn trong sách giáo khoa đại số vào cuối thế kỷ XVI.I

Carl Friedrich Gauss vào năm 1810 đã nghĩ ra kí hiệu cho phép thử đối xứng. Về sau, Gauss đã đưa ra những mô tả chi tiết nhất về phương pháp này vào năm 1801, trong cuốn *"Theoria Motus corporum coelestium sectionibus solem ambientium"*.

2.2 Ý tưởng phương pháp

Phương pháp Gauss (Gauss Elimination) là một phương pháp giải đúng hệ phương trình tuyến tính. Nó sử dụng những phép biến đổi cơ bản, biến ma trận mở rộng của hệ phương trình tuyến tính về dạng ma trận hình thang, sau đó sẽ giải lần lượt từng phương trình của hệ theo thứ tự từ dưới lên, khi mà các phương trình ở dưới sẽ đơn giản hơn (ít ẩn hơn), và khi đó việc tìm từng giá trị của ẩn thỏa mãn hệ phương trình sẽ dễ dạng hơn.

2.3 Nội dung phương pháp

Mảng Ind (index): Xét ma trận $\overline{A} = (a)_{m\times(n+1)}$ là là một mảng có m phần tử lưu các vị trí ở mỗi hàng mà tại đó, phần tử a_{ij}là phần tử đầu tiên khác 0 trong hàng của ma trận hệ số A. Từ mảng index ta có thể rút ra:

- Tại một vị trí trong mảng index, nếu giá trị $ind[i] = n + 1$ thì tức là tại hàng i, tất cả hệ số bên phải dấu bằng đều bằng 0, còn giá trị bên trái dấu bằng khác 0 nên phương trình vô nghiệm hay hệ phương trình vô nghiệm.
- Nhìn vào mảng Ind ta có thể nhận ra các nghiệm phụ thuộc và nghiệm độc lập.

Để thuận tiện cho việc mô tả nội dung phương pháp, ta xét trường hợp hệ có n phương trình và n ẩn chắc chắn có nghiệm duy nhất.

$$\begin{cases} E1: & a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ E2: & a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ En: & a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{cases} \tag{2.1}$$

Viết lại hệ phương trình (2.1) dưới dạng ma trận:

$$AX = B$$

Với:

$$A = [a_{ij}] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Từ đây ta xây dựng ma trận mở rộng của hệ phương trình (2.1)

$$\overline{A} = [A \mid B] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & a_{1,n+1} \\ a_{21} & a_{22} & \cdots & a_{2n} & a_{2,n+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & a_{n,n+1} \end{array}\right]$$

Ở đây, các giá trị ở cột chứ $n+1$ là các giá trị của ma trận B hay $a_{i,n+1} = b_i$ với $i = 1, 2, \ldots, n$. Đến đây, ta chia phương pháp Guass thành hai giai đoạn:

a) **Quá trình thuận:**

Giả sử $a_{11} \neq 0$, ta thực hiện các phép biến đổi sơ cấp sau:

$$E_j - \left(\frac{a_{j1}}{a_{11}} E_1\right) \to E_j \qquad \text{với} \quad j = 2, ..., n$$

Việc thực hiện các phép biển đổi sơ cấp này sẽ khiến các hệ số của ân x_1 bị loại bỏ. Mặc dù các hệ số a_{ij} sẽ thay đổi sau các phép biến đổi sơ cấp này, ta ký hiệu giá trị hàng i và côt j là a_{ij}. Tiếp tục thực hiện các phép biến đổi sơ cấp tương tự với $i = 2, 3, \ldots, n-1$ (giả sử $a_{ii} \neq 0$):

$$E_j - \left(\frac{a_{ji}}{a_{ii}} E_i\right) \rightarrow E_j \qquad \text{với} \quad j = i+1, i+2, ..., n$$

Các thao tác này sẽ khử hệ số của biến x_i trong mỗi hàng dưới hàng thứ i với mỗi $i = 1, 2, \ldots, n-1$. Ma trận cuối cùng sẽ có dạng như sau:

$$A = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & a_{1,n+1} \\ 0 & a_{22} & \cdots & a_{2n} & a_{2,n+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn} & a_{n,n+1} \end{array}\right]$$

Sau cùng, những giá trị a_{ij} của ma trận $\overline{A}$ sẽ khác so với ban đầu. Ma trận $\overline{A}$ lúc này sẽ biểu diễn dưới dạng một hệ phương trình tuyến tính tương đương với hệ phương trình ban đầu.

$$\begin{cases} E_1: & a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & a_{1,n+1} \\ E_2: & & & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & a_{2,n+1} \\ \vdots & & & & & \ddots & & \vdots & & \vdots \\ E_n: & & & & & & & a_{nn}x_n & = & a_{n,n+1} \end{cases}$$

b) **Quá trình nghịch:**

Giải phương trình thứ n với ẩn a_n, ta thu được:

$$x_n = \frac{a_{n,n+1}}{a_{nn}}$$

Giải phương trình thứ $n-1$ với ẩn x_{n-1}, và nhớ rằng giá trị x_n đã biết ta thu được:

$$x_n = \frac{a_{n-1,n+1} - a_{n-1,n}x_n}{a_{n-1,n-1}}$$

Tiếp tục quá trình nhưu vậy, ta thu được:

$$x_i = \frac{a_{i,n+1} - a_{i,n}x_n - a_{i,n-1}x_{n-1} - ... - a_{i,i+1}x_{i+1}}{a_{ii}} = \frac{a_{i,n+1} - \displaystyle\sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}}$$

Với $i = n-1, n-2, ..., 2, 1$

3 Thuật toán của phương pháp Guass

3.1 Quá trình thuận

Algorithm 1 Quá trình thuận

Input: Ma trận mở rộng của một hệ m phương trình tuyến tính n ẩn.

Output: Ma trận được sắp xếp dưới dạng bậc thang.

```
Function KiemTraVoNghiem
    for i = 1 to n + 1 do
        if ind[i] = n + 1 then
            Phương trình vô nghiệm
            break

Function QuaTrinhThuan(a_{m×n})
    for i = 1 to m do
        for j = i to n do
            if a_{ij} ≠ 0 then
                ind[i]=j
                for k = i+1 to m do
                    for l = j to n do
                        a_{kl} = a_{kl} - (a_{kj} / a_{ij}).a_{il}
                break
            else
                for t = i + 1 to m do
                    if a_{tj} ≠ 0 then
                        doivitrihang(i,t)
                        j - -
                        KiemTraVoNghiem
                        break
```

3.2 Quá trình nghịch

Algorithm 2 Quá trình nghịch

Input: Ma trận đã được sắp xếp dạng bậc thang.

Output: Nghiệm/Báo vô nghệm/vector nghiệm nếu hệ phương trình có vô số nghiệm.

```
Function rank A
    for i = m down to 1 do
        if ind[i] ≠ 0 then
            rank = i
            break

Function GiaiNghiem
    if ind[rank] = n+1 then
        Phương trình vô nghiệm
    if rank == n                          ▷ phương trình có nghiệm duy nhất then
        for i = m down to 1 do
            x_i = a_{i,n} / a_{i,i}
            for j = m - 1 down to 1 do
                a_{j,n+1} -= a_{j,i} . x_i
    if rank < n                           ▷ phương trình có vô số nghiệm then
        ndl = n - rank
        for i = rank down to 1            ▷ Hệ số tự do trong biểu diễn nghiệm phụ thuộc do
            for j = rank down to 1 do
                a_{j,n} -= a_{ji} × a_{ind[i]-1,n} / a_{ind[i]-1,ind[i]-1}
        for i = rank down to 1            ▷ Hệ số của các ndl trong biểu diễn nghiệm phụ thuộc do
            for j = 0 to ndl  do
                for k = i down to 1 do
                    a_{k-1,ndl[j]} -= a_{k-1,ind[i]-1} × a_{i,ndl[j]} / a_{i,ind[i]-1}
        for int i = rank down to 1        ▷ Biểu diễn nghiệm phụ thuộc do
            for j = 0 to  do
                x[i] += -(a_{i,ndl[j]} / a_{i,ind[i]-1}) × x[ndl[j]]     ▷ cộng xâu
```

4 Kết quả thu được

4.1 Các ví dụ cơ bản

Ví dụ 1 Giải hệ phương trình:

$$\begin{cases} x - y + z = 8 \\ 2x + 3y - z = -2 \\ 3x - 2y - 9z = 9 \end{cases}$$

Kết quả

Giải theo phương pháp thông thường:

Đầu tiên, chúng ta viết hệ phương trình dưới dạng ma trận:

$$\left[\begin{array}{ccc|c} 1 & -1 & 1 & 8 \\ 2 & 3 & -1 & -2 \\ 3 & -2 & -9 & 9 \end{array}\right]$$

Sau đó, Thực hiện các phép biến đổi sơ cấp để có được ma trận dạng bậc thang.

$$\xrightarrow{R_1+R_2=R_2} \left[\begin{array}{ccc|c} 1 & -1 & 1 & 8 \\ 0 & 5 & -3 & -18 \\ 3 & -2 & -9 & 9 \end{array}\right] \xrightarrow{-3R_1+R_3=R_3} \left[\begin{array}{ccc|c} 1 & -1 & 1 & 8 \\ 0 & 5 & -3 & -18 \\ 0 & 1 & -12 & -15 \end{array}\right]$$

$$\xrightarrow{R1 \leftrightarrow R2} \left[\begin{array}{ccc|c} 1 & -1 & 1 & 8 \\ 0 & 1 & -12 & -15 \\ 0 & 5 & -3 & -18 \end{array}\right]$$

Sau đó:

$$\xrightarrow{-5R_2+R_3=R_3} \left[\begin{array}{ccc|c} 1 & -1 & 1 & 8 \\ 0 & 1 & -12 & -15 \\ 0 & 0 & 57 & 57 \end{array}\right] \xrightarrow{-\frac{1}{57}R_3=R_3} \left[\begin{array}{ccc|c} 1 & -1 & 1 & 8 \\ 0 & 1 & -12 & -15 \\ 0 & 0 & 1 & 1 \end{array}\right]$$

Ma trận cuối tương đương với hệ sau:

$$\begin{cases} x - y + z = 8 \\ y - 12z = -15 \\ z = 1 \end{cases}$$

Sử dụng quá trình nghịch, ta có được bộ nghiệm: $(4, -3, 1)$

Sử dụng code, ta thu được kết quả như sau:

```
Hệ phương trình bạn vừa nhập là:
x1-x2+x3 = 8
2*x1+3*x2 -x3 = -2
3*x1-2*x2 -9*x3 = 9

Hạng của ma trận là: 3

Phương trình có nghiệm duy nhất:
X1 = 4
X2 = -3
X3 = 1
```

Ví dụ 2 Giải hệ phương trình:

$$\begin{cases} -x_1 - 2x_2 + x_3 = -1 \\ 2x_1 + 3x_2 = 2 \\ x_2 - 2x_3 = 0 \end{cases}$$

Kết quả

Giải theo phương pháp thông thường:

Đầu tiên, chúng ta viết hệ phương trình dưới dạng ma trận:

$$\left[\begin{array}{ccc|c} -1 & -2 & 1 & -1 \\ 2 & 3 & 0 & 2 \\ 0 & 1 & -2 & 0 \end{array}\right] \xrightarrow{-R_1 \to R_1} \left[\begin{array}{ccc|c} 1 & 2 & -1 & 1 \\ 2 & 3 & 0 & 1 \\ 0 & 1 & -2 & 0 \end{array}\right] \xrightarrow{R_2 \leftrightarrow R_3} \left[\begin{array}{ccc|c} 1 & 2 & -1 & 1 \\ 0 & 1 & -2 & 0 \\ 2 & 3 & 0 & 2 \end{array}\right]$$

$$\xrightarrow{-2R_1+R_3=R_3} \left[\begin{array}{ccc|c} 1 & 2 & -1 & 1 \\ 0 & 1 & -2 & 0 \\ 0 & -1 & 2 & 0 \end{array}\right] \xrightarrow{R_2+R_3=R_3} \left[\begin{array}{ccc|c} 1 & 2 & -1 & 1 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

Ma trận cuối cho ta một hệ phương trình như sau:

$$\begin{cases} x_1 + 2x_2 - x_3 = 1 \\ x_2 - 2x_3 = 0 \\ 0 = 0 \end{cases}$$

Nhận thấy rằng có đồng nhất thức $0 = 0$, đây chính là dấu hiệu của phương trình có vô số nghiệm. Bằng cách đặt $x_3 = t_3$ là tham số sau đó giải phương trình thứ hai cho x_2 và thế vào phương trình thứ nhất ta có thể tìm ra x_1 theo x_3.

Từ đó ta có bộ nghiệm: $X = (1 - 3t_3,\ \ 2t_3,\ \ t_3)$

Sử dụng code, ta thu được kết quả như sau:

```
Hệ phương trình bạn vừa nhập là:
-x1-2*x2+x3 = -1
2*x1+3*x2 = 2
+x2 -2*x3 = 0


Hạng của ma trận là: 2

Phương trình có vô số nghiệm, với nghiệm dạng tham số là:
X1 = 1 + -3t3
X2 = 0 + 2t3
X3 = t3
```

Ví dụ 3 Giải hệ phương trình:

$$\begin{cases} x_1 + x_2 - x_3 = 0 \\ 2x_1 - x_2 - x_3 = -2 \\ 4x_1 + x_2 - 3x_3 = 5 \end{cases}$$

Kết quả

Giải theo phương pháp thông thường:

Đầu tiên, chúng ta viết hệ phương trình dưới dạng ma trận:

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ 2 & -1 & -1 & -2 \\ 4 & 1 & -3 & 5 \end{array}\right] \xrightarrow[R_2=R_2-2R_1]{R_3=R_3-2R_2} \left[\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ 0 & -3 & 1 & -2 \\ 0 & 3 & -1 & 9 \end{array}\right] \xrightarrow{R_3=R_3+R_2} \left[\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ 0 & -3 & 1 & -2 \\ 0 & 0 & 0 & 7 \end{array}\right]$$

Hàng thứ 3 tương ứng với phương trình

$$0x_1 + 0x_2 + 0x_3 = 7$$

phương trình này vô nghiệm.

Do đó, hệ phương trình vô nghiệm.

Sử dụng code, ta thu được kết quả như sau:

```
Hệ phương trình bạn vừa nhập là:
x1+x2 -x3 = 0
2*x1-x2 -x3 = -2
4*x1+x2 -3*x3 = 5


Phương trình vô nghiệm
```

4.2 Một số ví dụ ứng dụng

Ví dụ 4 An đầu tư tổng cộng 10000$ vào ba tài khoản, tài khoản một có lãi 5%, tài khoản thứ hai lãi 8% và tài khoản thứ 3 lãi 9%. Tiền lãi hằng năm kiếm được từ ba tài khoản đầu tư sau một năm là 770$. Số tiền đầu tư ở tài khoản có lãi 9% gấp đôi số tiền ở tài khoản có lãi 5%. Hỏi mỗi tài khoản đầu tư bao nhiêu tiền?

Kết quả

Giải theo phương pháp thông thường:

Từ đề bài, có được hệ phương trình 3 ẩn x, y, z, với x, y, z lần lượt là số tiền trong tài khoản thứ 1, thứ 2 và thứ 3:

$$\begin{cases} x + y + z = 10000 \\ 0.05x + 0.08y + 0.09z = 770 \\ 2x - z = 0 \end{cases}$$

Đưa về dạng ma trận, ta có: $\left[\begin{array}{ccc|c} 1 & 1 & 1 & 10000 \\ 0.05 & 0.08 & 0.09 & 770 \\ 2 & 0 & -1 & 0 \end{array}\right]$

Bây giờ, sử dụng các phép biến đổi sơ cấp trên các hàng:

$$\xrightarrow{-0.05R_1+R_2=R_2} \left[\begin{array}{ccc|c} 1 & 1 & 1 & 10000 \\ 0 & 0.03 & 0.04 & 270 \\ 2 & 0 & -1 & 0 \end{array}\right] \xrightarrow{-2R_1+R_3=R_3} \left[\begin{array}{ccc|c} 1 & 1 & 1 & 10000 \\ 0 & 0.03 & 0.04 & 270 \\ 0 & -2 & -3 & -20000 \end{array}\right]$$

$$\xrightarrow{\frac{1}{0.03}R_2=R_2} \left[\begin{array}{ccc|c} 1 & 1 & 1 & 10000 \\ 0 & 1 & \frac{4}{3} & 9000 \\ 0 & -2 & -3 & -20000 \end{array}\right] \xrightarrow{2R_2+R_3=R_3} \left[\begin{array}{ccc|c} 1 & 1 & 1 & 10000 \\ 0 & 1 & \frac{4}{3} & 9000 \\ 0 & 0 & -\frac{1}{3} & -2000 \end{array}\right]$$

Từ hàng thứ ba, có: $-\frac{1}{3}z = -2000$ do đó $z = 6000$.
Từ hàng thứ hai, có: $y + \frac{4}{3}z = 9000$ thay $z = 6000$ ta được:

$$y + \frac{4}{3}(6000) = 9000$$

$$y + 8000 = 9000$$

$$y = 1000$$

Từ hàng thứ nhất ta có $x + y + z = 10000$. Thay $y = 1000$ và $z = 6000$, ta có

$$x + 1000 + 6000 = 10000 \to x = 3000$$

Như vậy, tài khoản 1 là 3000$, tài khoản 2 là 1000$ và tài khoản 3 là 6000$.

Sử dụng code, ta thu được kết quả như sau:

```
Hệ phương trình bạn vừa nhập là:
x1+x2+x3 = 10000
0.05*x1+0.08*x2+0.09*x3 = 770
2*x1 -x3 = 0


Hạng của ma trận là: 3

Phương trình có nghiệm duy nhất:
X1 = 2999.99999999998
X2 = 1000.0000000000048
X3 = 5999.999999999997
```

Nhận xét: Kết quả này xấp xỉ so với kết quả tính bằng tay là do có sai số tính toán, và sai số này ở mức chấp nhận được.

Ví dụ 5 Cân bằng phương trình hóa học sau:

$K_4Fe(CN)_6 + KMnO_4 + H_2SO_4 \to KHSO_4 + Fe_2\,(SO_4)_3 + MnSO_4 + HNO_3 + CO_2 + H_2O$

Kết quả Đặt $y_i (i = 1, 2, \ldots 9)$ là hệ số của mỗi chất hóa học trong phương trình nhu sau:

$\mathbf{y}_1 K_4Fe(CN)_6 + \mathbf{y}_2 KMnO_4 + \mathbf{y}_3 H_2SO_4 \to \mathbf{y}_4 KHSO_4 + \mathbf{y}_5 Fe_2\,(SO_4)_3 + \mathbf{y}_6 MnSO_4 + \mathbf{y}_7 HNO_3 + \mathbf{y}_8 \mathrm{CO_2} + \mathbf{y}_9 H_2\mathrm{O}$

Tương ứng với 8 phân tử, ta có thể nhận được tám phương trình tuyến tính như sau:

$$\begin{aligned}
\mathrm{K}: &\quad 4\mathrm{y}_1 + y_2 = y_4 \\
\mathrm{Fe}: &\quad \mathrm{y}_1 = 2y_5 \\
\mathrm{C}: &\quad 6\mathrm{y}_1 = y_8 \\
\mathrm{N}: &\quad 6\mathrm{y}_1 = y_7
\end{aligned}$$

Mn : $y_2 = y_6$

O : $4\,y_2 + 4y_3 = 4y_4 + 12y_3 + 4y_6 + 3y_7 + 2y_8 + y_9$

H : $2\,y_3 = y_4 + y_7 + 2y_9$

S : $y_3 = y_4 + 3y_5 + y_6$

Trong các phương trình trên, chỉ số đại diện cho tổng số nguyên tử của một nguyên tử. Viết lại các được hệ phương trình sau:

$$\begin{cases} 4y_1 + y_2 - y_4 = 0 \\ y_1 - 2y_5 = 0 \\ 6y_1 - y_8 = 0 \\ 6y_1 - y_7 = 0 \\ y_2 - y_6 = 0 \\ 4y_2 + 4y_3 - 4y_4 - 12y_3 - 4y_6 - 3y_7 - 2y_8 - y_9 = 0 \\ 2y_3 - y_4 - y_7 - 2y_9 = 0 \\ y_3 - y_4 - 3y_5 - y_6 = 0 \end{cases}$$

Giải hệ phương trình bằng phương pháp khử Guass: $AY = B$

$$A = \begin{bmatrix} 4 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 4 & 4 & -4 & -12 & -4 & -3 & -2 & -1 \\ 0 & 0 & 2 & -1 & 0 & 0 & -1 & 0 & -2 \\ 0 & 0 & 1 & -1 & -3 & -1 & 0 & 0 & 0 \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Khi đó ma trận mở rộng $\overline{A} = [A \mid B] = \left[\begin{array}{ccccccccc|c} 4 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 4 & 4 & -4 & -12 & -4 & -3 & -2 & -1 & 0 \\ 0 & 0 & 2 & -1 & 0 & 0 & -1 & 0 & -2 & 0 \\ 0 & 0 & 1 & -1 & -3 & -1 & 0 & 0 & 0 & 0 \end{array}\right]$

Giải ma trận này bằng phương pháp khử Guass. Tham khảo các cách biến đổi trên web: http://matrixcalc.org/vi/slu.html, ta có kết quả:

$$Y = \begin{pmatrix} 0.05 \times t_9 \\ 0.65 \times t_9 \\ 1.59 \times t_9 \\ 0.86 \times t_9 \\ 0.03 \times t_9 \\ 0.65 \times t_9 \\ 0.32 \times t_9 \\ 0.32 \times t_9 \\ t_9 \end{pmatrix}$$

Như vậy đây là một hệ phương trình vô số nghiệm, để cho các hệ số là số nguyên.

Ta chọn $y_9 = t_9 = 188$;

từ đó có được $y_1 = 10, y_2 = 122\ y_3 = 299, y_4 = 162, y_5 = 5, y_6 = 122, y_7 = 60, y_8 = 60$

Đây sẽ là phương trình sau khi cân bằng:

10$K_4Fe(CN)_6 +$ **122**$KMnO_4 +$ **299**$H_2SO_4 =$

162$\mathrm{KHSO_4} +$ **5**$\mathrm{Fe_2\left(SO_4\right)_3} +$ **122**$\mathrm{MnSO_4}+$ **60**$\mathrm{HNO_3} +$ **60**$\mathrm{CO_2} +$ **188**$\mathrm{H_2O}$

Sử dụng code, ta thu được kết quả như sau:

```
Hệ phương trình bạn vừa nhập là:
4*x1+x2 -x4 = 0
x1 -2*x5 = 0
6*x1 -x8 = 0
6*x1 -x7 = 0
+x2 -x6 = 0
+4*x2+4*x3 -4*x4 -12*x5 -4*x6 -3*x7 -2*x8 -x9 = 0
+2*x3 -x4 -x7 -2*x9 = 0
+x3 -x4 -3*x5 -x6 = 0


Hạng của ma trận là: 8

Phương trình có vô số nghiệm, với nghiệm dạng tham số là:
X1 = 0 + 0.05319148936170215t9
X2 = 0 + 0.6489361702127661t9
X3 = 0 + 1.5904255319148937t9
X4 = 0 + 0.8617021276595747t9
X5 = -0 + 0.026595744680851064t9
X6 = -0 + 0.6489361702127661t9
X7 = 0 + 0.3191489361702128t9
X8 = 0 + 0.3191489361702128t9
X9 = t9
```

Nhận xét:** **Kết quả nhận được qua việc chạy code xấp xỉ kết quả giải thông thường. Xảy ra việc phương pháp giải đúng nhận được kết quả xấp xỉ là do trong các bước tính toán có xảy ra việc làm tròn dẫn đến có sai số tính toán. Trong bài toán này, sai số tính toán ở mức chấp nhận được.

4.3 Một số trường hợp đặc biệt

Ví dụ 6 Giải hệ phương trình:

$$\begin{cases} x_1 + 2x_2 + 10^{-4}x_3 + x_4 + 5x_5 = 10000 \\ 2x_1 + x_2 + 5x_3 + 6x_4 + x_5 = 0.00001 \\ 5x_1 + x_2 + 2x_3 + 6x_4 + x_5 = 6 \\ 5x_1 + 8x_2 + 5.10^{-3}x_3 + x_4 + 4x_5 = 5 \\ x_1 + 2x_2 + 5x_3 + x_4 + 6x_5 = 7 \end{cases}$$

Kết quả

Sử dụng code, ta thu được kết quả như sau:

```
Hệ phương trình bạn vừa nhập là:
x1+2*x2+0.0004*x3+x4+5*x5 = 10000
2*x1+x2+5*x3+6*x4+x5 = 1E-05
5*x1+x2+2*x3+6*x4+x5 = 6
5*x1+8*x2+0.005*x3+x4+4*x5 = 5
x1+2*x2+5*x3+x4+6*x5 = 7


Hạng của ma trận là: 5

Phương trình có nghiệm duy nhất:
X1 = -2376.2357953192513
X2 = 236.3517885888226
X3 = -2378.235791985919
X4 = 2418.345184394413
X5 = 1897.2276656127976
Thử lại ta được vector B = A.X là:
10000
9.999999520005076E-06
6.000000000003411
4.9999999999972715
6.999999999996362
```

Nhận xét: Kết quả này sau khi thử lại ta được một vector B' sát với vector B của đề bài, sai số xảy ra là do việc chênh lệch hệ số, có những hệ số lớn (như là 10000) nhưng cũng có những hệ số nhỏ (như là 0.0004). Như vậy sẽ có sai số khi tính toán, dẫn đến nghiệm không còn là nghiệm đúng nữa mà là nghiệm xấp xỉ. Nhưng trong ví dụ này, sai số ở mức chấp nhận được.

Ví dụ 7 Giải hệ phương trình:

$$\begin{cases} x + 2y = 4 \\ 2x + 3.999y = 7.999 \end{cases}$$

Kết quả

Ta có ma trận mở rộng tương ứng:

$$\left[\begin{array}{cc|c} 1.000 & 2.000 & 4.000 \\ 2.000 & 3.999 & 7.999 \end{array}\right] \xrightarrow{R_2 = R_2 - 2.000 \times R_1} \left[\begin{array}{cc|c} 1.000 & 2.000 & 4.000 \\ 0 & -0.001 & -0.001 \end{array}\right] \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Thực hiện một thay đổi nhỏ trong vectơ bên phải của phương trình:

$$\left[\begin{array}{cc|c} 1.000 & 2.000 & 4.001 \\ 2.000 & 3.999 & 7.998 \end{array}\right] \xrightarrow{R_2 = R_2 - 2.000 \times R_1} \left[\begin{array}{cc|c} 1.000 & 2.000 & 4.000 \\ 0 & -0.001 & -0.004 \end{array}\right] \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -3.999 \\ 4 \end{bmatrix}$$

Thực hiện một thay đổi nhỏ trong ma trận hệ số của các phương trình:

$$\left[\begin{array}{cc|c} 1.001 & 2.001 & 4.000 \\ 2.001 & 3.998 & 7.999 \end{array}\right] \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 0.001388 \end{bmatrix}$$

Nhận xét: Như vậy, với ví dụ này nhìn có vẻ là "trường hợp xấu" bởi khi thực hiện một thay đổi rất nhỏ ở ma trận hệ số thì nghiệm nhận được lại có thay đổi rất lớn. Mà trong khi tính toán trên máy tính, kết quả của mỗi phép toán được lưu lại dưới dạng số thập phân, và khi kết quả là số vô tỷ, hoặc xảy ra tràn số thì việc làm tròn số khi lưu chính là việc tác động một thay đổi nhỏ đến mỗi phần tử hệ số từ đó dẫn ra sự thay đổi về nghiệm. Như vậy, nghiệm thu được sau nhiều bước tính toán không còn đáng tin nữa.

5 Đánh giá bài báo cáo

5.1 Đánh giá thuật toán

Tính đúng đắn và sai số

Phương pháp khử Guass là phương pháp giải đúng hệ phương trình tuyến tính. Do đó không có sai số phương pháp mà chỉ có sai số do tính toán. Do phương pháp khử Gauss thực hiện trực tiếp lên ma trận hệ số, và chỉ có những phép biến đổi sơ cấp, không có chọn lọc phần tử khử nên việc sai số sau các bước tính toán là không thể kiểm soát. Đối với những ma trận có hệ số đơn giản, hoặc có thể coi là "ước", "bội" của nhau thì kết quả nhận được của chúng ta sẽ hoàn toàn đúng, các phép tính toán đơn giản sẽ không có ảnh hưởng bởi sai số. Nhưng nếu như các phần tử trong ma trận hệ số có sự chênh lệch gấp nhau quá nhiều lần, hoặc tồn tại những phép tính toán có kết quả là số vô tỷ thì sẽ xảy ra việc làm tròn số (do tràn số khi lưu trữ), từ đó kết quả nhận được là kết quả gần đúng. Sai số của việc tính toán là không thể kiểm soát vì vậy phương pháp khử Guass có những điểm hạn chế cần được khắc phục nhất định.

Trường hợp tốt và trường hợp xấu

Trường hợp tốt: Khi đầu vào là các ma trận hệ số là các ma trận tam giác trên hoặc ma trận bậc thang thì chúng ta có thể sử dụng trực tiếp quá trình nghịch mà không cần thực hiện các biến đổi của quá trình thuận. Vì vậy các phép tính toán cũng như thời gian tính toán sẽ được cải thiện đáng kể.

Trường hợp xấu:

- Ma trận hệ số có các phần tử chênh lệch nhau quá nhiều lần, khi đó trong các phép tính toán sẽ xảy ra những sai số rất lớn. Ví dụ như việc chia cho số xấp xỉ 0, khi đó ta sẽ nhận được kết quả là dương vô cùng, cùng với việc lưu trữ trong máy tính thì các bước tính toán sau đó có thể là cộng hoặc trừ thì sẽ vô nghĩa, vì vậy nghiệm nhận được có sai số rất lớn so với nghiệm đúng của phương trình.
- Ma trận thuộc loại không ổn định "illcondition". Đối với những ma trận như vậy, khi ta thực hiện một tác động rất nhỏ vào các phần tử thuộc ma trận hệ số thì ta lại nhận được một sự thay đổi rất lớn của vector nghiệm. Nhưng trong việc lưu trữ các kết quả tính toán, kết quả sẽ được lưu dưới dạng số thập phân. Như vậy, khi mà kết quả của một phép tính là số vô tỷ, hay số có phần thập phân "tràn" so với độ dài mà máy tính có thể lưu trữ thì chính việc làm tròn khi lưu trữ đó đã tác động đến các phần tử của ma trận hệ số một tác động nhỏ. Như vậy, nghiệm mà chúng ta nhận được sẽ thay đổi rất lớn so với nghiệm đúng mà đáng ra ta sẽ nhận được, và khi đó nghiệm không còn đáng tin nữa. Những hệ phương trình như vậy sẽ là vô nghĩa khi giải bằng phương pháp Guass. Và đây được coi là trường hợp xấu của phương pháp.

Tổng hợp các thuật toán chính sử dụng trong báo cáo

Ở đây, ta giả sử trường hợp "xấu nhất", đó chính là hệ phương trình n ẩn m phương trình và có nghiệm duy nhất. Từ đó ta có bảng tổng hợp sau:

STT	Thuật toán	INPUT	OUTPUT	Độ phức tạp
1	Quá trình thuận	Ma trận mở rộng A	Ma trận A ở dạng bậc thang	$O(n^3)$
2	Tìm nghiệm	Ma trận mở rộng A đã được đưa về dạng bậc thang	Nghiệm của hệ phương trình	$O(n^3)$
3	Giải ma trận ba đường chéo	Ma trận ba đường chéo	Nghiệm của phương trình	$O(n)$
4	Tính định thức của ma trận	Ma trận vuông cấp n	Định thức củama trận	$O(n)$
5	Kiểm tra vô nghiệm	Ma trận mở rộng	Hệ phương trình có vô nghiệm hay không	$O(n)$

Từ đó, rút ra đánh giá chung của phương pháp Gauss:

- Là một phương pháp giải đúng nghiệm của phương trình (nghĩa là không có sai số phương pháp, chỉ có sai số tính toán).
- Ý tưởng đơn giản, tận dụng được mảng index (lưu vị trí đầu tiên của phần tử đầu tiên khác 0 trong mỗi hàng).
- Hạn chế: Sai số rất lớn, dẫn đến kết quả sai như một số phép chia cho số xấp xỉ 0, các phần tử trong hàng cần khử rất lớn so với phần tử khử. Để khắc phục hạn chế này người ta sử dụng Gauss - Jordan (chọn phần tử khử).
- Có độ phức tạp chung là $O(n^3)$.

5.2 So sánh một số phương pháp khác

So sánh với một số phương pháp khác:

Đặc điểm	**Gauss**	**Gauss - Jordan**	**Choleski**
Điều kiện	Không có	Không có	Có điều kiện ràng buộc
Độ phức tạp	$\frac{2n^3}{3}$	n^3	$\frac{n^3}{3}$, xấu nhất $\frac{4n^3}{3}$
Nghiệm duy nhất	Có	Có	Có
Vô nghiệm	Có	Có	Không
Vô số nghiệm	Có	Có	Không
Nhận xét	Giải được hệ phương trình bất kì, nhưng không kiểm soát được sai số.	Giải được hệ phương trình bất kì, kiểm soát được sai số.	Có yêu cầu đầu vào, nếu hệ thỏa mãn, thời gian chạy nhanh hơn nhưng vẫn không kiểm soát được sai số.

6 Một số kết quả mở rộng

6.1 Ý tưởng "Pivot Row" giảm thiểu sai số tính toán

Bước 1: Ở mỗi hàng chọn ra phần tử có trị tuyệt đối lớn nhất. Đặt là M_i.

Bước 2: Ta tính tỉ lệ $r_i = \left|\frac{a_{1i}}{M_i}\right|$ (tỉ lệ giữa phần tử cột 1 với GTLN trong hàng của nó).

Bước 3: Chọn r_i max. Lấy hàng i đó làm hàng để khử (tức là đẩy lên hàng 1 để khử).

Bước 4: Sau đó tiếp tục đi tìm hàng thứ 2 khử các hàng dưới bằng phương pháp trên.

Ví dụ 8 Đưa ma trận sau về dạng bậc thang:

$$\left[\begin{array}{ccc|c} -4 & -3 & 5 & 0 \\ 6 & 7 & -3 & 2 \\ 2 & -1 & 1 & 6 \end{array}\right]$$

Kết quả

$$r_1 = \frac{|-4|}{|5|} = \frac{4}{5}, \quad r_2 = \frac{|6|}{|7|} = \frac{6}{7}, \quad r_3 = \frac{|2|}{|2|} = 1$$

$$\xrightarrow{R_1 \leftrightarrow R_3} \left[\begin{array}{ccc|c} 2 & -1 & 1 & 6 \\ 6 & 7 & -3 & 2 \\ -4 & -3 & 5 & 0 \end{array}\right] \xrightarrow[R_3+2\times R_1=R_3]{R_2-3\times R_1=R_2} \left[\begin{array}{ccc|c} 2 & -1 & 1 & 6 \\ 0 & 10 & -6 & -16 \\ 0 & -5 & 7 & 12 \end{array}\right]$$

Làm tương tự với ma ma trận $\left[\begin{array}{cc|c} 10 & -6 & -16 \\ -5 & 7 & 12 \end{array}\right]$

có $r_1 = \frac{|10|}{|-16|} = \frac{5}{8}, \quad r_2 = \frac{|-5|}{|12|} = \frac{5}{12}$

$$\Rightarrow \left[\begin{array}{cc|c} 10 & -6 & -16 \\ -5 & 7 & 12 \end{array}\right] \xrightarrow{2\times R_2+R_1=R_2} \left[\begin{array}{cc|c} 10 & -6 & -16 \\ 0 & 4 & 4 \end{array}\right]$$

như vậy, $\left[\begin{array}{ccc|c} -4 & -3 & 5 & 0 \\ 6 & 7 & -3 & 2 \\ 2 & -1 & 1 & 6 \end{array}\right]$ trở thành $\left[\begin{array}{ccc|c} -2 & -1 & 1 & 6 \\ 0 & 10 & -6 & -16 \\ 0 & 0 & 4 & 4 \end{array}\right]$

Từ đó khi sử dụng phương pháp này ta sẽ giảm thiểu được sai số trong lúc tính toán, nhưng vẫn có thể giữ được bản chất của Gauss.

6.2 Giải ma trận 3 đường chéo

Định nghĩa: Ma trận ba đường chéo là một ma trận có các phần tử khác 0 trên đường chéo chính, các phần tử của đường chéo đầu tiên bên trên và đường chéo đầu tiên bên dưới đường chéo này cũng có giá trị khác 0, các phần tử còn lại trên ma trận bằng 0.

$$\begin{bmatrix} b_0 & c_0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ a_1 & b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{n-3} & b_{n-3} & c_{n-3} & 0 \\ 0 & 0 & 0 & \cdots & 0 & a_{n-2} & b_{n-2} & c_{n-2} \\ 0 & 0 & 0 & \cdots & 0 & 0 & a_{n-1} & b_{n-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-3} \\ x_{n-2} \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-3} \\ d_{n-2} \\ d_{n-1} \end{bmatrix}$$

Trong đó, có thể thấy dạng tổng quát là: $a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$ với $a_1 = 0$ và $c_n = 0$

Algorithm 3 Thuật toán cải tiến Gauss giải ma trận ba đường chéo

Input: Ma trận ba đường chéo

Output: Số nghiệm và biểu diễn nghiệm của phương trình

Function *TriDiagonalMatrix*

for i = 1 to m do

if i = 1 then

$c'_i = \dfrac{c_i}{b_i}$

$d'_i = \dfrac{d_i}{b_i}$

else

$c'_i = \dfrac{c_i}{b_i - a_i c'_{i-1}}$

$d'_i = \dfrac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}}$

$x_n = d'_n$

for i = n - 1 dow to n do

$x_i = d'_i - c'_i x_{i+1}$

Đối với các ma trận như vậy, cách giải sẽ có độ phức tạp là $O(n)$ thay vì độ phức tạp của Gauss là $O(n^3)$.

Để thực hiện kiểm tra và so sánh việc sử dụng giải bằng Gauss thông thường và áp dụng thuật toán giải cho ma trận ba đường chéo thì chúng em sử dụng một hàm đo thời gian, và đầu vào mình sẽ random một ma trận vuông. Từ đó đo thời gian tính toán từ khi có được ma trận mở rộng đến khi ra được nghiệm.

> **Ví dụ 9** Random một hệ phương trình có 1000 ẩn, giải bằng hai phương pháp, Gauss thông thường và Gauss dành cho ma trận ba đường chéo.

Kết quả Sử dụng code, ta thu được kết quả như sau:

```
+------------------------------------------------------------+
|                       NHẬP MA TRẬN                         |
+---+--------------------------------------------------------+
| 1 |                NHẬP TỪ CONSOLE                         |
+---+--------------------------------------------------------+
| 2 |                NHẬP TỪ FILE                            |
+---+--------------------------------------------------------+
| 3 |                RANDOM RA MA TRẬN BẤT KÌ                |
+---+--------------------------------------------------------+
| 4 |                RANDOM RA MA TRẬN 3 ĐƯỜNG CHÉO          |
+---+--------------------------------------------------------+

Nhập lựa chọn của bạn: 4

Nhập số ẩn: 1000

Nhập khoảng giá trị trong ma trận: 100

Ma trận bạn vừa nhập là ma trận 3 đường chéo

Thời gian giải ma trận 3 đường chéo pp rút gọn: 5891ms
Hạng của ma trận là: 1000

Phương trình có nghiệm duy nhất:

Thời gian giải với pp Gauss: 8569ms
```

Như vậy với hệ phương trình này, thời gian giải bằng Gauss thông thường mất 8569ms (tức xấp xỉ 1,43 phút), còn giải bằng Gauss khi đã cải tiến cho phù hợp với ma trận ba đường chéo thì chỉ còn 5891ms (tức 0,98 phút). Thời gian này được cải thiện đáng kể.

6.3 Tính định thức của ma trận

Định thức của ma trận là một hàm cho mỗi ma trận vuông A, tương ứng với số vô hướng, ký hiệu là **det(A)**. Ý nghĩ hình học của định thức là tỷ lệ xích cho thể tích khi A được coi là một biến đổi tuyến tính. Định thức được sử dụng để giải (và biện luận) các hệ phương trình đại số tuyến tính.

Định thức chỉ được xác định trong các ma trận vuông. Nếu định thức của một ma trận bằng 0, ma trận này được gọi là ma trận suy biến, nếu định thức bằng 1, ma trận này được gọi là ma trận đơn modul.

Định nghĩa: Định thức của ma trận vuông cấp n là tổng đại số của n! số hạng, mỗi số hạng là tích của n phần tử lấy trên các hàng và các cột khác nhau của ma trận A, mỗi tích được nhân với phần tử dấu là $+1$ hoặc -1 theo phép thế tạo bởi các chỉ số hàng và chỉ số cột của các phần tử trong tích. Gọi S_n là nhóm các hoán vị của n phần tử $1, 2, \ldots, n$ ta có:

$$det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}$$

Algorithm 4 Thuật toán cải tiến Gauss giải ma trận ba d

Input: Ma trận đã được sắp xếp dạng bậc thang

Output: Định thức của ma trận

Function *DetMaTran*

```
for i = 1 to m do
    det = det × a_ij
det = det × (−1)^SoLanDoiHang
```

6.4 Tính ma trận nghịch đảo của ma trận

Định nghĩa ma trận nghịch đảo: Ma trận A vuông cấp n được gọi là khả nghịch trên vành V nếu tồn tại ma trận A' cùng cấp n sao cho $A \times A' = A \times A' \times A = E$. Khi đó A' được gọi là ma trận nghịch đảo của ma trận A, ký hiệu là A^1.

Cách tích ma trận nghịch đảo thông thường:

a) *Định thức con và phần bù đại số:*

Cho ma trận vuông A cấp n và phần tử a_{ij}. Định thức của ma trận cấp $n-1$ suy ra từ A bằng cách xóa đi dòng thứ i, cột thứ j được gọi là định thức con của A ứng với phần tử a_{ij}, ký hiệu là M_{ij}.

Định thức con M_{ij} với dấu bằng $(-1)^{i+j}$ được gọi là phần bù đại số của phần tử a_{ij}, ký hiệu là A_{ij}.

b) *Công thức tính ma trận nghịch đảo:*

Nếu định thức của ma trận A là khả nghịch thì ma trận nghịch đảo của A được tính bằng công thức:

$$A^{-1} = \frac{1}{det(A)} \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix}$$

c) *Các bước tìm ma trận nghịch đảo:*

- *Bước 1*:
  - Tính định thức của ma trận A.
  - Nếu $det(A) = 0$ thì A không có ma trận nghịch đảo A^{-1}.
  - Nếu $det(A) \neq 0$ thì A có ma trận nghịch đảo A^{-1}, chuyển sang bước 2.
- *Bước 2*:
  - Lập ma trận chuyển vị A' của A.
- *Bước 3*:
  - Lập ma trận phụ hợp của A' được định nghĩa như sau $A^* = (A'_{ij})_{nn}$
    với $A' = A'_{ij}$ là phần bù đại số của phần tử ở hàng i, cột j trong ma trận A'.
- *Bước 4*:
  - Tính ma trận $A^{-1} = \dfrac{1}{det(A)} A^*$.

Ý tưởng cải tiến từ Gauss

- Đầu vào phải là ma trận khả nghịch (tức là vuông và $det \neq 0$).
- Dùng Gauss xử lý ma trận mở rộng $[A \mid E]$ đưa về ma trận $[E \mid A^{-1}]$.

6.5 Hàm kiểm tra kết quả và kiểm tra tính an toàn của đầu vào

6.5.1 Hàm kiểm tra kết quả

Đặt vấn đề: do thuật toán Gauss gặp sai số rất lớn trong một số trường hợp nên dẫn tới việc nghiệm nhận được không chính xác.

Ý tưởng: Thiết lập 1 hàm để đánh giá nghiệm thông qua việc tính chuẩn của vectơ $AX_0 - B$. Trong đó X_0 là nghiệm mình vừa tìm được theo phương pháp Gauss. Mình sẽ đi so sánh chuẩn của vecto đó với sai số ϵ mình cho trước (sai số cho phép). Càng gần ϵ thì nghiệm càng đáng tin.

6.5.2 Kiểm tra hệ phương trình không ổn định (ill - conditioned)

Đặt vấn đề: Khi giải hệ phương trình ta sẽ gặp một số bài toán mà khi ta tác động 1 thay đổi nhỏ vào ma trận ban đầu thì sẽ dẫn tới sự thay đổi lớn của nghiệm X cần tìm.
Ý tưởng: Thiết lập 1 hàm để kiểm tra đầu vào có ổn định hay không thông qua đại lượng $cond(A)$ (số điều kiện của ma trận A) xác định bởi:

$$Cond(A) = \|A\|.\|A^{-1}\|$$

Giá trị $Cond(A)$ càng gần 1 thì hệ càng ổn định.

7 Lời kết và tài liệu tham khảo

Trong bài báo cáo này chúng em đã trình bày những kiến thức mà nhóm đã tìm hiểu về phương pháp Gauss. Bài báo cáo gồm các nội dung cơ bản của phương pháp: ý tưởng, thuật toán, các ví dụ và đưa ra đánh giá cho phương pháp. Ngoài ra nhóm còn trình bày về một số các ứng dụng mở rộng và cải tiến cho phương pháp này.

Phương pháp Gauss là môt phương pháp giải đúng, đơn giản, dễ hiểu và có thể được sử dụng trên máy tính với hàng ngàn hệ phương trình và ẩn số. Tuy nhiên, phương pháp cũng có môt số hạn chế về tốc đô, khả năng thực hiện song song, đặc biệt là rất khó kiểm soát sai số trong một vài trường hợp cụ thể chúng em đã nêu trên. Do đó, phương pháp này không thích hợp với hệ có hàng triệu phương trình. Những hệ lớn như vậy thường được giải bằng các phương pháp lặp để tận dụng lợi ích về tốc đô mà những phương pháp đó mang lại, điển hình như phương pháp lặp Jacobi, phương pháp lặp Gauss- Siedel ...

Chúng em xin cảm ơn cô Hà Thị Ngọc Yến đã giảng dạy môn Giải tích số và hướng dẫn để nhóm chúng em có thể hoàn thành bài báo cáo này. Mặc dù đã có nhiều cố gắng trong việc tìm hiểu, song chúng em vẫn còn hạn chế về kiến thức và ngôn ngữ, nên báo cáo có thể còn một vài thiếu sót. Rất mong nhận được sự góp ý của cô và các bạn đọc giả để bài báo cáo của chúng em được hoàn chỉnh hơn.

Dưới đây là tài liệu tham khảo mà chúng em dùng để hoàn thiện báo cáo của mình:

1. Numerical Methods in Scientific Computing (volume I) - Germund Dahlquist and Ake BJorck.

2. Numerical Methods for Engineers and Scientists Using MATLAB (second Edition) - Ramin S.Esfandiari, phD.

3. Atkinson, Kendall A. An Introduction to Numerical Analysis, 2nd edition, John Wiley Sons, New York, 1989.

4. Golub, Gene H., and Van Loan, Charles F. Matrix computations, 3rd edition, Johns Hopkins, Baltimore, 1996.

5. Burden, R. L., Faires, J. D. (2011). Numerical Analysis. Cengage Learning.

6. Slide bài giảng TS. Hà Thị Ngọc Yến - Đại học Bách Khoa Hà Nội.

7. Bài Giảng Đại số thầy Bùi Xuân Diệu - Đại học Bách Khoa Hà Nội.

8. Trang web http://matrixcalc.org/vi/slu.html.