ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

TIỂU LUẬN

Phương pháp số cho đại số tuyến tính
Một số phương pháp lặp giải hệ phương trình tuyến tính

Đại học Quốc gia Hà Nội
Đại học Khoa học Tự nhiên
Khoa Toán cơ tin

Giảng viên: Phan Trung Hiếu

Học viên: Nguyễn Mạnh Linh

Mục lục

1 Tổng quan 1
2 GMRES 3
2.1 Thuật giải Arnoldi 3
2.2 GMRES 4
3 MINRES 6
3.1 Giải thuật Lanczos 6
3.2 MINRES 7
3.3 Ma trận quay và phân tích QR 8
3.4 Giải thuật MINRES chi tiết 9
3.5 MINES code 10

Tóm tắt

Tiểu luận này trình bày một số phương pháp lặp giải hệ phương trình tuyến tính lớn bằng cách sử dụng kĩ thuật tối thiểu hóa phần dư tại mỗi bước lặp (minimal residual (MR)). Ma trận của hệ tuyến tính được xem xét ở dạng tổng quát (sử dụng phương pháp GMRES (Generalized minimal residual method)) và dạng đối xứng xác định dương (sử dụng phương pháp MINRES (Minimal residual method)). Tiệu luận tập trung vào giải quyết bài toán ma trận đối xứng như một trường hợp riêng (lấy ý tưởng từ việc giải bài toán tổng quát). Kỹ thuật MINRES được sử dụng như 1 cách hiệu quả để giảm thiểu số bước lặp cũng như không gian lưu trữ khi so sánh với GMRES. Không gian con Krylov được xem xét để tìm nghiệm gần đúng của hệ. Bài này cũng giới thiệu về phương pháp lặp Arnoldi và Lanczos. Cuối cùng ta sử dụng ngôn ngữ lập trình Python để minh họa các thuật giải và so sánh hiệu năng của chúng.

1 Tổng quan

Xem xét hệ phương trình tuyến tính $Ax = b$ trong đó $A \in \mathbb{C}^{m\times m}$ và $b \in \mathbb{C}^m$.

Nghiệm chính xác của hệ $x_* = A^{-1}b$. Dương nhiên việc tính ma trận nghịch đảo A^{-1} hoặc dùng phương pháp khử Gauss là rất tốn chi phí. Ta biết rằng độ phức tạp của thuật toán khi sử dụng phương pháp khủ Gauss là $O(m^3)$. Để có cái nhìn trực quan về độ phức tạp này, ta có thể xem qua bảng sau

Năm	m	m^3
1950	20	2×10^3
1965	200	2×10^6
1980	2000	2×10^9

Khi kích thước của ma trận tăng lên, số phép tính cũng tăng lên rất nhanh. Cùng với sự phát triển của phần cứng chúng ta cũng tính toán được với những ma trận với kích thước lớn hơn rất nhiều so với những năm 1950. Nhưng ta cũng thấy rằng, khi kích thước của ma trận là hàng nghìn thì số phép tính đã là hàng tỉ. Với những bài toán lớn như mô phỏng thời tiết hay thiên hà trong thiên văn học, phần cứng máy tính không thể đuổi kịp để tính toán cũng như lưu trữ.

Điều này đòi hỏi chúng ta cần tìm ra những phương pháp tốt hơn nhằm giảm độ phức tạp của thuật toán. Tiểu luận này giới thiệu 1 vài phương pháp với độ phức tạp $O(m^2)$.

Phương pháp lặp cho phép chúng ta tiến gần đến nghiệm chính xác của phương trình qua mỗi bước lặp. Đến một lúc nào đó phần dư là đủ chấp nhận được, ta có nghiệm gần đúng của phương trình.

Gọi x_n là nghiệm gần đúng của phương trình tại bước lặp thứ n. Ta định nghĩa phần dư

$$r_n = b - Ax_n \tag{1.1}$$

Khi $r_n < \epsilon$ (là một sai số nhỏ chấp nhận được) ta dừng lại và thu được nghiệm gần đúng x_n

Trong bài này chúng ta sẽ tìm nghiệm $x_n \in \kappa_n$, trong đó κ_n là không gian con Krylov

Không gian con Krylov

Cho ma trận $A \in \mathbb{C}^{m\times m}$ và vector $b \in \mathbb{C}^m$.

Dãy Krylov được định nghĩa là tập các vectors $b, Ab, A^2b, ...$

Không gian con Krylov là không gian được sinh bởi các vectors này

Ma trận Krylov

$$K_n = \left[\begin{array}{c|c|c|c|c} & & & & \\ b & Ab & A^2b & ... & A^{n-1}b \\ & & & & \end{array} \right] \tag{1.2}$$

Trở lại bài toán, ta muốn tìm nghiệm x_n trong không gian κ_n hay nói các khác $x_n \in range(K_n)$. Ta có thể sử dụng phân tích QR cho ma trận Krylov và đặt $x_n = Q_n y$ trong đó Q_n là ma trận unitary.

Bài toán cực tiểu hóa r_n đưa về việc cực tiểu hóa $\|AQ_n y - b\|$

Chúng ta sẽ tiếp cận bài toán với việc giải tổng quát với phương pháp GMRES và sử giải bài toán ma trận đối xứng như một trường hợp riêng với phương pháp MINRES. Trước hết, thuật giải Arnoldi sẽ được giới thiệu sau đây.

2 GMRES

2.1 Thuật giải Arnoldi

Phương pháp GMRES sử dụng quá trình Gram-Schmitz chỉnh sửa để thiết lập một cơ sở trực chuẩn cho không gian Krylov $span\{r_0, Ar_0, ..., A^k r_0\}$. Quá trình Gram-Schmitz được áp dụng cho không gian này được gọi là phương pháp *Arnoldi* Giải thuật này nhằm phân tích $A = QHQ^*$ hoặc $AQ = QH$, trong đó Q là ma trận trực chuẩn (unitary) còn H là ma trận Hessenberg[1].

Gọi Q_n (ma trận $m \times n$) là ma trận được tạo thành bởi n cột đầu tiên của ma trận Q

$$Q_n = \left[\begin{array}{c|c|c|c|c} & & & & \\ q_1 & q_2 & q_3 & ... & q_n \\ & & & & \end{array}\right] \tag{2.1}$$

Gọi $\tilde{H}_n$ là phần trên bên trái $(n+1) \times n$ của ma trận H. $\tilde{H}_n$ đương nhiên cũng là 1 ma trận Hessenberg.

$$\tilde{H}_n = \begin{bmatrix} h_{11} & & \cdots & h_{1n} \\ h_{21} & h_{22} & & \vdots \\ & \ddots & \ddots & \vdots \\ & & h_{n,n-1} & h_{nn} \\ & & & h_{n+1,n} \end{bmatrix} \tag{2.2}$$

Ta có phương trình:

$$AQ_n = Q_{n+1}\tilde{H}_n \tag{2.3}$$

Ta có thể viết riêng cho cột thứ n của phương trình như sau:

$$Aq_n = h_{1n}q_1 + ... + h_{nn}q_n + h_{n+1,n}q_{n+1} \tag{2.4}$$

Có nghĩa là q_{n+1} có thể tìm được từ n vectors $q_1, q_2, ..., q_n$. q_{n+1} được tính theo phương pháp lặp bằng giải thuật dưới đây (thuật giải Arnoldi)

Arnoldi Algorithm

[1] *Ma trận H vuông kích thước $n \times n$ được gọi là ma trận upper Hessenberg nếu $h_{ij} = 0$ với mọi i, j mà $i > j + 1$*

```
Given q_1 with ||q_1|| = 1
For j = 1, 2, ...
    q̃_{j+1} = A q_j
    For i = 1, ..., j
        h_{ij} = ⟨q̃_{j+1}, q_i⟩
        q̃_{j+1} ← q̃_{j+1} − h_{ij} q_i
    h_{j+1,j} = ||q̃_{j+1}||
    q_{j+1} = q̃_{j+1} / h_{j+1,j}
```

Trong đó $\langle v, u\rangle$ là tích vô hướng của 2 vectors v và u.

Thực chất, ta có thể chứng minh phương trình (2.3) bằng cách sử dụng quy nạp và tính chất của ma trận Hessenberg dựa vào cách chọn các vectors q_j và h_{ij} trong thuật giải Arnoldi ở trên (chi tiết trong phần phụ lục).

Chúng ta có thể tìm được ma trận Q với các ngôn ngữ lập trình bậc cao như MATLAB một cách tương đối dễ đàng. Ma trận A ở đây chỉ xuất hiện trong tích Aq_j và MATLAB cũng tích hợp sẵn chương trình nhân ma trận. Tuy nhiên trong bài này, tác giả sử dụng ngôn ngữ lập trình pyhthon cũng là một ngôn ngữ bậc cao hiện đại và có nhiều bộ thư viện giúp ta tính toán với ma trận tương đối dễ dàng và tường minh như *numpy* chẳng hạn.

2.2 GMRES

Trong phương pháp GMRES, nghiệm gần đúng x_k được chọn dưới dạng $x_k = x_0 + Q_k y_k$ với một vector y_k nào đó cần tìm. Nghĩa là x_k là x_0 cộng với một tổ hợp tuyến tính của các vectors cơ sở trực chuẩn cho không gian Krylov. Chúng ta sẽ tìm vector y_k sao cho phần dư $r_k = r_0 - AQ_k y_k$ có chuẩn-2 tối thiểu. Như vậy vector y phải thỏa mãn bài toán bình phương tối thiểu:

$$\begin{aligned}&\min_y \|r_0 - AQ_k y\| = \min_y \|r_0 - Q_{k+1} H_k y\| \\ &= \min_y \|Q_{k+1}(\beta e_1 - H_k y)\| = \min_y \|\beta e_1 - H_k y\|\end{aligned}$$

Trong đó, $\beta = \|r_0\|$, e_1 là vector đơn vị $(k+1)$ chiều $(1, 0, ..., 0)^T$.

Từ đây, ta có thể đưa ra những bước cơ bản để thực hiện giải nghiệm gần đúng x_k bằng phương pháp GMRES như sau:

1. Lấy một điểm bắt đầu, vector x_0, tính phần dư $r_0 = b - A_x 0$, đặt $q_1 = r_0/\|r_0\|$

2. Cho k chạy từ 1: $k = 1, 2, ...$: tính q_{k+1} và $h_{i,k}$ với $i = 1, 2, ..., k+1$ sử dụng giải thuật Arnoldi.

3. Tính $x_k = x_0 + Q_k y_k$ với y_k là nghiệm của bài toán bình phương tối thiểu $\min_y \|\beta e_1 - H_k y\|$

Việc giải bài toán GMRES tổng quát được mô tả ở trên là khá tốn công sức và thiếu thực tế vì khối lượng tính toán cũng như không gian lưu trữ khá lớn, đặc biệt khi kích thước của hệ là lớn. Ta nhận thấy số vòng lặp $\sim k^2$ tuy nhiên tại thời điểm này ta chưa nhận định được sau bao nhiêu bước k thì thu được nghiệm gần đúng chấp nhận được. Ta có thể dùng giải thuật GMRES(j) được định nghĩa đơn giản là *khởi động lại* GMRES qua mỗi j bước, sử dụng bước lặp gần nhất làm giá trị khởi tạo cho vòng tiếp theo. Chúng ta sẽ không bàn quá sâu về GMRES ở đây nữa vì trọng tâm phần này, tác giả muốn giới thiệu về ý tưởng chính của phương pháp và tìm ra 1 cách hiệu quả để giải bài toán ma trận đối xứng. Chi tiết cho bài toán bình phương tối thiểu (*least square*) cũng sẽ được trình bày ở phần sau.

3 MINRES

3.1 Giải thuật Lanczos

Khi ma trận A là một ma trận Hermintian (trong không gian thực ta có thể nói A là ma trận đối xứng), giải thuật Arnoldi được trình bày ở phần trước có thể được đơn giản hóa bằng giải thuật Lanczos. Các phương trình cho hệ số hơi khác một chút (nhưng vẫn tương đương về mặt toán học) thường được dùng cho trường hợp ma trận Hermintian.

Lanczos Algorithm (for Hermintian matrices A)

Given q_1 with $\|q_1\| = 1$, set $\beta_0 = 0$
For $j = 1, 2, ...$
$\tilde{q}_{j+1} = Aq_j - \beta_{j-1}q_{j-1}$
set $\alpha_j = \langle \tilde{q}_{j+1}, q_j \rangle$
$\tilde{q}_{j+1} \leftarrow \tilde{q}_{j+1} - \alpha_j q_j$
$\beta_j = \|\tilde{q}_{j+1}\|$
$q_{j+1} = \tilde{q}_{j+1}/\beta_j$

Một cách trực quan ta có thể thấy ngay rằng số vòng lặp đã được giảm đi một bậc so với giải thuật Arnoldi. Đây thực chất không phải điều "kì diệu" mà do ta đang xem xét bài toán trong trường hợp ma trận A là đối xứng dẫn đến việc ma trận Hessenberg H là "thưa" hơn rất nhiều so với trường hợp tổng quát.

Có thể thấy rằng các vectors được hình thành trong giải thuật trên là tương đương với các vectors trong giải thuật Arnoldi khi ma trận A là ma trận Hermintian. Ngoài ra chúng ta cũng cần chỉ ra rằng các vectors q_j cũng lập thành một cơ sở trực chuẩn cho không gian Krylov (sinh bởi A và q_1).

Ta cùng chứng minh khẳng định này bằng phương pháp quy nạp như sau.

Trước hết ta thấy rằng từ cách chọn vectors q và hệ số β, các vectors này nằm trong không gian Krylov và mỗi vector đều có chuẩn bằng 1.

Ta có:

$$\begin{aligned} \alpha_j &= \langle \tilde{q}_{j+1}, q_j \rangle \\ \leftrightarrow 0 &= \langle \tilde{q}_{j+1}, q_j \rangle - \alpha_j \langle q_j, q_j \rangle \\ &= \langle \tilde{q}_{j+1} - \alpha_j q_j, q_j \rangle = \beta_j \langle q_{j+1}, q_j \rangle \end{aligned}$$

Hay $\langle q_{j+1}, q_j \rangle = 0$. Giả sử rằng $\langle q_k, q_i \rangle = 0$ với mọi $i \neq k$ mà $k, i \leq j$. Ta có:

$$\begin{aligned}\langle \tilde{q}_{j+1}, q_{j-1}\rangle &= \langle Aq_j - \alpha_j q_j - \beta_{j-1}q_{j-1}, q_{j-1}\rangle = \langle Aq_j, q_{j-1}\rangle - \beta_{j-1} \\ &= \langle q_j, Aq_{j-1} - \beta_{j-1}\rangle \\ &= \langle q_j, \tilde{q}_j + \alpha_{j-1}q_{j-1} + \beta_{j-2}q_{j-2}\rangle - \beta_{j-1} = \langle q_j, \tilde{q}_j\rangle - \beta_{j-1} = 0\end{aligned}$$

Với $i < j-1$, Ta có (lưu ý rằng $A = A^T$):

$$\begin{aligned}\langle \tilde{q}_{j+1}, q_i\rangle &= \langle Aq_j - \alpha_j q_j - \beta_{j-1}q_{j-1}, q_i\rangle = \langle Aq_j, q_i\rangle \\ &= \langle q_j, Aq_i\rangle \\ &= \langle q_j, \tilde{q}_{i+1} + \alpha_i q_i + \beta_{i-1}q_{i-1}\rangle = 0\end{aligned}$$

Như vậy tập các vectors $(q_1, ..., q_{j+1})$ lập thành 1 cơ sở trực chuẩn của không gian Krylov $span(q_1, Aq_1, ..., A^j q_1)$.

Giải thuật Lanczos có thể viết gọn dưới dạng ma trận như sau:

$$AQ_k = Q_k T_k + \beta_k q_{k+1} e_k^T = Q_{k+1} T_{k+1,k} \tag{3.1}$$

Trong đó Q_k là ma trận $m \times k$ gồm k cột đầu tiên của các vectors cơ sở trực chuẩn $q_1, ..., q_k$, e_k là vectors đơn vị thứ k và T_k là ma trận Hermintian *3 đường chéo* (tridiagonal) với các hệ số như sau:

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix} \tag{3.2}$$

Ma trận $(k+1) \times k$ $T_{k+1,k}$ là ma trận với T_k là khối $k \times k$ phía trên và cột cuối cùng là $\beta_k e_k^T$.

3.2 MINRES

Như đã trình bày ở phần trên, ý tưởng chung cho GMRES hay MINRES là tìm cách xấp xỉ nghiệm x_k biểu diễn trong không gian Krylov với chuẩn-2 của phần dư là cực tiểu.

Nghĩa là nếu q_1 được chọn trong thuật giải Lanczos là r_0/β với $\beta = \|r_0\|$ thì nghiệm gần đúng sẽ có dạng $x_k = x_0 + Q_k y_k$. Trong đó y_k được tính toán để làm cực tiểu hóa phần dư. Nói cách khác, y_k là nghiệm của bài toán bình phương tối thiểu:

$$\begin{aligned}\min_y \|r_0 - AQ_k y\| &= \min_y \|r_0 - Q_{k+1}T_{k+1,k}y\| \\ &= \min_y \|Q_{k+1}(\beta e_1 - T_{k+1,k}y)\| \\ &= \min_y \|\beta e_1 - T_{k+1,k}y\|\end{aligned} \tag{3.3}$$

tương tự như thuật giải GMRES ở phần trước. Tuy nhiên phương trình này đơn giản hơn nhiều vì ma trận $T_{k+1,k}$ là rất thưa khi so sánh với ma trận H_k.

Giải thuật MINRES cũng không yêu cầu cần phải lưu trữ các vectors cơ sở trực chuẩn khi dùng Lanczos (so sánh với Arnoldi).

Tiếp theo, ta sẽ cùng tìm cách giải bài toán bình phương tối thiểu này để tìm nghiệm gần đúng x_k.

Gọi $R_{k\times k}$ là khối $(k \times k)$ phía trên của ma trận R trong phân tích QR của ma trận T ($T_{k+1,k} = F^H R$). Trong đó ma trận F^H là ma trận trực chuẩn (unitary) kích thước $(k+1) \times (k+1)$ và ma trận R là ma trận tam giác trên kích thước $(k+1) \times k$ với khối $k \times k$ và các hàng dưới đều bằng 0.

Đối với trường hợp đặc biệt này (ma trận $T_{k+1,k}$ - tridiagonal) thì $R_{k\times k}$ chỉ có 2 đường chéo khác 0. Ta định nghĩa $P_k \equiv (p0, ..., p_{k-1}) \equiv Q_k R_{k\times k}^{-1}$. Do đó, p_0 là tích của q_1 với các cột của ma trận P_k có thể được tính như sau:

$$p_{k-1} = \left(q_k - b_{k-2}^{(k-1)} p_{k-2} - b_{k-3}^{(k-1)} p_{k-3}\right) / b_{k-1}^{(k-1)} \tag{3.4}$$

Trong đó $b_{k-1}^{(k-l)}$ là phần tử $(k-l+1, k)$ của ma trận $R_{k\times k}$. Lưu ý rằng các phần tử $b_{k-1}^{(k-l)}$ là khác 0. Nghiệm gần đúng x_k được cập nhật từ x_{k-1} như sau:

$$x_k = x_0 + P_k \beta (Fe_1)_{k\times 1} = x_{k-1} + a_{k-1} p_{k-1} \tag{3.5}$$

Với a_{k-1} là phần tử thứ k của $\beta(Fe_1)$.

3.3 Ma trận quay và phân tích QR

Để tính các phần tử $b_{k-1}^{(k-l)}$ của ma trận $R_{k\times k}$, ta tiền hành phân tích QR ma trận $T_{k+1,k}$. Ở đây ta sẽ dùng 1 cách nhẹ nhàng hơn so với việc phân tích QR đầy đủ hoặc ít nhất là không phải lưu trữ cả ma trận R vì ta chỉ quan tâm tới 3 hàng cuối cùng của cột thứ k của ma trận R mà thôi.

Nếu cho biết được phân tích QR của ma trận $T_{k+1,k}$ ta có thể tính được phân tích QR cho ma trận tiếp theo $T_{k+2,k+1}$ một cách tiết kiệm công sức.

Θ_i được định nghĩa là một ma trận quay vector đơn vị e_i và e_{i+1} đi 1 góc θ_i

$$\Theta_i = \begin{bmatrix} I & & & \\ & c_i & s_i & \\ & -\bar{s_i} & c_i & \\ & & & I \end{bmatrix} \tag{3.6}$$

Trong đó $c_i \equiv cos(\theta_i)$ và $s_i \equiv sin(\theta_i)$. Kích thước của ma trận phụ thuộc vào thành phần đơn vị I thứ 2 và tùy thuộc vào bối cảnh cụ thể.

Trong trường hợp tổng quát khi phân tích QR cho 1 ma trận Hessenberg $H_{k+2,k+1}$ từ phân tích QR của ma trận $H_{k+1,k}$ trong bước trước đó, giả sử rằng các ma trận quay Θ_i với $i = 1, ..., k$ đã được tác động lên $H_{k+1,k}$, ta có:

$$(\Theta_k \Theta_{k-1} ... \Theta_1) H_{k+1,k} = R^{(k)} = \begin{bmatrix} z & z & \dots & z \\ & z & \dots & z \\ & & \ddots & \vdots \\ & & & z \\ 0 & 0 & \dots & 0 \end{bmatrix} \tag{3.7}$$

Trong đó kí hiệu z cho các thành phần khác 0

Để thu được $R^{(k+1)}$ hay là phần tam giác trên của $H_{k+2,k+1}$, đầu tiên ta tác động các ma trận quay trước đó vào $H_{k+2,k+1}$

$$(\Theta_k \Theta_{k-1} ... \Theta_1) H_{k+2,k+1} = R^{(k+1)} = \begin{bmatrix} z & z & \dots & z & z \\ & z & \dots & z & z \\ & & \ddots & \vdots & \vdots \\ & & & z & z \\ 0 & 0 & \dots & 0 & d \\ 0 & 0 & \dots & 0 & h \end{bmatrix} \tag{3.8}$$

Trong đó phần từ $(k+2, k+1)$ (kí hiệu h) trong ma trận là $h_{k+2,k+1}$ không bị ảnh hưởng với tác động quay này. Tác động Θ_{k+1} trong bước tiếp theo được chọn để triệt tiêu phần tử này bằng cách chọn $c_{k+1} = |d|/\sqrt{|d|^2 + |h|^2}$, $\bar{s}_{k+1} = c_{k+1} h/d$ nếu $d \neq 0$ và $c_{k+1} = 0$, $s_{k+1} = 1$ nếu $d = 0$.

3.4 Giải thuật MINRES chi tiết

Ta có giải thuật MINRES được viết chi tiết như sau:

MINRES Algorithm (for Hermintian matrices A)

```
Given vector x_0
Compute r_0 = b - A x_0
Set q_1 = r_0 ||r_0||
Initialize e = (1, 0, ..., 0)^T ,  β = ||r_0||
For k = 1,2,...
    Compute q_{k+1} ,  α_k ≡ T(k,k) ,  β_k ≡ T(k+1,k) ≡ T(k,k+1)
    using Lanczos algorithm

    Apply F_{k-2} and F_{k-1} to the last column of T:
```

$$\begin{pmatrix} T(k-2,k) \\ T(k-1,k) \end{pmatrix} \leftarrow \begin{pmatrix} c_{k-2} & s_{k-2} \\ -\bar{s}_{k-2} & c_{k-2} \end{pmatrix} \begin{pmatrix} 0 \\ T(k-1,k) \end{pmatrix}, \text{ if } k > 2$$

$$\begin{pmatrix} T(k-1,k) \\ T(k,k) \end{pmatrix} \leftarrow \begin{pmatrix} c_{k-1} & s_{k-1} \\ -\bar{s}_{k-1} & c_{k-1} \end{pmatrix} \begin{pmatrix} T(k-1,k) \\ T(k,k) \end{pmatrix}, \text{ if } k > 1$$

Compute the kth rotation, c_k and s_k to annihilate the $(k+1,k)$ entry of T

Apply kth rotation to e and to the last column of T

$$\begin{pmatrix} e(k) \\ e(k+1) \end{pmatrix} \leftarrow \begin{pmatrix} c_k & s_k \\ -\bar{s}_k & c_k \end{pmatrix} \begin{pmatrix} e(k) \\ 0 \end{pmatrix}$$

$$T(k,k) \leftarrow c_k T(k,k) + s_k T(k+1,k)$$
$$T(k+1,k) \leftarrow 0$$

Commpute $p_{k-1} = [q_k - T(k-1,k)p_{k-2} - T(k-2,k)p_{k-3}]/T(k,k)$ where undefined terms are zero for $k \leq 2$

Set $x_k = x_{k-1} + a_{k-1}p_{k-1}$, where $a_{k-1} = \beta e(k)$

3.5 MINES code

Để có cái nhìn trực quan về giải thuật MINRES, chúng ta sẽ dùng python với thư viện scipy để giải thử một hệ phương trình với ma trận đối xứng kích thước 20×20.

Thư viện scipy đã cung cấp hàm dựng sẵn để giải hệ, ta cùng xem đầu ra và xem xét tính hội tụ của phần dư:

```
1     import numpy as np
2     from scipy.sparse.linalg import minres
3
4     np.random.seed(100)
5     A = np.random.randint(1, 10, (20,20))
6     A = A + A.T
7     print(A)
8     b = np.random.randint(1, 10, 20)
9     x, exitCode = minres(A, b, maxiter=1000, show=True)
10    print(exitCode)
```

Output:

```
[[18 11 11 16 12  9 11  4  8  4  8  9  7  7 10 10  8 16  8 13]
 [11 12  7 12 11  6 12  8  4 15 12  5 11  8 11  8 13  5  5 17]
 [11  7  2 13  8 12  5 13  3 12 13  6 17  9 15 10  8  9  7  7]
 [16 12 13  2  8  9 10 14 11 10 11 14  8 12  5 13 11  3 16 12]
 [12 11  8  8 10  5 15  4  8  7 10 15  8 14  6 10 12  6 11 12]
 [ 9  6 12  9  5 14  8 11 13 10 14  8  5 14 12  6 12 10  7  4]
 [11 12  5 10 15  8  4  6  6  7 10 14 12  4  9 11  7 10 16  3]
```

```
 [ 4  8 13 14  4 11  6  4 13  9  9 14 13  8 11 15  7  4  6 10]
 [ 8  4  3 11  8 13  6 13 10 12  8  7 10  8  8  8 12  8 12 10]
 [ 4 15 12 10  7 10  7  9 12  8 14 10 14  6  9 14  7 12 16  9]
 [ 8 12 13 11 10 14 10  9  8 14 18  6 15 16  6 11  9  3 15 17]
 [ 9  5  6 14 15  8 14 14  7 10  6  4 11  9 12 13 14 12  6  7]
 [ 7 11 17  8  8  5 12 13 10 14 15 11  2 11  9  9  9 12  9  7]
 [ 7  8  9 12 14 14  4  8  8  6 16  9 11  6 11  8  8 11  9  6]
 [10 11 15  5  6 12  9 11  8  9  6 12  9 11 12  8  4  5 10  7]
 [10  8 10 13 10  6 11 15  8 14 11 13  9  8  8  8  6  3 13 12]
 [ 8 13  8 11 12 12  7  7 12  7  9 14  9  8  4  6  6 14  4 14]
 [16  5  9  3  6 10 10  4  8 12  3 12 12 11  5  3 14  2 10  7]
 [ 8  5  7 16 11  7 16  6 12 16 15  6  9  9 10 13  4 10  4  7]
 [13 17  7 12 12  4  3 10 10  9 17  7  7  6  7 12 14  7  7  2]]
Enter minres.    Solution of symmetric Ax = b
Enter minres.    n      =   20     shift  =     0.00000000000000e+00
Enter minres.    itnlim =  1000     rtol   =     1.00e-05


   Itn     x(1)      Compatible    LS        norm(A)  cond(A) gbar/|A|
     1  3.11286e-02  5.684e-01  9.861e-01  1.7e+02  1.0e+00  8.5e-01
     2  2.53166e-02  3.489e-01  7.919e-02  1.9e+02  9.5e+00  2.2e-02
     3  2.29511e-02  8.665e-02  7.995e-02  2.0e+02  9.5e+00 -7.8e-02
     4 -1.00864e-01  5.486e-02  5.118e-02  2.0e+02  1.3e+01 -2.9e-02
     5  6.82543e-02  3.450e-02  4.995e-02  2.0e+02  1.3e+01  4.4e-02
     6  7.07606e-02  3.127e-02  6.089e-02  2.0e+02  1.3e+01  2.4e-02
     7  1.07082e-01  2.178e-02  4.968e-02  2.0e+02  1.3e+01 -4.7e-02
     8  1.02874e-01  1.815e-02  5.627e-02  2.0e+02  1.3e+01 -3.6e-02
     9  6.72500e-02  1.310e-02  4.263e-02  2.0e+02  1.4e+01  3.6e-02
    10  6.59159e-02  1.016e-02  6.389e-02  2.0e+02  1.4e+01  4.8e-02

    11  2.27907e-02  8.223e-03  3.633e-02  2.0e+02  1.6e+01 -2.6e-02
    12 -1.45653e-02  6.747e-03  3.219e-02  2.0e+02  1.8e+01 -2.5e-02
    13 -2.01979e-02  6.625e-03  3.708e-02  2.0e+02  1.8e+01  9.9e-03
    14 -3.82966e-02  5.447e-03  4.020e-02  2.0e+02  1.8e+01  3.9e-02
    15 -5.85959e-02  2.988e-03  3.406e-01  2.6e+02  1.8e+01 -3.0e-01
    16 -6.55580e-02  2.456e-03  2.160e-02  2.8e+02  2.1e+01 -1.3e-02
    17 -5.45825e-02  2.341e-03  1.265e-02  2.8e+02  2.5e+01  6.9e-03
    18 -9.76390e-02  1.589e-03  4.082e-03  2.8e+02  9.5e+01  3.8e-03
    19 -9.64683e-02  1.559e-03  2.039e-02  2.8e+02  9.5e+01 -5.1e-03
    20 -1.31843e-01  4.592e-04  1.787e-02  2.8e+02  9.5e+01 -1.8e-02

    21 -1.51048e-01  1.925e-09  1.351e-02  2.8e+02  9.5e+01 -1.4e-02

Exit  minres.    istop  =    1              itn    =   21
Exit  minres.    Anorm  =    2.8122e+02     Acond  =    9.4687e+01
Exit  minres.    rnorm  =    1.7053e-06     ynorm  =    3.1505e+00
Exit  minres.    Arnorm =    1.3878e+00
Exit  minres.    A solution to Ax = b was found, given rtol
0
```

Phân tích kết quả

Như ta thấy trong code, tham số maxiter được set là 1000 nghĩa là ta cho phép tối đa 1000 vòng lặp. Tolerent mặc định của hàm minres trong thư viện scipy được lấy là 10^{-5}. rnorm là phần dư có giá trị là $1.7053e-06$ và đạt được sau 21 vòng lặp. Nếu so sánh với phương pháp khử Gauss, số bước tính $\sim 20^3 = 8000$