TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Báo cáo

Phương pháp lặp Jacobi

Giảng viên hướng dẫn: TS. Hà Thị Ngọc Yến

Nhóm 3:

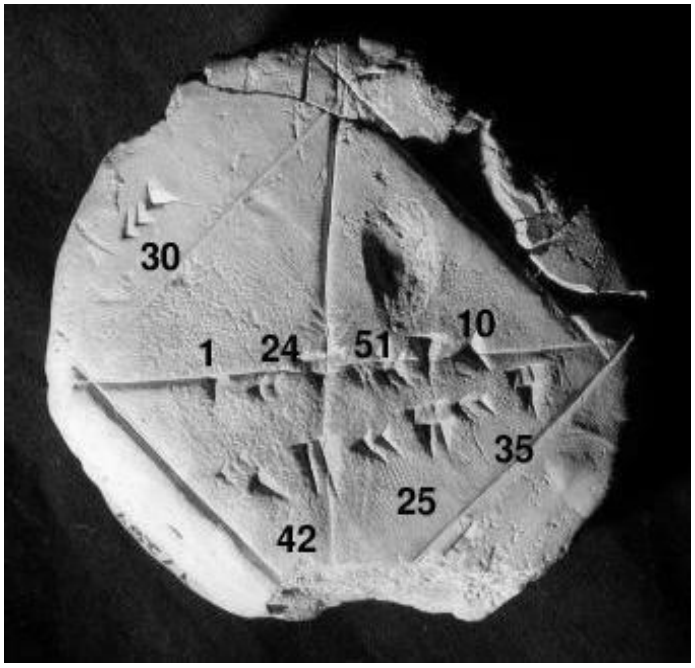
Họ tên	**MSSV**
Nguyễn Hải Đăng	20185333
Đỗ Quang Hùng	20185365
Hoàng Phi Long	20185375
Nguyễn Kim Long	20185379
Trần Hải Phong	20185393

MỤC LỤC

1. Mở đầu

1.1. Giới thiệu chung

- Giải tích số là ngành nghiên cứu về thuật toán sử dụng các số xấp xỉ đối với hàm liên tục cho các vấn đề phân tích toán học (phân biệt với toán học rời rạc).
- Giải tích số được ứng dụng trong tất cả các lĩnh vực kỹ thuật và khoa học vật lý; và trong thế kỷ 21 là cả khoa học đời sống, khoa học xã hội, y học, kinh doanh, và thậm chí là cả nghệ thuật...



Một trong những bản ghi chép toán học sớm nhất về giải tích số: bản ghi Babylon YBC 7289, trong đó nêu một phép tính xấp xỉ √2, độ dài đường chéo của hình vuông đơn vị.

- Sự tăng trưởng về sức mạnh tính toán đã cách mạng hóa việc sử dụng các mô hình toán học thực tế trong khoa học và kỹ thuật, và giải tích số là công cụ cần thiết để thực hiện các mô hình chi tiết này. Ví dụ: phương trình vi phân thông thường xuất hiện trong cơ học thiên thể (dự đoán chuyển động của các hành tinh, sao và thiên hà); đại số tuyến tính số là quan trọng để phân tích dữ liệu; phương trình vi phân ngẫu nhiên và chuỗi Markov rất cần thiết trong việc mô phỏng các tế bào sống cho y học và sinh học....

1.2. Giới thiệu đề tài

- Thay vì các câu trả lời tượng trưng chính xác, yêu cầu khối lượng tính toán lớn cũng như tồn tại những sai lệch ảnh hưởng rất lớn trong quá trình tính, giải tích số đưa ra các nghiệm gần đúng trong giới hạn lỗi được chỉ định, nhất là đối với các bài toán có kích cỡ lớn.
- Trong bài báo cáo này nhóm chúng em sẽ nghiên cứu về một trong những phương pháp được sử dụng để giải gần đúng nghiệm của hệ phương trình tuyến tính bậc nhất AX = B: phương pháp lặp đơn và phương pháp lặp Jacobi.
- Phương pháp Jacobi là một thuật toán lặp để xác định các nghiệm của một ma trận đường chéo trội của phương trình tuyến tính. Mỗi phần tử đường chéo được giải ra và một giá trị gần đúng được đưa vào. Quá trình sau đó được lặp lại cho đến khi nó hội tụ. Phương pháp được đặt tên theo nhà toán học người Đức Jacobi.

Carl Gustav Jacob Jacobi

(1804 – 1851)

2. Cơ sở lý thuyết

2.1. Phương pháp lặp đơn

2.1.1. Chuẩn của véc tơ

Chuẩn của véc tơ X, kí hiệu là $\|X\|$, là một số không âm thỏa mãn:

- $\|X\| \geq 0$, dấu "=" <=> X = 0
- $\|kX\| = |k|.\|X\|$ ∀k∈R
- $\|X + Y\| \leq \|X\| + \|Y\|$

2.1.2. Chuẩn của ma trận

Chuẩn của ma trận A, kí hiệu là $\|A\|$, là một số không âm thỏa mãn:

- $\|A\| \geq 0$, dấu "=" <=> A = 0 (ma trận không)
- $\|kA\| = |k|. \|A\|$ ∀k∈R
- $\|A + B\| \leq \|A\| + \|B\|$

2.1.3. Một số chuẩn thường dùng

Chuẩn	ma trận	véc tơ
∞	$\max \sum_{j=1} \lvert a_{ij} \rvert$	$\max_i \{\lvert x_i \rvert\}$
1	$\max \Sigma_{i=1} \lvert a_{ij} \rvert$	$\sum_{i=1} \lvert x_i \rvert$
2	$(\sum_{i=1}\sum_{j=1} \lvert a_{ij} \rvert 2)^{1/2}$	$(\sum_{i=1} (x_i)^2)^{1/2}$

2.1.4. Ý tưởng

- Đưa phương trình AX=B (1) về dạng X=αX + β, thực hiện phép lặp cho đến khi thu được nghiệm với sai số ε
- Xây dựng công thức

$$AX = B$$
$$\leftrightarrow X = X - AX + B$$
$$\leftrightarrow X = (E-A)X + B$$

đặt $\alpha=E-A$, $\beta=B$

$$\rightarrow X=\alpha X+\beta \quad (2)$$

→ Chọn xấp xỉ đầu vào $X=(0,0,..,0)_t$

→ Thu được công thức: $X_n=\alpha X_{n-1}+\beta$

2.1.5. Sự hội tụ của phương pháp

- **Định lý:**

Sau khi đưa phương trình về dạng (2), nếu có một chuẩn p nào đó của α thỏa mãn:

$$\|\alpha\|_{(p)} \leq q < 1 \ (p = \infty,1,2)$$

thì (1) sẽ có nghiệm duy nhất X* và dãy $X^{(k)}$ sẽ hội tụ về nghiệm X* theo chuẩn p

$$\|X^{(k)} - X^*\|_{(p)} \rightarrow 0 \text{ khi } k \rightarrow \infty \text{ hay } \lim_{k\rightarrow\infty} X^{(k)} = X^*$$

2.1.6. Đánh giá sai số

- Công thức tiên nghiệm:

$$\left\|X^{(k)} - X^*\right\|_{(p)} \leq \frac{q^k}{1-q}\left\|X^{(1)} - X^{(0)}\right\|_{(p)}$$

- Công thức hậu nghiệm:

$$\left\|X^{(k)} - X^*\right\|_{(p)} \leq \frac{q}{1-q}\left\|X^{(k)} - X^{(k-1)}\right\|_{(p)}$$

2.2. Phương pháp lặp Jacobi

2.2.1. Định nghĩa

- Phương pháp lặp Jacobi là phương pháp lặp đơn với A là ma trận chéo trội

2.2.2. Ma trận chéo trội

- Ma trận chéo trội: là ma trận vuông sao cho giá trị tuyệt đối của các phần tử trên đường chéo chính lớn hơn tổng các giá trị tuyệt đối của các phần tử còn lại nằm cùng hàng (hoặc cột)
- Ví dụ: Ma trận chéo trội hàng:

$$A = \begin{bmatrix} 3 & -2 & 1 \\ 1 & -4 & 2 \\ -1 & 2 & 4 \end{bmatrix}$$

$$|a_{11}| > |a_{12}| + |a_{13}| \qquad |+3| > |-2| + |+1|$$
$$|a_{22}| > |a_{21}| + |a_{23}| \qquad |-4| > |+1| + |+2|$$
$$|a_{33}| > |a_{31}| + |a_{32}| \qquad |+4| > |-1| + |+2|$$

a. Trường hợp ma trận chéo trội hàng

$$|aii| \geq \sum_{j=1, j\neq i}^{n} |aij| \qquad i = 1, n$$

B1: Chia các hàng cho phần tử trội của hàng đó

=> A'X=B'

B2: Biến đổi

A'X=B' <=> X=X-A'X+B'

<=> X=(E-A')X+B' => Thu được α=E-A', β=B'

b. Trường hợp ma trận chéo trội cột

$$|ajj| \geq \sum_{i=1, i\neq j}^{n} |aij| \qquad j = 1, n$$

Có T=diag ($\frac{1}{a_{11}}, \frac{1}{a_{22}}, \dots, \frac{1}{a_{nn}}$)

Biến đổi:

AX = B <=> TAX = TB

<=> X = X – TAX + TB

<=> X + (E – TA)X + TB

Thu được α = E – TA, β = TB

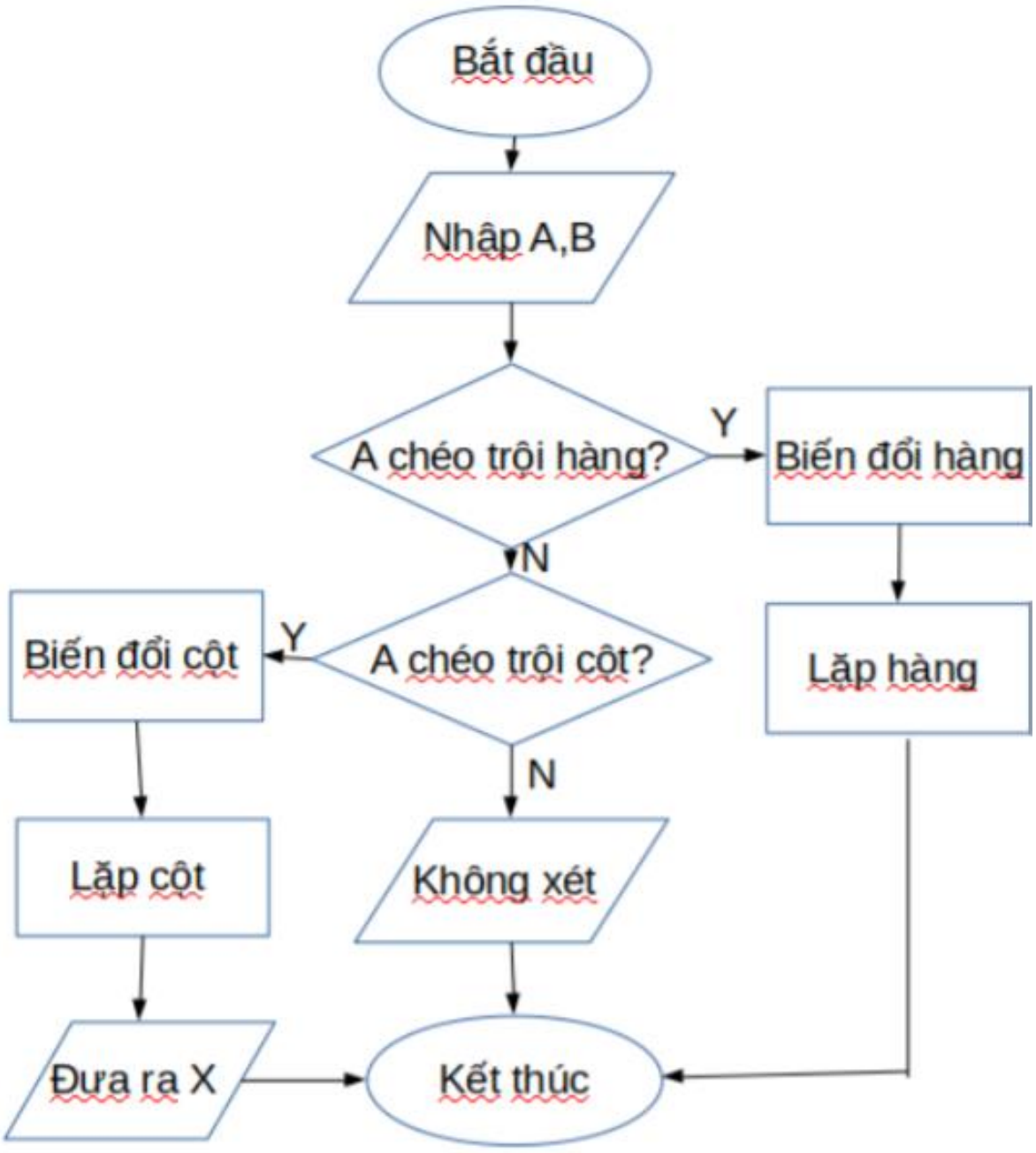
2.2.3. Đánh giá sai số

- Công thức hậu nghiệm:

$$\left\|X^{(k)} - X^{*}\right\|_{(p)} \leq \frac{max|aii|}{min|aii|} \frac{q}{1-q} \left\|X^{(k)} - X^{(k-1)}\right\|_{(p)}$$

3. Giải thuật và lập trình

- Chương trình được code bằng ngôn ngữ C
- Phạm vi những bài toán AX = B có thể giải quyết:
  - A là ma trận chéo trội hàng hoặc cột
  - Chỉ cần đổi chỗ các hàng là thu được ma trận chéo trội hàng hoặc cột
- Thuật toán tổng thể được thể hiện bằng sơ đồ khối sau:



- Khởi tạo

```
7   void enter();
8   void write(float X[100][100],float Y[100]);
9   int kiemtracheotroihang();
10  int kiemtracheotroicot();
11  float chuan(float M[100][100],int p);
12  float chuanx(float M[100],int p);
13  void jacobihang();
14  void jacobicot();
15  int main()
16  {
17      enter();
18      if (kiemtracheotroihang()==1)
19      {
20          printf("Ma tran cheo troi hang \n");
21          jacobihang();
22      }
23          else if (kiemtracheotroicot()==1)
24          {
25              printf("Ma tran cheo troi cot \n");
26              jacobicot();
27          }
28      getch();
29      return 0;
```

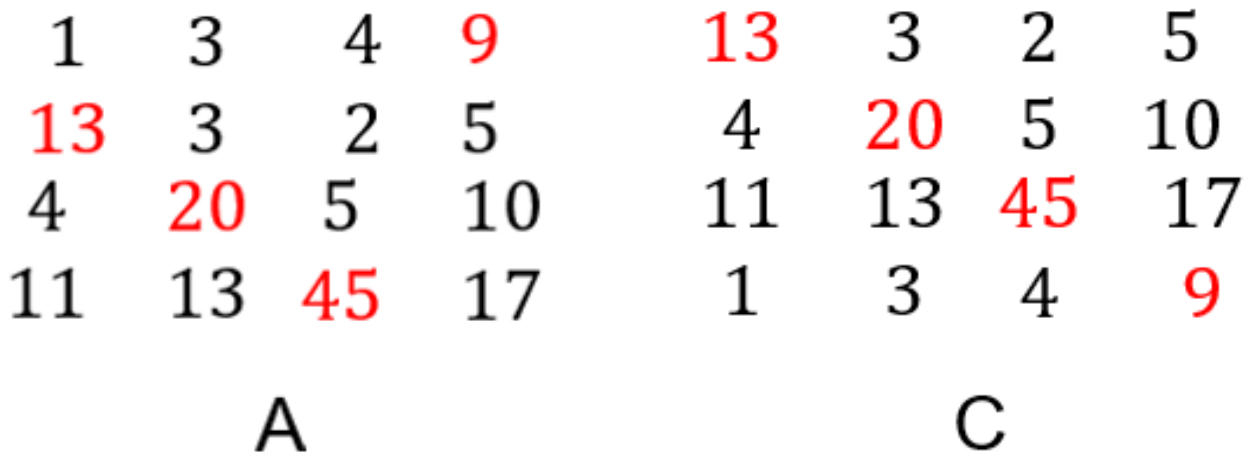
```
31  void enter()
32  {
33      printf("Nhap kich thuoc cua ma tran A va epsilon: ");
34      scanf("%d %f",&n,&epsilon);
35      printf("Nhap ma tran A|B\n");
36      for (int i=1;i<=n;i++)
37          for (int j=1;j<=n+1;j++)
38          {
39              if (j<=n) scanf ("%f",&A[i][j]);
40                  else scanf("%f",&B[i]);
41          }
42  }
```

3.1. Kiểm tra chéo trội hàng

- Ý tưởng thuật toán kiểm tra ma trận chéo trội hàng:
  - Dùng thêm một mảng vitri
  - Kiểm tra phần tử trội của từng hàng
  - Nếu vitri[j] = i thì a[i][j] là phần tử trội
  - Đưa hàng vitri[i] về hàng i để biến đổi A về dạng ma trận chéo trội hàng

```
int kiemtracheotroihang()
{
    int i=0,j=0,sum=0,key,vitri[100]={0};
    float C[100][100],D[100],max=-9999999;
    for (i=1;i<=n;i++)
    {
        max=-9999999; sum=0;
        for (j=1;j<=n;j++)
        {
            sum=sum+abs(A[i][j]);
            if (abs(A[i][j])>max)
            {
                max=abs(A[i][j]);
                key=j;
            }
        }
        if ((max>sum-max) && (vitri[key]==0)) vitri[key]=i;
            else return 0;
    }
    for (i=1;i<=n;i++)
        for (j=1;j<=n;j++) C[i][j]=A[vitri[i]][j];
    for (i=1;i<=n;i++)
        for (j=1;j<=n;j++) A[i][j]=C[i][j];
    for (i=1;i<=n;i++) D[i]=B[vitri[i]];
    for (i=1;i<=n;i++) B[i]=D[i];
    return 1;
```

- Ví dụ:



  - Mảng Vitri = [4,1,2,3]
  - Xây dựng mảng C sao cho C[i][j] = A[Vitri[i]][j]
  - Gán A = C
- Lặp Jacobi theo hàng
  - Xây dựng ma trận α sao cho α [i][j] = - A[i][j]/A[i][i]
  - Xây dựng ma trận β sao cho β [i] = B[i]/A[i][i]
  - Thực hiện lặp đơn với biểu thức: x = α *x + β
  - In kết quả ra màn hình.

```
167  void jacobihang()
168  {
169      float duongcheo[100];
170      int i=0,j=0;
171      for (i=1;i<=n;i++)
172      {
173          duongcheo[i]=A[i][i]; A[i][i]=0;
174          for (j=1;j<=n;j++) A[i][j]=-A[i][j]/duongcheo[i];
175          B[i]=B[i]/duongcheo[i];
176      }
177      printf("Ta se thuc hien lap don ma tran X=Alpha*X+Beta, voi ma tran Alpha|Beta=\n");
178      write(A,B);
179      float q=0,x[100]={0},x0[100]={0},z[100]={0};
180      q=chuan(A,0);
181      int continueloop=1;
182      while (continueloop==1)
183      {
184          for (i=1;i<=n;i++) x[i]=0;
185          for (i=1;i<=n;i++)
186          {
187              for (j=1;j<=n;j++) x[i]=x[i]+A[i][j]*x0[j];
188              x[i]=x[i]+B[i];
189          }
190          for (i=1;i<=n;i++) z[i]=x[i]-x0[i];
191          if (q*chuanx(z,0)/(1-q)<epsilon) continueloop=0;
192              else for (i=1;i<=n;i++) x0[i]=x[i];
193      }
194      printf("Nghiem cua he phuong trinh la: ");
195      for (i=1;i<=n;i++) printf("%f ",x[i]);
```

3.2. Kiểm tra chéo trội cột

- Ý tưởng thuật toán kiểm tra ma trận chéo trội cột:
  - Dùng thêm một mảng vitri
  - Kiểm tra phần tử trội của từng cột
  - Nếu vitri[j] = i thì a[i][j] là phần tử trội
  - Đưa hàng vitri[i] về hàng i để biến đổi A về dạng ma trận chéo trội cột

```
int kiemtracheotroicot()
{
    int i=0,j=0,sum=0,key,vitri[100]={0};
    float C[100][100],D[100],max=-9999999;
    for (j=1;j<=n;j++)
    {
        max=-9999999; sum=0;
        for (i=1;i<=n;i++)
        {
            sum=sum+abs(A[i][j]);
            if (abs(A[i][j])>max)
            {
                max=abs(A[i][j]);
                key=i;
            }
        }
        if ((max>sum-max) && (vitri[key]==0)) vitri[key]=j;
            else return 0;
    }
    for (i=1;i<=n;i++)
        for (j=1;j<=n;j++) C[vitri[i]][j]=A[i][j];
    for (i=1;i<=n;i++)
        for (j=1;j<=n;j++) A[i][j]=C[i][j];
    for (i=1;i<=n;i++) D[vitri[i]]=B[i];
    for (i=1;i<=n;i++) B[i]=D[i];
    return 1;
```

- Ví dụ:

30	20	100	110	100	90	50	20
100	90	50	20	20	150	25	15
20	150	25	15	30	20	100	110
40	10	10	150	40	10	10	150

A C

  - Mảng Vitri = [2,3,1,4]
  - Xây dựng mảng C sao cho C[vitri[i]][j] = A[i][j]
  - Gán A = C
- Lặp Jacobi theo hàng
  - Biến đổi A sao cho A[i][j] = A[i][j]/A[i][i]
  - Xây dựng ma trận αsao cho α [i][j] = I – A[i][j]
  - Xây dựng ma trận βsao cho β [i] = B[i]
  - Lặp Jacobi với biểu thức x’ = α*x’ + β với x’[i] = A[i][i]*x
  - In kết quả ra màn hình.

```
197  void jacobicot()
198  {
199      float duongcheo[100];
200      int i=0,j=0;
201      for (j=1;j<=n;j++)
202      {
203          duongcheo[j]=A[j][j]; A[j][j]=0;
204          for (i=1;i<=n;i++) A[i][j]=-A[i][j]/duongcheo[j];
205      }
206      printf("Ta se thuc hien lap don ma tran X=Alpha*X+Beta, voi ma tran Alpha|Beta=\n");
207      write(A,B);
208      float q=0,x[100]={0},x0[100]={0},z[100]={0};
209      q=chuan(A,1);
210      int continueloop=1;
211      while (continueloop==1)
212      {
213          for (i=1;i<=n;i++) x[i]=0;
214          for (i=1;i<=n;i++)
215          {
216              for (j=1;j<=n;j++) x[i]=x[i]+A[i][j]*x0[j];
217              x[i]=x[i]+B[i];
218          }
219          for (i=1;i<=n;i++) z[i]=x[i]-x0[i];
220          if (q*chuanx(z,1)/(1-q)<epsilon) continueloop=0;
221              else for (i=1;i<=n;i++) x0[i]=x[i];
222      }
223      printf("Nghiem cua he phuong trinh la: ");
224      for (i=1;i<=n;i++) printf("%f ",x[i]/duongcheo[i]);
225  }
```

3.3. Các hàm tính chuẩn

```
140  float chuanx(float M[100],int p)
141  {
142      float max=-999999,sum=0;
143      int i=0,j=0;
144      switch(p)
145      {
146      case 0: //chuan vo cung;
147          for (i=1;i<=n;i++)
148              if (fabs(M[i])>max) max=fabs(M[i]);
149          return max;
150      case 1:
151           //Chuan1
152          for (i=1;i<=n;i++)
153          {
154              sum=0;
155              sum=sum+fabs(M[i]);
156          }
157          return sum;
158      case 2:
159          for (i=1;i<=n;i++)
160          {
161              sum=0;
162              sum=sum+fabs(pow(M[i],2));
163          }
164          return sqrt(sum);
165      }
```

4. Ví dụ

4.1. Xử lý bài toán

- Ma trận chéo trội hàng:

$$\begin{pmatrix} 15 & 5 & 4 & 6 \\ 2 & 10 & 7 & 8 \\ 3 & 11 & 16 & 4 \end{pmatrix}$$

```
Nhap kich thuoc cua ma tran A va epsilon: 3 0.01
Nhap ma tran A|B
15 5 4 6
2 10 7 8
3 11 16 4
Ma tran cheo troi hang
Ta se thuc hien lap don ma tran X=Alpha*X+Beta, voi ma tran
Alpha|Beta=
-0.000000 -0.333333 -0.266667 0.400000
-0.200000 -0.000000 -0.700000 0.800000
-0.187500 -0.687500 -0.000000 0.250000
Nghiem sau lan lap thu 1 la: 0.400000 0.800000 0.250000
Nghiem sau lan lap thu 2 la: 0.066667 0.545000 -0.375000
Nghiem sau lan lap thu 3 la: 0.318333 1.049167 -0.137188
Nghiem sau lan lap thu 4 la: 0.086861 0.832365 -0.530990
Nghiem sau lan lap thu 5 la: 0.264142 1.154320 -0.338537
Nghiem sau lan lap thu 6 la: 0.105503 0.984148 -0.593122
Nghiem sau lan lap thu 7 la: 0.230117 1.194085 -0.446383
Nghiem sau lan lap thu 8 la: 0.121007 1.066445 -0.614080
Nghiem sau lan lap thu 9 la: 0.208273 1.205655 -0.505870
Nghiem sau lan lap thu 10 la: 0.133014 1.112454 -0.617939
Nghiem sau lan lap thu 11 la: 0.193966 1.205954 -0.539752
Nghiem sau lan lap thu 12 la: 0.141949 1.139034 -0.615462
Nghiem sau lan lap thu 13 la: 0.184445 1.202434 -0.559701
Nghiem sau lan lap thu 14 la: 0.148442 1.154902 -0.611257
Nghiem sau lan lap thu 15 la: 0.178035 1.198191 -0.571828
Nghiem sau lan lap thu 16 la: 0.153090 1.164672 -0.607138
Nghiem sau lan lap thu 17 la: 0.173679 1.194378 -0.579417
Nghiem sau lan lap thu 18 la: 0.156385 1.170856 -0.603700
Nghiem sau lan lap thu 19 la: 0.170701 1.191313 -0.584286
Nghiem sau lan lap thu 20 la: 0.158705 1.174860 -0.601034
Nghiem sau lan lap thu 21 la: 0.168656 1.188983 -0.587473
Nghiem sau lan lap thu 22 la: 0.160332 1.177500 -0.599049
Nghiem sau lan lap thu 23 la: 0.167246 1.187268 -0.589593
Nghiem sau lan lap thu 24 la: 0.161469 1.179266 -0.597605
Nghiem sau lan lap thu 25 la: 0.166273 1.186030 -0.591021
Nghiem sau lan lap thu 26 la: 0.162262 1.180460 -0.596572
Nghiem sau lan lap thu 27 la: 0.165599 1.185148 -0.591991
Nghiem sau lan lap thu 28 la: 0.162815 1.181274 -0.595839
Nghiem sau lan lap thu 29 la: 0.165133 1.184524 -0.592653
Nghiem sau lan lap thu 30 la: 0.163199 1.181831 -0.595323
Nghiem sau lan lap thu 31 la: 0.164809 1.184086 -0.593109
Nghiem sau lan lap thu 32 la: 0.163467 1.182214 -0.594961
Nghiem sau lan lap thu 33 la: 0.164585 1.183779 -0.593422
Nghiem sau lan lap thu 34 la: 0.163653 1.182479 -0.594708
Nghiem sau lan lap thu 35 la: 0.164429 1.183565 -0.593639
Nghiem cua he phuong trinh la: 0.164429 1.183565 -0.593639

--------------------------------
Process exited after 54.01 seconds with return value 0
Press any key to continue . . .
```

- Ma trận chéo trội cột A
- Ta có 2 ma trận A|B:

$$\begin{pmatrix} 10 & 12 & 6 & 7 \\ 3 & 14 & 5 & 9 \\ 6 & 1 & 17 & 4 \end{pmatrix}$$

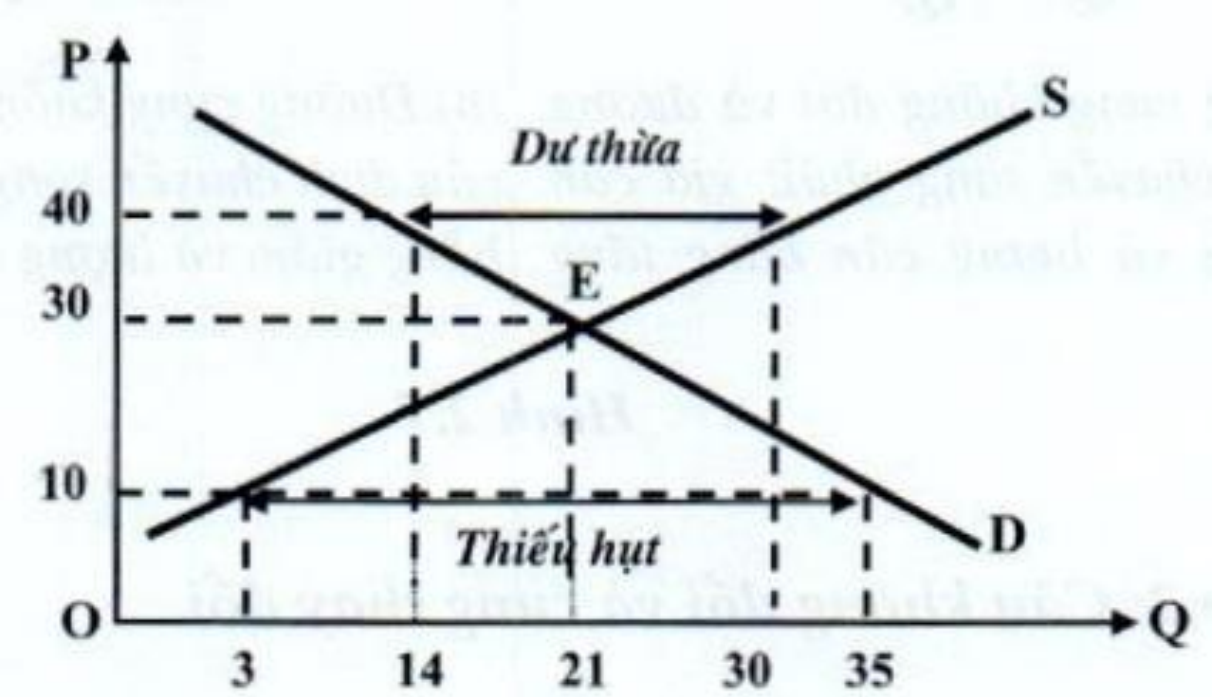
```
Nhap kich thuoc cua ma tran A va epsilon: 3 0.01
Nhap ma tran A|B
10 12 6 7
3 14 5 9
6 1 17 4
Ma tran cheo troi cot
Ta se thuc hien lap don ma tran X=Alpha*X+Beta, voi ma tran Alpha|Beta=
-0.000000 -0.857143 -0.352941 7.000000
-0.300000 -0.000000 -0.294118 9.000000
-0.600000 -0.071429 -0.000000 4.000000
Nghiem sau lan lap thu 1 la: 7.000000 9.000000 4.000000
Nghiem sau lan lap thu 2 la: -2.126051 5.723529 -0.842857
Nghiem sau lan lap thu 3 la: 2.391597 9.885715 4.866807
Nghiem sau lan lap thu 4 la: -3.191166 6.851107 1.858919
Nghiem sau lan lap thu 5 la: 0.471533 9.410609 5.425335
Nghiem sau lan lap thu 6 la: -2.981061 7.262853 3.044894
Nghiem sau lan lap thu 7 la: -0.299971 8.998761 5.269861
Nghiem sau lan lap thu 8 la: -2.573175 7.540032 3.537214
Nghiem sau lan lap thu 9 la: -0.711313 8.731596 5.005332
Nghiem sau lan lap thu 10 la: -2.250813 7.741238 3.803102
Nghiem sau lan lap thu 11 la: -0.977618 8.556684 4.797542
Nghiem sau lan lap thu 12 la: -2.027552 7.882244 3.975379
Nghiem sau lan lap thu 13 la: -1.159284 8.439036 4.653513
Nghiem sau lan lap thu 14 la: -1.875876 7.979105 4.092782
Nghiem sau lan lap thu 15 la: -1.283744 8.359003 4.555590
Nghiem sau lan lap thu 16 la: -1.772715 8.045244 4.173175
Nghiem sau lan lap thu 17 la: -1.368809 8.304410 4.488968
Nghiem sau lan lap thu 18 la: -1.702408 8.090358 4.228113
Nghiem sau lan lap thu 19 la: -1.426867 8.267159 4.443562
Nghiem sau lan lap thu 20 la: -1.654453 8.121130 4.265609
Nghiem sau lan lap thu 21 la: -1.466477 8.241745 4.412591
Nghiem sau lan lap thu 22 la: -1.621738 8.142122 4.291190
Nghiem sau lan lap thu 23 la: -1.493500 8.224407 4.391463
Nghiem sau lan lap thu 24 la: -1.599420 8.156443 4.308642
Nghiem sau lan lap thu 25 la: -1.511934 8.212578 4.377048
Nghiem sau lan lap thu 26 la: -1.584193 8.166213 4.320548
Nghiem sau lan lap thu 27 la: -1.524510 8.204509 4.367215
Nghiem sau lan lap thu 28 la: -1.573807 8.172878 4.328670
Nghiem sau lan lap thu 29 la: -1.533091 8.199004 4.360507
Nghiem sau lan lap thu 30 la: -1.566721 8.177425 4.334211
Nghiem sau lan lap thu 31 la: -1.538944 8.195249 4.355931
Nghiem sau lan lap thu 32 la: -1.561886 8.180527 4.337992
Nghiem sau lan lap thu 33 la: -1.542936 8.192686 4.352808
Nghiem sau lan lap thu 34 la: -1.558588 8.182643 4.340570
Nghiem sau lan lap thu 35 la: -1.545660 8.190938 4.350678
Nghiem sau lan lap thu 36 la: -1.556337 8.184087 4.342329
Nghiem sau lan lap thu 37 la: -1.547519 8.189746 4.349225
Nghiem sau lan lap thu 38 la: -1.554803 8.185072 4.343529
Nghiem sau lan lap thu 39 la: -1.548786 8.188932 4.348234
Nghiem sau lan lap thu 40 la: -1.553756 8.185743 4.344348
Nghiem sau lan lap thu 41 la: -1.549651 8.188377 4.347558
Nghiem sau lan lap thu 42 la: -1.553041 8.186202 4.344907
Nghiem sau lan lap thu 43 la: -1.550241 8.187999 4.347096
Nghiem sau lan lap thu 44 la: -1.552554 8.186515 4.345288
Nghiem sau lan lap thu 45 la: -1.550644 8.187740 4.346781
Nghiem sau lan lap thu 46 la: -1.552221 8.186728 4.345548
Nghiem sau lan lap thu 47 la: -1.550918 8.187564 4.346567
Nghiem sau lan lap thu 48 la: -1.551994 8.186873 4.345725
Nghiem sau lan lap thu 49 la: -1.551105 8.187444 4.346420
Nghiem cua he phuong trinh la: -0.155111 0.584817 0.255672
```

(giải với phương pháp lặp Jacobi)

4.2. Bài toán thực tế

- Trong kinh tế, nếu thị trường lưu hành 1 sản phẩm, hàm cung cầu sẽ được biểu diễn dạng: a+bp. Điểm cân bằng thị trường chính là giao điểm của các đường này.

- Yêu cầu: Xác định điểm cân bằng thị trường bằng cách áp dụng phương pháp lặp Jacobi



- Giải pháp:

- Trước tiên ta xét trường hợp thị trường lưu hành một sản phẩm, sản phẩm đó có thông tin như sau:
  Hàm cung: A = $-a_0 + a_1 p$
  Hàm cầu: B = $b_0 - b_1 p$
  a_0 , a_1, b_0 , b_1 : hằng số dương, p: giá mặt hàng.

- Khi đó mô hình cân bằng thị trường có dạng:

$$\begin{cases} A = -a_0 + a \\ B = b_0 - b_1 p \\ A = B \end{cases} \leftrightarrow \begin{cases} A = -a_0 + a \\ B = b_0 - b_1 p \\ -a_0 + a = b_0 - b_1 p \end{cases}$$

- Giải hệ ta được: $\overline{p} = \frac{a_0 + b_0}{a_1 + b_1}$
- Sản lượng cân bằng:

  $A - B = -a_0 + \frac{a_0 + b_0}{a_1 + b_1} = \frac{a_1 b_0 + a_0 b_1}{a_1 + b_1}$
- Giả sử có n (n ∈ N*) sản phẩm cùng lưu thông trên thị trường. Sản phẩm thứ i có lượng cung, cầu, giá bán là: A_i; B_i; p_i. Giả định rằng các yếu tố khác không thay đổi, hàm cung, hàm cầu sản phẩm thứ i là:

  $A_i = a_{io} + a_{i1} p_1 + a_{i2} p_2 + \ldots + a_{in} p_n$ ∀i = 1,2,....,n

  $B_i = b_{io} + b_{i1} p_1 + b_{i2} p_2 + \ldots + b_{in} p_n$ ∀i = 1,2,....,n
- Thị trường cân bằng khi: A=B ∀i = 1,2,....,n

Ta viết lại điều kiện cân bằng thị trường về dạng sau:

$$\begin{cases} a_{10} + a_{11}p_1 + a_{12}p_2 + \cdots + a_1 p_n = b_{10} + b_{11}p_n + b_{12}p_2 + \cdots + b_1 p_n \\ a_{20} + a_{21}p_1 + a_{22}p_2 + \cdots + a_2 p_n = b_{20} + b_{21}p_n + b_{22}p_2 + \cdots + b_2 p_n \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.. \\ a_{n0} + a_{n1}p_1 + a_{n2}p_2 + \cdots + a_n p_n = b_{n0} + b_{n1}p_n + b_{n2}p_2 + \cdots + b_n p_n \end{cases}$$

Đặt $c_{ik} = a_{ik} - b_{ik}$ $\forall i = 1, 2, \ldots, n,\ \forall k = 0, 1, 2, \ldots, n$

Ta được hệ: $\begin{cases} c_{11}p_1 + c_{12}p_2 + \cdots + c_{1n}p_n = -c_{10} \\ c_{21}p_1 + c_{22}p_2 + \cdots + c_{2n}p_n = -c_{20} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.. \\ c_{n1}p_1 + c_{n2}p_2 + \cdots + c_{nn}p_n = -c_{n0} \end{cases}$

- Áp dụng phương pháp lặp Jacobi để giải hệ phương trình cụ thể, từ đó rút ra các giá trị p_i

- <u>Ví dụ:</u>

- Giả sử hàm cung và hàm cầu được biểu diễn theo các hệ sau:

$$\text{A} = \begin{cases} 8 + 20p_1 + 10p_2 + 8p_3 \\ 12 + 5p_1 + 14p_2 + 8p_3 \\ 9 + 8p_1 + 14p_2 + 18p_3 \end{cases} \quad \text{B} = \begin{cases} 2 + 5p_1 + 5p_2 + 4p_3 \\ 4 + 3p_1 + 4p_2 + 1p_3 \\ 5 + 5p_1 + 3p_2 + 2p_3 \end{cases}$$

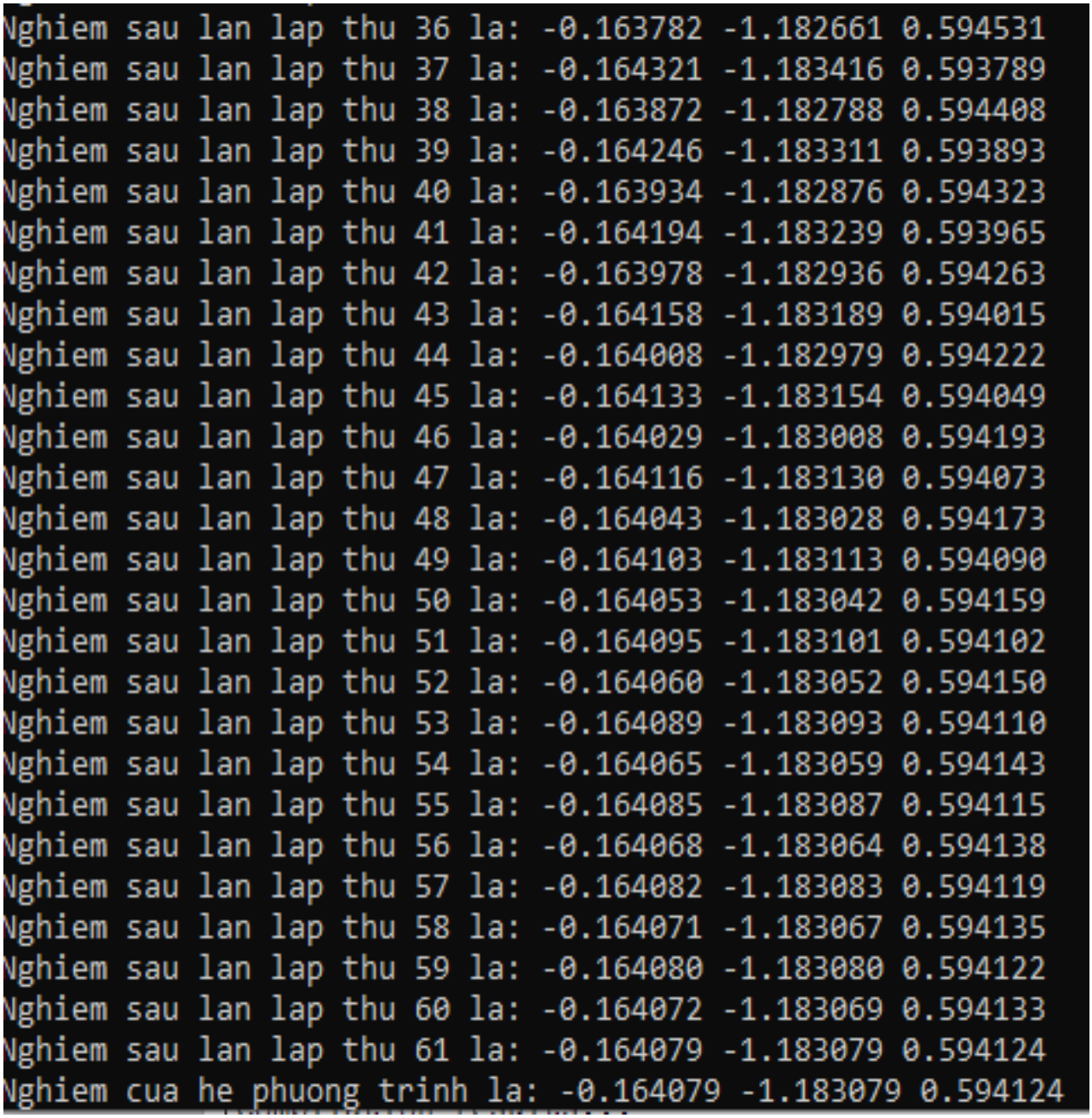
- Thay $c_{ik} = a_{ik} - b_{ik}$, ta thu được hệ:

$$\begin{cases} 6 + 15p_1 + 5p_2 + 4p_3 \\ 8 + 2p_1 + 10p_2 + 7p_3 \\ 4 + 3p_1 + 11p_2 + 16p_3 \end{cases} \leftrightarrow \begin{cases} 15p_1 + 5p_2 + 4p_3 = -6 \\ 2p_1 + 10p_2 + 7p_3 = -8 \\ 3p_1 + 11p_2 + 16p_3 = -4 \end{cases}$$

- Như vậy với các hệ số như trên, ta dễ dàng thiết lập một ma trận chéo trội hàng A|B như sau:

$$\begin{matrix} 15 & 5 & 4 & -6 \\ 2 & 10 & 7 & -8 \\ 3 & 11 & 16 & -4 \end{matrix}$$

- Chạy thuật toán và kết quả của p_1, p_2, p_3 lần lượt là:



(giải với phương pháp lặp Jacobi)

- Như vậy với ma trận trên sau 61 lần lặp cho ta kết quả bộ số (p_1, p_2, p_3) như trong hình.
- Các con số ở trên thể hiện được thị trường cung cầu của dòng sản phẩm mà ta đã đề cập

4.3. So sánh phương pháp Gauss và lặp Jacobi

- Xét hệ phương trình sau:

$$\begin{cases} 0.00001p_1 + 0.00001p_2 + 3p_3 = 4 \\ 5p_1 + 0.00001p_2 + 0.00001p_3 = 4 \\ 0.00001p_1 + 4p_2 + 0.000002p_3 = 2 \end{cases}$$

- Ta thu được ma trận A|B:

$$\begin{matrix} 0.000001 & 0.000001 & 3 & 4 \\ 5 & 0.000001 & 0.000001 & 4 \\ 0.000001 & 4 & 0.000002 & 2 \end{matrix}$$

- Giải với phương pháp Gauss:

```
Hệ phương trình bạn vừa nhập là:
1E-06*x1+1E-06*x2+3*x3 = 4
5*x1+1E-06*x2+1E-06*x3 = 4
1E-06*x1+4*x2+2E-06*x3 = 2


Hạng của ma trận là: 3

Phương trình có nghiệm duy nhất:
X1 = 1.0305114
X2 = 0.40000007
X3 = 1.3333329

 Det của ma trận là: 59.999996

C:\Users\Lenovo\Desktop\So Sanh\Guass_Drap_File\bin\Debug\netcoreapp3.1\Guass_Drap_File.exe (process 1840) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

- Kết quả trả về: $\begin{cases} p_1 = 1.0305114 \\ p_2 = 0.40000007 \\ p_3 = 1.3333329 \end{cases}^{(1)}$

- **Giải với phương pháp lặp Jacobi:**

- Nhập dữ liệu theo code:

- Dòng 1: nhập n (ma trận A kích cỡ n*n) và epsilon:

```
3 0.0000001
0.000001 0.000001 3 4
5 0.000001 0.000001 4
0.000001 4 0.000002 2
```

```
Ma tran cheo troi hang
Ta se thuc hien lap don ma tran X=Alpha*X+Beta, voi ma tran Alpha|Beta=
-0.0000000000 -0.0000002000 -0.0000002000 0.8000000119
-0.0000002500 -0.0000000000 -0.0000005000 0.5000000000
-0.0000003333 -0.0000003333 -0.0000000000 1.3333333731
Nghiem sau lan lap thu 1 la: 0.8000000119 0.5000000000 1.3333333731
Nghiem sau lan lap thu 2 la: 0.7999996543 0.4999991357 1.3333328962
Nghiem cua he phuong trinh la: 0.7999996543 0.4999991357 1.3333328962
```

- Kết quả trả về: $\begin{cases} p_1 = 0.7999996543 \\ p_2 = 0.4999991357 \\ p_3 = 1.3333328962 \end{cases}^{(2)}$

- **Giải bằng máy tính cầm tay CASIO fx-570VN PLUS:**

- Kết quả trả về: $\begin{cases} p_1 = 0.7999996333 \\ p_2 = 0.4999991333 \\ p_3 = 1.3333329 \end{cases}^{(3)}$

- **So sánh các kết quả:**

- Kết quả trả về: $\begin{cases} p_1 = 1.0305114 \\ p_2 = 0.40000007^{(1)} \\ p_3 = 1.3333329 \end{cases}$ (Gauss)

- Kết quả trả về: $\begin{cases} p_1 = 0.7999996543 \\ p_2 = 0.4999991357^{(2)} \\ p_3 = 1.3333328962 \end{cases}$ (Jacobi)

- Kết quả trả về: $\begin{cases} p_1 = 0.7999996333 \\ p_2 = 0.4999991333^{(3)} \\ p_3 = 1.3333329 \end{cases}$ (CASIO)

⇨ Sử dụng phương pháp Jacobi trong trường hợp này cho sai số ít hơn, đáp án đáng tin cậy hơn

- Sai số trong phương pháp lặp Jacobi nhiều hay ít là do việc ta lấy giá trị epsilon: epsilon nhỏ vừa đủ

⇨ Phương pháp lặp Jacobi kiểm soát sai số tốt hơn.

5. Kết luận

- Ưu điểm của phương pháp lặp Jacobi:
  - Nghiệm cần tìm có độ chính xác cao hơn đối với các ma trận có hệ số nhỏ (nguyên nhân là do thực hiện kiểm tra độ chính xác sau mỗi lần lặp).
  - Kết quả trả về đáng tin hơn so với các phương pháp khác (phương pháp giải đúng Gauss).
- Nhược điểm của phương pháp lặp Jacobi:
  - Chỉ có thể sử dụng phương pháp lặp Jacobi nếu ma trận là ma trận chéo trội (hàng/cột).