

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



CÁC PHƯƠNG PHÁP
TÌM GẦN ĐÚNG MA TRẬN NGHỊCH ĐẢO

Bùi Tiến Thành - MSSV 20190081

Chu Thị Ngân - MSSV 20195904

GIẢNG VIÊN HƯỚNG DẪN

TS. Hà Thị Ngọc Yến

Hà Nội, tháng 11, 2020

Mục lục

	Mục lục	1
1	Đặt vấn đề	3
1.1	Phái biểu bài toán	3
1.2	Tại sao phải giải gần đúng?	3
1.3	Các phương pháp giải	3
1.4	Chuẩn ma trận và sự hội tụ của dãy ma trận	4
2	Phương pháp Newton	4
2.1	Ý tưởng	4
2.2	Công thức lặp	5
2.3	Điều kiện hội tụ	5
2.4	Công thức sai số	6
2.5	Thuật toán và chương trình	6
3	Phương pháp lặp Jacobi	7
3.1	Công thức lặp	7
3.2	Điều kiện hội tụ	8
3.3	Công thức sai số	8
3.4	Thuật toán và chương trình	9
4	Phương pháp Gauss - Seidel	12
4.1	Công thức lặp	12
4.2	Công thức sai số	12
4.3	Thuật toán và chương trình	13
5	Phân tích, tổng kết các phương pháp và hệ thống ví dụ	17
5.1	Ưu, nhược điểm của các phương pháp	17
5.2	Chọn xấp xỉ đầu cho các phương pháp	17
5.3	Hệ thống ví dụ	19
5.4	Ứng dụng	26

Phụ lục	27
A Các chương trình được sử dụng	27
B Mã nguồn báo cáo và bài trình chiếu	27
C Hướng dẫn sử dụng chương trình	27
C.1 Lưu ý trước khi sử dụng	27
C.2 Hướng dẫn sử dụng chi tiết	27
Danh mục tài liệu tham khảo	30

1 Đặt vấn đề

1.1 Phái biểu bài toán

Bài toán: Cho ma trận A vuông cấp n , khả nghịch. Tìm ma trận nghịch đảo của A .

Trong báo cáo này, nhóm sẽ nghiên cứu các phương pháp giải gần đúng nghịch đảo của ma trận trên ma trận thực vuông cấp n , khả nghịch. Nói cách khác, ta có ma trận A thỏa mãn các tính chất sau:

$$A \in \mathbf{M}_{n \times n}(\mathbb{R}) \text{ và } \det(A) \neq 0$$

ở đó $\mathbf{M}_{n \times n}(\mathbb{R})$ là tập các ma trận vuông cấp n trên tập số thực.

1.2 Tại sao phải giải gần đúng?

Các phương pháp giải chính xác thường cho ra lời giải sau hữu hạn bước và thậm chí cho lời giải chính xác nếu máy tính có thể xử lý số thực với cấp chính xác là vô cùng. Tuy nhiên, các phương pháp này gặp khó khăn với những ma trận có kích thước rất lớn và có nhiều phần tử không (hay còn gọi là ma trận thưa) và do hạn chế của tính toán số trên máy tính, các phương pháp giải đúng có thể sinh ra sai số rất lớn. Hơn nữa, độ phức tạp của các phương pháp tìm chính xác ma trận nghịch đảo là $\mathcal{O}(n^3)$ và khối lượng tính toán sẽ tăng rất nhanh khi tăng kích cỡ ma trận. Do đó, phương pháp xấp xỉ được phát minh để phần nào giải quyết những vấn đề mà phương pháp chính xác gặp phải.

1.3 Các phương pháp giải

Ta thấy đây là một trường hợp riêng của bài toán giải phương trình $AX = B$, trong đó $B = E$ là ma trận đơn vị cấp n , do đó có thể áp dụng các phương pháp giải gần đúng phương trình ma trận vào bài toán này. Ngoài ra, nhóm còn đề cập thêm một phương pháp riêng là phương pháp Newton để tìm gần đúng ma trận nghịch đảo.

(i) Phương pháp Newton

(ii) Phương pháp lặp đơn và lặp Jacobi

(iii) Phương pháp lặp Gauss-Seidel

1.4 Chuẩn ma trận và sự hội tụ của dãy ma trận

Điểm chung của các phương pháp này là đều xuất phát từ một xấp xỉ ban đầu $X^{(0)}$, tìm cách hiệu chỉnh dần sau một số bước ta thu được dãy $\{X^{(k)}\}$ mà $\{X^{(k)}\}$ tiến gần đến A^{-1} . Do đó, cần đưa vào khái niệm về chuẩn và sự hội tụ của dãy ma trận

Về chuẩn của ma trận: Trong báo cáo này, nhóm sẽ sử dụng một số chuẩn thông dụng để xác định sai số và độ hội tụ của các phương pháp. Các chuẩn đó là chuẩn Euclid (chuẩn 2), chuẩn cột (chuẩn 1) và chuẩn hàng (chuẩn vô cùng) và chúng được định nghĩa như sau:

$$\|A\|_2 = \sqrt{\sum_{i=1}^n \sum_{j=1}^n |A_{ij}|^2}$$

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |A_{ij}|$$

$$\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |A_{ij}|$$

Dễ thấy các "chuẩn" này là một chuẩn trên $M_{n \times n}(\mathbb{R})$ và thỏa mãn các tiên đề của chuẩn (*bạn đọc tự chứng minh như một câu hỏi*)

Sự hội tụ của dãy ma trận: Từ chuẩn của ma trận, ta xây dựng định nghĩa về sự hội tụ của dãy ma trận như sau: Dãy ma trận $\{A^{(k)}\} \subset M_{n \times n}(\mathbb{R})$ hội tụ về ma trận A , hay $\lim_{k \rightarrow \infty} A^{(k)} = A$ nếu:

$$\lim_{k \rightarrow \infty} \|A^{(k)} - A\| = 0$$

2 Phương pháp Newton

2.1 Ý tưởng

Giả sử cho số thực a khác 0, tìm x để $ax = 1 \Rightarrow a = \frac{1}{x}$ với $a \neq 0$. Ta đặt:

$$f(x) = a - \frac{1}{x} = 0$$

Theo công thức Newton:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{a - \frac{1}{x_k}}{\frac{1}{x_k^2}}$$

hay:

$$x_{k+1} = x_k(2 - ax_k) \text{ với } k \in \mathbb{N}$$

2.2 Công thức lặp

Vận dụng ý tưởng trên, xem a là ma trận A , x là ma trận X , cần tìm X sao cho $AX = XA = E$ ta có công thức lặp như sau:

$$X_{k+1} = X_k(2E - AX_k) \text{ với } k \in \mathbb{N} \quad (1)$$

ở đó E là ma trận đơn vị cùng cấp và X_0 là xấp xỉ đầu

2.3 Điều kiện hội tụ

Ta cần tìm điều kiện để

$$\lim_{k \rightarrow \infty} X_k = A^{-1}$$

Ta đặt $G_k = E - AX_k$. Theo công thức (1) ta được:

$$\begin{aligned} G_k &= E - AX_{k+1}(2E - AX_{k+1}) \\ &= E - 2AX_{k+1} + (AX_{k+1})^2 \\ &= (E - AX_{k+1})^2 = G_{k+1}^2 \end{aligned}$$

Theo truy hồi ta có: $G_k = G_{k-1}^2 = (G_{k-2}^2)^2 = (G_{k-2})^4 = \dots = G_0^{2^k}$

Mặt khác $A^{-1} - X_k = A^{-1}(E - AX_k) = A^{-1}G_k = A^{-1}G_0^{2^k}$ nên:

$$\|A^{-1} - X_k\| \leq \|A^{-1}\| \|G_0\|^{2^k} \quad (2)$$

ở đó $G_0 = E - AX_0$. Cũng từ (2) ta thấy nếu $\|G_0\| < 1$ thì:

$$\|A^{-1} - X_k\| \xrightarrow{k \rightarrow \infty} 0 \text{ hay } \lim_{k \rightarrow \infty} X_k = A^{-1}$$

Vậy $\|G_0\| < 1$ là điều kiện để quá trình lặp (1) hội tụ. Tuy nhiên, trên máy tính ta không thể cho k lặp đến vô cùng mà chỉ lặp đến một khoảng sai số nhất định nên vì vậy, cần phải xây dựng công thức sai số.

2.4 Công thức sai số

Giả sử $\|G_0\| \leq q < 1$. Ta lại có:

$$\begin{aligned} G_0 &= E - AX_0 \Leftrightarrow X_0 = A^{-1}(E - G_0) \\ &\Leftrightarrow A^{-1} = X_0(E - G_0)^{-1} \\ &\Leftrightarrow A^{-1} = X_0(E + G_0 + G_0^2 + \dots) \end{aligned}$$

$$\text{Suy ra } \|A^{-1}\| \leq \|X_0\| (1 + q + q^2 + \dots) = \frac{\|X_0\|}{1-q}$$

Thay vào (2) ta được:

$$\|A^{-1} - X_k\| \leq \frac{\|X_0\|}{1-q} \|G_0\|^{2^k} = \frac{\|X_0\|}{1-q} q^{2^k}$$

Từ đó ta có công thức đánh giá sai số như sau:

$$\|A^{-1} - X_k\| \leq \frac{\|X_0\|}{1-q} q^{2^k} \quad (3)$$

2.5 Thuật toán và chương trình

Input, Output và chương trình:

Input: Ma trận A , xấp xỉ đầu X_0 và sai số ε . Giả thiết coi như các công thức ở 2.3 và 2.4 thỏa mãn

Output: Ma trận xấp xỉ A^{-1}

Chương trình: `newton.py`

Mã giả:

Algorithm 1: Phương pháp Newton tìm ma trận nghịch đảo

Input: Ma trận A, X_0, ε

Output: Ma trận $X^* = A^{-1}$ là ma trận nghịch đảo

begin

 Nhập A, X_0, ε ;

$q \leftarrow \|E - AX_0\|$;

$k \leftarrow 0$;

$X \leftarrow X_0$;

while $\frac{\|X_0\| * q^{2^k}}{1-q} > \varepsilon$ **do**

$X \leftarrow X(2E - AX)$;

$k \leftarrow k + 1$;

end

 Đưa ra X chính là ma trận nghịch đảo;

end

Giải thích biến trong thuật toán:

- A : Ma trận cần nghịch đảo
- X_0 : Xấp xỉ đầu vào
- ε : Sai số cho phép
- E : Ma trận đơn vị cùng cấp với A
- X : Dãy ma trận X_k , tuy nhiên chúng ta chỉ cần xét phần tử X_{n-1} để tính X_k nên chúng ta chỉ cần 1 ma trận X
- q : Giá trị q trong công thức sai số

3 Phương pháp lập Jacobi

3.1 Công thức lập

Như kiến thức đã biết trong phần lập Jacobi, với $T = \text{diag}\{\frac{1}{A_{11}}, \frac{1}{A_{22}}, \dots, \frac{1}{A_{nn}}\}$ là ma trận đường chéo cấp n , ta có:

$$X_{k+1} = (E - TA)X_k + T \quad (4)$$

Do phần này (cùng với phương pháp lặp Gauss-Seidel) nằm trong phần kiến thức đã biết nên trong báo cáo chỉ áp dụng trực tiếp và không chứng minh các công thức.

3.2 Điều kiện hội tụ

Phương pháp này cùng với phương pháp lặp Gauss-Seidel (sẽ được nêu sau) hội tụ nếu $\|E - TA\| < 1$, tức là ma trận A phải chéo trội:

$$\sum_{j=1, j \neq i}^n |A_{ij}| < |A_{ii}| \quad \forall i = \overline{1, n} \quad (\text{chéo trội hàng})$$

$$\text{hoặc} \quad \sum_{i=1, i \neq j}^n |A_{ij}| < |A_{jj}| \quad \forall j = \overline{1, n} \quad (\text{chéo trội cột})$$

3.3 Công thức sai số

Ta vẫn đặt $T = \text{diag}\{\frac{1}{A_{11}}, \frac{1}{A_{22}}, \dots, \frac{1}{A_{nn}}\}$ như trong phần 3.2. Ta xét 2 trường hợp:

(i) **Trường hợp chéo trội hàng:** Đặt $B = E - TA$, dễ thấy tồn tại một số q để:

$$\|B\|_{\infty} \leq q < 1$$

Do đó, ta có công thức sai số như sau:

$$\|X_k - X^*\|_{\infty} \leq \frac{q}{1 - q} \|X_k - X_{k-1}\|_{\infty} \quad (5)$$

$$\|X_k - X^*\|_{\infty} \leq \frac{q^k}{1 - q} \|X_1 - X_0\|_{\infty} \quad (6)$$

(ii) **Trường hợp chéo trội cột:** Đặt $B_1 = E - AT$ và $\lambda = \frac{\max |A_{ii}|}{\min |A_{ii}|}$, $\|B_1\|_1 \leq q < 1$, dễ thấy tồn tại một số q để:

$$\|B_1\|_1 \leq q < 1$$

Do đó, ta có công thức sai số như sau:

$$\|X_k - X^*\|_1 \leq \lambda \frac{q}{1-q} \|X_k - X_{k-1}\|_1 \quad (7)$$

$$\|X_k - X^*\|_1 \leq \lambda \frac{q^k}{1-q} \|X_1 - X_0\|_1 \quad (8)$$

3.4 Thuật toán và chương trình

Mã giả thuật toán chính:

Algorithm 2: Phương pháp Jacobi tìm ma trận nghịch đảo

Input: Ma trận A chéo trội, sai số ε

Output: Ma trận $X^* = A^{-1}$ là ma trận nghịch đảo

begin

Nhập A, ε ;

$p \leftarrow \text{checkDomination}(A)$;

$T \leftarrow \text{diag}\{\frac{1}{A_{11}}, \frac{1}{A_{22}}, \dots, \frac{1}{A_{nn}}\}$;

$B \leftarrow E - TA, B_1 \leftarrow E - AT$;

$q \leftarrow \text{getNorm}(B, B_1, p)$;

$\lambda \leftarrow \text{getLambda}(A, p)$;

$X^* \leftarrow \text{iterate}(X_0 \leftarrow A, B, T, q, \lambda, p, \varepsilon)$;

Trả về X^* là ma trận nghịch đảo;

end

Gói chọn chuẩn

Function getNorm(A, A_1, p)

Input: Ma trận A , giá trị kiểm tra p

Output: $\|A\|_\infty$ nếu $p = 1$, $\|A_1\|_1$ nếu $p = -1$

begin

if $p = 1$ **then return** $\|A\|_\infty$;

if $p = -1$ **then return** $\|A_1\|_1$;

end

Gói kiểm tra ma trận chéo trội**Function** checkDomination(A)**Input:** Ma trận A **Output:** Giá trị kiểm tra p bằng 1 nếu A chéo trội hàng, -1 nếu A chéo trội cột, 0 khi A không chéo trội**begin** $row_dom \leftarrow 1, col_dom \leftarrow 1;$ **for** $i = 1$ **to** n **do****if** $\sum_{j=1, j \neq i}^n |A_{ij}| \geq |A_{ii}|$ **then** $row_dom \leftarrow 0;$ **if** $\sum_{j=1, j \neq i}^n |A_{ji}| \geq |A_{ii}|$ **then** $col_dom \leftarrow 0;$ **end****if** $row_dom = 1$ **then return** 1;**if** $col_dom = 1$ **then return** -1;**return** 0;**end****Gói chọn giá trị lambda****Function** getLambda(A, p)**Input:** Ma trận A , giá trị kiểm tra p **Output:** $\lambda = 1$ nếu $p = 1$, $\lambda = \frac{\max |A_{ii}|}{\min |A_{ii}|}$ nếu $p = -1$ **begin****if** $p = 1$ **then return** 1; $max_A \leftarrow |A_{11}|;$ $min_A \leftarrow |A_{11}|;$ **for** $i = 1$ **to** n **do** $max_A = \max(max_A, |A_{ii}|);$ $min_A = \min(min_A, |A_{ii}|);$ **end****return** $\frac{max_A}{min_A}$ **end**

Gói lặp - Đánh giá tiên nghiệm**Function** iterate($X_0, B, T, q, \lambda, p, \varepsilon$)**Input:** Ma trận xấp xỉ đầu X_0, B, T , hệ số q, λ , giá trị kiểm tra p và sai số ε **Output:** X^* là ma trận nghịch đảo theo đánh giá tiên nghiệm**begin** $qk \leftarrow 1, X \leftarrow X_0;$ $predecessor_norm \leftarrow \text{getNorm}((BX_0 + T) - X_0, p);$ **while** $\frac{\lambda * qk * predecessor_norm}{1 - q} > \varepsilon$ **do** $X \leftarrow BX + T;$ $qk \leftarrow qk * q;$ **end****return** X **end****Gói lặp - Đánh giá hậu nghiệm****Function** iterate($X_0, B, T, q, \lambda, p, \varepsilon$)**Input:** Ma trận xấp xỉ đầu X_0, B, T , hệ số q, λ , giá trị kiểm tra p và sai số ε **Output:** X^* là ma trận nghịch đảo theo đánh giá hậu nghiệm**begin** $old_X \leftarrow X_0;$ $new_X \leftarrow BX_0 + T;$ **while** $\frac{\lambda * q * \text{getNorm}(new_X - old_X, p)}{1 - q} > \varepsilon$ **do** $old_X \leftarrow new_X;$ $new_X \leftarrow B * old_X + T;$ **end****return** new_X **end****Chương trình:** jacobi.py

4 Phương pháp Gauss - Seidel

Phương pháp Gauss-Seidel là một cải tiến theo ý tưởng của phương pháp lặp đơn và Jacobi, cho phép tính toán theo dòng của ma trận từ đó tiết kiệm bộ nhớ hơn các phương pháp thông thường. Do phần này (cùng với phương pháp lặp Jacobi) nằm trong phần kiến thức đã biết nên trong báo cáo chỉ áp dụng trực tiếp và không chứng minh các công thức. Phần này cũng không nêu điều kiện hội tụ bởi điều kiện hội tụ của phương pháp này hoàn toàn tương tự như phương pháp Jacobi.

4.1 Công thức lặp

Đặt $B = E - TA$ với $T = \text{diag}\{\frac{1}{A_{11}}, \frac{1}{A_{22}}, \dots, \frac{1}{A_{nn}}\}$ tương tự như trên. Như kiến thức đã biết trong phần lặp Gauss-Seidel, ta có công thức lặp sau[4]:

$$X_i^{(k+1)} = \sum_{j=1}^{i-1} B_{ij} X_j^{(k+1)} + \sum_{j=i+1}^n B_{ij} X_j^{(k)} + T_i \quad (9)$$

với mọi $i = \overline{1, n}$ và A_i là dòng i của ma trận A

4.2 Công thức sai số

Ta vẫn đặt $T = \text{diag}\{\frac{1}{A_{11}}, \frac{1}{A_{22}}, \dots, \frac{1}{A_{nn}}\}$ như trong phần 3.2. Ta xét 2 trường hợp:

(i) **Trường hợp chéo trội hàng:** Đặt $B = E - TA$, dễ thấy tồn tại một số q để:

$$q = \max_{1 \leq i \leq n} \frac{\sum_{j=i}^n |B_{ij}|}{1 - \sum_{j=1}^{i-1} |B_{ij}|} \leq \|B\|_{\infty} < 1$$

Do đó, ta có công thức sai số như sau:

$$\|X_k - X^*\|_{\infty} \leq \frac{q}{1 - q} \|X_k - X_{k-1}\|_{\infty} \quad (10)$$

$$\|X_k - X^*\|_{\infty} \leq \frac{q^k}{1 - q} \|X_1 - X_0\|_{\infty} \quad (11)$$

(ii) **Trường hợp chéo trội cột:** Đặt $B_1 = E - AT$, ta có:

$$q = \max_{1 \leq i \leq n} \frac{\sum_{j=1}^i |B_{1_{ji}}|}{1 - \sum_{j=i+1}^n |B_{1_{ji}}|} \leq \|B_1\|_1 < 1$$

$$S = \max_{1 \leq i \leq n} \sum_{j=i+1}^n |B_{1_{ji}}|$$

Do đó, ta có công thức sai số như sau[4]:

$$\|X_k - X^*\|_1 \leq \frac{q}{(1-S)(1-q)} \|X_k - X_{k-1}\|_1 \quad (12)$$

$$\|X_k - X^*\|_1 \leq \frac{q^k}{(1-S)(1-q)} \|X_1 - X_0\|_1 \quad (13)$$

4.3 Thuật toán và chương trình

Mã giả thuật toán chính:

Algorithm 3: Phương pháp Gauss-Seidel tìm ma trận nghịch đảo

Input: Ma trận A chéo trội, sai số ε , hệ số điều chỉnh ω

Output: Ma trận $X^* = A^{-1}$ là ma trận nghịch đảo

begin

Nhập A, ε ;

$p \leftarrow \text{checkDomination}(A)$;

$T \leftarrow \text{diag}\{\frac{1}{A_{11}}, \frac{1}{A_{22}}, \dots, \frac{1}{A_{nn}}\}$;

$B \leftarrow E - TA$;

$S \leftarrow \text{getSCoeff}(B, p)$;

$q \leftarrow \text{getqCoeff}(B, p)$;

$X^* \leftarrow \text{iterate}(X_0 \leftarrow A, B, T, S, q, p, \omega, \varepsilon)$;

Trả về X^* là ma trận nghịch đảo;

end

Gói lấy hệ số S**Function** getSCoeff(A, p)**Input:** Ma trận A , giá trị kiểm tra p **Output:** $S = 0$ nếu $p = 1$, $\max_{1 \leq i \leq n} \sum_{j=i+1}^n |A_{ji}|$ nếu $p = -1$ **begin** **if** $p = 1$ **then return** 0; $S \leftarrow 0$; **for** $i = 1$ **to** n **do** $S \leftarrow \max(S, \sum_{j=i+1}^n |A_{ji}|)$; **return** S **end****Gói lấy hệ số q****Function** getqCoeff(A, p)**Input:** Ma trận A , giá trị kiểm tra p **Output:** $q = \max_{1 \leq i \leq n} \frac{\sum_{j=i}^n |A_{ij}|}{1 - \sum_{j=1}^{i-1} |A_{ij}|}$ nếu $p = 1$, $\max_{1 \leq i \leq n} \frac{\sum_{j=1}^i |A_{ji}|}{1 - \sum_{j=i+1}^n |A_{ji}|}$ nếu $p = -1$ **begin** $q \leftarrow 0$; **for** $i = 1$ **to** n **do** $Q1 \leftarrow 0, Q2 \leftarrow 0$; **if** $p = 1$ **then** $Q1 \leftarrow \sum_{j=i}^n |A_{ij}|, Q2 \leftarrow \sum_{j=1}^{i-1} |A_{ij}|$; **else** $Q1 \leftarrow \sum_{j=1}^i |A_{ji}|, Q2 \leftarrow \sum_{j=i+1}^n |A_{ji}|$; $q \leftarrow \max(q, \frac{Q1}{1-Q2})$; **end** **return** q **end**

Gói lắp - Đánh giá tiên nghiệm**Function** iterate($X_0, B, T, S, q, p, \omega, \varepsilon$)**Input:** Ma trận xấp xỉ đầu X_0, B, T , hệ số S, q , giá trị kiểm tra p , hệ số điều chỉnh ω và sai số ε **Output:** X^* là ma trận nghịch đảo theo đánh giá tiên nghiệm**begin** $qk \leftarrow 1;$ $X \leftarrow X_0;$ $X_1 \leftarrow \text{nextIteration}(X_0, B, T, \omega);$ $\text{predecessor_norm} \leftarrow \text{getNorm}(X_1 - X_0, p);$ **while** $\frac{qk * \text{predecessor_norm}}{(1-q)*(1-S)} > \varepsilon$ **do** $X \leftarrow \text{nextIteration}(X, B, T, \omega);$ $qk \leftarrow qk * q;$ **end****return** X **end****Gói lắp - Đánh giá hậu nghiệm****Function** iterate($X_0, B, T, S, q, p, \omega, \varepsilon$)**Input:** Ma trận xấp xỉ đầu X_0, B, T , hệ số S, q , giá trị kiểm tra p , hệ số điều chỉnh ω và sai số ε **Output:** X^* là ma trận nghịch đảo theo đánh giá hậu nghiệm**begin** $\text{old_}X \leftarrow X_0;$ $\text{new_}X \leftarrow \text{nextIteration}(X_0, B, T, \omega);$ **while** $\frac{\lambda * q * \text{getNorm}(\text{new_}X - \text{old_}X, p)}{1-q} > \varepsilon$ **do** $\text{old_}X \leftarrow \text{new_}X;$ $\text{new_}X \leftarrow \text{nextIteration}(X, B, T, \omega);$ **end****return** $\text{new_}X$ **end**

Gói lấy ma trận tiếp theo thu được từ ma trận đưa vào theo công thức lặp Gauss-Seidel

Function nextIteration(*old_X*, *B*, *T*, ω)

Input: Ma trận *old_X*, *B*, *T*, và hệ số ω

Output: Đưa ra ma trận tiếp theo thu được từ ma trận đưa vào *old_X* theo công thức lặp Gauss-Seidel

begin

new_X \leftarrow Ma trận không cấp *n*;

for *i* = 1 **to** *n* **do**

$$new_X_i = \sum_{j=1}^{i-1} B_{ij} * new_X_j + \sum_{j=i+1}^n B_{ij} * old_X_j + T_i;$$

end

return $(1 - \omega) * old_X + \omega * new_X$

end

5 Phân tích, tổng kết các phương pháp và hệ thống ví dụ

5.1 Ưu, nhược điểm của các phương pháp

Ưu điểm chung

- So với các phương pháp giải đúng, các phương pháp xấp xỉ có tính chất "self-correcting"[2] - tự sửa lỗi, tức là sai số tính toán được sửa lại sau mỗi bước lặp.
- Trong một vài trường hợp, các phương pháp này hội tụ rất nhanh và có thời gian chạy nhanh hơn hẳn các phương pháp giải đúng.

Phương pháp Newton

Ưu điểm:

- Tốc độ hội tụ rất nhanh khi xấp xỉ đầu thỏa mãn.
- Dễ cài đặt, thuật toán đơn giản, dễ nhớ.

Nhược điểm: Rất khó tìm xấp xỉ đầu cho phương pháp này do yêu cầu của hệ số co q .

Phương pháp lặp Jacobi và lặp Gauss-Seidel

Ưu điểm:

- Có thể chọn xấp xỉ đầu bất kỳ
- Xấp xỉ đầu ảnh hưởng lớn đến hệ số co nên ta có thể điều chỉnh tốc độ hội tụ bằng xấp xỉ đầu.

Nhược điểm:

- Yêu cầu ma trận phải chéo trội.
- Thuật toán phức tạp, ít dùng trong thực tế khi lấy nghịch đảo ma trận.

5.2 Chọn xấp xỉ đầu cho các phương pháp

Trong báo cáo này, nhóm đề xuất 2 cách chọn xấp xỉ đầu cho các phương pháp nêu trên như sau:

- (i) **Sử dụng kết quả của các phương pháp giải đúng:** Do tính self-correcting của các phương pháp giải gần đúng, chúng ta có thể "chữa lại" kết quả của các phương pháp giải

đúng như Gauss-Jordan, Cholesky,... bằng cách lấy kết quả của các phương pháp giải đúng làm đầu vào và xấp xỉ đầu cho các phương pháp giải gần đúng để cải thiện độ chính xác về tính toán cho các phương pháp giải đúng. Cách chọn xấp xỉ đầu này phù hợp với cả ba phương pháp Newton, Gauss-Seidel và Jacobi.

- (ii) **Một cách chọn xấp xỉ đầu cho phương pháp Newton:** Năm 1986, Victor Pan cùng với John H. Reif [3] đề xuất một phương pháp xấp xỉ đầu vào cho phương pháp Newton. Theo đó, ma trận xấp xỉ ban đầu sẽ được tính từ ma trận đề bài A như sau:

$$X_0 = \frac{A^T}{\|A\|_1 \|A\|_\infty}$$

Trên cơ sở của mục (ii), nhóm đề xuất gói tìm xấp xỉ đầu của phương pháp Newton như sau:

Function getFirstApprox(A)

Input: Ma trận A

Output: Ma trận X_0 thỏa mãn điều kiện hội tụ

begin

$t1 \leftarrow \|A\|_1$;

$t2 \leftarrow \|A\|_\infty$;

$X_0 \leftarrow (\frac{A}{t1*t2})^T$;

while $\|E - AX_0\| \geq 1$ **do**

$X_0 \leftarrow X_0(2E - AX_0)$;

end

return X_0

end

5.3 Hệ thống ví dụ

Ví dụ 1: Nghịch đảo ma trận sau:

$$\begin{bmatrix} 15 & 1 & 4 & -8 \\ 2 & 16 & 9 & 3 \\ 1 & -9 & 12 & 1 \\ 1 & 0 & 5 & 23 \end{bmatrix}$$

Đây là ma trận chéo trội hàng, thỏa mãn điều kiện hội tụ của phương pháp Jacobi cũng như Gauss-Seidel. Khi chạy các thuật toán với ma trận này với sai số 10^{-15} , ta được kết quả kèm ma trận tích của kết quả thu được với ma trận ban đầu như sau:

```

D:\bulth4nh\OneDrive - Hanoi University of Science and Technology\Courses\TALENTED-K64MI\MI3040\report-code\Newton>python newton_fromfile.py
Đang tải ma trận....
Tải ma trận thành công.
Nhập sai số eps: 1e-15

-----
Ma trận A:
[[15.  1.  4. -8.]
 [ 2. 16.  9.  3.]
 [ 1. -9. 12.  1.]
 [ 1.  0.  5. 23.]]

-----
Phương pháp Newton kết thúc sau 6 bước lặp
Ma trận nghịch đảo của A:
[[ 0.0685159 -0.01648494 -0.02169369  0.02692503]
 [-0.00364951  0.04556805 -0.03050675 -0.00588667]
 [-0.00834981  0.03614485  0.06332976 -0.0103723 ]
 [-0.00116378 -0.00714084 -0.01282413  0.04456245]]

-----
Kiểm tra nhân ngược:
[[ 1.00000000e+00 -2.77555756e-17 -1.11022302e-16  3.33066907e-16]
 [ 2.86229374e-17  1.00000000e+00  4.16333634e-17 -2.77555756e-17]
 [ 2.60208521e-17  1.11022302e-16  1.00000000e+00 -5.55111512e-17]
 [ 1.38777878e-17 -1.38777878e-17  5.55111512e-17  1.00000000e+00]]

D:\bulth4nh\OneDrive - Hanoi University of Science and Technology\Courses\TALENTED-K64MI\MI3040\report-code\Newton>

```

Phương pháp Newton với xấp xỉ đầu theo mục 5.2(ii)

```

Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\bu1th4nh\OneDrive - Hanoi University of Science and Technology\Co
urses\TALENTED-K64MI\MI3040\report-code\Jacobi>python jacobi_fromfil
e.py
Đang tải ma trận.....
Tải ma trận thành công.
Nhập sai số eps: 1e-15

-----
Ma trận A:
[[15.  1.  4. -8.]
 [ 2. 16.  9.  3.]
 [ 1. -9. 12.  1.]
 [ 1.  0.  5. 23.]]

-----
A chéo trội hàng
Phương pháp Jacobi đánh giá tiên nghiệm kết thúc sau 468 bước lặp
Ma trận nghịch đảo của A theo PP Jacobi tiên nghiệm:
[[ 0.0685159 -0.01648494 -0.02169369  0.02692503]
 [-0.00364951  0.04556805 -0.03050675 -0.00588667]
 [-0.00834981  0.03614485  0.06332976 -0.0103723 ]
 [-0.00116378 -0.00714084 -0.01282413  0.04456245]]

-----
Kiểm tra nhân ngược:
[[ 1.00000000e+00  0.00000000e+00  5.55111512e-17  0.00000000e+00]
 [ 6.93889390e-18  1.00000000e+00  3.81639165e-17 -2.77555756e-17]
 [ 0.00000000e+00  1.11022302e-16  1.00000000e+00  5.55111512e-17]
 [ 0.00000000e+00 -1.38777878e-17  0.00000000e+00  1.00000000e+00]]

-----
A chéo trội hàng
Phương pháp Jacobi đánh giá hậu nghiệm kết thúc sau 93 bước lặp
Ma trận nghịch đảo của A theo PP Jacobi hậu nghiệm:
[[ 0.0685159 -0.01648494 -0.02169369  0.02692503]
 [-0.00364951  0.04556805 -0.03050675 -0.00588667]
 [-0.00834981  0.03614485  0.06332976 -0.0103723 ]
 [-0.00116378 -0.00714084 -0.01282413  0.04456245]]

-----
Kiểm tra nhân ngược:
[[ 1.00000000e+00  1.38777878e-16  1.66533454e-16  1.11022302e-16]
 [-4.68375339e-17  1.00000000e+00  8.32667268e-17  0.00000000e+00]
 [ 1.66533454e-16  2.22044605e-16  1.00000000e+00  3.33066907e-16]
 [-2.08166817e-17  1.11022302e-16  0.00000000e+00  1.00000000e+00]]

D:\bu1th4nh\OneDrive - Hanoi University of Science and Technology\Co
urses\TALENTED-K64MI\MI3040\report-code\Jacobi>

```

Phương pháp Jacobi, xấp xỉ đầu là ma trận A

```

Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\bu1th4nh\OneDrive - Hanoi University of Science and Technology\Co
urses\TALENTED-K64MI\MI3040\report-code\Gauss-Seidel>python gauss_se
idel_fromfile.py
Đang tải ma trận.....
Tải ma trận thành công.
Nhập sai số eps: 1e-15

-----
Ma trận A:
[[15.  1.  4. -8.]
 [ 2. 16.  9.  3.]
 [ 1. -9. 12.  1.]
 [ 1.  0.  5. 23.]]

-----
A chéo trội hàng
Phương pháp Gauss-Seidel đánh giá tiên nghiệm kết thúc sau 427 bước
lập
Ma trận nghịch đảo của A theo PP Gauss-Seidel tiên nghiệm:
[[ 0.0685159 -0.01648494 -0.02169369  0.02692503]
 [-0.00364951  0.04556805 -0.03050675 -0.00588667]
 [-0.00834981  0.03614485  0.06332976 -0.0103723 ]
 [-0.00116378 -0.00714084 -0.01282413  0.04456245]]

-----
Kiểm tra nhân ngược:
[[ 1.00000000e+00  0.00000000e+00  5.55111512e-17  0.00000000e+00]
 [ 7.80625564e-18  1.00000000e+00  4.16333634e-17  0.00000000e+00]
 [ 0.00000000e+00  1.11022302e-16  1.00000000e+00  5.55111512e-17]
 [ 0.00000000e+00 -1.38777878e-17  0.00000000e+00  1.00000000e+00]]

-----
A chéo trội hàng
Phương pháp Gauss-Seidel đánh giá hậu nghiệm kết thúc sau 32 bước lậ
p
Ma trận nghịch đảo của A theo PP Gauss-Seidel hậu nghiệm:
[[ 0.0685159 -0.01648494 -0.02169369  0.02692503]
 [-0.00364951  0.04556805 -0.03050675 -0.00588667]
 [-0.00834981  0.03614485  0.06332976 -0.0103723 ]
 [-0.00116378 -0.00714084 -0.01282413  0.04456245]]

-----
Kiểm tra nhân ngược:
[[ 1.00000000e+00  2.77555756e-17  0.00000000e+00  0.00000000e+00]
 [-1.47451495e-17  1.00000000e+00 -2.08166817e-17 -5.55111512e-17]
 [-3.46944695e-18  1.11022302e-16  1.00000000e+00 -2.77555756e-17]
 [ 0.00000000e+00 -1.38777878e-17  0.00000000e+00  1.00000000e+00]]

D:\bu1th4nh\OneDrive - Hanoi University of Science and Technology\Co
urses\TALENTED-K64MI\MI3040\report-code\Gauss-Seidel>

```

Phương pháp Gauss-Seidel, xấp xỉ đầu là ma trận A

Có thể thấy với xấp xỉ đầu là ma trận A, các phương pháp Jacobi và Gauss-Seidel có số bước lặp khá lớn do hệ số co phụ thuộc xấp xỉ đầu. Đây cũng là chủ đề nhóm sẽ cải tiến trong tương lai.

Ví dụ 2: Nghịch đảo ma trận sau:

$$\begin{bmatrix} 0 & 5 & 0 & 0 \\ 0 & 0.00013 & 40 & 0 \\ 0 & 0 & 0.152 & 0.00067 \\ 0.00001 & 0 & 0 & 0 \end{bmatrix}$$

Đây là ma trận không chéo trội và có các định thức con chính bằng 0, tức là không thể chạy được với phương pháp viền quanh (chưa cải tiến) và các phương pháp Jacobi cũng như Gauss-Seidel. Khi chạy thuật toán Newton kèm cải tiến xấp xỉ đầu với ma trận này với sai số 10^{-15} , ta được kết quả kèm ma trận tích của kết quả thu được với ma trận ban đầu như sau:

```

(c) 2020 Microsoft Corporation. All rights reserved.

D:\bui\th4nh\OneDrive - Hanoi University of Science and Technology\Courses\TALENTED-K64MI\MI3040\report-code\Newton>python newton
fromfile.py
Đang tải ma trận....
Tải ma trận thành công.
Nhập sai số eps: 1e-15

-----
Ma trận A:
[[0.00e+00 5.00e+00 0.00e+00 0.00e+00]
 [0.00e+00 1.30e-04 4.00e+01 0.00e+00]
 [0.00e+00 0.00e+00 1.52e-01 6.70e-04]
 [1.00e-05 0.00e+00 0.00e+00 0.00e+00]]

-----
Phương pháp Newton kết thúc sau 47 bước lặp
Ma trận nghịch đảo của A:
[[ 0.00000000e+000  0.00000000e+000  0.00000000e+000  1.00000000e+005]
 [ 2.00000000e-001 -6.29018435e-235 -1.61028719e-232  0.00000000e+000]
 [-6.50000000e-007  2.50000000e-002  6.14275815e-238  0.00000000e+000]
 [ 1.47462687e-004 -5.67164179e+000  1.49253731e+003  0.00000000e+000]]

-----
Kiểm tra nhân ngược:
[[ 1.00000000e+000  0.00000000e+000  0.00000000e+000  0.00000000e+000]
 [ 0.00000000e+000  1.00000000e+000 -4.96371027e-233 -1.07889242e-235]
 [ 0.00000000e+000  4.23516474e-022  1.00000000e+000  4.11564796e-241]
 [ 0.00000000e+000  0.00000000e+000 -2.84217094e-014  1.00000000e+000]]

D:\bui\th4nh\OneDrive - Hanoi University of Science and Technology\Courses\TALENTED-K64MI\MI3040\report-code\Newton>

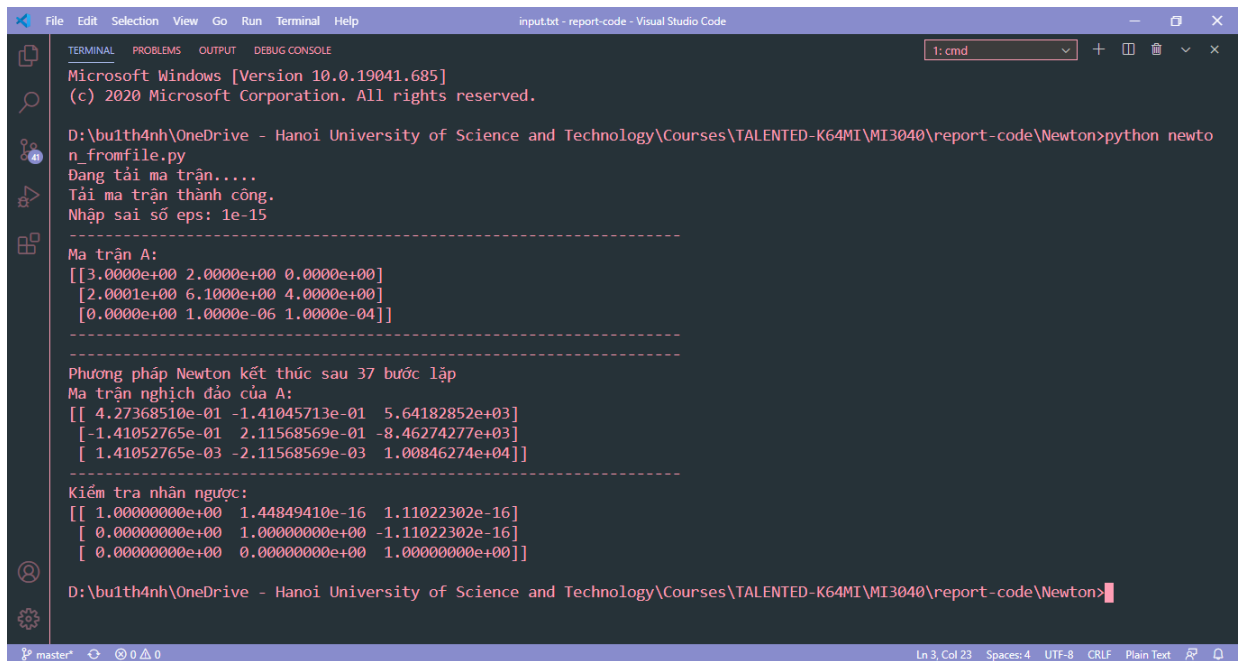
```

Ví dụ 3: Nghịch đảo ma trận sau: (*Ma trận gần suy biến*)

$$\begin{bmatrix} 3 & 2 & 7 \\ 2.0001 & 6.1 & 4 \\ 0 & 0.000001 & 0.001 \end{bmatrix}$$

Đây là ma trận chéo trội hàng và là ma trận gần suy biến, thỏa mãn điều kiện hội tụ của phương pháp Jacobi cũng như Gauss-Seidel. Khi chạy các thuật toán với ma trận này với sai số 10^{-15} ,

ta được kết quả kèm ma trận tích của kết quả thu được với ma trận ban đầu như sau:



```
Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\bu1th4nh\OneDrive - Hanoi University of Science and Technology\Courses\TALENTED-K64MI\MI3040\report-code\Newton>python newton_fromfile.py
Đang tải ma trận.....
Tải ma trận thành công.
Nhập sai số eps: 1e-15

-----
Ma trận A:
[[3.0000e+00 2.0000e+00 0.0000e+00]
 [2.0001e+00 6.1000e+00 4.0000e+00]
 [0.0000e+00 1.0000e-06 1.0000e-04]]
-----

Phương pháp Newton kết thúc sau 37 bước lặp
Ma trận nghịch đảo của A:
[[ 4.27368510e-01 -1.41045713e-01  5.64182852e+03]
 [-1.41052765e-01  2.11568569e-01 -8.46274277e+03]
 [ 1.41052765e-03 -2.11568569e-03  1.00846274e+04]]
-----

Kiểm tra nhân ngược:
[[ 1.00000000e+00  1.44849410e-16  1.11022302e-16]
 [ 0.00000000e+00  1.00000000e+00 -1.11022302e-16]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]]

D:\bu1th4nh\OneDrive - Hanoi University of Science and Technology\Courses\TALENTED-K64MI\MI3040\report-code\Newton>
```

Phương pháp Newton với xấp xỉ đầu theo mục 5.2(ii)


```

Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\bu1th4nh\OneDrive - Hanoi University of Science and Technology\Co
urses\TALENTED-K64MI\MI3040\report-code\Jacobi>python jacobi_fromfil
e.py
Đang tải ma trận.....
Tải ma trận thành công.
Nhập sai số eps: 1e-15

-----
Ma trận A:
[[3.0000e+00 2.0000e+00 0.0000e+00]
 [2.0001e+00 6.1000e+00 4.0000e+00]
 [0.0000e+00 1.0000e-06 1.0000e-04]]

-----
A chéo trội hàng
Phương pháp Jacobi đánh giá tiên nghiệm kết thúc sau 2899 bước lặp
Ma trận nghịch đảo của A theo PP Jacobi tiên nghiệm:
[[ 4.27368510e-01 -1.41045713e-01 5.64182852e+03]
 [-1.41052765e-01 2.11568569e-01 -8.46274277e+03]
 [ 1.41052765e-03 -2.11568569e-03 1.00846274e+04]]

-----
Kiểm tra nhân ngược:
[[ 1.00000000e+00 -7.71951947e-17 1.11022302e-16]
 [ 5.55111512e-17 1.00000000e+00 -1.11022302e-16]
 [-8.67361738e-19 -1.73472348e-18 1.00000000e+00]]

-----
A chéo trội hàng
Phương pháp Jacobi đánh giá hậu nghiệm kết thúc sau 55 bước lặp
Ma trận nghịch đảo của A theo PP Jacobi hậu nghiệm:
[[ 4.27368510e-01 -1.41045713e-01 5.64182852e+03]
 [-1.41052765e-01 2.11568569e-01 -8.46274277e+03]
 [ 1.41052765e-03 -2.11568569e-03 1.00846274e+04]]

-----
Kiểm tra nhân ngược:
[[ 1.00000000e+00 -7.71951947e-17 1.11022302e-16]
 [ 5.55111512e-17 1.00000000e+00 -1.11022302e-16]
 [-8.67361738e-19 -1.73472348e-18 1.00000000e+00]]

D:\bu1th4nh\OneDrive - Hanoi University of Science and Technology\Co
urses\TALENTED-K64MI\MI3040\report-code\Jacobi>

```

Phương pháp Jacobi, xấp xỉ đầu là ma trận A

```

Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\bu1th4nh\OneDrive - Hanoi University of Science and Technology\Co
urses\TALENTED-K64MI\MI3040\report-code\Gauss-Seidel>python gauss_se
idel_fromfile.py
Đang tải ma trận.....
Tải ma trận thành công.
Nhập sai số eps: 1e-15

-----
Ma trận A:
[[3.0000e+00 2.0000e+00 0.0000e+00]
 [2.0001e+00 6.1000e+00 4.0000e+00]
 [0.0000e+00 1.0000e-06 1.0000e-04]]

-----
A chéo trội hàng
Phương pháp Gauss-Seidel đánh giá tiên nghiệm kết thúc sau 2899 bước
lặp
Ma trận nghịch đảo của A theo PP Gauss-Seidel tiên nghiệm:
[[ 4.27368510e-01 -1.41045713e-01 5.64182852e+03]
 [-1.41052765e-01 2.11568569e-01 -8.46274277e+03]
 [ 1.41052765e-03 -2.11568569e-03 1.00846274e+04]]

-----
Kiểm tra nhân ngược:
[[ 1.00000000e+00 -7.71951947e-17 1.11022302e-16]
 [ 5.55111512e-17 1.00000000e+00 -1.11022302e-16]
 [-8.67361738e-19 -1.73472348e-18 1.00000000e+00]]

-----
A chéo trội hàng
Phương pháp Gauss-Seidel đánh giá hậu nghiệm kết thúc sau 28 bước lặ
p
Ma trận nghịch đảo của A theo PP Gauss-Seidel hậu nghiệm:
[[ 4.27368510e-01 -1.41045713e-01 5.64182852e+03]
 [-1.41052765e-01 2.11568569e-01 -8.46274277e+03]
 [ 1.41052765e-03 -2.11568569e-03 1.00846274e+04]]

-----
Kiểm tra nhân ngược:
[[ 1.00000000e+00 -7.71951947e-17 1.11022302e-16]
 [ 5.55111512e-17 1.00000000e+00 -1.11022302e-16]
 [-8.67361738e-19 -1.73472348e-18 1.00000000e+00]]

D:\bu1th4nh\OneDrive - Hanoi University of Science and Technology\Co
urses\TALENTED-K64MI\MI3040\report-code\Gauss-Seidel>

```

Phương pháp Gauss-Seidel, xấp xỉ đầu là ma trận A

Với hai phương pháp Jacobi và Gauss-Seidel, ví dụ này cho thấy sự khác biệt rõ rệt về số bước lặp giữa hai phương pháp tiên nghiệm và hậu nghiệm. Ở cả 2 phương pháp, cách đánh giá tiên nghiệm có số lần lặp gần 2900 bước, trong khi cách đánh giá hậu nghiệm có số lần lặp chưa đến 60 mà vẫn đảm bảo kết quả tương tự cách đánh giá tiên nghiệm. Ngoài ra, cả 3 phương pháp đều cho thấy sự ổn định khi ma trận có trạng thái gần suy biến.

5.4 Ứng dụng

Nghịch đảo ma trận có rất nhiều ứng dụng trong thực tế. Chẳng hạn, có thể kể đến các ứng dụng như sau

- **Đồ họa máy tính:** Theo dõi một đối tượng trong không gian 3 chiều, có ứng dụng rất lớn trong ngành công nghiệp trò chơi điện tử, VFX và các ứng dụng VR,AR.
- **Kỹ thuật điện, điện tử:** Giải các hệ thống mạch điện
- **An toàn thông tin:** Mã hóa, giải mã thông tin

Đặc biệt, phương pháp Newton còn có thể sử dụng để tìm nghiệm của phương trình ma trận với ma trận hệ số không chéo trội - nhược điểm của các phương pháp lặp Jacobi, Gauss-Seidel để giải phương trình ma trận $AX = B$.

A Các chương trình được sử dụng

Các chương trình trong báo cáo và slide được lưu tại các liên kết này: <https://github.com/bulth4nh/TALENTED-K64MI/blob/master/MI3040/report-code/>

B Mã nguồn báo cáo và bài trình chiếu

Mã nguồn báo cáo và slide được lưu tại đây: <https://github.com/bulth4nh/TALENTED-K64MI/tree/master/MI3040/report>

C Hướng dẫn sử dụng chương trình

C.1 Lưu ý trước khi sử dụng

- Phần thuật toán chính được đóng gói trong các file thư viện, có mẫu là `lib_*.py`. Các file này cung cấp các gói để chạy thuật toán thông qua các file chương trình chính `interface_*.py` với `*` là tên phương pháp. Do đó, file thư viện phải để cùng thư mục với file chương trình chính.
- Các chương trình nhập đầu vào từ file `input.txt` và trả kết quả qua đầu ra chuẩn (standard output)
- Các chương trình và thuật toán có sử dụng thư viện `numpy` và `scipy` và cần cài đặt các thư viện này trước khi chạy bằng lệnh sau:

```
pip install numpy
pip install scipy
```

C.2 Hướng dẫn sử dụng chi tiết

Phương pháp Newton

Các file trong thư mục: `Newton`

- **Chương trình chính:** `interface_newton.py`
- **Gói thuật toán độc lập:** `lib_newton.py`
- **File dữ liệu đầu vào:** `input.txt`

Các bước sử dụng

- **Bước 1:** Nhập ma trận vào file `input.txt`
- **Bước 2:** Chạy file `interface_newton.py`. Chương trình sẽ tự động dò kích cỡ ma trận dựa theo ma trận người dùng nhập vào và báo lỗi nếu ma trận nhập vào không hợp lệ.
- **Bước 3:** Nhập sai số, sau đó chương trình sẽ đưa ra số lần lặp, ma trận nghịch đảo và kết quả nhân ngược. Trường hợp đầu vào không hợp lệ, chương trình sẽ báo lỗi và đưa ra ma trận NaN

Phương pháp Jacobi

Các file trong thư mục: `Jacobi`

- **Chương trình chính:** `interface_jacobi.py`
- **Gói thuật toán độc lập:** `lib_jacobi.py`
- **File dữ liệu đầu vào:** `input.txt`

Các bước sử dụng

- **Bước 1:** Nhập ma trận vào file `input.txt`
- **Bước 2:** Chạy file `interface_jacobi.py`. Chương trình sẽ tự động dò kích cỡ ma trận dựa theo ma trận người dùng nhập vào và báo lỗi nếu ma trận nhập vào không hợp lệ.
- **Bước 3:** Nhập sai số, sau đó chương trình sẽ đưa ra kiểu ma trận, số lần lặp, ma trận nghịch đảo và kết quả nhân ngược với 2 cách đánh giá tiên nghiệm và hậu nghiệm. Trường hợp đầu vào không hợp lệ, chương trình sẽ báo lỗi và đưa ra ma trận NaN

Phương pháp Gauss-Seidel

Các file trong thư mục: `Gauss-Seidel`

- **Chương trình chính:** `interface_gauss_seidel.py`
- **Gói thuật toán độc lập:** `lib_gauss_seidel.py`
- **File dữ liệu đầu vào:** `input.txt`

Các bước sử dụng

- **Bước 1:** Nhập ma trận vào file `input.txt`
- **Bước 2:** Chạy file `interface_gauss_seidel.py`. Chương trình sẽ tự động dò kích

cỡ ma trận dựa theo ma trận người dùng nhập vào và báo lỗi nếu ma trận nhập vào không hợp lệ.

- **Bước 3:** Nhập sai số, sau đó chương trình sẽ đưa ra kiểu ma trận, số lần lặp, ma trận nghịch đảo và kết quả nhân ngược với 2 cách đánh giá tiên nghiệm và hậu nghiệm. Trường hợp đầu vào không hợp lệ, chương trình sẽ báo lỗi và đưa ra ma trận NaN

Tài liệu tham khảo

- [1] Richard L. Burden J. Douglas Faires. *Numerical Methods, 4th Edition*. Brooks / Cole, Cengage Learning, 2013, 2003, 1998.
- [2] The Pennsylvania State University Jaan Kiusalaas. *Numerical Methods in Engineering With MATLAB*. Cambridge University Press, 2005.
- [3] Thomas E. Phipps Jr. “The inversion of large matrices: The Pan and Reif algorithm provides a solution”. In: *Byte Magazine* 11.04 (4-1986).
- [4] Lê Trọng Vinh. *Giáo trình Giải tích số*. NXB Khoa học và kỹ thuật, 2007.
- [5] Hà Thị Ngọc Yến. *Phương pháp lặp Jacobi*. 2019.