GIẢI TÍCH SỐ

Phương pháp lặp đơn - lặp Jacobi

Giảng viên hướng dẫn: Hà Thị Ngọc Yến

Nguyễn Văn Đại	20195847
Bùi Khương Duy	20195864
Nguyễn Gia Giang	20195867
Trương Đăng Thắng	20195916
Lê Văn Thẩm	20195914

Nhóm sinh viên: Nhóm 3 - lớp 125001 - học kỳ 20202

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

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC
School of Applied Mathematics and Informatics

Mục lục

I

Mở đầu

1 Giới thiệu chung

Giải tích số (Numerial Analysis) là một môn khoa học nghiên cứu cách giải gần đúng, chủ yếu là giải số, các phương trình, các bài toán xấp xỉ hàm số và các bài toán tối ưu.

Cho tới thế kỷ 21 hầu hết các lĩnh vực trong đời sống như: các ngành kỹ thuật, khoa học tự nhiên - xã hội, y học, tài chính - kinh doanh và thậm chí cả nghệ thuật, ... cũng đã áp dụng các yếu tố của tính toán khoa học. Sự phát triển về sức mạnh tính toán đã tạo ra một cuộc cách mạng trong việc sử dụng các mô hình toán học thực tế, trong khoa học và kỹ thuật, và giải tích số là lĩnh vực tinh tế để thực hiện các mô hình chi tiết này của thế giới. Ví dụ,các phương trình vi phân thông thường xuất hiện trong cơ học thiên thể (dự đoán chuyển động của các hành tinh, ngôi sao và thiên hà). Phương trình vi phân ngẫu nhiên và chuỗi Markov rất cần thiết trong việc mô phỏng tế bào sống cho y học và sinh học.

Giải tích số đặc biệt quan tâm đến các vấn đề: thời gian máy, bộ nhớ cần sử dụng, tốc độ hội tụ và sự ổn định của thuật toán. Cùng với các vấn đề: Xấp xỉ hàm số, giải gàn đúng các phương trình, hệ phương trình và giải gần đúng các bài toán tối ưu là ba nhiệm vụ chính vô cùng quan trọng của Giải tích số. Trong báo cáo này, chúng em sẽ trình bày một phương pháp được sử dụng để giải gần đúng nghiệm của hệ phương trình tuyến tính bậc nhất $Ax = b$, đó là phương pháp lặp đơn và lặp Jacobi.

Báo cáo được chuẩn bị trong thời gian ngắn nên chúng em không thể tránh được những sai sót ngoài ý muốn, rất mong được sự góp ý của cô và các bạn. Chúng em xin chân thành cảm ơn cô Hà Thị Ngọc Yến đã giảng dạy và hướng dẫn chúng em hoàn thành bài báo cáo này.

2 Giới thiệu đề tài

Vấn đề thực tế: Các phương pháp trực tiếp giải đúng hệ phương trình đại số tuyến tính thường phải thực hiện một số lượng tính toán khổng lồ. Khi thực hiện các phép tính, ta luôn phải thực hiện làm tròn số, tính toán với số gần đúng. Mà với hệ phương trình đại số tuyến tính, một chút thay đổi trong hệ số có thể dẫn tới nghiệm của hệ có sai số lớn khi tính ra nghiệm. Ví dụ ta xét hệ sau:

$$\begin{cases} 2x_1 & + & x_2 & = 2 \\ 2.01x_1 & + & x_2 & = 2 \end{cases} \Rightarrow \begin{cases} x_1 = 0 \\ x_2 = 2 \end{cases}$$

Tuy nhiên:

$$\begin{cases} 2x_1 + & x_2 & = 2 \\ 2.01x_1 + 1.01x_2 & & = 2 \end{cases} \Rightarrow \begin{cases} x_1 = 2 \\ x_2 = -2 \end{cases}$$

Điều này được gọi là sự bất ổn định của hệ phương trình đại số tuyến tính.
Bởi vậy, với những bài toán có kích thước lớn, ta phải dùng các phương pháp lặp và xấp xỉ thay vì dùng các phương pháp trực tiếp, phần nhỏ trong số đó là phương pháp lặp đơn và lặp Jacobi.

II

Cơ sở lý thuyết

1 Chuẩn vector

a) Định nghĩa chuẩn: Chuẩn là một ánh xạ $\|.\| : \mathbb{R}^n \to \mathbb{R}^+$ và thỏa mãn các tính chất như sau:

- $\|u\| \geq 0 \quad "=" \Leftrightarrow u = 0$
- $\|ku\| = |k|\|u\| \quad \forall k \in \mathbb{R}$
- $\|u + v\| \leq \|u\| + \|v\|$

b) Chuẩn vector

- $\|x\|_\infty = \max\limits_{i=\overline{1,n}}\{|x_i|\}$ <Chuẩn cực đại>
- $\|x\|_1 = \sum\limits_{i=1}^{n} |x_i|$ <Chuẩn tuyệt đối>
- $\|x\|_2 = \sqrt{\sum\limits_{i=1}^{n} x_i^2}$ <Chuẩn Euclid>

2 Sự hội tụ của dãy vector

- Ta nói dãy $x_n \xrightarrow{n\to\infty} x^* \Leftrightarrow \|x_n - x^*\| \xrightarrow{n\to\infty} 0 \Leftrightarrow x_{n_i} \xrightarrow{n\to\infty} x_i^* \;\; \forall i$

- Nếu $x_n \xrightarrow{n\to\infty} x^*$ theo một chuẩn nào đó thì $\Rightarrow x_n \xrightarrow{n\to\infty} x^*$ theo mọi chuẩn

  Chứng minh: Trong không gian định chuẩn hữu hạn chiều, mọi chuẩn đều tương đương.

  Ta chỉ cần chứng minh, mọi chuẩn đều tương đương với chuẩn Euclid sau:

  Đặt $\{e_i\}_{i=1}^n$ là cơ sở của X. Như vậy $\forall x \in X$: $x = \sum\limits_{i=1}^{n} x_i e_i, \quad \begin{cases} x_i \in R \;\; \forall i \\ e_i \in X \;\; \forall i \end{cases}$,

$$\Rightarrow ||x||_2 = \left(\sum_{i=1}^{n} x_i^2\right)^{\frac{1}{2}}$$

* Ta có $||x|| = \left|\left|\sum\limits_{i=1}^{n} x_i e_i\right|\right| \leq \sum\limits_{i=1}^{n} |x_i|.||e_i|| \leq \left(\sum\limits_{i=1}^{n} |x_i|^2\right)^{\frac{1}{2}} . \left(\sum\limits_{i=1}^{n} ||e_i||^2\right)^{\frac{1}{2}} = ||x||_2.C_2$

** Xét hàm số n biến $f(x_1, x_2, \ldots, x_n) = ||x|| = \left|\left|\sum\limits_{i=1}^{n} e_i x_i\right|\right|$, hàm số này liên tục theo các biến

$$\begin{aligned}|f(x_1,x_2,\dots,x_n)-f(y_1,y_2,\dots,y_n)| &= \left\|\sum_{i=1}^{n}(x_i-y_i)e_i\right\| \le \sum_{i=1}^{n}|x_i-y_i|.||e_i|| \\ &\le \left(\sum_{i=1}^{n}|x_i-y_i|^2\right)^{\frac{1}{2}}.\left(\sum_{i=1}^{n}||e_i||^2\right)^{\frac{1}{2}} \\ &= C_2.||x-y||_2 \to 0,\ \ (|x-y|\to 0)\end{aligned}$$

Ta xét $M = \{(x_1,x_2,\dots,x_n)\in R^n \ |\ x_1^2+x_2^2+\dots+x_n^2 = 1\}$ là tập đóng và bị chặn trong $R^n \Rightarrow M$ là tập compact.
Ta thấy $f : M \to R$ là hàm số liên tục trên tập compact
$\Rightarrow \exists C_1 = \min\limits_{M}\{f(x_1,x_2,\dots,x_n)\} \ge 0$
Nếu $C_1 = 0 \Rightarrow \exists (x_1,x_2,\dots,x_n)\in M$ sao cho $f(x_1,x_2,\dots,x_n) = 0$
Tức là $x_1e_1+x_2e_2+\dots+x_ne_n = 0$. Mà $(e_1,e_2,\dots,e_n)$ là cơ sở của $X \Rightarrow$ nó độc lập tuyến tính
$\Rightarrow x_1 = x_2 = \dots = x_n = 0 \Rightarrow$ mâu thuẫn vì không thuộc tập $M \Rightarrow C_1 > 0$

Lấy $\begin{cases} x\neq 0 \\ x = x_1e_1+x_2e_2+\dots+x_ne_n\end{cases}$

ta đặt $x_i' = \dfrac{x_i}{\sqrt{x_1^2+x_2^2+\dots+x_n^2}}$ thì $x' = (x_1',x_2',\dots,x_n') \in\ M$

$$\begin{aligned}\Rightarrow C_1 \le f(x_1',x_2',\dots,x_n') &= ||x_1'e_1+x_2'e_2+\dots+x_n'e_n|| \\ &= \frac{1}{\sqrt{x_1^2+x_2^2+\dots+x_n^2}}.||x_1e_1+x_2e_2+\dots+x_ne_n||\end{aligned}$$

$\Rightarrow C_1\sqrt{x_1^2+x_2^2+\dots+x_n^2} \le ||x||$

$\left.\begin{array}{l} C_1||x||_2 \le ||x|| \\ \|x\| \le C_2\|x\|_2\end{array}\right\} \Rightarrow C_1\|x\|_2 \le \|x\| \le C_2\|x\|_2$

$\Rightarrow$ Mọi chuẩn đề tương đương với chuẩn Euclid trong không gian hữu hạn chiều.

3 Chuẩn và sự hội tụ của ma trận

- Chuẩn theo hàng: $\|A\|_\infty = \max\limits_{i=\overline{1,n}}\sum\limits_{j=1}^{n}|a_{ij}|$
- Chuẩn theo cột: $\|A\|_1 = \max\limits_{j=\overline{1,n}}\sum\limits_{i=1}^{n}|a_{ij}|$
- Chuẩn Euclid: $\|A\|_2 = \left[\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}a_{ij}^2\right]^{\frac{1}{2}}$
- Chuẩn theo trị riêng: $\|A\|_2 = \sqrt{\lambda_{max}(\alpha^T\alpha)}$

Dãy ma trận $\{A^{(k)}\} = \{a_{ij}^{(k)}\}$ được gọi là hội tụ đến ma trận $\{A\} = a_{ij}$ khi $k \to \infty$ nếu:

$$\lim_{k\to\infty} a_{ij}^{(k)} = a_{ij} \;\; \forall i, j = \overline{1, n} \quad \text{hoặc} \quad \lim_{k\to\infty} \|A^{(k)} - A\| = 0$$

Ví dụ: Xét ma trận

$$A = \begin{bmatrix} 1 & 2 & 1 \\ -2 & 3 & -2 \\ -1 & 4 & 3 \end{bmatrix} \Rightarrow \begin{cases} \|A\|_\infty & = & 8 \\ \|A\|_1 & = & 9 \\ \|A\|_2 & = & 7 \end{cases}$$

III Phương pháp lặp đơn

1 Ý tưởng phương pháp

Tương tự như phương pháp lặp đơn để giải lớp phương trình $f(x) = 0$ ta cũng có thể mở rộng ra các không gian khác có cấu trúc phức tạp hơn như $\mathbb{R}^n$
Xét nhóm phương trình tuyến tính: $Ax = b$
Ta sẽ đưa phương trình trên về dạng: $x = \alpha x + \beta$ điển hình là cách biến đổi sau:

C1: $Ax = b \Leftrightarrow x = Ax + x - b \Leftrightarrow x = (A+I)x - b$

Đặt $\begin{cases} \alpha &= A + I \\ \beta &= -b \end{cases} \Rightarrow (1) \Leftrightarrow x = \alpha x + \beta$

C2: $Ax = b \Leftrightarrow x = -Ax + x + b \Leftrightarrow x = (I - A)x + b$

Đặt $\begin{cases} \alpha = I - A \\ \beta = b \end{cases} \Rightarrow x = \alpha x + \beta$

Từ đó ta xây dựng được dãy lặp cho phương pháp: $x_n = \alpha x_{n-1} + \beta$

2 Sự hội tụ của phương pháp và tính duy nhất nghiệm

Từ dãy lặp đã được xây dựng ở trên, ta có những vấn đề cơ bản sau cần được giải quyết:

$$\text{Nếu } x^* \text{ là nghiệm của hệ thì } \Rightarrow \begin{cases} x^{(n)} \xrightarrow{n\to\infty} x^* \\ x^* \text{ là duy nhất} \end{cases}$$

2.1 Chứng minh sự hội tụ của phương pháp

Nguyên lý ánh xạ co

Nếu $\|\alpha\| \leq q < 1 \Rightarrow x_n \xrightarrow{n\to\infty} x^*$ (với x^* là nghiệm của hệ)

Chứng minh: Từ $x^{(n)} = \alpha x^{(n-1)} + \beta \Leftrightarrow x_{(n)} = f(x_{(n-1)})$

$$\Rightarrow \forall x^{(n)}, y^{(n)} \text{ thì } \|f(x^{(n)}) - f(y^{(n)})\| = \|\alpha x^{(n)} + \beta - (\alpha y^{(n)} + \beta)\| = \|\alpha(x^{(n)} - y^{(n)})\|$$
$$\Rightarrow \|f(x^{(n)}) - f(y^{(n)})\| \leq \|\alpha\|.\|x^{(n)} - y^{(n)}\| \leq q.\|x^{(n)} - y^{(n)}\|$$

Ta xét:

$$\|x^{(n+p)} - x^{(n)}\| = \|f(x^{(n+p-1)}) - f(x^{(n-1)})\|$$
$$\Rightarrow \|x^{(n+p)} - x^{(n)}\| \leq q.\|x^{(n+p-1)} - x^{(n-1)}\| \leq \cdots \leq q^n\|x^{(p)} - x^{(0)}\|$$

Vì $q < 1 \Rightarrow q^n \xrightarrow{n\to\infty} 0 \Rightarrow \|x^{(n+p)} - x^{(n)}\| \xrightarrow{n\to\infty} 0$
$\Rightarrow \{x^{(n)}\}$ là dãy Cauchy $\Rightarrow x^{(n)} \xrightarrow{n\to\infty} x^*$
(Do $\mathbb{R}^n$ là không gian định chuẩn đủ, mọi dãy Cauchy trong $\mathbb{R}^n$ đều hội tụ)

2.2 Chứng minh sự duy nhất nghiệm

Sự duy nhất của x^*

Khi $x^{(n)} \xrightarrow{n\to\infty} x^*$ thì x^* là nghiệm duy nhất của phương trình $Ax = b$

Ta có $x^{(n)} = \alpha x^{(n-1)} + \beta \Rightarrow n \to \infty$ thì $\begin{cases} x^{(n)} \to x^* \\ x^{(n-1)} \to x^* \end{cases} \Rightarrow x^* = \alpha x^* + \beta$

C/m tính duy nhất: Giả sử $\begin{cases} x^* & = \alpha x^* + \beta \Leftrightarrow x^* = f(x^*) \\ y^* & = \alpha y^* + \beta \Leftrightarrow y^* = f(y^*) \end{cases}$

$$\Rightarrow \|x^* - y^*\| = \|f(x^*) - f(y^*)\| \leq q.\|x^* - y^*\|$$
$$\Leftrightarrow (1-q)\|x^* - y^*\| \leq 0 \Rightarrow \|x^* - y^*\| = 0 \Rightarrow x^* = y^*$$

Như vậy: Nghiệm x^* là duy nhất
$\Rightarrow$ *Ta có điều phải chứng minh*

3 Sai số

Trong phương pháp lặp , muốn có nghiệm đúng ta phân tích theo công thức $x^{(k)} = \alpha x^{(k-1)} + \beta$ với $k \to \infty$. Tuy nhiên điều đó là không thể thực hiện được nên ta phải dừng ở bước thứ k, khi mà $x^{(k)} \sim x^*$.
$\Rightarrow$ Ta cần phải thiết lập công thức sai số cho phương pháp lặp.

3.1 Thiết lập công thức sai số hậu nghiệm

Ta xét:

$$x^{(k+1)} = \alpha x^{(k)} + \beta$$
$$x^{(k)} = \alpha x^{(k-1)} + \beta$$

Lấy $x^{(k+1)} - x^{(k)}$:

$$\Rightarrow x^{(k+1)} - x^{(k)} = \alpha(x^{(k)} - x^{(k-1)})$$
$$\Rightarrow \|x^{(k+1)} - x^{(k)}\| \leq \|\alpha\|.\|x^{(k)} - x^{(k-1)}\|$$

Mặt khác

$$\begin{aligned} x^{(k+m)} - x^{(k)} &= (x^{(k+m)} - x^{(k+m-1)}) + (x^{(k+m-1)} - x^{(k+m-2)}) + \cdots + (x^{(k+1)} - x^{(k)}) \\ \|x^{(k+m)} - x^{(k)}\| &= \|x^{(k+m)} - x^{(k+m-1)}\| + \|x^{(k+m-1)} - x^{(k+m-2)}\| + \cdots + \|x^{(k+1)} - x^{(k)}\| \\ &\leq (1 + \|\alpha\| + \cdots + \|\alpha\|^{m-1}).\|x^{(k+1)} - x^{(k)}\| \\ &\leq \sum_{i=o}^{m-1} \|\alpha\|^i \|x^{(k+1)} - x^{(k)}\| = \frac{1 - \|\alpha\|^m}{1 - \|\alpha\|}\|x^{(k+1)} - x^{(k)}\| \end{aligned}$$

$\Rightarrow \quad \|x^{(k+m)} - x^{(k)}\| \leq \dfrac{1 - \|\alpha\|^m}{1 - \|\alpha\|}\|x^{(k+1)} - x^{(k)}\|$

Cố định k ta cho $m \to \infty$. Vì $\|\alpha\| < 1$ nên $\|\alpha\|^m \xrightarrow{m\to\infty} 0$ ta có:

$$\|x^* - x^{(k)}\| \leq \frac{1}{1 - \|\alpha\|} \|x^{(k+1)} - x^{(k)}\|$$

Hay

$$\boxed{\left\|x^* - x^{(k)}\right\| \leq \frac{\|\alpha\|}{1 - \|\alpha\|} \left\|x^{(k)} - x^{(k-1)}\right\| < \epsilon} \tag{1}$$

Công thức sai số hậu nghiệm (1) thuận lợi cho việc tính toán trên máy tính.

3.2 Thiết lập công thức sai số tiên nghiệm

Khai triển tiếp tục công thức (1) ta được:

$$\begin{aligned}
\|x^* - x^{(k)}\| &\leq \frac{\|\alpha\|}{1 - \|\alpha\|}.\|\alpha\|.\|x^{(k-1)} - x^{(k-2)}\| \\
&\leq \frac{\|\alpha\|^2}{1 - \|\alpha\|}.\|\alpha\|.\|x^{(k-2)} - x^{(k-3)}\| \\
&\leq \ldots \leq \frac{\|\alpha\|^k}{1 - \|\alpha\|}\|x^{(1)} - x^{(0)}\|
\end{aligned}$$

Vậy

$$\boxed{\left\|x^* - x^{(k)}\right\| \leq \frac{\|\alpha\|^k}{1 - \|\alpha\|} \left\|x^{(1)} - x^{(0)}\right\| < \epsilon} \tag{2}$$

Công thức sai số tiên nghiệm (2) thuận lợi khi tìm số lần lặp k cần thiết để đạt được sai số mong muốn.
Với $\epsilon > 0$, đặt $q = \|\alpha\|$ ta có:

$$\left\|x^{(k)} - x^*\right\| < \epsilon \Rightarrow k \geq \log_q \frac{(1-q)\epsilon}{\|x^{(1)} - x^{(0)}\|} \tag{3}$$

Công thức (3) này sẽ được dùng để tính số k lần lặp sử dụng trong thuật toán của phương pháp.

4 Thuật toán

4.1 Thuật toán bằng mã giả

Trên thực tế ta đang xét hệ phương trình $Ax = b$ nên ta cần một hàm thủ tục để biến đổi hệ từ $Ax = b \Rightarrow x = \alpha x + \beta$ như đã trình bày ở trên:

```
FUNTION: BOOLE CHECKLOOP():
    STEP 1: nhan vao A, b
    STEP 2: alpha <-- I + A
            beta  <-- -b
            k <-- 0
    STEP 3: if ||alpha|| < 1
                return true
    STEP 4: alpha <-- I - A
            beta  <-- b
    STEP 5: if ||alpha|| < 1
                return true
            else
                output("SINGLE LOOP ERROR")
                return false
```

Đến đây nếu hàm CHECKLOOP() trả về **true** thì hệ $x = \alpha x + \beta$ đã thỏa mãn điều kiện lặp đơn $\|q\| = \|\alpha\| < 1$:

Lặp đơn với công thức sai số hậu nghiệm (1):

```
FUNTION: SINGLOOP1():
    STEP 1: input A, b, x0, epsilon
    STEP 2: if CHECKLOOP() == true:
                q <-- ||alpha||
                k <-- 0
                while(!(q.||x-x0|| < epsilon*(1-q)))
                    x  <-- alpha*x0 + beta
                    x0 <-- x
            else
                end().
    STEP 3: output(x,k)
    STEP 4: end().
```

Lặp đơn với công thức sai số tiên nghiệm (2):
(Ở đây ta sẽ dùng đến công thức (3) xây dựng ở trên)

```
FUNTION: SINGLOOP2():
    STEP 1: input alpha, beta, x0, epsilon
    STEP 2: if CHECKLOOP() == true
                q <-- ||alpha||
                x <-- alpha*x0 + beta
                J <-- log_{q}[(1-q)*epsilon]/||x-x0||
                k <-- Ceil(J)
                for:i from 2 to k
                    x  <-- alpha*x0 + beta
                    x0 <-- x
```

```
            else
                end().
    STEP 3: output(x, k)
    STEP 4: end().
```

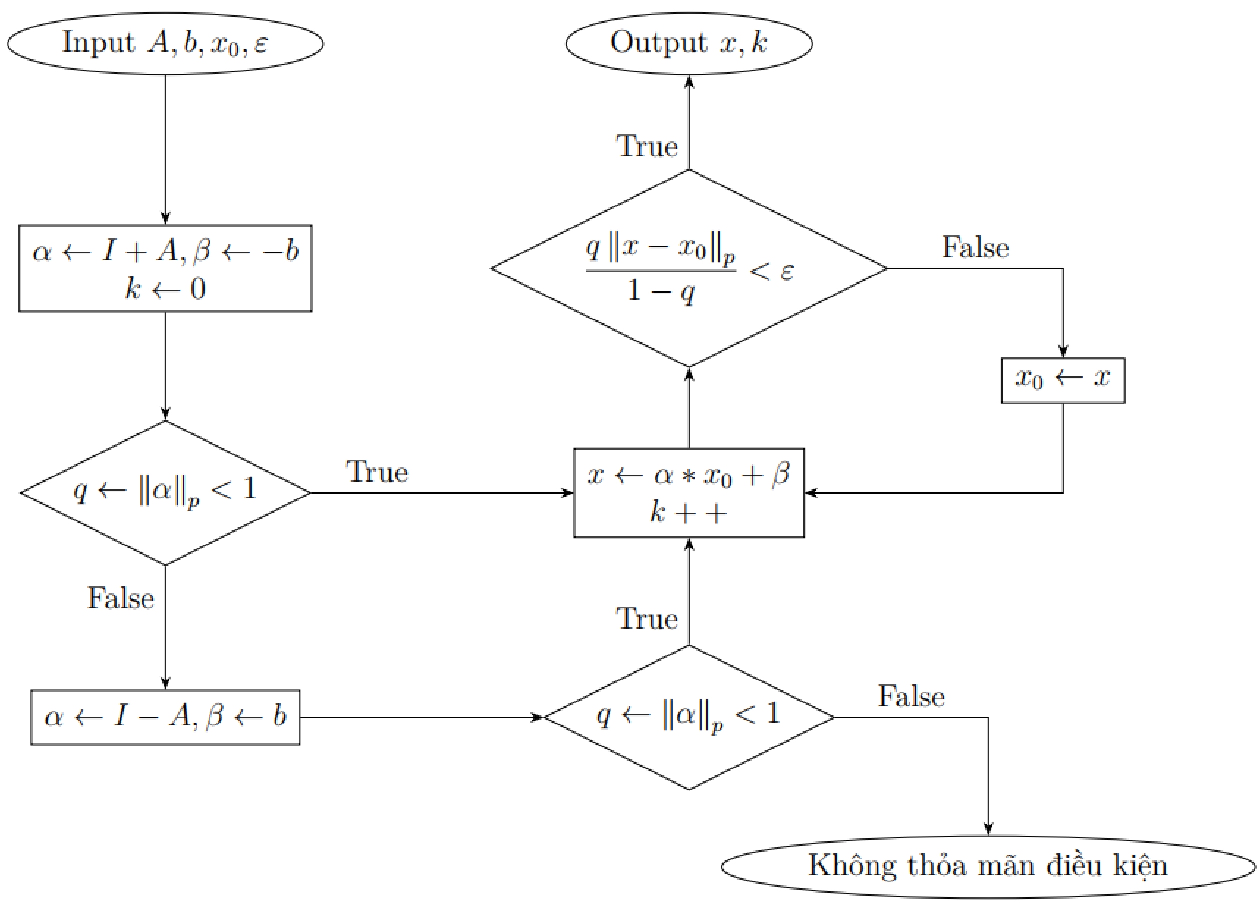
Kết hợp các hàm trên lại ta có mã giả của phương pháp lặp đơn như sau:

```
#SINGLE LOOP METHOD:
#Bien doi ma tran va kiem tra chuan alpha
FUNTION: BOOLE CHECKLOOP():
    STEP 1: nhan vap A, b
    STEP 2: alpha <-- I + A
            beta  <-- -b
            k <-- 0
    STEP 3: if ||alpha|| < 1
                return true
    STEP 4: alpha <-- I - A
            beta  <-- b
    STEP 5: if ||alpha|| < 1
                return true
            else
                output("SINGLE LOOP ERROR")
                return false

#Cong thuc hau nghiem
FUNTION: SINGLOOP1():
    STEP 1: input A, b, x0, epsilon
    STEP 2: if CHECKLOOP() == true:
                q <-- ||alpha||
                k <-- 0
                while(!(q.||x-x0|| < epsilon*(1-q)))
                    x  <-- alpha*x0 + beta
                    x0 <-- x
            else
                end().
    STEP 3: output(x,k)
    STEP 4: end().

#Cong thuc tien nghiem
FUNTION: SINGLOOP2():
    STEP 1: input alpha, beta, x0, epsilon
    STEP 2: if CHECKLOOP() == true
                q <-- ||alpha||
                x <-- alpha*x0 + beta
                J <-- log_{q}[(1-q)*epsilon]/||x-x0||
                k <-- Ceil(J)
                for:i from 2 to k
```

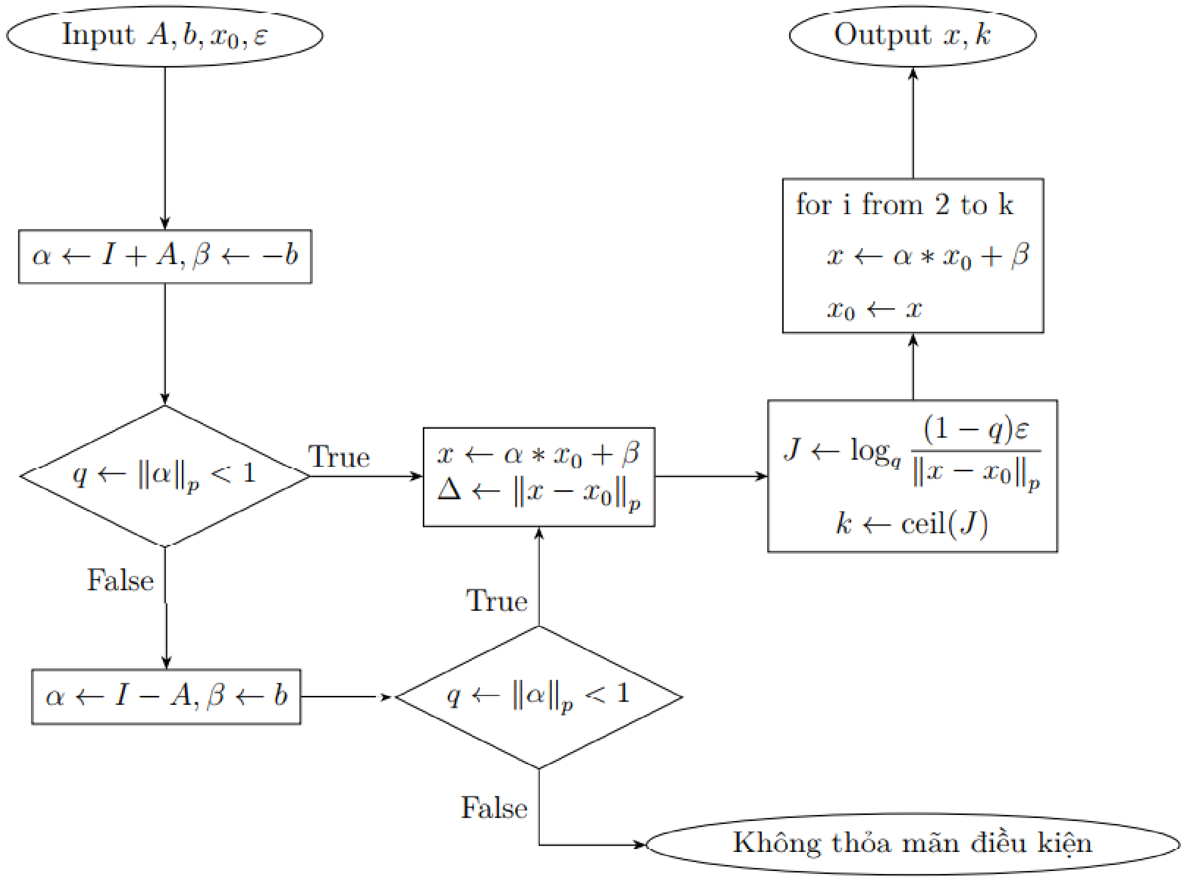
```
                    x   <-- alpha*x0 + beta
                    x0  <-- x
            else
                end().
    STEP 3: output(x, k)
    STEP 4: end().

#HAM MAIN():
FUNTION: MAIN()
    STEP 1: print("Chon loai cong thuc")
            print("1. Hau nghiem")
            print("2. Tien nghiem")
    STEP 2: input(type)
    STEP 3: if type == 1:
                SINGLELOOP1()
            else if type == 2
                SINGLELOOP2()
    STEP 4: end.
```

4.2 Thuật toán bằng sơ đồ khối

Sơ đồ khối cho sai số hậu nghiệm:



Sơ đồ khối cho công thức tiên nghiệm:



4.3 Một cách khác để tìm được xấp xỉ nghiệm $x^{(n)}$

Xuất phát từ dãy lặp với xấp xỉ đầu:

$$\begin{aligned}
x^{(1)} &= \alpha x^{(0)} + \beta \\
x^{(2)} &= \alpha x^{(1)} + \beta = \alpha(\alpha x^{(0)} + \beta) + \beta \\
&= \alpha^2 x^{(0)} + (\alpha + I)\beta \\
\ldots &= \ldots \\
x^{(n)} &= \alpha^n x^{(0)} + (\alpha^{n-1} + \alpha^{n-2} + \cdots + I)\beta
\end{aligned}$$

Ta có:

$$\begin{bmatrix} \alpha & I \\ O & I \end{bmatrix}^n = \begin{bmatrix} \alpha^n & \alpha^{n-1} + \alpha^{n-2} + \cdots + I \\ O & I \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \alpha & I \\ O & I \end{bmatrix}^n \cdot \begin{bmatrix} x^{(0)} \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha^n x^{(0)} + (\alpha^{n-1} + \alpha^{n-2} + \cdots + I)\beta \\ \beta \end{bmatrix}$$

Dễ thấy vế trái (VT) có xấp xỉ ban đầu $x^{(0)}$ và vế phải (VP) có chứa xấp xỉ nghiệm $x^{(n)}$ cần tìm.

IV Phương pháp lặp Jacobi

1 Ma trận chéo trội và ý tưởng phương pháp

1.1 Ma trận chéo trội

Định nghĩa chéo trội:
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)

Chéo trội hàng	Chéo trội cột
$\begin{bmatrix} 1 & 0.5 & 0.4 \\ 0,2 & 1.5 & 0.8 \\ 1 & 0.6 & 2 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 0.4 \\ 0.1 & 3.3 & 0.2 \\ 0.5 & 0.3 & 0.9 \end{bmatrix}$

1.2 Ma trận chéo trội không suy biến

Trường hợp trội hàng

$$A \in M_n(\mathbb{K}) \text{ có: } |a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad i = \overline{1,n}$$

Chứng minh rằng: A không suy biến

Chứng minh: Giả sử A không khả nghịch

$$\Rightarrow \text{Hệ thuần nhất: } \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = 0 \\ a_{21}x_2 + a_{22}x_2 + \cdots + a_{2n}x_n = 0 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = 0 \end{cases} \text{ có vô số nghiệm}$$

Trong đó có nghiệm không tầm thường, giả sư nghiệm đó là:

$$X_0(x_1^0, x_2^0, \dots, x_n^0) \text{ và đặt } |x_i^0| = max\{|x_1^0|; |x_2^0|; \dots; |x_n^0|\}$$

Xét phương trình thứ i của hệ trên ta có:

$$\sum_{j=1}^{n} a_{ij}x_j^0 = 0$$

$$\Leftrightarrow \quad a_{ii}x_i^0 = -\sum_{j=1, j\neq i}^{n} a_{ij}x_i^0$$

$$\Rightarrow \quad |a_{ii}x_i^0| \leq \sum_{j=1, j\neq i}^{n} |a_{ij}||x_j^0|$$

$$\Leftrightarrow \quad |a_{ii}| \leq \sum_{j=1, j\neq i}^{n} |a_{ij}| \text{ (Vô lý)}$$

Do đó A khả nghịch $\Rightarrow \det A \neq 0$

Tương tự với ma trận chéo trội cột ta đưa về A^T rồi xử lý tương tự như trên.

1.3 Ý tưởng phương pháp

Phương pháp lặp Jacobi = Phương pháp lặp đơn cho $Ax = b$ với A là ma trận chéo trội:

$$Ax = b \Rightarrow x = Bx + d, \quad \|B\|_{(\infty,1,2)} = q < 1 \rightarrow \text{ lặp đơn}$$

2 Ma trận chéo trội hàng

2.1 Định nghĩa và biến đổi ma trận

Định nghĩa:

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad i = \overline{1,n}$$

- Với giả thiết ma trận A có tính chéo trội khi đó $a_{ii} \neq 0$ ta xét ma trận bổ xung của hệ phương trình như sau:

$$[A|b] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array}\right] \xrightarrow{\frac{r_i}{a_{ii}} \to r_i} \left[\begin{array}{cccc|c} 1 & \frac{a_{12}}{a_{11}} & \dots & \frac{a_{1n}}{a_{11}} & \frac{b_1}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 1 & \dots & \frac{a_{2n}}{a_{22}} & \frac{b_2}{a_{22}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \dots & 1 & \frac{b_n}{a_{nn}} \end{array}\right]$$

Hệ được viết lại thành: $A'x = d$ với

$$A' = \begin{bmatrix} 1 & \frac{a_{12}}{a_{11}} & \dots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 1 & \dots & \frac{a_{2n}}{a_{22}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \dots & 1 \end{bmatrix} \qquad \text{và} \qquad d = \begin{bmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{bmatrix}$$

$$\Rightarrow x = Bx + d \text{ với } B = I - A' = \begin{bmatrix} 0 & \frac{-a_{12}}{a_{11}} & \dots & \frac{-a_{1n}}{a_{11}} \\ \frac{-a_{21}}{a_{22}} & 0 & \dots & \frac{-a_{2n}}{a_{22}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{-a_{n1}}{a_{nn}} & \frac{-a_{n2}}{a_{nn}} & \dots & 0 \end{bmatrix}$$

Như vậy ta luôn có:

$$\|B\|_\infty = \max_{i=\overline{1,n}} \left\{ \sum_{j=1, j\neq i}^{m} \left| \frac{a_{ij}}{a_{ii}} \right| \right\} < 1$$

Ta cùng lấy một ví dụ sau để có cái nhìn rõ hơn về ma trận chéo trội hàng:

Ví dụ

$$\begin{bmatrix} 15 & -5 & 2 \\ 10 & 35 & -18 \\ 8 & 4 & 15 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 12 \\ 5 \end{bmatrix}$$

Ta có:

$$[A|b] = \left[\begin{array}{ccc|c} 15 & -5 & 2 & 7 \\ 10 & 35 & -18 & 12 \\ 8 & 4 & 15 & 5 \end{array}\right] \Rightarrow [A'|d] = \left[\begin{array}{ccc|c} 1 & \frac{-5}{15} & \frac{2}{15} & \frac{7}{15} \\ \frac{10}{35} & 1 & \frac{-18}{35} & \frac{12}{15} \\ \frac{8}{15} & \frac{4}{15} & 1 & \frac{5}{15} \end{array}\right]$$

$$\Rightarrow x = (I - A')x + d = \begin{bmatrix} 0 & \frac{5}{15} & \frac{-2}{15} \\ \frac{-10}{35} & 0 & \frac{18}{35} \\ \frac{-8}{15} & \frac{4}{15} & 0 \end{bmatrix} x + \begin{bmatrix} \frac{7}{15} \\ \frac{12}{15} \\ \frac{5}{15} \end{bmatrix}$$

$$\Rightarrow q = \|B\|_\infty = \max\left\{\frac{7}{15}; \frac{28}{35}; \frac{12}{15}\right\} = 0.8 < 1$$

- Với ý tưởng như trên ta có thể thực hiện đơn giản hơn như sau:

  Đặt $T = diag\left(\frac{1}{a_{11}}; \frac{1}{a_{22}}; \dots; \frac{1}{a_{nn}}\right)$ ta có:

  $$Ax = b \Leftrightarrow TAx = Tb \Leftrightarrow x = (I - TA)x + Tb$$

  Với $B = I - TA,\ \ d = Tb$ ta đã xây dựng được dãy lặp:

  $$x^{(n)} = Bx^{(n-1)} + d$$

2.2 Sai số

Ta thấy đây là dãy lặp trực tiếp cho x nên các công thức sai số áp dụng trực tiếp cho dãy lặp $x^{(n)}$, với các công thức hậu nghiệm (1) và công thức tiên nghiệm (2) tuy nhiên lúc này:

$$\|\alpha\|_p = \|B\|_p = \|I - TA\|_p$$

3 Ma trận chéo trội cột

3.1 Định nghĩa và biến đổi ma trận

Định nghĩa:

$$|a_{jj}| > \sum_{i=1, i\neq j}^{n} |a_{ij}| \quad j = \overline{1, n}$$

- Ta tách $Ax = b$ thành tổ hợp của các cột như sau:

  $$x_1[a_{i1}] + x_2[a_{i2}] + \dots + x_n[a_{in}] = b$$

  $$\Leftrightarrow a_{11}x_1\frac{1}{a_{11}}[a_{i1}] + a_{22}x_2\frac{1}{a_{22}}[a_{i2}] + \dots + a_{nn}x_n\frac{1}{a_{nn}}[a_{in}] = b$$

  Đặt ẩn phụ: $\begin{cases} y_1 = a_{11}x_1 \\ y_2 = a_{22}x_2 \\ \dots \\ y_n = a_{nn}x_n \end{cases} \Longrightarrow \begin{cases} y_1 + \frac{a_{12}}{a_{22}}y_2 + \dots + \frac{a_{1n}}{a_{nn}}y_n = b_1 \\ \frac{a_{21}}{a_{11}}y_1 + y_2 + \dots + \frac{a_{2n}}{a_{nn}}y_n = b_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \frac{a_{n1}}{a_{11}}y_1 + \frac{a_{n2}}{a_{22}}y_2 + \dots + y_n = b_n \end{cases}$

Như vậy ta có thể viết lại hệ dưới dạng: $y = B_1 y + b$

$$B_1 = \begin{bmatrix} 0 & \frac{-a_{12}}{a_{22}} & \cdots & \frac{-a_{1n}}{a_{nn}} \\ \frac{-a_{21}}{a_{11}} & 0 & \cdots & \frac{-a_{2n}}{a_{nn}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{-a_{n1}}{a_{11}} & \frac{-a_{n2}}{a_{22}} & \cdots & 0 \end{bmatrix} \qquad \text{và} \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{bmatrix}$$

Dễ dàng nhận thấy: $\|B_1\|_1 = \max\limits_{j=\overline{1,n}} \left\{ \sum\limits_{i=1,i\neq j}^{n} \left| \frac{a_{ij}}{a_{jj}} \right| \right\} < 1$

Ví dụ

$$\begin{bmatrix} 10 & 8 & 7 \\ 3 & -15 & 14 \\ -5 & 2 & 24 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ 13 \end{bmatrix}$$

Đặt $\begin{cases} y_1 = 10x_1 \\ y_2 = -15x_2 \\ y_3 = 24x_3 \end{cases}$ hay $\quad y = diag(10; -15; 24)x \Leftrightarrow x = Ty$

$$\Rightarrow y = \begin{bmatrix} 0 & \frac{8}{15} & \frac{-7}{14} \\ \frac{-3}{10} & 0 & \frac{-14}{24} \\ \frac{5}{10} & \frac{2}{15} & 0 \end{bmatrix} y + \begin{bmatrix} 2 \\ 6 \\ 13 \end{bmatrix}$$

$\Rightarrow \|B_1\|_1 = \max\{0.8; 0.67; 0.875\} = 0.875 < 1$

- Cùng một ý tưởng như trên ta có cách thể hiện đơn giản như sau:

$T = diag\left(\frac{1}{a_{11}}; \frac{1}{a_{22}}; \ldots; \frac{1}{a_{nn}}\right) \Rightarrow \begin{cases} D = T^{-1} = diag(a_{11}; a_{22}; \ldots; a_{nn}) \\ x = Ty \end{cases}$

$$Ax = b \Leftrightarrow ATy = b \Leftrightarrow y = (I - AT)y + b = B_1 y + b$$

Với xấp xỉ đầu $y^{(0)} \in \mathbb{R}^n$ ta xây dựng được dãy lặp:

$$\boxed{y^{(n+1)} = B_1 y^{(n)} + b}$$

3.2 Phương pháp xấp xỉ nghiệm đầu và sai số

- **Hai cách xấp xỉ nghiệm đầu:**

C1: xấp xỉ cho biến trung gian y_0 các công thức sai số chỉ áp dụng được cho y

Ta có:

$$y = y^{(0)} \mapsto x^{(0)} = Ty^{(0)}$$
$$y^{(1)} = B_1 y^{(0)} + d \mapsto x^{(1)} = Ty^{(1)}$$
$$\Rightarrow \|x^{(1)} - x^*\| \leq \|T\|.\|y^{(1)} - y^*\| \leq \|T\|\frac{q}{1-q}\|y^{(1)} - y^{(0)}\|$$
$$y^{(2)} = B_1 y^{(1)} + d \mapsto x^{(2)} = Ty^{(2)} \Rightarrow \|x^{(2)} - x*\| \leq \|T\|\frac{q}{1-q}\|y^{(2)} - y^{(1)}\|$$

C2: Xây dựng công thức lặp trực tiếp x và xây dựng sai số trực tiếp cho x mà không cần qua y

Xuất phát từ dãy lặp:

$$y^{(n)} = B_1 y^{(n-1)} + b$$
$$\Leftrightarrow Ty^{(n)} = TB_1 y^{(n-1)} + Tb$$
$$\Leftrightarrow Ty^{(n)} = T(I - AT)T^{-1}Ty^{(n-1)} + Tb$$
$$\Leftrightarrow x^{(n)} = (I - TA)x^{(n-1)} + Tb$$
$$\Leftrightarrow x^{(n)} = Bx^{(n-1)} + Tb$$

Vậy ta có: $x^{(n)} = Bx^{(n-1)} + d$ với $\begin{cases} B = I - TA \\ d = Tb \\ q = \|B_1\|_1 \end{cases}$

- **Thiết lập công thức sai số:** Xuất phát từ:

$$\begin{aligned}
\|x^{(n)} - x^*\| = \|Ty^{(n)} - Ty^*\| &\leq \|T\|.\|y^{(n)} - y^*\| \\
&\leq \|T\|\frac{q}{1-q}\|y^{(n)} - y^{(n-1)}\| \\
&= \|T\|\frac{q}{1-q}\|T^{-1}x^{(n)} - T^{-1}x^{(n-1)}\| \\
&\leq \|T\|.\|T^{-1}\|.\frac{q}{1-q}.\|x^{(n)} - x^{(n-1)}\| \\
&\leq \frac{\max|a_{ii}|}{\min|a_{ii}|}.\frac{q}{1-q}\|x^{(n)} - x^{(n-1)}\|
\end{aligned}$$

Tương tự với khai triển của phương pháp lặp đơn ta có hai công thức sai số với $\lambda = \dfrac{\max|a_{ii}|}{\min|a_{ii}|}$

Công thức hậu nghiệm:

$$\boxed{\|x^{(n)} - x^*\| \leq \frac{\lambda q}{1-q}\|x^{(n)} - x^{(n-1)}\|}$$

Công thức tiên nghiệm:

$$\boxed{\|x^{(n)} - x^*\| \leq \frac{\lambda q^n}{1-q}\|x^{(1)} - x^{(0)}\|}$$

4 Thuật toán

4.1 Thuật toán bằng mã giả

Trước tiên ta sẽ xây dựng hai hàm thủ tục để kiểm tra điều kiện chéo trội với độ ưu tiên trước là trường hợp chéo trội hàng, cả hai hàm đều nhận đầu vào là hệ $Ax = b$

Hàm kiểm tra tính chéo trội hàng:

```
#check if row diagonal diminance
Funtion:
checkrow(matrix):
    check <-- true
    for i from 1 to len(matrix):
      max <-- matrix[i][i]
      for j from 1 to len(matrix):
         if i == j: continue
         max <-- max - abs(matrix[i][j])
      if max <= 0:
         check <-- false
    return check
```

Hàm kiểm tra tính chéo trội cột:

```
#check if column diagonal diminance
Funtion:
checkcol(matrix):
    check <-- true
    for i from 1 to len(matrix):
        max <-- abs(matrix[i][i])
        for j from 1 to len(matrix):
            if i == j: continue
            max <-- max - abs(matrix[j][i])
        if max <= 0:
            check <-- false
    return check
```

Sau đó sẽ thực hiện biến đổi để đưa về dạng lặp đơn $x = \alpha x + \beta$ như trên đã trình bày:

Hàm biến đổi Jacobi:

```
funtion: change(matrix)
    STEP 1: if checkrow():
                goto STEP 2
            else if checkcol():
                    goto STEP 3
                 else:
                    end().
    STEP 2: for i from 1 to len(matrix):
                for j from 1 to len(matrix)
                    a[i][j] <-- a[i][j]/a[i][i]
                    b[i] <-- b[i]/a[i][i]
            alpla <-- I-A
            beta  <-- b
            end().
    STEP 3: for j from 1 to len(matrix)
                for i from 1 to n
                    if i = j: T[i][i] <-- 1/a[i][i]
                    else:  T[i][j] <-- 0
            alpha <-- I-TA
             beta <-- Tb
                q  <-- ||I-AT||
           lambda <-- max|a[i][i]|/min|a[i][i]|
           end().
```

Sau khi kết thúc thủ tục chuyển đổi **change()** thì hệ đã trở thành dạng $x = \alpha x + \beta$ và ta tiến hành lặp đơn để tìm $x^{(n)}$.

4.2 Thuật toán đưa ma trận trội không trên đường chéo về chéo trội

Trong thực tế ta cũng rất hay gặp phải các phương trình có thể lặp Jacobi được nhưng các hàng, các cột sắp xếp chưa hợp lí để lặp Jacobi hay gọi là trường hợp ma trận trội theo hàng, cột nhưng sắp xếp không trên đường chéo chính:

Ví dụ trội không đúng đường chéo

$$\begin{bmatrix} 3 & 7.2 & 19.5 & 2.4 \\ 4 & 15.7 & -2.5 & -6 \\ 1.9 & 12.4 & -3.7 & 36.6 \\ 8 & 1.4 & -2.5 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Như vậy để mở rộng phạm vi bài toán ta sẽ xây dựng thuật toán để căn chỉnh lớp ma trận trên về ma trận chéo trội.

Thuật toán sắp xếp lại ma trận trội hàng:

```
#INPUT: matrix
FOR i from 1 to len(matrix):
    a[i] = address_maxrow(matrix(row_i))
```

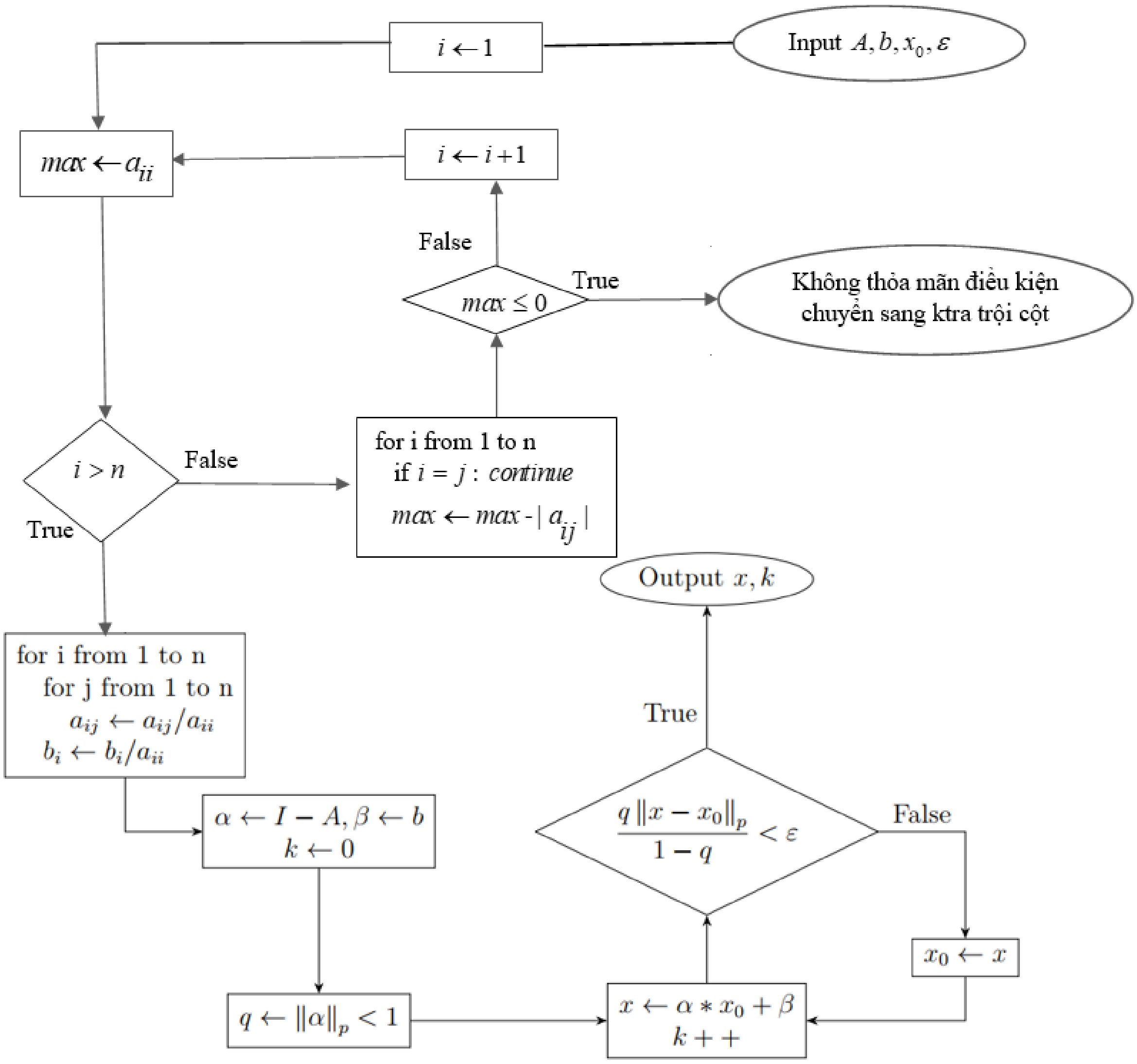
```
IF SAME_CHECK(a[]):
    end().
ELSE:
    for i from 1 to len(matrix):
      if a[i] != i:
        for j from i to len(matrix):
           if a[j] = i:
              TEMP[] = matrix(row_i)
              matrix(row_i) = matrix(row_j)
              matrix(row_j) = TEMP[]
```

Xử lí tương tự đối với ma trận trội cột

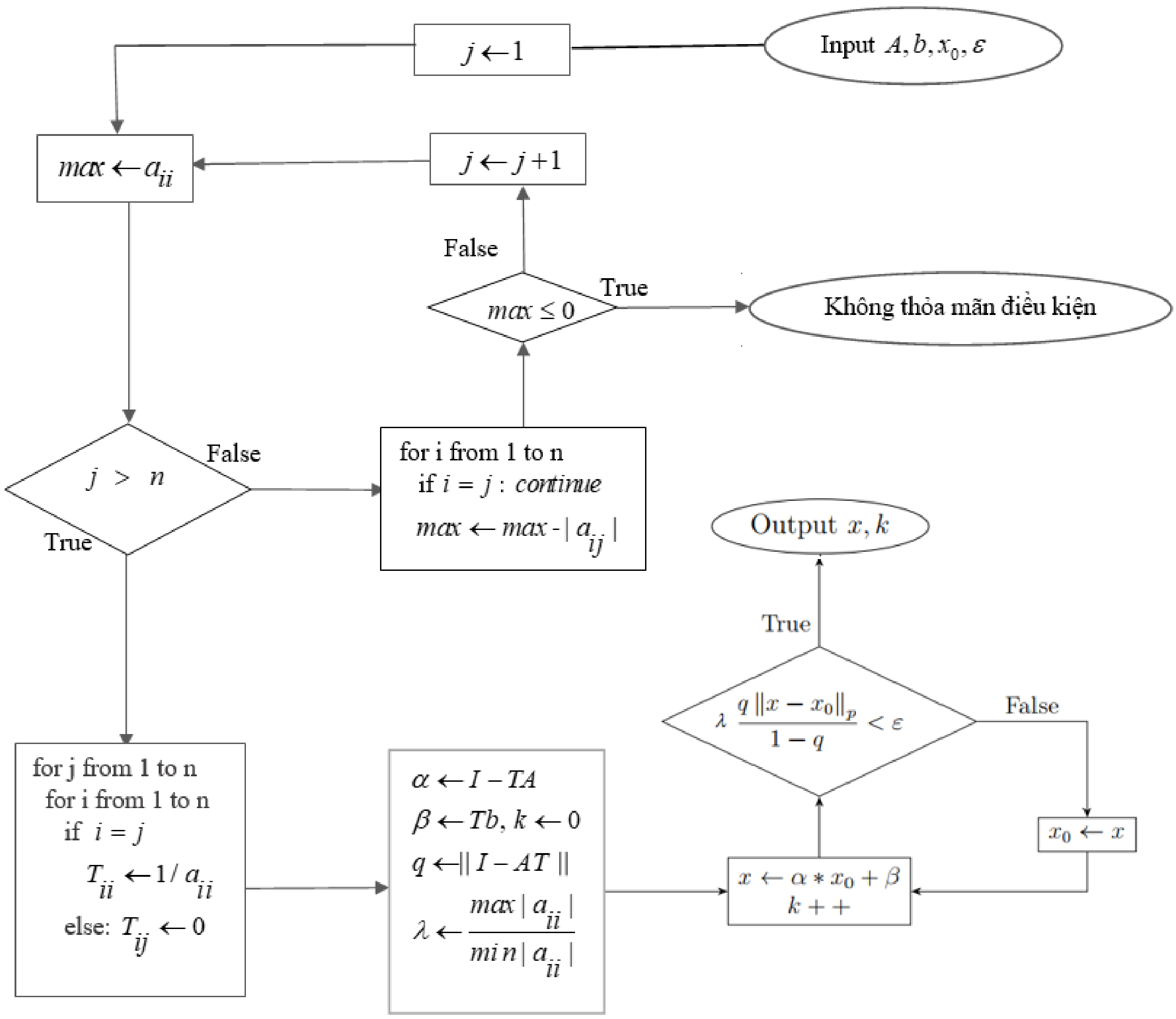
4.3 Thuật toán bằng sơ đồ khối

Ta cũng sẽ đi kiểm tra điều kiện chéo trội hàng trước, nếu không thỏa mãn điều kiện chéo trội hàng thì sẽ chuyển sang thuật toán kiểm tra điều kiện chéo trội cột.

Sơ đồ khối cho trường hợp chéo trội hàng với sai số hậu nghiệm:



Sơ đồ khối cho trường hợp chéo trội cột với sai số hậu nghiệm:



Với công thức sai số tiên nghiệm cũng tương tự như phần lặp đơn đã trình bày ở trên.

V

Chạy các ví dụ

1 Ví dụ phần lặp đơn

Ví dụ 1: Giải hệ phương trình

$$\begin{bmatrix} 1 & 0.4 & -0.3 \\ 0.3 & 1.1 & -0.3 \\ 0.2 & -0.1 & 1.3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.4 \\ 0.7 \end{bmatrix}$$

Giải bằng máy tính Casio ta có nghiệm: $\begin{cases} x_1 = 0.5027027027 \\ x_2 = 0.3598455598 \\ x_3 = 0.4888030888 \end{cases}$

Biến đổi ma trận thành $x = \alpha x + \beta$ với $\alpha = I - A$ ta có:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & -0.4 & 0.3 \\ -0.3 & -0.1 & 0.3 \\ -0.2 & 0.1 & -0.3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.4 \\ 0.7 \end{bmatrix}$$

$q = \|\alpha\|_\infty = 0.7 \Rightarrow$ có thể tiến hành lặp đơn cho phương trình
Với sai số $\epsilon = 10^{-5}$ chọn xấp xỉ đầu

$$x^{(0)} = [0, 0, 0]^t \Rightarrow x^{(1)} = [0.5, 0.4, 0.7]^t$$

$$\Rightarrow J = \log_{0.7} \frac{(1-0.7)10^{-5}}{\|x^{(1)} - x^{(0)}\|_\infty} \sim 34.65$$

$k = Ceil(J) = 35$ là số lần lặp với sai số tiên nghiệm để tìm được $x^{(n)}$

a) Chương trình chạy với thuật toán tiên nghiệm:

Ket qua vong lap thu 33
$x_1 = 0.50270270$
$x_2 = 0.35984556$
$x_3 = 0.48880309$
Ket qua vong lap thu 34
$x_1 = 0.50270270$
$x_2 = 0.35984556$
$x_3 = 0.48880309$
Ket qua vong lap thu 35
$x_1 = 0.50270270$
$x_2 = 0.35984556$
$x_3 = 0.48880309$
so vong lap la = 35

b) Chương trình chạy với thuật toán hậu nghiệm:

Ket qua cua vong lap thu 13
$x_1 = 0.50270617$
$x_2 = 0.35984905$
$x_3 = 0.48880485$
Ket qua cua vong lap thu 14
$x_1 = 0.50270183$
$x_2 = 0.35984470$
$x_3 = 0.48880222$
Ket qua cua vong lap thu 15
$x_1 = 0.50270279$
$x_2 = 0.35984564$
$x_3 = 0.48880344$
So vong lap la : 15

Dễ thấy, chương trình chạy bằng công thức tiên nghiệm, tìm số lần lặp rồi mới tìm nghiệm, chậm hơn so với chương trình chạy vừa lặp tìm nghiệm vừa xét điều kiện dừng. Lí do là chương trình thứ nhất cho số lần lặp chắc chắn ra được nghiệm, còn thực tế không cần đến từng đấy vòng lặp.

Ví dụ 2: Giải hệ phương trình.

$$\begin{bmatrix} 0.47 & 0.034 & 0.033 & 0.028 & 0.038 & 0.004 & 0.037 \\ 0.029 & 0.187 & 0.054 & 0.045 & 0.01 & 0.041 & 0.018 \\ 0.057 & 0.075 & 0.203 & 0.071 & 0.057 & 0.021 & 0.064 \\ 0.068 & 0.075 & 0.084 & 0.310 & 0.057 & 0.041 & 0.048 \\ 0.031 & 0.30 & 0.015 & 0.047 & 0.13 & 0.028 & 0.03 \\ 0.018 & 0.007 & 0.007 & 0.017 & 0.093 & 0.165 & 0.035 \\ 0.033 & 0.029 & 0.031 & 0.022 & 0.028 & 0.042 & 0.143 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 0.75 \\ 0.375 \\ 0.3 \\ 0.207 \\ 0.2 \\ 0.25 \\ 0.27 \end{bmatrix}$$

Sai số: $\epsilon = 10^-6$, xấp xỉ đầu $x^{(0)} = [0, 0, 0, 0, 0, 0, 0]^t$

a) Chạy chương trình với sai số tiên nghiệm:

Ket qua cua vong lap thu 96
x_1 = 1.553435848
x_2 = 1.114542644
x_3 = 0.773698971
x_4 = -0.205949226
x_5 = -2.121811603
x_6 = 2.289287762
x_7 = 0.910642289
Ket qua cua vong lap thu 97
x_1 = 1.553438987
x_2 = 1.114527257
x_3 = 0.773711045
x_4 = -0.205948382
x_5 = -2.121840596
x_6 = 2.289332880
x_7 = 0.910642176
Ket qua cua A * x -b la :
-0.000002690
0.000014848
-0.000011251
-0.000000317
0.000022186
-0.000040327
0.000001231
So vong lap la : 97
q = 0.860448720

b) Chạy chương trình với sai số hậu nghiệm:

Ket qua cua vong lap thu 96
x_1 = 1.553435848
x_2 = 1.114542644
x_3 = 0.773698971
x_4 =-0.205949226
x_5 = -2.121811603
x_6 = 2.289287762
x_7 = 0.910642289
Ket qua cua vong lap thu 97 ($x^{(n)}$)
x_1 = 1.553438987
x_2 = 1.114527257
x_3 = 0.773711045
x_4 = -0.205948382
x_5 = -2.121840596
x_6 = 2.289332880
x_7 = 0.910642176
Ket qua cua A * x -b la :
-0.000002690
0.000014848
-0.000011251
-0.000000317
0.000022186
-0.000040327
0.000001231
So vong lap la : 97
q =0.860448720

Ở ví dụ này lại cho số lần lặp của hai phương thức xét điều kiện dừng là như nhau trong khi

$q \sim 0.86$. Ta xét thêm một ví dụ nữa với q cũng trong khoảng > 0.85 xem có sự khác nhau không?

Ví dụ 3: Sự sai khác lớn giữa tiên nghiệm và hậu nghiệm

$$\begin{bmatrix} -1.1 & 0.1 & 0 & -0.3 & 0.1 \\ 0 & -0.3 & 0.1 & 0 & -0.1 \\ -0.1 & 0 & -0.5 & -0.2 & 0 \\ 0.2 & 0 & -0.2 & -1 & 0.3 \\ -0.1 & -0.1 & 0 & 0.4 & -0.8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} -0.2 \\ 1.3 \\ -2 \\ 0.7 \\ 0.8 \end{bmatrix}$$

Với sai số $\epsilon = 10^{-5}$, xấp xỉ đầu $x^{(0)} = [0,0,0,0,0]^t$

Dễ dàng biến đổi tương tự ví dụ 1 và tính được $\|\alpha\|_\infty = 0.9$

$x^{(0)} = [0,0,0,0,0]^t \Rightarrow x^{(1)} = [-0.2, 1.3, -2, 0.7, 0.8]^t$

$$J = \log_{0.9} \frac{(1-0.9)(10^{-5})}{\|x^{(1)} - x^{(0)}\|_\infty} \sim 137.7$$

$\Rightarrow k = Ceil(J) = 138$ là số lần lặp với sai số tiên nghiệm để tìm được $x^{(n)}$

Ta sẽ kiểm tra xem với sai số hậu nghiệm thì kết quả sẽ như thế nào?

a) Chạy chương trình với sai số tiên nghiệm:

Ket qua cua vong lap thu 136
x_1 = 0.40096432015429
x_2 = -2.12671166827387
x_3 = 4.77126325940212
x_4 = -2.12864030858245
x_5 = -1.84860173577628
Ket qua cua vong lap thu 137
x_1 = 0.40096432015429
x_2 = -2.12671166827387
x_3 = 4.77126325940212
x_4 = -2.12864030858245
x_5 = -1.84860173577628
Ket qua cua vong lap thu 138
x_1 = 0.40096432015429
x_2 = -2.12671166827387
x_3 = 4.77126325940212
x_4 = -2.12864030858245
x_5 = -1.84860173577628
Ket qua cua A * x -b la :
0.0000000000000000
0.0000000000000000
0.0000000000000000
-0.0000000000000003
0.0000000000000000
So vong lap la : 138
$q = 0.9$

b) Chạy chương trình với sai số hậu nghiệm:

Ket qua cua vong lap thu 42
x_1 = 0.40096371421250
x_2 = -2.12671676233101
x_3 = 4.77126313069246
x_4 = -2.12863985799399
x_5 = -1.84860032093893
Ket qua cua vong lap thu 43
x_1 = 0.40096387764995
x_2 = -2.12671538846857
x_3 = 4.77126316552378
x_4 = -2.12863997957767
x_5 = -1.84860070257353
Ket qua cua vong lap thu 44
x_1 = 0.40096399700410
x_2 = -2.12671438511827
x_3 = 4.77126319091243
x_4 = -2.12864006834682
x_5 = -1.84860098126391
Ket qua cua A * x -b la :
0.00000008716132
0.00000073275311
0.00000001851274
-0.00000006481402
-0.00000020351618
So vong lap la : 44
$q = 0.9$

Số lần lặp sai số hậu nghiệm giảm đi đáng kể so với tiên nghiệm, ở đây $q = \|\alpha\|$ đang gần 1 nên sự bất ổn và lệch lớn về số lần lặp giữa hai công thức sai số.

Ví dụ 4: Khi chuẩn của α quá gần 1

$$\begin{bmatrix} 1 & 0.4999 & 0.5 \\ 0.4999 & 1 & 0.5 \\ 0.5 & 0.4999 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.1 \\ 1.2 \end{bmatrix}$$

Với sai số $\epsilon = 10^{-6}$, xấp xỉ đầu $x^{(0)} = [0,0,0]^t$

Sau khi biến đổi về dang $x = \alpha x + \beta$ thì ta được: $\|\alpha\|_\infty = 0.9999 \sim 1$:

$$x^{(0)} = [0,0,0]^t \Rightarrow x^{(1)} = [2.0, 2.1, 2.2]^t$$

$$J = \log_{0.9999} \frac{(1-0.9999)(10^{-9})}{\|x^{(1)} - x^{(0)}\|_\infty} \sim 232070.12$$

$$\Rightarrow k = Ceil(J) = 232071 \text{ là số lần lặp với sai số tiên nghiệm để tìm được } x^{(n)}$$

Một con số khá lớn. Ta sẽ chạy chương trình để kiểm nghiệm lại con số trên và xem số vòng lặp hậu nghiệm là bao nhiêu?.

a) Chạy chương trình với sai số tiên nghiệm:

Ket qua vong lap thu 232069
$x_1 = 0.350037497921532$
$x_2 = 0.549997505919933$
$x_3 = 0.750037497921532$
Ket qua vong lap thu 232070
$x_1 = 0.350037497829859$
$x_2 = 0.549997505828260$
$x_3 = 0.750037497829859$
Ket qua vong lap thu 232071
$x_1 = 0.350037497921523$
$x_2 = 0.549997505919924$
$x_3 = 0.750037497921523$
Ket qua cua A * x -b la :
0.000000000091655
0.000000000091655
0.000000000091655
So vong lap la : 232071
$q = 0.9999$

b) Chạy chương trình với sai số hậu nghiệm:

Ket qua cua vong lap thu 231198
$x_1 = 0.350037497825682$
$x_2 = 0.549997505824083$
$x_3 = 0.750037497825682$
Ket qua cua vong lap thu 231199
$x_1 = 0.350037497925700$
$x_2 = 0.549997505924100$
$x_3 = 0.750037497925700$
Ket qua cua vong lap thu 231200
$x_1 = 0.350037497825692$
$x_2 = 0.549997505824093$
$x_3 = 0.750037497825692$
Ket qua cua A * x -b la :
-0.000000000099998
-0.000000000099998
-0.000000000099998
So vong lap la: 231200
$q = 0.9999$

Số lần lặp của thuật toán hậu nghiệm cũng rất lớn, vì thế khi $\|\alpha\| \to 1$ thì độ bất ổn và số lần lặp của hệ sẽ tăng lên rất cao.

2 Ví dụ phần lặp Jacobi

Ví dụ 1: Ma trận trội hàng

$$\begin{bmatrix} 7.7 & 1.2 & 1.1 & 2.1 \\ 2.4 & 17.4 & 2.6 & 2.2 \\ 1.1 & 1.3 & 8.3 & 1.0 \\ 4.2 & 2.2 & 2.0 & 9.8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 9.8 \\ 4.6 \\ 2.1 \\ 14 \end{bmatrix}$$

Với sai số $\epsilon = 10^{-5}$,xấp xỉ đầu $x^{(0)} = [0, 0, 0, 0]^t$

Biến đổi ma trận như đã trình bày ở phần lý thuyết ta được:

$$\alpha = B = -\begin{bmatrix} 0 & \frac{1.2}{7.7} & \frac{1}{7} & \frac{2.1}{7.7} \\ \frac{2.4}{17.4} & 0 & \frac{2.6}{17.4} & \frac{2.2}{17.4} \\ \frac{1.1}{8.3} & \frac{1.3}{8.3} & 0 & \frac{1}{8.3} \\ \frac{3}{7} & \frac{1.1}{4,9} & \frac{1}{4.9} & 0 \end{bmatrix} \quad \text{và} \quad \beta = \begin{bmatrix} 9.8 \\ 4.6 \\ 2.1 \\ 14 \end{bmatrix}$$

Với $\|\alpha\|_\infty = \dfrac{6}{7} \sim 0.8571428571$ Tiến hành lặp cho bài toán ta thu được:

a) Sai số tiên nghiệm

Ket qua vong lap thu 88
x_1 = 1.0000000000
x_2 = 0.0000000000
x_3 = 0.0000000000
x_4 = 1.0000000000
Ket qua vong lap thu 89
x_1 = 1.0000000000
x_2 = 0.0000000000
x_3 = 0.0000000000
x_4 = 1.0000000000
Ket qua vong lap thu 90
x_1 = 1.0000000000
x_2 = 0.0000000000
x_3 = 0.0000000000
x_4 = 1.0000000000
ket qua cua A * X - B la :
0.0000000000
0.0000000000
-0.0000000000
0.0000000000
So vong lap la : 90
0.8571428571

b) Sai số hậu nghiệm

Ket qua cua vong lap thu 23
x_1 = 1.0000012153
x_2 = 0.0000008741
x_3 = 0.0000008606
x_4 = 1.0000015752
Ket qua cua vong lap thu 24
x_1 = 0.9999993112
x_2 = -0.0000004954
x_3 = -0.0000004878
x_4 = 0.9999991073
Ket qua cua vong lap thu 25
x_1 = 1.0000003903
x_2 = 0.0000002808
x_3 = 0.0000002764
x_4 = 1.0000005059
Ket qua cua A * x - b la :
0.0000047091
0.0000076538
0.0000035947
0.0000077681
So vong lap la : 25
0.8571428571

Ví dụ 2: Giải phương trình với cả trội hàng và cột

$$\begin{bmatrix} 5.1 & 1.1 & 1.2 & 1.0 \\ 1.1 & 6.1 & 1.0 & 1.1 \\ 1.2 & 1.0 & 7.1 & 1.0 \\ 1.0 & 1.1 & 1.0 & 5.1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 6.3 \\ 2.1 \\ 8.3 \\ 2.0 \end{bmatrix}$$

Với sai số $\epsilon = 10^{-5}$, xấp xỉ đầu $x^{(0)} = [0, 0, 0, 0]^t$

Nhận thấy ma trận trên vừa chéo trội hàng và vừa chéo trội cột. Ta chỉnh sửa chương trình rồi chạy cả hai trường hợp chéo trội hàng và cột để so sánh kết quả số lần lặp
Chạy chương trình với ưu tiên trội hàng:

a) Sai số tiên nghiệm:

Ket qua cua vong lap thu 29
x_1 = 1.0000000219
x_2 = 0.0000000189
x_3 = 1.0000000168
x_4 = 0.0000000210
Ket qua cua vong lap thu 30
x_1 = 0.9999999879
x_2 = -0.0000000105
x_3 = 0.9999999907
x_4 = -0.0000000116
So vong lap la : 30
q= 0.6470588235

b) Sai số hậu nghiệm:

Ket qua cua vong lap thu 21
x_1 = 1.0000024288
x_2 = 0.0000020972
x_3 = 1.0000018635
x_4 = 0.0000023315
Ket qua cua vong lap thu 22
x_1 = 0.9999986520
x_2 = -0.0000011639
x_3 = 0.9999989657
x_4 = -0.0000012940
So vong lap la : 22
q = 0.6470588235

Chạy chương trình với ưu tiên trội cột:

a) Sai số tiên nghiệm:

Ket qua cua vong lap thu 32
x_1 = 0.9999999963
x_2 = -0.0000000032
x_3 = 0.9999999971
x_4 = -0.0000000036
Ket qua cua vong lap thu 33
x_1 = 1.0000000021
x_2 = 0.0000000018
x_3 = 1.0000000016
x_4 = 0.0000000020
So vong lap la : 33
q = 0.6470588235

b) Sai số hậu nghiệm:

Ket qua cua vong lap thu 24
x_1 = 0.9999995848
x_2 = -0.0000003585
x_3 = 0.9999996814
x_4 = -0.0000003986
Ket qua cua vong lap thu 25
x_1 = 1.0000002304
x_2 = 0.0000001990
x_3 = 1.0000001768
x_4 = 0.0000002212
Ket qua cua A * x - b la :
So vong lap la : 25
q = 0.6470588235

Ta có thể thấy, với phương pháp lặp theo trội hàng và phương pháp lặp theo trội cột cho ta số lần lặp sấp sỉ nhau khi q không quá lớn khi mà ma trận A ở ví dụ này có tính đối xứng qua đường chéo chính.

Ví dụ 3: Ma trận trội không trên đường chéo chính

$$\begin{bmatrix} 3 & 7.2 & 19.5 & 2.4 \\ 4 & 15.7 & -2.5 & -6 \\ 1.9 & 12.4 & -3.7 & 36.6 \\ 8 & 1.4 & -2.5 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Với sai số $\epsilon = 10^{-4}$, xấp xỉ đầu $x^{(0)} = [0, 0, 0]^t$

Khi đưa vào chương trình, ma trận trên sẽ được đưa về dạng chéo trội như bình thường của phương pháp lặp Jacobi. Ta sẽ tiến hành chạy chương trình như sau:

a) Chạy chương trình với sai số tiên nghiệm:

Ket qua cua vong lap thu 70
x_1 = 0.4679914
x_2 = 0.0207455
x_3 = -0.0341843
x_4 = 0.0471883
Ket qua cua vong lap thu 71
x_1 = 0.4679914
x_2 = 0.0207455
x_3 = -0.0341843
x_4 = 0.0471883
Ket qua cua A * x -b la :
0.0000000
0.0000000
0.0000000
0.0000000
So vong lap la : 71
q =0.8625000

b) Chạy chương trình với sai số hậu ngiệm:

Ket qua cua vong lap thu 11
x_1 = 0.4679937
x_2 = 0.0207451
x_3 = -0.0341795
x_4 = 0.0471899
Ket qua cua vong lap thu 12
x_1 = 0.4679923
x_2 = 0.0207462
x_3 = -0.0341847
x_4 = 0.0471888
Ket qua cua A * x -b la :
0.0000014
0.0000143
0.0000312
0.0000112
So vong lap la : 12
q = 0.8625000

Nhận thấy ở thuật toán tiên nghiệm, với số lần lặp là 71 lần nhiều hơn gấp ~ 6 lần so với thuật toán hậu nghiệm, bởi lẽ có sự chênh lệch này là do $q = 0.86250$ khá gần so với 1. Và giá trị sai lệch của thuật toán tiên nghiệm trong phương trình đã trả về $[0.0000000, 0.0000000, 0.0000000]^t$ đúng đến hơn 7 chữ số 0 ở phần thập phân nó rất nhỏ so với mức chấp nhận được xong nó lại tốn số lần lặp hơn khi nhìn sang thuật toán hậu nghiệm chỉ mất 12 lần lặp đã cho ta sai số mong muốn.

VI Chương trình

Dưới đây là những hàm chính trong thuật toán của chúng em với sự hỗ trợ của ngôn ngữ **C++** và phương pháp lập trình hướng đối tượng.
Chúng em xem cảm ơn cô đã góp ý cho chúng em phần lỗi nhỏ ở khâu đọc file input, chúng em đã sửa lại với việc kiểm tra phần đọc file đầu vào như sau:

chương trình đã sữa lại file .inp bằng cách thêm hàm tính số phần tử trong file .inp vào so sánh với số phần tử cần thiết để chạy chương trình nếu chúng không khác nhau chương trình sẽ đưa ra lỗi "Input khong hop le"

```
int soptu()
{
    fstream ab;
    double a;
    int dem=0;
    ab.open(Vao, ios::in);
     while(!ab.eof())
    {
        ab>>a;
        dem++;
    }
    ab.close();
    dem=dem-1;
    return dem;
}

void checkfile(){
    int a;
    a = A.row * A.col + B.row * B.col + 7;
    int b = soptu();
    if(a != b)
    {
        cout<<"Input khong hop le";
        exit(0);
    }
}
```

1 Hàm lặp khi hệ đã ở dạng dãy lặp

```
template<typename T>
matrix<T> loop(matrix<T> &A, matrix<T> &B, matrix<T> &X0, double &eps, int &type, double
    &q, int &normType, double w = 1)//ham lap
{
    fstream wi;
    matrix<T> X(X0.row, X0.col);
    switch (type)
    {
    case 1:
    {
        wi.open("A.csv", ios::out);
        matrix<T> X1(X0.row, X0.col);
```

```
        double w1 = q / (1 - q);
        X = X0;
        int loopNumber = 0;
        do
        {
            ++loopNumber;
            X1 = X;
            X = (A * X1) + B;
            wi << "Ket qua cua vong lap thu " << loopNumber  << endl;
            wi << fixed << setprecision(-log10(eps) + 3) << X;
        } while (w * w1 * getNorm((X-X1),normType) > eps);
        cout << "So vong lap la : " << loopNumber << "\n";
        cout << q << endl;
        wi << "Sai so : " << (w * w1 * getNorm((X-X1),normType));
        wi << endl;
        wi.close();
        return X;
        break;
    }
    case 2:
    {
        wi.open("A1.csv", ios::out);
        matrix<T> X1(X0.row, X0.col);
        X = X0;
        X1 = (A * X) + B;
        int loopNumber = ceil((log((eps * (1 - q)) / (getNorm((X-X1),normType) * w))) /
    (log(q)));
        for (int i = 1; i <= loopNumber; ++i)
        {
            X = (A * X) + B;
            wi << "Ket qua vong lap thu " << i << endl;
            wi << fixed << setprecision(-log10(eps) + 3) << X;
        }
        cout << "So vong lap la : " << loopNumber << "\n";
        cout << q << endl;
        wi << "Sai so la : " << (w*(pow(q, loopNumber)/(1-q))*getNorm((X1 - X0), normType));
        wi << endl;
        wi.close();
        return X;
        break;
    }
    case 3:
    {
        wi.open("A2.csv", ios::out);
        matrix<T> X1(X0.row, X0.col);
        X = X0;
        X1 = (A * X) + B;
        int loopNumber = ceil((log((eps * (1 - q)) / (getNorm((X-X1),normType) * w))) /
    (log(q)));
        matrix<matrix<T> > P(2,2);
        matrix<T> O(A.row, A.col);
        matrix<T> I(A.row, A.col);
        for (int i = 0; i < A.row; i++) I[i][i] = 1;
        P[0][0] = A; P[0][1] = I;
        P[1][0] = O;     P[1][1] = I;
        P = Pow(P,loopNumber);
        X = P[0][0] * X + P[0][1] * B;
        cout << "So vong lap la : " << loopNumber << "\n";
```

```
        cout << q << endl;
        wi << "Sai so la : " << (w*(pow(q, loopNumber)/(1-q))*getNorm((X1 - X0), normType));
        wi << endl;
        wi.close();
        return X;
        break;
    }
    }
    wi.close();
    return X;
}
```

2 Chương trình phương pháp lặp đơn

```
template<typename T>
matrix<T> singleloop(matrix<T> &A, matrix<T> B,
           matrix<T> &X0, double &eps)
{
    Check(A, B, X0, eps);
    int type;
    double q;
    int normType = 0;
    matrix<T> C(A.row, A.col);
    for(int i = 0; i < A.row; ++i)
      for(int j = 0; j < A.col; ++j)
      {
          if(i == j)
            C[i][j] = 1 - A[i][j];
          else
            C[i][j] = -A[i][j];
      }
    //normType = MinNorm(C, q);
    for(int i = 1; i <= SumNorm; i++)
    {
        if(getNorm(C, i) < 1)
        {
            normType = i;
            q = getNorm(C, i);
            break;
        }
    }
    if (normType != 0)
    {
        cout << endl << "1.Hau nghiem" << endl;
        cout << "2.Tien nghiem" << endl;
        cout << "3.Tien nghiem 2" << endl;
        cout << "Chon cong thuc sai so : ";
        cin >> type;
        return loop(C, B, X0, eps, type, q, normType);
    }
    for(int i = 0; i < A.row; ++i)
    {
      for(int j = 0; j < A.col; ++j)
      {
          if(i == j)
```

```
            C[i][j] = 1 + A[i][j];
          else
             C[i][j] = A[i][j];
      }
    }
    //normType = MinNorm(C, q);
    for(int i = 1; i <= SumNorm; i++)
    {
        if(getNorm(C, i) < 1)
        {
            normType = i;
            q = getNorm(C, i);
            break;
        }
    }
    if(normType == 0)
    {
        cout << "Khong the giai bang lap don!!";
        exit(0);
    }
    else
    {
        for(int i = 0; i < B.row; ++i)
          for(int j = 0; j < B.col; ++j)
            {
              B[i][j] = -B[i][j];
            }
    }
    cout << endl << "1.Hau nghiem" << endl;
    cout << "2.Tien nghiem" << endl;
    cout << "3.Tien ngiem 2" << endl;
    cout << "Chon cong thuc sai so : ";
    cin >> type;
    return loop(C, B, X0, eps, type, q, normType);
}
```

3 Chương trình phương pháp lặp Jacobi

```
template<typename T>
matrix<T> jacobiloop(matrix<T> A, matrix<T> B, matrix<T> &X0, double &eps)
{
    Check(A, B, X0, eps);
    double q;
    int type;
    int normType = dominant(A);
    if (normType == 0)
    {
        cout << "Ma tran khong troi";
        exit(0);
    }
    cout << endl << "1.Hau nghiem" << endl;
    cout << "2.Tien nghiem" << endl;
    cout << "3.Tien nghiem 2" << endl;
    cout << "Chon cong thuc sai so : ";
    cin >> type;
```

```
    matrix<T> C(A.row, A.col), D(B.row, B.col);
    if(normType == 3 || normType ==4)
    {
        for(int i = 0; i < A.row; ++i)
        {
            int a = 0;
            if(a != arr[a])
            {
               A[a].swap(A[arr[a]]);
               B[a].swap(B[arr[a]]);
               Swap(arr[a], arr[arr[a]]);
            }
           else a++;
        }
        normType = normType - 2;
    }
    int Domi = normType;
    for (int i = 0; i < A.row; ++i)
    {
        for (int j = 0; j < A.col; ++j)
        {
            if(i!=j)
            {
                C[i][j] = -(A[i][j]) / (A[i][i]);
            }
        }
    }
    for (int i = 0; i < B.row; ++i)
    {
        for (int j = 0; j < B.col; ++j)
        {
            D[i][j] = (B[i][j]) / (A[i][i]);
        }
    }
    double w = 1;
    q = getNorm(C, normType);
    //normType = MinNorm(C, q);
    if(Domi == 2)
    {
        matrix<T> F(A.row, A.col);
      for (int i = 0; i < A.row; ++i)
        for (int j = 0; j < A.col; ++j)
        {
            if(i!=j){
                F[i][j] = -(A[i][j]) / (A[j][j]);
            }
        }
        q = getNorm(F,normType);
        //normType = MinNorm(F, q);
        double t1 = abs(A[0][0]);
        double t2 = abs(A[0][0]);
        for (int i = 1; i < A.col; i++)
        {
            t1 = max(t1, abs(A[i][i]));
            t2 = min(t2, abs(A[i][i]));
        }
        w = t1 / t2;
    }
```

```
    return loop(C, D, X0, eps, type, q, normType, w);
}
```

4 Các hàm quan trọng khác trong chương trình

Hàm tính chuẩn

```
template<typename T>
double getNorm(const matrix<T> &A, const int &normType)
/*ham tinh chuan cua ma tran:
    1 la chuan hang
  2 la chuan cot
  3 la chuan euclid
  4 la chuan tri rieng*/
{
    double norm1;
    double Max = 0;
    switch (normType)
    {
    case 1:
        for (int i = 0; i < A.row; ++i)
        {
            norm1 = 0;
            for (int j = 0; j < A.col; ++j)
            {
                norm1 = norm1 + fabs(A[i][j]);
            }
            Max = max(Max, norm1);
        }
        return Max;
    case 2:
        for (int i = 0; i < A.col; ++i)
        {
            norm1 = 0;
            for (int j = 0; j < A.row; ++j)
            {
                norm1 = norm1 + fabs(A[j][i]);
            }
            Max = max(Max, norm1);
        }
        return Max;
    case 3:
        for (int i = 0; i < A.row; ++i)
        {
            norm1 = 0;
            for (int j = 0; j < A.col; ++j)
            {
                norm1 = norm1 + A[i][j] * A[i][j];
            }
        }
        Max = sqrt(norm1);
        return Max;
    case 4:
    {
        matrix<T> B = !A;
```

```
        B = B * A;
        matrix<T> X(A.row, 1, 1);
        double t0 = 0;
        double t1 = 0;
        do
        {
            t0 = t1;
            X = B * X;
            double s = X[0][0];
            int j;
            for(int i = 1; i < A.row; ++i)
            {
                if(fabs(X[i][0]) > s)
                {
                    j = i;
                    s = fabs(X[i][0]);
                }
            }
            t1 = X[j][0];
            for(int i = 0; i < A.row; ++i)
            {
                X[i][0] = X[i][0] / t1;
            }
        } while (fabs((t1 - t0)) > alpha);
        return sqrt(t1);
    }
    default:
        return -1;
        break;
    }
}
```

Hàm kiểm tra tính trội của ma trận

```
template<typename T>
bool checkRow(const matrix<T> &A) // ham kiem tra xem ma tran co troi hay khong(bao gom ca
    dung va khong dung vi tri)
{
    arr.resize(A.row + 1);
    vector<int> use;
    use.resize(A.row + 1);
    for(int i = 0; i < use.size(); i++) use[i] = 0;
    double norm1;
    double Max;
    int k = -1;
    for (int i = 0; i < A.row; ++i)
    {
        Max = 0;
        norm1 = 0;
        for (int j = 0; j < A.col; ++j)
        {
            {
                norm1 = norm1 + abs(A[i][j]);
                if(abs(A[i][j]) > Max)
                {
                    Max = abs(A[i][j]);
```

```
                    k = j;
                }
            }
        }
        if(abs(A[i][k]) <= norm1 / 2 || use[k] == 1)
        {
            return false;
        }
        else
        {
            arr[i] = k;
            use[k] = 1;
        }

    }
    return true;
}
```

Hàm kiểm tra tính chéo trội của ma trận

```
template<typename T>
int dominant(const matrix<T> &A)
{
    double a = 0;
    bool check = true;
    for (int i = 0; i < A.row; ++i)
    {
        a = abs(A[i][i]);
        for (int j = 0; j < A.col; ++j)
        {
            if (i == j)
                continue;
            a = a - abs(A[i][j]);
            if (a <= 0)
            {
                check = false;
                break;
            }
        }
    }
    if (check) return 1;
    check = true;
    for (int i = 0; i < A.col; ++i)
    {
        a = abs(A[i][i]);
        for (int j = 0; j < A.row; ++j)
        {
            if (i == j)
                continue;
            a = a - abs(A[j][i]);
            if (a <= 0)
            {
                check = false;
                break;
            }
        }
```

```
    }
    if (check) return 2;
    if(checkRow(A)) return 3;
    if(checkCol(A)) return 4;
    return 0;
}
```

VII Đánh giá

Qua thuật toán và hệ thống ví dụ, ta rút ra kết luận sau:

- Phương pháp lặp đơn giải quyết được vấn đề về sự bất ổn của hệ khi giải bằng phương pháp nghiệm đúng.
- Tối ưu hơn được bộ nhớ khi xử lí trên máy tính so với phương pháp giải đúng.
- Phương pháp lặp đơn hội tụ phụ thuộc vào $q = \|\alpha\|$, và bất ổn về sự co khi $q \to 1$.
- Hai phương pháp chéo trội hàng và cột có số lần lặp xấp xỉ nhau khi ma trận đối xứng qua đường chéo chính.
- Lặp jacobi cung cấp phương pháp xử lý khi chuẩn của α không thỏa mãn điều kiện lặp đơn, nhưng chỉ khi dựa trên tính chéo trội của ma trận, nghĩa là chỉ lặp Jacbi được khi ma trận chéo trội.

$\Rightarrow$ Vì vậy lớp hệ phương trình phương pháp xử lý được cũng tương đối hẹp.

VIII Tài liệu tham khảo

1. Giáo trình "Giải tích sô" - Lê Trọng Vinh -2007
2. A Friendly Introduction to Numerical Analysis - Brian Bradie
3. https://en.wikipedia.org/