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

BÁO CÁO GIẢI TÍCH SỐ

<u>CHỦ ĐỀ 22:</u> TÌM GIÁ TRỊ RIÊNG TRỘI VÀ GIÁ TRỊ RIÊNG TRỘI TIẾP THEO

Nhóm sinh viên:	Trần Thị Thanh Tươi (MSSV: 20185423)
	Nhâm Đỗ Hải Ninh (MSSV: 20182714)
Lớp:	CTTN Toán Tin K63
Giảng viên hướng dẫn:	TS. Hà Thị Ngọc Yến

HÀ NỘI - 2019

LỜI NÓI ĐẦU

Giải tích số là tên thường gọi của một lĩnh vực toán học chuyên nghiên cứu các phương pháp số giải gần đúng các bài toán thực tế được mô hình hóa bằng ngôn ngữ toán học. Đặc biệt trong thời đại công nghệ 4.0, khi nhân loại hướng tới hầu như bất kể hoạt động nào của con người đều có thể được thay thế thông minh bởi máy móc, tự động hóa; khi sự nghiên cứu thuật toán là thách thức cho con người, thì vai trò giải tích số lại càng quan trọng. Nó đóng vai trò cơ sở, nền tảng cho xây dựng thuật toán trong tương lai.

Để có được lời giải cho bất kì bài toán nào cũng đòi hỏi phải có các dữ kiện của bài toán được thu thập bằng cách đo đạc, thống kê,.. và sau đó là xây dựng mô hình bài toán rồi thực thi chúng. Nhiều bài toán trong thực tế được quy về việc giải hệ thống các phương trình, có thể với kích cỡ đầu vào ma trận lớn. Nếu giải bằng các phương pháp trong đại số tuyến tính, thì việc tính toán là không thể với ma trận kích cỡ lớn vì độ phức tạp. Do vậy việc nghiên cứu giải số là hết sức cần thiết. Có rất nhiều phương pháp số trong việc tìm giá trị riêng và vector riêng, nói chung được chia thành hai loại: giải đúng và giải gần đúng. Trong khuôn khổ bài viết này, chúng em xin phép được trình bày về phương pháp tìm gần đúng giá trị riêng trội và giá trị riêng trội tiếp theo và các véc-tơ riêng tương ứng. Các giá trị riêng trội của ma trận cùng các véc-tơ riêng tương ứng được áp dụng trong nhiều ứng dụng như: xử lí ảnh, mô hình kinh tế động,..

Trong bài viết về giải thuật lũy thừa này chúng em chia làm 5 phần chính: Phần 1: Giới thiệu cơ sở toán học của phương pháp. Phần này sẽ cho chúng ta thấy rõ được bài toán được giải quyết như thế nào qua một vài kết quả quan trọng trong đại số tuyến tính. Phần tiếp theo, phần 2 chúng em xin trình bày về chi tiết phương pháp và xây dựng công thức của phương pháp lũy thừa. Phần 3 trình bày nội dung phương pháp xuống thang để tìm giá trị riêng trội tiếp theo. Phần 4, chúng em đưa ra thuật toán cùng một vài ví dụ điển hình để minh họa cho phương pháp, đồng thời cho bạn đọc nắm được phương pháp dễ dàng hơn và trình bày 2 chương trình được viết trên ngôn ngữ Cpp.

Chúng em xin chân thành cảm ơn TS. Hà Thị Ngọc Yến vì những bài giảng của cô cũng như việc hướng dẫn chúng em hoàn thành tài liệu này.

Cảm ơn mọi người đã đón đọc bài làm của nhóm mình.

Trong quá trình làm báo cáo, dù rất cẩn thận và tập trung, nhóm mình vẫn không tránh khỏi những sai sót và khiếm khuyết, rất mong nhận được sự góp ý của cô cùng các bạn trong lớp.

MỤC LỤC

CHƯƠNG 1. CƠ SỞ TOÁN HỌC CỦA PHƯƠNG PHÁP LŨY THỪA

1.1. Nội dung bài toán

Phương pháp Danilevsky là phương pháp tìm trị riêng đúng; tuy nhiên với những phương trình đặc trưng giải nghiệm gần đúng thì ta chỉ được giá trị riêng gần đúng. Do đó ta cần đến phương pháp tìm các trị riêng và vector riêng gần đúng. Tuy nhiên việc giải đúng đôi khi rất phức tạp và hơn nữa nhu cầu của chúng ta có thể chỉ cần các giá trị gần đúng để tính toán, do đó vấn đề đặt ra là chúng ta có thể tìm gần đúng giá trị riêng và vector riêng gần đúng bằng cách đơn giản và hiệu quả. Trong bản báo cáo này chúng em xin đề cập đến phương pháp lũy thừa để tìm giá trị riêng trội kết hợp với phương pháp xuống thang để tìm các giá trị riêng tiếp theo.

Ở đây bài toán cơ bản đặt ra là: **"Cho ma trận A vuông cấp n. Tìm giá trị riêng trội của ma trận và các giá trị riêng trội tiếp theo cùng với các vector riêng tương ứng."**

1.2. Một số khái niệm

1.2.1. Khái niệm trị riêng trội và vector riêng

Cho ma trận A là ma trận vuông cấp n trên trường số $(K = R\ hoặc\ C)$. Số $\lambda \in K$ được gọi là giá trị riêng của ma trận A nếu tồn tại một vector $X \neq 0, X \in K^n$ sao cho $AX = \lambda X$. Khi đó vector X được gọi là vector riêng của ma trận A ứng với trị riêng λ.

Giả sử ma trận A cấp n có đủ n trị riêng thực hoặc phức (đơn hoặc bội) và chúng thoả mãn điều kiện:

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$$

Khi đó λ_1 được gọi là giá trị riêng trội của ma trận A. Vector ứng với λ_1 được gọi là vector riêng trội của ma trận A. Ma trận có thể không có giá trị trội.

1.2.2. Tính chất của giá trị riêng và vector riêng

- Giá trị riêng λ chính là nghiệm của phương trình $det\, det\,(A - \lambda I) = 0$, hay còn được gọi là phương trình đặc trưng của ma trận A.

- Một giá trị riêng có thể có nhiều vector riêng.
- Mỗi vector riêng chỉ ứng với 1 giá trị riêng duy nhất.
- Nếu ma trận A có giá trị riêng là 0 thì A không khả nghịch. Ngược lại, nếu mọi giá trị riêng của A khác 0 thì A khả nghịch.

1.2.3. Một số cách tìm trị riêng và vector riêng

- Giải phương trình $det\, det\,(A - \lambda I) = 0$ tìm các giá trị riêng. Ứng với mỗi giá trị riêng λ ta giải hệ phương trình tuyến tính thuần nhất $(A - \lambda I)X = 0$.
- Sử dụng phương pháp Danilevsky đưa ma trận A về dạng ma trận có phương trình đặc trưng theo công thức để giải và tìm giá trị riêng.
- Sử dụng phương pháp luỹ thừa để tính gần đúng giá trị riêng trội và vector riêng tương ứng.

CHƯƠNG 2. PHƯƠNG PHÁP LŨY THỪA TÌM GIÁ TRỊ RIÊNG TRỘI

***Ý tưởng cơ bản của phương pháp:** *Nếu như các phương pháp tính đúng dùng một số hữu hạn phép biến đổi ma trận ban đầu về ma trận có cấu trúc đơn giản, từ đó dễ tính đa thức đặc trưng và vector riêng thì các phương pháp tính gần đúng lại sử dụng ý tưởng khuếch đại sự khác biệt của các giá trị riêng bằng cách sử dụng lũy thừa bậc cao* $A^k x$.

Giá trị riêng trội

Giả sử ma trận A cấp n có đủ n trị riêng thực hoặc phức (đơn hoặc bội) và chúng thoả mãn điều kiện:

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n| \qquad (1)$$

Khi đó λ_1 được gọi là giá trị riêng trội của ma trận A. Vector ứng với λ_1 được gọi là vector riêng trội của ma trận A. Và không phải ma trận nào cũng có hoặc có một giá trị riêng trội. Ma trận B sau có thể coi là không có giá trị riêng trội khi hai giá trị riêng của nó bằng và trái dấu nhau:

$$B = [2\;0\;0\;-2]$$

Trở lại với ma trận A tổng quát, ta gọi $X_1, X_2, X_3, \dots, X_n$ là hệ vector độc lập tuyến tính tương ứng với các giá trị riêng ở trên.

Khi đó, ta có vector Y bất kỳ đều là tổ hợp tuyến tính của hệ các vector riêng của A (với C_i là các hằng số):

$$Y = \sum_{i=1}^{n} C_i X_i = C_1 X_1 + C_2 X_2 + \cdots + C_n X_n$$

Ta thực hiện việc tính dãy:

$$AY = A \sum_{i=1}^{n} C_i X_i = \sum_{i=1}^{n} C_i A X_i = \sum_{i=1}^{n} C_i \lambda_i X_i$$

do ta đã có $AX_i = \lambda X_i$. Tiếp tục tính:

$$A^2 Y = A \sum_{i=1}^{n} C_i \lambda_i X_i = \sum_{i=1}^{n} C_i \lambda_i^2 X_i$$

$$A^3Y = A\sum_{i=1}^{n} C_i\lambda_i^2X_i = \sum_{i=1}^{n} C_i\lambda_i^3X_i$$

$$\dots\dots\dots$$

Suy ra:

$$A^mY = A(A^{m-1}Y) = \sum_{i=1}^{n} C_i\lambda_i^mX_i \qquad (2)$$

Trước hết ta sẽ tìm giá trị riêng trội của ma trận với các số trường hợp và đi tìm các giá trị riêng tiếp theo.

2.1. Trường hợp trị riêng trội thực, đơn (bội một)

Với điều kiện (1) : $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$ và chọn Y sao cho $C_1 \neq 0$, từ đẳng thức (2) ta có:

$$A^mY = C_1\lambda_1^mX_1 + \sum_{i=2}^{n} C_i\lambda_i^mX_i = \lambda_1^m\left[C_1X_1 + \sum_{i=2}^{n} C_i\frac{\lambda_i^m}{\lambda_1^m}X_i\right]$$

khi $m \to \infty$ thì $\left(\frac{\lambda_i}{\lambda_1}\right)^m \to 0\ với\ i = 2 \dots n$. Và ta có:

$$A^mY \to C_1\lambda_1^mX_1$$

Hay với m đủ lớn thì $A^mY \approx C_1\lambda_1^mX_1$. Và:

$$A^{m+1}Y \approx C_1\lambda_1^{m+1}X_1$$

$$A(A^mY) \approx \lambda_1(A^mY)$$

Vậy, đẳng thức trên chứng tỏ A^mY là vector riêng ứng với giá trị riêng λ_1 của ma trận A. Và λ_1 có thể được tính theo tỷ số $\frac{(A^{m+1}Y)_j}{(A^mY)_j}$ $(*)$ (là các thành phần thứ j của vector A^mY và $A^{m+1}Y$).

Từ đó, một cách tổng quát ta sẽ chọn véc-tơ Y bất kỳ có C_1 khác 0, đi tính dãy vécto AY, $A^2Y, \dots,$ $A^{m+1}Y$ cho đến khi các tỉ số (*) xấp xỉ nhau thì là m đủ lớn và có trị riêng trội của ma trận A. Các vécto riêng tương ứng có thể chọn là A^mY hoặc $A^{m+1}Y$ đều được.

Ta lấy một ví dụ đơn giản để thể hiện phương pháp:

<u>*Ví dụ 1:*</u> Tìm trị riêng trội của ma trận A sau:

$$\begin{bmatrix} 2 & 3 & 2 \\ 4 & 3 & 5 \\ 3 & 2 & 9 \end{bmatrix}$$

Giải

Ta chọn vector riêng Y bất kì, ở đây lấy $Y = (1{,}1{,}1)^t$. Ta tính được AY, A^2Y,... được viết thành bảng sau:

A	Y	AY	A^2Y	A^3Y	A^4Y	A^5Y	A^6Y
2 3 2 4 3 5	1 1 1	7 12 14	78 134 1	900 156	10589 1	125128	1480345

Ta thấy:

$$\frac{(A^6Y)_j}{(A^5Y)_j} \approx 11{,}83 \text{ với } j = 1{,}2{,}3$$

Do đó có thể lấy $\lambda_1 \approx 11{,}83$ và vector riêng là A^6Y. Và các vector riêng chỉ khác nhau một hằng số nhân, nên ta lấy A^6Y *thành* X_1=$(1;\ 1{,}750;\ 2{,}991)^t$.

Từ lý thuyết và ví dụ, ta có một vài nhận xét:

- Ta có thể nhận thấy rằng việc nhân liên tiếp ma trận A với một vector Y bất kỳ giống như việc tác động liên tục một ánh xạ co lên 1 điểm. Ta nhắc lại một bài toán trong giải tích hàm:

Cho ánh xạ co f có điểm bất động a, chứng minh: $$f^n(x) = a$$

Trong bài toán luỹ thừa, ánh xạ f chính là ánh xạ $f(X) = AX$ và $f^m(X) = A^mX$. Chỉ khác rằng, thay vì hội tụ đến 1 vector thì $f^m(X)$ lại hội tụ đến 1 họ vector mà mỗi 1 vector là vector riêng ứng với trị riêng λ_1 của ma trận A.

- Ta có thể và nên thu nhỏ các vector A^mX sau mỗi lần tính để giảm khối lượng tính toán vì nếu như m lớn thì các hệ số của vector A^mX cũng sẽ rất lớn và việc xảy ra tràn số là khó tránh khỏi.
- Về tốc độ hội tụ:

Ta sẽ xét ví dụ sau:

Xét ma trận $A = [4\ 5\ 6\ 5\ ]$ quá trình tính trị riêng trội chính xác đến chữ số thứ 3 sau dấu phẩy được thể hiện qua bảng sau:

A	Y	AY	A^2Y	A^3Y	A^4Y	A^5Y	A^6Y
4 5 6 5	1 1	9 11	91 109	909 1091	9091 109	90909 10	909091 10

Tương tự với ma trận $B = [-3\ 8\ 7\ 4\ ]$ ta có bảng kết quả sau:

Y	BY	B^2Y	…	$B^{92}Y$	$B^{93}Y$	$B^{94}Y$
1	5	73	…	457.45	4007.88	35114.75
1	11	79	…	672.53	5892.30	51624.39

Nhận xét: Với ma trận A, chỉ cần 6 lần nhân, ta đã thu được dãy vector hội tụ đến một xấp xỉ vừa ý. Thế nhưng với ma trận B thì lại mất 94 lần mới đạt được điều tương tự. Lý do là vì ma trận A có 2 giá trị riêng là $\lambda_1 = 10$ và $\lambda_2 = -1$. Còn ma trận B có 2 giá trị riêng là $\lambda_1 = 8.7614$ và $\lambda_2 = -7.7613$. Ở trên ta đã rút ra kết luận khi $m \to \infty$ thì $\left(\frac{\lambda_i}{\lambda_1}\right)^m \to 0$. Và khi đó, ta có:

$$A^mY \to C_1\lambda_1^mX_1\ khi\ (m \to \infty)$$

Do đó tỷ số $\left|\frac{\lambda_i}{\lambda_1}\right|$ càng nhỏ thì ta càng dễ dàng tìm được m đủ lớn để thỏa mãn yêu cầu tức là tốc độ hội tụ càng nhanh. Đó là lý do tại sao đối với ma trận A, phương pháp luỹ thừa chỉ cần 4 lần tính, nhưng với ma trận B thì lại cần đến gần 70 lần tính.

Đó chính là vấn đề của phương pháp lũy thừa**: tốc độ hội tụ không ổn định.** Tức là phương pháp lũa thừa phù hợp hơn với những ma trận mà các giá trị riêng sai khác nhau lớn, khi đó thông qua các bước lặp lũy thừa thì sự sai khác đó dễ bộc lộ hơn.

Thêm nữa, có một vài trường hợp phương pháp không thể áp dụng, do việc xấp xỉ nhiều lần có thể dẫn tới kết quả thu được không hội tụ như mong muốn, ví dụ với các ma trận đường chéo

2.2. Trường hợp trị riêng trội thực, bội r:

Giả sử λ_1 thực và bội r, tức là:

$$\{\lambda_1 = \lambda_2 = \cdots = \lambda_r \, |\lambda_r| > |\lambda_{r+1}| \geq |\lambda_{r+2}| \geq \cdots \geq |\lambda_n| \qquad (3)$$

Khi đó, biểu thức (2) sẽ có dạng:

$$A^m Y = \lambda_1^m (C_1 X_1 + C_2 X_2 + \cdots + C_r X_r) + \sum_{i=r+1}^{n} C_i \lambda_i^m X_i$$

Bằng những lập luận như trên, khi m đủ lớn, ta có:

$$A(A^m Y) \approx \lambda_1 (A^m Y)$$

$$hay\ \lambda_1 \approx \frac{(A^{m+1}Y)_j}{(A^m Y)_j} \text{ với } j = 1 \dots n$$

Tương tự với trường hợp λ_1 thực, đơn ta sẽ tìm được giá trị riêng và vector riêng tương ứng.

Ta thấy trong trường hợp λ_1 bội r, sẽ có r vector riêng độc lập tuyến tính ứng với λ_1 nhưng ta chỉ có thể tìm được 1 vector. Và trong quá trình tính toán, dựa vào các vector $A^m Y$, sẽ không thể xác định được liệu λ_1 là đơn hay bội r vì trong cả 2 trường hợp, biểu hiện của các vector đều như nhau. Đều này khiến cho việc tìm các giá trị riêng còn lại gặp nhiều khó khăn do ta không biết ma trận có n trị riêng đơn hay n trị riêng có bội.

Tuy nhiên trong phần thuật toán và chay chương trình, ta có thể thay đổi Y đầu vào để tìm ra các vector riêng khác ứng với giá trị riêng là λ_1.

2.3. Trường hợp λ_1 và λ_2 thực trái dấu

Giả sử $\lambda_1 = -\lambda_2$, khi đó ta có hệ điều kiện:

$$\{\lambda_1 = -\lambda_2 \, |\lambda_1| = |\lambda_2| > |\lambda_3| \geq \cdots \geq |\lambda_n| \qquad (4)$$

Khi đó:

$$AY = \lambda_1 (C_1 X_1 - C_2 X_2) + \sum_{i=3}^{n} C_i \lambda_i X_i$$

$$AY^2 = \lambda_1^2 (C_1 X_1 + C_2 X_2) + \sum_{i=3}^{n} C_i \lambda_i^2 X_i$$

$$AY^3 = \lambda_1^3 (C_1 X_1 - C_2 X_2) + \sum_{i=3}^{n} C_i \lambda_i^3 X_i$$

...

$$A^{2k-1}Y = \lambda_1^{2k-1}(C_1X_1 - C_2X_2) + \sum_{i=3}^{n} C_i\lambda_i^{2k-1}X_i$$

$$A^{2k}Y = \lambda_1^{2k}(C_1X_1 + C_2X_2) + \sum_{i=3}^{n} C_i\lambda_i^{2k}X_i$$

Theo giả thiết (4), khi k đủ lớn:

$$A^{2k-2}Y \approx \lambda_1^{2k-2}(C_1X_1 + C_2X_2)$$
$$A^{2k-1}Y \approx \lambda_1^{2k-1}(C_1X_1 - C_2X_2)$$
$$A^{2k}Y \approx \lambda_1^{2k}(C_1X_1 + C_2X_2)$$

Và:

$$A^{2k}Y \approx \lambda_1^{2k}(C_1X_1 + C_2X_2) = \lambda_1^2\lambda_1^{2k-2}(C_1X_1 + C_2X_2) = \lambda_1^2A^{2k-2}Y$$

$$\Rightarrow A^{2k}Y \approx \lambda_1^2A^{2k-2}Y \quad (5)$$

Hay: $$\frac{(A^{2k}Y)_j}{(A^{2k-2}Y)_j} \approx \lambda_1^2 \qquad j = 1 \dots n \qquad (*)$$

Ta nhận thấy rằng, khi $k \to \infty$ thì các bước lũy thừa liền nhau có các tỉ số các thành phần không có xu hướng gần nhau, nhưng ở các bước cùng chẵn hoặc cùng lẻ thì các tỷ số dạng (*) lại có xu hướng trùng nhau.

Từ (5), ta có thể tính được xấp xỉ của λ_1^2 tương tự như cách làm ở 2 trường hợp trước rồi từ đó tính được xấp xỉ của λ_1^2. Để tính được vector riêng tương ứng với λ_1 và $\lambda_2 = -\lambda_1$ ta sẽ đưa các biểu thức trên về dạng $AX = \lambda X$. Ta xét hệ:

$$\begin{cases} A^{2k-1}Y \approx \lambda_1^{2k-1}(C_1X_1 - C_2X_2) \\ A^{2k}Y \approx \lambda_1^{2k}(C_1X_1 + C_2X_2) \end{cases}$$

$$\Leftrightarrow \begin{cases} A^{2k}Y + \lambda_1A^{2k-1}Y \approx \lambda_1^{2k}2C_1X_1 \\ A^{2k}Y - \lambda_1A^{2k-1}Y \approx \lambda_1^{2k}2C_2X_2 \end{cases}$$

$$\Leftrightarrow \begin{cases} A\left(A^{2k}Y + \lambda_1A^{2k-1}Y\right) \approx \lambda_1^{2k}2C_1AX_1 \\ A\left(A^{2k}Y - \lambda_1A^{2k-1}Y\right) \approx \lambda_1^{2k}2C_2AX_2 \end{cases}$$

$$\Leftrightarrow \begin{cases} A\left(A^{2k}Y + \lambda_1A^{2k-1}Y\right) \approx \lambda_1(\lambda_1^{2k}2C_1X_1) \\ A\left(A^{2k}Y - \lambda_1A^{2k-1}Y\right) \approx \lambda_2(\lambda_1^{2k}2C_2X_2) \end{cases}$$

$$\Leftrightarrow \begin{cases} A\left(A^{2k}Y + \lambda_1A^{2k-1}Y\right) \approx \lambda_1\left(A^{2k}Y + \lambda_1A^{2k-1}Y\right) \\ A\left(A^{2k}Y - \lambda_1A^{2k-1}Y\right) \approx \lambda_2\left(A^{2k}Y - \lambda_1A^{2k-1}Y\right) \end{cases}$$

Vậy ta đã có thể kết luận $A^{2k}Y + \lambda_1 A^{2k-1}Y$ và $A^{2k}Y - \lambda_1 A^{2k-1}Y$ lần lượt là các vector riêng ứng với λ_1 và $\lambda_2 = -\lambda_1$.

2.4. Trường hợp λ_1 và λ_2 là phức liên hợp

Giả sử ma trận A có $\lambda_1 = \underline{\lambda_2}$ (phức liên hợp) và:

$$|\lambda_1| = |\overline{\lambda_2}| > |\lambda_3| \geq \cdots \geq |\lambda_n| \quad (6)$$

Biểu thức (2) được viết lại thành:

$$A^m Y = \lambda_1^m C_1 X_1 + \lambda_2^m C_2 X_2 + \sum_{i=3}^{n} C_i \lambda_i^m X_i$$

Cùng với các lập luận tương tự, từ giả thiết (6), khi m đủ lớn, ta có:

$$\{A^m Y \approx \lambda_1^m C_1 X_1 + \lambda_2^m C_2 X_2 \; A^{m+1} Y \approx \lambda_1^{m+1} C_1 X_1 + \lambda_2^{m+1} C_2 X_2 \; A^{m+2} Y \approx \lambda_1^{m+2} C_1 X_1 + \lambda_2^{m+2} C_2 X_2$$

$$\Rightarrow A^{m+2} Y - (\lambda_1 + \lambda_2) A^{m+1} Y + \lambda_1 \lambda_2 A^m Y = 0 \quad (7)$$

Ta thấy trong suốt quá trình tính toán, dãy các vector không hề hội tụ và các dãy con của nó cũng vậy. Thế nhưng khi m đạt được một giá trị nhất định, 3 vector liên tiếp có dấu hiệu tổ hợp tuyến tính với nhau.

Đặt $p = -(\lambda_1 + \lambda_2)$, $q = \lambda_1 \lambda_2$. Khi đó, λ_1 và λ_2 là nghiệm của phương trình:

$$Z^2 + pZ + q = 0 \quad (8)$$

Viết lại phương trình (7): $$A^{m+2} Y + pA^{m+1} Y + qA^m Y = 0 \quad (9)$$

Công việc cần làm là tìm λ_1 và λ_2, tức là tìm nghiệm của phương trình (8).

Từ phương trình (9), ta chọn 2 toạ độ bất kì của các vector (chẳng hạn như $j = r$ và $j = s$, $r \neq s$), kết hợp với phương trình (8) ta có hệ 3 phương trình:

$$\{Z^2 + pZ + q = 0 \; \left(A^{m+2}Y\right)_r + p\left(A^{m+1}Y\right)_r + q\left(A^m Y\right)_r = 0 \; \left(A^{m+2}Y\right)_s + p\left(A^{m+1}Y\right)_s + q\left(A^m Y\right)_s = 0 \quad (10)$$

Hệ phương trình (10) có 3 ẩn là 1, p và q và là hệ phương trình thuần nhất. Để hệ có nghiệm khác không thì định thức phải bằng 0. Tức là:

$$\left| Z^2 \; Z \; 1 \; \left(A^{m+2}Y\right)_r \; \left(A^{m+1}Y\right)_r \; \left(A^m Y\right)_r \; \left(A^{m+2}Y\right)_s \; \left(A^{m+1}Y\right)_s \; \left(A^m Y\right)_s \right| = 0$$

Với các toạ độ tính được từ dãy vector, phương trình trên chính là phương trình (8). Giải ta được cặp nhiệm phức chính là λ_1 và λ_2.

Để tìm vector riêng, ta sử dụng cách làm tương tự như khi λ_1 và λ_2 là thực trái dấu, ta có:

$$\{A^{m-1}Y \approx \lambda_1^{m-1}C_1X_1 + \lambda_2^{m-1}C_2X_2 \; A^mY \approx \lambda_1^m C_1X_1 + \lambda_2^m C_2X_2$$

$$\Leftrightarrow \{A^mY - \lambda_1 A^{m-1}Y \approx C_2\lambda_2^m(\lambda_2 - \lambda_1)X_2 \; A^mY - \lambda_2 A^{m-1}Y \approx C_1\lambda_1^m(\lambda_1 - \lambda_2)X_1$$

Nhân A vào các vế:

$$\Leftrightarrow \{A(A^mY - \lambda_1 A^{m-1}Y) \approx \lambda_2(A^mY - \lambda_1 A^{m-1}Y) \; A(A^mY - \lambda_2 A^{m-1}Y) \approx \lambda_1(A^mY - \lambda_2 A^{m-1}Y)$$

Vậy $A^mY - \lambda_1 A^{m-1}Y$ là vector riêng ứng với λ_2, $A^mY - \lambda_2 A^{m-1}Y$ là vector riêng ứng với λ_1.

CHƯƠNG 3: PHƯƠNG PHÁP XUỐNG THANG TÌM GIÁ TRỊ RIÊNG TRỘI TIẾP THEO

***Ý tưởng phương pháp:**

- Đối với bài toán tìm nghiệm của đa thức, sau khi tìm được $x = x_0$ là nghiệm của P(x), ta loại bỏ nghiệm đã tìm được bằng cách chia P(x) cho đơn thức tương ứng. Khi đó, ta xét:

$$Q(x) = \frac{P(x)}{x - x_0}$$

- Đối với bài toán tìm giá trị riêng trội tiếp theo của ma trận, ta sẽ đưa giá trị riêng vừa tìm được về 0, biến đổi ma trận A ban đầu thành một ma trận mới có các giá trị riêng (trừ giá trị riêng trội vừa tìm) giống A và giá trị riêng trội vừa tìm chuyển thành 0.

Nội dung phương pháp

Xét ma trận $A = [a_{ij}]$ là ma trận vuông cấp n có đủ n giá trị riêng là

$\lambda_1, \lambda_2, \dots, \lambda_n$ và các véc-tơ riêng tương ứng là $X_1, X_2, \dots, X_n$

Qua phương pháp lũy thừa, ta đã tìm được giá trị riêng trội của A là λ_1 và véc-tơ riêng tương ứng là X_1

Chuẩn hóa X_1,ta được véc-tơ có thành phần thứ i bằng 1.Ta vẫn gọi véc-tơ mới là $X_1 = (x_1, x_2, \dots, 1, \dots, x_n)^t$

Xét ma trận θ phụ thuộc vào véc-tơ X_1 và chỉ số i nên kí hiệu là $\theta(X_1, i)$

$$\theta(X_1,i) = [1\ 0\ \dots\ 0\ 1\ \dots\ 0\ -x_1\ 0\ 0\ -x_2\ 0\ \dots\ 0\ \dots\ 0\ \vdots\vdots\vdots\ 0\ 0\ \dots\ \vdots\ \ \vdots\ \ \vdots\ 0\ \ 0\ \ 0\ \vdots\ \vdots\ \dots\ 0\ \vdots\vdots\vdots\ 0\ 0\ \dots\ \vdots\vdots\vdots\ 0\ -x_n\ 0\ \vdots\vdots\ \dots\ 1\ ]$$

Với mọi véc-tơ $Z = (z_1, z_2, \dots, z_n)^t$ ta đều có:

$$\theta Z = [z_1 - x_1 z_i\ z_2 - x_2 z_i\ \dots\ 0\ \dots\ z_n - x_n z_i\ ] = Z - z_i X_1$$

Do đó, với véc-tơ X_1 ta có $\theta X_1 = 0$

Xét ma trận A_1 xác định như sau:

$$A_1 = \theta(X_1, i)A$$
$$= [a_{11} - x_1 a_{i1}\ a_{21} - x_2 a_{i1}\ \dots\ \dots\ a_{1n} - x_1 a_{in}\ a_{2n} - x_2 a_{in}\ \dots\ 0\ \dots\ \dots\ \dots\ 0\ \dots\ a_{n1} - x_n a_{i1}\ \dots\ \dots\ \dots\ a_{nn} - x_n a_{in}\ ]$$

Nhận xét: $A_1 = A - X_1 a_i$ với $a_i = (a_{i1}, a_{i2}, \dots, a_{in})$ là hàng thứ i của ma trận A

Ta sẽ chứng minh ma trận A_1 có các giá trị riêng $\lambda_2, \lambda_3, \dots, \lambda_n$và 0; các véc-tơ riêng tương ứng là $\theta X_2, \theta X_3, \dots, \theta X_n$ và X_1.

Đầu tiên, chứng minh 0 và X_1 lần lượt là trị riêng và véc-tơ riêng tương ứng của A_1

Thật vậy, ta có:

$$A_1 X_1 = (\theta A)X_1 = \theta(AX_1) = \theta\lambda_1 X_1 = \lambda_1(\theta X_1) = 0$$

Do đó, 0 và X_1 lần lượt là trị riêng và véc-tơ riêng tương ứng của A_1

Tiếp theo, với k =2,3,…,n, gọi $x_i^{(k)}$ là phần tử thứ i của X_k. Khi đó ta có:

$$\theta X_k = X_k - x_i^{(k)} X_1$$

Suy ra:

$A_1 = \phi A$

$$\begin{aligned} A_1(\theta X_k) &= (\theta A)(\theta X_k) \\ &= (\theta A)\left(X_k - x_i^{(k)} X_1\right) \\ &= \theta A X_k - x_i^{(k)} \underbrace{\theta A X_1}_{0} \\ &= \theta A X_k \\ &= \theta \lambda_k X_k \end{aligned}$$

$$= \lambda_k(\theta X_k)$$

Nên suy ra $\lambda_k$và θX_k lần lượt là trị riêng và véc-tơ riêng của ma trận A_1 với mọi k=2,3,…,n

Như vậy, sau khi biến đổi ma trận A thành ma trận A_1, thực hiện theo phương pháp lũy thừa một lần nữa, ta sẽ tìm được giá trị riêng trội tiếp theo là λ_2. Tuy nhiên, vấn đề đặt ra là *tìm véc-tơ riêng tương ứng với λ_2 của ma trận A* như thế nào.

Để tìm véc-tơ riêng tương ứng với λ_2 của ma trận A, ta xuất phát từ véc-tơ riêng θX_2 của ma trận A_1. Điều cần làm ở đây là tìm hệ số $x_i^{(2)}$

Ta có:

$$X_2 = \theta X_2 + x_i^{(2)} X_1$$

Suy ra

$$\begin{aligned} AX_2 &= A\left(\theta X_2 + x_i^{(2)} X_1\right) \\ &= A\theta X_2 + x_i^{(2)} AX_1 \\ &= A\theta X_2 + x_i^{(2)} \lambda_1 X_1 \end{aligned}$$

Vì X_2 là véc-tơ riêng ứng với λ_2 của ma trận A nên:

$$\begin{aligned} AX_2 &= \lambda_2 X_2 \\ &= \lambda_2\left(\theta X_2 + x_i^{(2)} X_1\right) \\ &= \lambda_2 \theta X_2 + x_i^{(2)} \lambda_2 X_1 \end{aligned}$$

Suy ra:

$$A\theta X_2 + x_i^{(2)} \lambda_1 X_1 = \lambda_2 \theta X_2 + x_i^{(2)} \lambda_2 X_1$$

$$=> \qquad (A - \lambda_2 E)\theta X_2 = x_i^{(2)}(\lambda_2 - \lambda_1) X_1$$

+ Nếu $\lambda_2 = \lambda_1$ ta có $(A - \lambda_2 E)\theta X_2 = 0$, suy ra θX_2 chính là véc-tơ riêng tương ứng với λ_2 của ma trận A. Hay nói cách khác, trong trường hợp này $x_i^{(2)} = 0$

+ Nếu $\lambda_2 \neq \lambda_1$. Vì X_1 có phần tử thứ i bằng 1, ta có thể tính $x_i^{(2)}$bằng cách chia phần tử thứ i của $(A - \lambda_2 E)\theta X_2$ cho $(\lambda_2 - \lambda_1)$, tức là ta có:

$$x_i^{(2)} = \frac{\left((A - \lambda_2 E)\theta X_2\right)_i}{\lambda_2 - \lambda_1}$$

Tổng kết: Bằng phương pháp lũy thừa, ta tìm được giá trị riêng trội λ_1 và véc-tơ riêng tương ứng X_1 của ma trận A. Áp dụng phương pháp xuống thang, ta biến đổi ma trận A thành ma trận A_1. Lại áp dụng phương phương pháp lũy thừa đối với ma trận A_1 ta tìm được λ_2 là giá trị riêng trội của A_1, đồng thời là giá trị riêng trội tiếp theo của A. Từ véc-tơ θX_2 là véc-tơ riêng ứng với λ_2 của ma trận A_1, ta tìm được véc-tơ X_2 ứng với λ_2 của ma trận A. Cứ như thế tiếp tục, sau (n-1) lần xuống thang, ta tìm được đủ n trị riêng và n véc-tơ riêng tương ứng của ma trận A.

CHƯƠNG 4. THUẬT TOÁN VÀ CHƯƠNG TRÌNH

4.1. Thuật toán và chương trình 1

4.1.1. Sơ lược về thuật toán

- B1: Đọc dữ liệu từ file : n, véc-tơ Y, ma trận
- B2: Tính các cột A^mY và kiểm tra điều kiện, chia trường hợp.
  Trường hợp thỏa mãn điều kiện nghiệm thực, đơn hoặc bội là th1
  Trường hợp thỏa mãn điều kiện nghiệm thực trái dấu là th2
  Trường hợp sau số lần lặp quy định mà vẫn không thõa mãn 2 trường hợp kia thì xếp vào trường hợp 3 (nghiệm phức)
- B3: Xử lí các trường hợp
  Th1 và th2: Đưa ra trị riêng, véc-tơ riêng tương ứng, xuống thang để tìm trị riêng trội tiếp theo
  Th3: Đưa ra nghiệm phức (nếu tìm được) và kết thúc chương trình. Trong trường hợp không tìm được nghiệm phức thì đưa thông báo ra màn hình

4.1.2. Các gói sử dụng

a, Tính A^mY: AmY(float a[M][M],float b[M][M],int n, int m)

- Lưu các cột A^mY liên tiếp thành một mảng 2 chiều b[M][M]
- Ngay từ đầu, véc-tơ Y được lưu thành cột đầu tiên là cột 0
- Đầu vào của gói này là véc-tơ A (mảng hai chiều a[M][M]), mảng b đã có m cột : 0,1,2,…,m-1.
- Gói này thực hiện việc tính và lưu cột thứ m ứng với các giá trị cột $A^{m-1}Y$ dựa vào cột thứ m-1 và ma trận A

b, Gói kiểm tra điều kiện: check(float b[M][M], int c1, int c2, int n)

- Đầu vào là mảng hai chiều b[M][M] gồm các cột A^mY liên tiếp; c1 và c2 là chỉ số 2 cột cần kiểm tra(c1<c2)

- Kiểm tra nếu thành phần thứ r của cột c1 khác không thì thực hiện chia thành phần thứ r của cột c2 cho cột c1. Trong các kết quả tìm được, xác định min, max. Hàm trả về giá trị max-min

c, Gói chuẩn hóa véc-tơ: standard(float x[M][M],int z[M],int n,int lap)

- Mảng x là mảng lưu liên tiếp theo cột các vector riêng tìm được qua mỗi lần thực hiện lũy thừa cho ra kết quả(không phải mảng lưu các vector riêng của ma trận A)
- Mảng z là mảng lưu chỉ số của thành phần bằng 1 tương ứng của các vector mảng x
- lap : biến chạy cho thuật toán xuống thang(đếm số nghiệm tìm được)
- Sau khi được lưu sơ bộ, gói sẽ tìm thành phần có mô-đun lớn nhất của vector riêng ở cột thứ lap vừa lưu, chia tất cả các thành phần của cột lap cho thành phần đó, tức là làm nhiệm vụ chuẩn hóa vector vừa tìm được, lưu lại
- Lưu số thứ tự của thành phần đó vào mảng z

d, Gói tìm vector riêng: vector(float c[M][M],float x[M][M],float lamda[M],int n,int lap)

- Mảng c là mảng lưu liên tiếp các hàng của ma trận A,A1,…(mỗi lần biến đổi ma trận chỉ cần lưu 1 hàng ứng với số thứ tự của phần tử bằng 1 ở vector riêng, chính là giá trị z[lap];
- Mảng lamda[M] là mảng lưu các trị riêng đã tìm được
- Chạy vòng lặp for(r=lap;r>0;r--) để "lùi" dần tìm ra vector riêng của A

e, Gói chuyển ma trận: matrix(float a[M][M],float x[M][M],int z[M],int lap,int n)

- Gói này thực hiện chuyển ma trận để xuống thang sau mỗi lần lặp, lưu lại ma trận mới vào mảng a[M][M]
- Thực hiện theo công thức:

$$A_1 = \theta(X_1, i)A$$
$$= [a_{11} - x_1 a_{i1}\ a_{21} - x_2 a_{i1}\ \dots\ \dots\ a_{1n} - x_1 a_{in}\ a_{2n} - x_2 a_{in}\ \dots\ 0\ \dots\ \dots\ \dots\ 0\ \dots\ a_{n1} - x_n a_{i1}\ \dots\ \dots\ \dots\ a_{nn} - x_n a_{in}\ ]$$

4.1.3. Chương trình

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define M 200
#define E 0.0001

int z[M],n,kt,th;
int i,j,k,lap,m,e;
float a[M][M],b[M][M],c[M][M],x[M][M],lamda[M];
//ham kiem tra
```

```
float check(float b[M][M], int c1, int c2, int n){
	int r;
	float min,max,tg;
	for (r=0;r<n;r++){
	 if (b[r][c1]!=0) {
	     min=b[r][c2]/b[r][c1];
	     break;
	    }
	}
	max=min;
	for (r=0;r<n;r++){
	 if (b[r][c1]!=0){
		tg=b[r][c2]/b[r][c1];
		if (tg>max) max=tg;
		if (tg<min) min=tg;
	 }
	}
	return max-min;
}
//chuyen ma tran A->A1
int matrix(float a[M][M],float x[M][M],int z[M],int lap,int n){
	int r,s,tg;
	tg=z[lap];
	for (r=0;r<n;r++){

	 if (r!=tg)
		for (s=0;s<n;s++) a[r][s]=a[r][s]-x[r][lap]*a[tg][s];
	}
	for (s=0;s<n;s++) a[tg][s]=0;

	for (r=0;r<n;r++){
	 for (s=0;s<n;s++) printf("%6.4f  ",a[r][s]);
	 printf("\n");
	}
	return 0;
}
//tim vector rieng cua ma tran ban dau
int vector(float c[M][M],float x[M][M],float lamda[M],int n,int lap){
	float v[M],yi,G,max;
	int r,s;
```

```
	for (r=0;r<n;r++) v[r]=x[r][lap];
	for (r=lap;r>0;r--){
	 if (lamda[r]!=lamda[r-1]){
	     G=0;
	     for (s=0;s<n;s++) G=G+c[r-1][s]*x[s][r];
	     yi=G/(lamda[r]-lamda[r-1]);
	 }
	 else yi=0;
	 for (s=0;s<n;s++) v[s]=v[s]+ yi*x[s][r-1];
	}
	printf("\nEigenvector: ");
	max=v[0];
	for (r=1;r<n;r++)
	    if (fabs(v[r])>fabs(max)) max=v[r];
	for (r=0;r<n;r++){
	 v[r]=v[r]/max;
	 printf("%6.4f  ",v[r]);
	}
	printf("\n\n");
	return 0;
}
//chuan hoa vector
int standard(float x[M][M],int z[M],int n,int lap){
	int r,tg;
	float max;
	max=x[0][lap];
	tg=0;
	for (r=0;r<n;r++)
	    if (fabs(x[r][lap])>fabs(max)) {max=x[r][lap];tg=r;}
	z[lap]=tg;
	for (r=0;r<n;r++) x[r][lap]=x[r][lap]/max;
	return 0;
}
//tinh A^mY
int AmY(float a[M][M],float b[M][M],int n, int m){
	int r,s;
	float S;
	for (r=0;r<n;r++){
	 S=0;
	 for (s=0;s<n;s++) S=S+a[r][s]*b[s][m-1];
```

```
 b[r][m]=S;
 }
 return 0;
}

int main(){

	float A,B,C,delta,maxi;
	FILE *f1,*f2;
	f1=fopen("input.txt","r");      // doc file
	fscanf(f1,"%d",&n);
	printf("n=%d",n);
	printf("\n");
	for (i=0;i<n;i++){
	 fscanf(f1,"%f",&b[i][0]);
	 printf("%6.4f ",b[i][0]);
	}
	printf("\n");
	printf("\n");
	for (i=0;i<n;i++){
	 for (j=0;j<n;j++){
		fscanf(f1,"%f",&a[i][j]);
		printf("%6.4f ",a[i][j]);
	 }
	 printf("\n");
	}
	printf("\n");
	fclose(f1);
	f2=fopen("output.txt","w");
	for (lap=0;lap<n;lap++){   // lap: bien dem cua pp xuong thang
	 kt=0;
	 th=0;
	 AmY(a,b,n,1);   // tinh hai cot AY,A^2Y
	 AmY(a,b,n,2);
	 for (i=0;i<3;i++) {   //in ra
		for (j=0;j<n;j++) printf("%15.4f  ",b[j][i]);
		printf("\n");
	 }
	   for (i=0;i<n;i++)     // ktra xem AY co thanh phan nao khac 0 ko
	      if (b[i][1]!=0) {kt=1;break;}
```

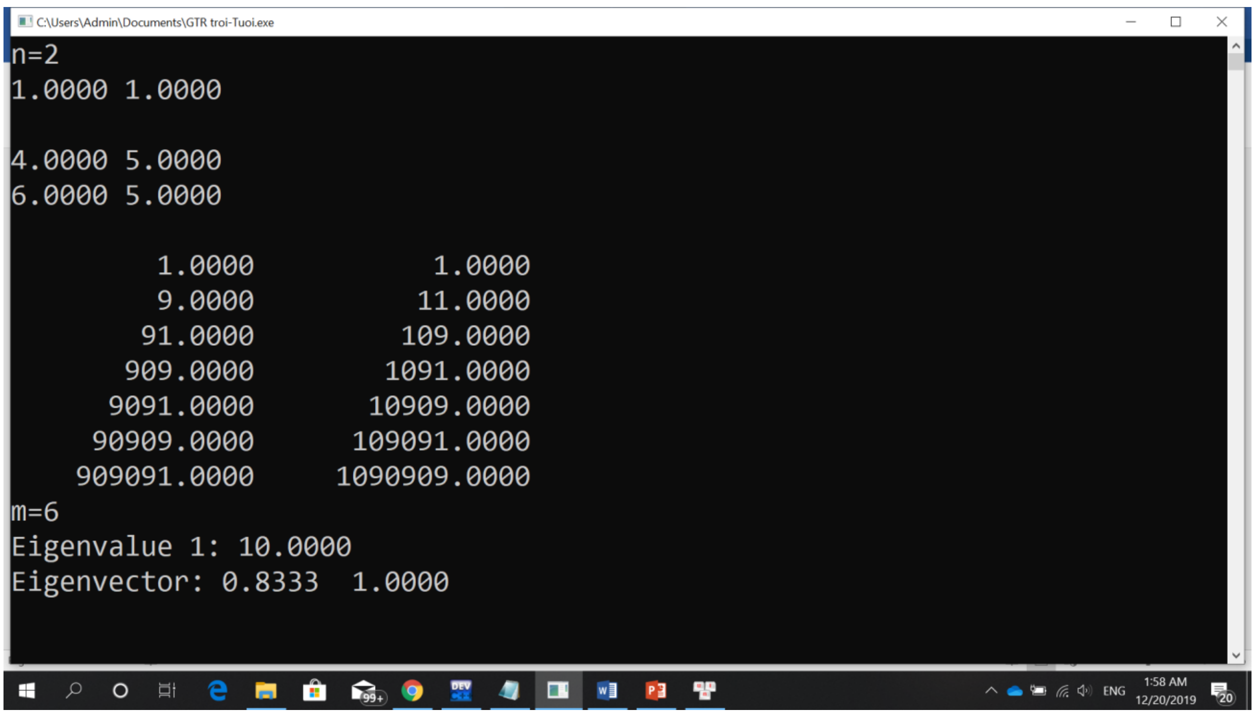
```
    if (kt==0) th=4;    //neu tat ca cac thanh phan cua AY deu =0 thi vector Y
ko phu hop
    else if (check(b,1,2,n)<E) {th=1;m=2;}  //ktra su hoi tu
    while (th==0){
        for (m=3;m<10;m++){    // tinh tu A^3Y -> A^9Y, kiem tra
            AmY(a,b,n,m);
            for (i=0;i<n;i++) {
                printf("%15.4f  ",b[i][m]);
            }
            printf("\n");
            if (check(b,m-1,m,n)<E) {th=1;break;}
            if (check(b,m-2,m,n)<E) {th=2;break;}
        }
        k=m;
        while (th==0) {
            for (e=3;e<(M/5+1);e++){     //Cu tinh dc 5 cot thi chia
                maxi=b[0][m-2];
              for     (j=0;j<n;j++)     if     (fabs(b[j][m-2])>fabs(maxi))
maxi=b[j][m-2];
                for    (j=0;j<n;j++)    if    (fabs(b[j][m-1])>fabs(maxi))
maxi=b[j][m-1];
                for (j=0;j<n;j++){
                    b[j][m-2]=b[j][m-2]/maxi;
                    b[j][m-1]=b[j][m-1]/maxi;
                }
                for (m=k;m<5*e;m++){
                    AmY(a,b,n,m);
               for (i=0;i<n;i++)
                   printf("%15.4f  ",b[i][m]);
            printf("\n");
               if (check(b,m-1,m,n)<E) {th=1;break;}
               if (check(b,m-2,m,n)<E) {th=2;break;}
                }
                k=m;
                if ((th==1)||(th==2)) break;
            }
            if (m>=M) th=3;  // sau 100 lan, neu ko phai th1,th2 thi la th3
        }
        if (m>=M) th=3;
    }
```

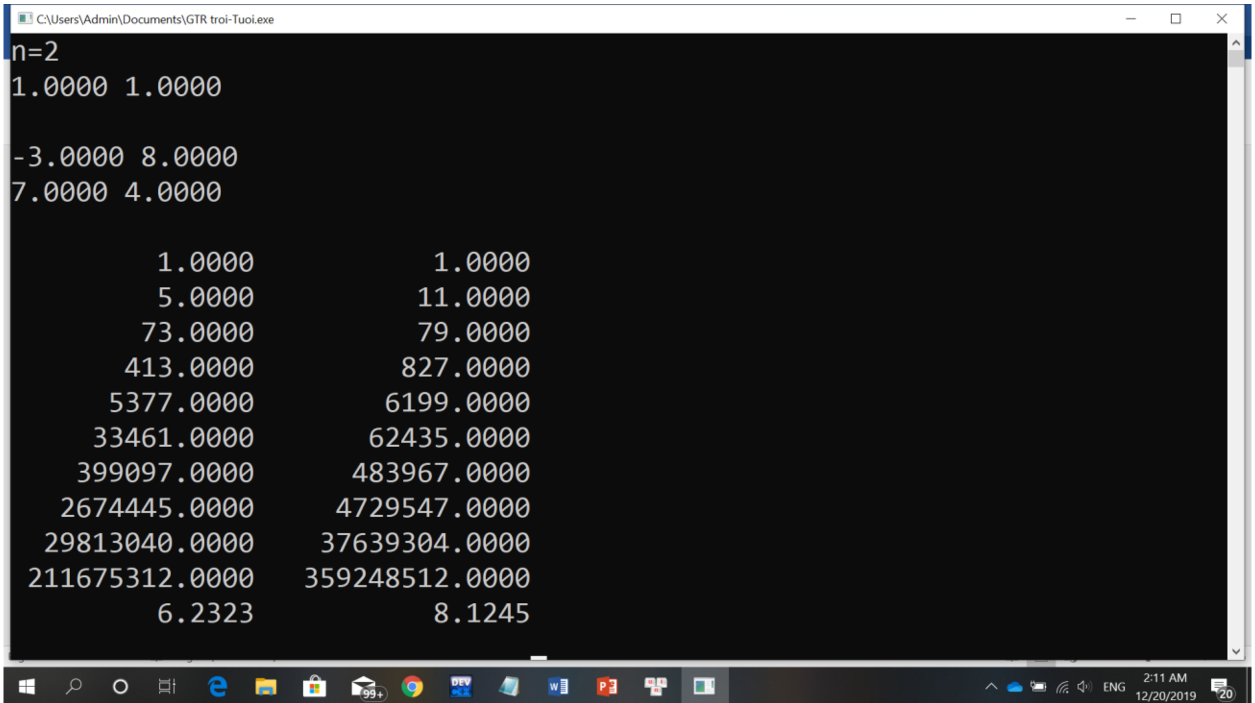
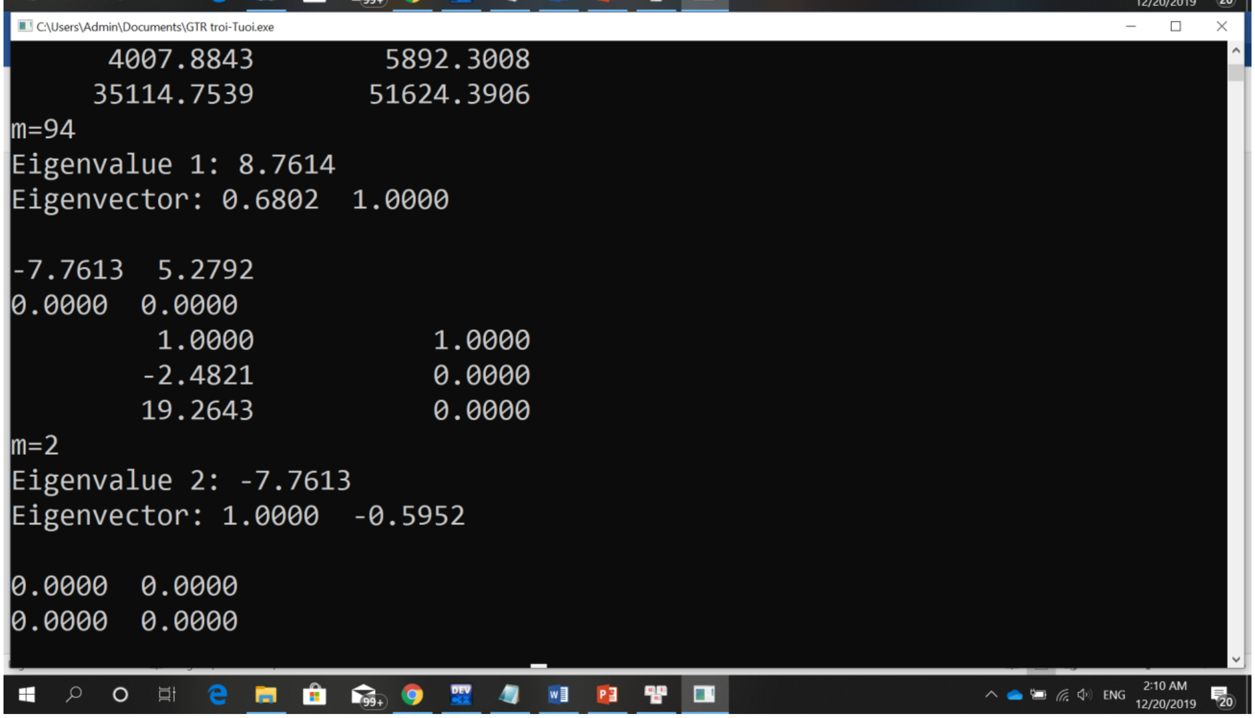
```
  printf("m=%d",m);
switch(th){
        case 1:
                for(i=0;i<n;i++)    //tinh lamda
                    if (b[i][m-1]!=0){lamda[lap]=b[i][m]/b[i][m-1];break;}
                printf("\nEigenvalue %d: %6.4f",lap+1,lamda[lap]);
                for (i=0;i<n;i++) x[i][lap]=b[i][m-1];  //tinh vector rieng
                standard(x,z,n,lap);
                k=z[lap];    // Luu lai 1 hang cua A vao mang c[m][m]
                for (i=0;i<n;i++){
                        if (i!=k) c[lap][i]=a[k][i];
                        else c[lap][i]=a[k][i]-lamda[lap];
                }
                vector(c,x,lamda,n,lap);  // tim vector rieng cua mtran ban dau
                matrix(a,x,z,lap,n);   // chuyen ma tran de xuong thang
                break;
        case 2:  //tuong tu case 1
                for(i=0;i<n;i++)
                    if (b[i][m-2]!=0){lamda[lap]=sqrt(b[i][m]/b[i][m-2]);break;}
                printf("\nEigenvalue %d: %6.4f",lap+1,lamda[lap]);
                for (i=0;i<n;i++) x[i][lap]=lamda[lap]*b[i][m-1]+b[i][m];
                standard(x,z,n,lap);
                k=z[lap];
                for (i=0;i<n;i++){
                        if (i!=k) c[lap][i]=a[k][i];
                        else c[lap][i]=a[k][i]-lamda[lap];
                }
                vector(c,x,lamda,n,lap);
                matrix(a,x,z,lap,n);
                lap=lap+1;  //lap+1
                for(i=0;i<n;i++)
                    if (b[i][m-2]!=0){lamda[lap]=-sqrt(b[i][m]/b[i][m-2]);break;}
                printf("\nEigenvalue %d: %6.4f",lap+1,lamda[lap]);
                for (i=0;i<n;i++) x[i][lap]=lamda[lap]*b[i][m-1]+b[i][m];
                standard(x,z,n,lap);
                k=z[lap];
                for (i=0;i<n;i++){
                        if (i!=k) c[lap][i]=a[k][i];
                        else c[lap][i]=a[k][i]-lamda[lap];
                }
```

```
			vector(c,x,lamda,n,lap);
			matrix(a,x,z,lap,n);
			break;
		case 3:
			for (i=0;i<n;i++)    //chon 2 hang
				if ((b[i][m]!=0)||(b[i][m-1]!=0)||(b[i][m-2]!=0)) break;
			for (j=i+1;j<n;j++)
				if ((b[j][m]!=0)||(b[j][m-1]!=0)||(b[j][m-2]!=0)) break;
			A=b[i][m-1]*b[j][m-2]-b[j][m-1]*b[i][m-2];  //tinh cac he so
			B=b[i][m-2]*b[j][m]-b[j][m-2]*b[i][m];
			C=b[i][m]*b[j][m-1]-b[j][m]*b[i][m-1];
			if (A==0) {    //ktra dkien de co nghiem phuc
				printf("\nA=0");
			}
			else {
				B=B/A;
				C=C/A;
				printf("\nZ^2+ %6.4f Z+ %6.4f =0",B,C);
				delta=pow(B,2)-C*4;
				if (delta>=0) {
				   printf("\nDelta>=0");
			   }
				else { //tinh nghiem phuc va vector rieng
					delta=sqrt(-delta)/2;
					B=-B/2;
					printf("\nEigenvalue         %d:         %6.4f         +
%6.4fi",lap+1,B,delta);
					printf("\nEigenvector: ");
					for (k=0;k<n;k++)
					   printf("%6.4f  +  %6.4fi    ",b[k][m-1]-B*b[k][m-
2],b[k][m-1]-delta*b[k][m-2]);
					printf("\nEigenvalue         %d:         %6.4f         +
%6.4fi",lap+2,B,-delta);
					printf("\nEigenvector: ");
					for (k=0;k<n;k++)
					   printf("%6.4f  +  %6.4fi    ",b[k][m-1]-B*b[k][m-
2],b[k][m-1]+delta*b[k][m-2]);
				}
			}
			break;
```

```
            case 4:
                printf("\nVector Y chon khong phu hop");
                break;
        }
         if ((th==4)||(th==3)){
            printf("\nChuong trinh ket thuc tai day");
            break;
        }
       }
       return 0;
}
```

Ví dụ về tốc độ hội tụ




C:\Users\Admin\Documents\GTR troi-Tuoi.exe
n=2
1.0000 1.0000
-3.0000 8.0000
7.0000 4.0000
1.0000 1.0000
5.0000 11.0000
73.0000 79.0000
413.0000 827.0000
5377.0000 6199.0000
33461.0000 62435.0000
399097.0000 483967.0000
2674445.0000 4729547.0000
29813040.0000 37639304.0000
211675312.0000 359248512.0000
6.2323 8.1245
ENG
2:11 AM
12/20/2019

C:\Users\Admin\Documents\GTR troi-Tuoi.exe
4007.8843 5892.3008
35114.7539 51624.3906
m=94
Eigenvalue 1: 8.7614
Eigenvector: 0.6802 1.0000
-7.7613 5.2792
0.0000 0.0000
1.0000 1.0000
-2.4821 0.0000
19.2643 0.0000
m=2
Eigenvalue 2: -7.7613
Eigenvector: 1.0000 -0.5952
0.0000 0.0000
0.0000 0.0000
ENG
2:10 AM
12/20/2019

Ví dụ sgk:

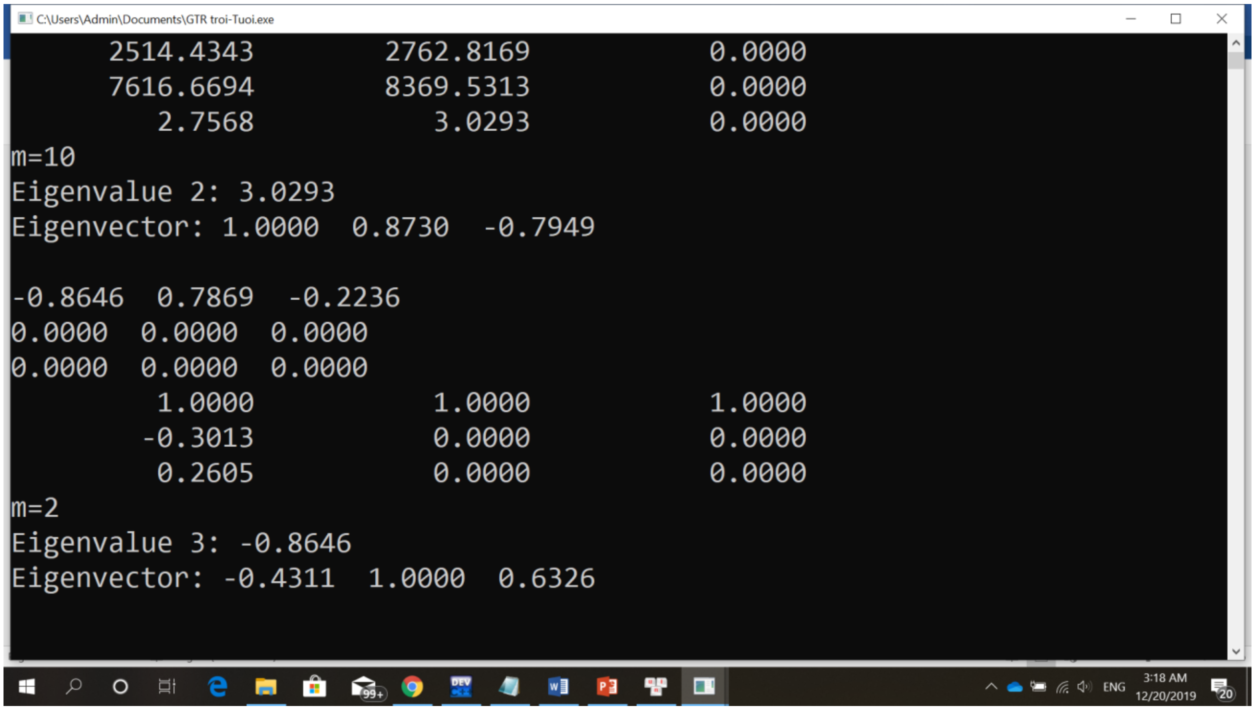
C:\Users\Admin\Documents\GTR troi-Tuoi.exe

```
n=3
1.0000 1.0000 1.0000

2.0000 3.0000 2.0000
4.0000 3.0000 5.0000
3.0000 2.0000 9.0000

          1.0000           1.0000           1.0000
          7.0000          12.0000          14.0000
         78.0000         134.0000         171.0000
        900.0000        1569.0000        2041.0000
      10589.0000       18512.0000       24207.0000
     125128.0000      218927.0000      286654.0000
    1480345.0000     2590563.0000     3393124.0000
   17518628.0000    30658688.0000    40160276.0000
  207333872.0000   362851968.0000   475315776.0000
 2453855232.0000  4294470144.0000  5625547776.0000
```

C:\Users\Admin\Documents\GTR troi-Tuoi.exe

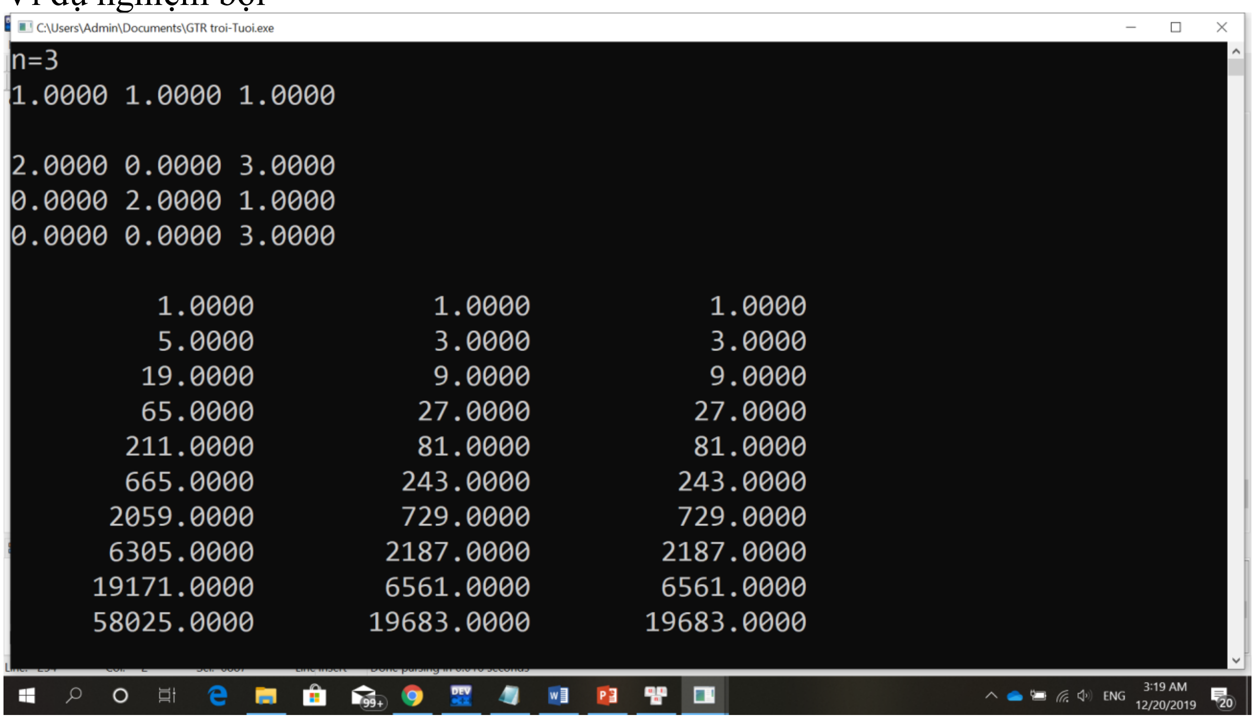
```
  207333872.0000   362851968.0000   475315776.0000
 2453855232.0000  4294470144.0000  5625547776.0000
          5.1626           9.0350          11.8354
m=10
Eigenvalue 1: 11.8353
Eigenvector: 0.4362  0.7634  1.0000

0.6914  2.1276  -1.9258
1.7098  1.4732  -1.8705
0.0000  0.0000  0.0000
          1.0000           1.0000           1.0000
          0.8932           1.3126           0.0000
          3.4102           3.4610           0.0000
          9.7215          10.9298           0.0000
         29.9757          32.7242           0.0000
         90.3494          99.4637           0.0000
        274.0873         301.0154           0.0000
```

3:18 AM 12/20/2019

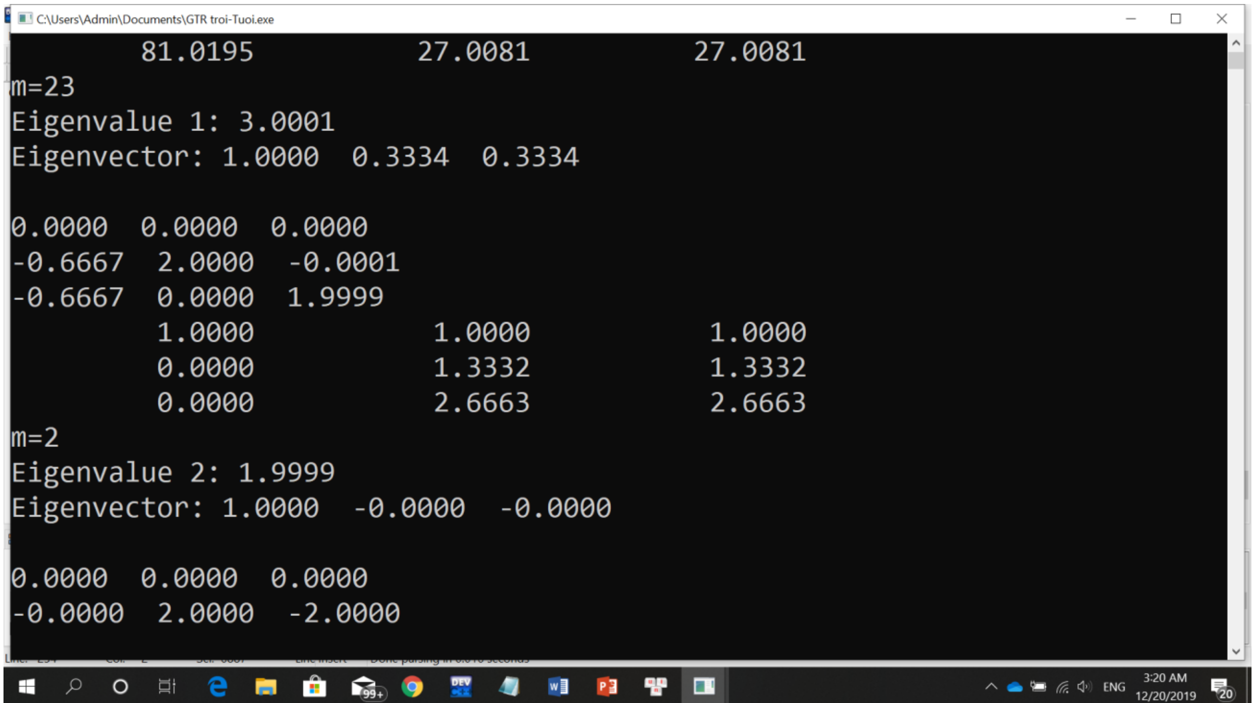


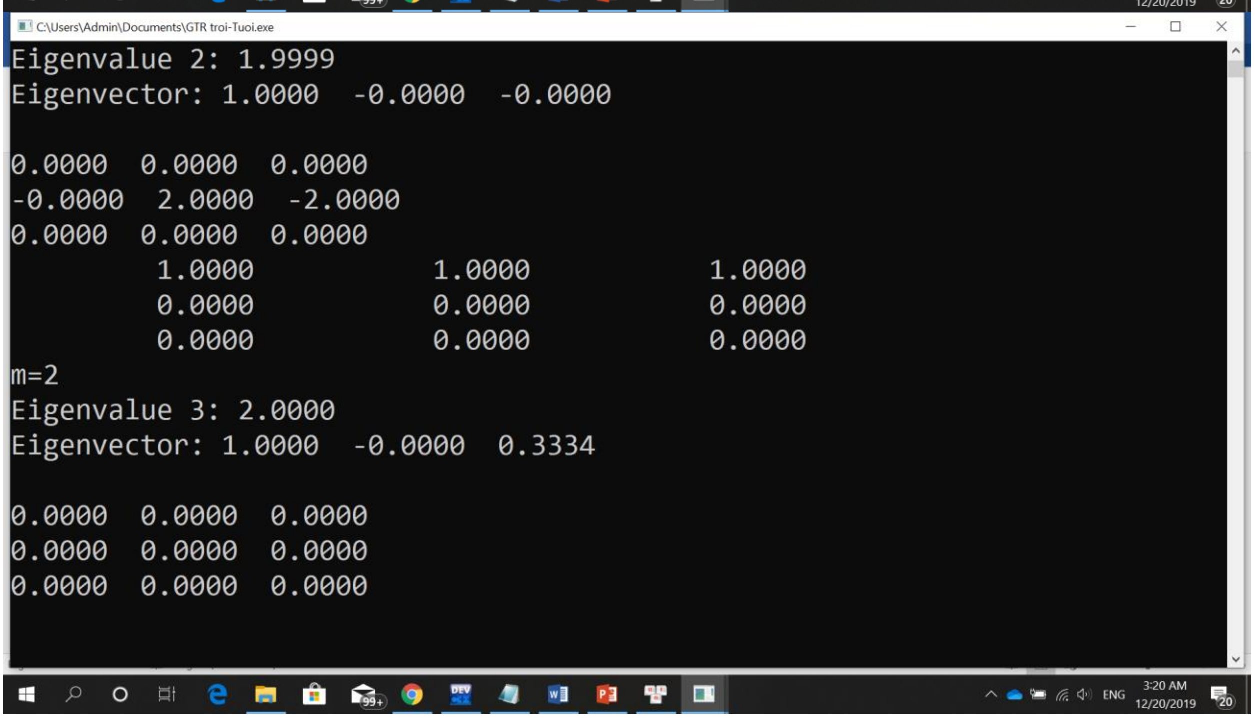
```
C:\Users\Admin\Documents\GTR troi-Tuoi.exe
      2514.4343         2762.8169           0.0000
      7616.6694         8369.5313           0.0000
         2.7568            3.0293           0.0000
m=10
Eigenvalue 2: 3.0293
Eigenvector: 1.0000  0.8730  -0.7949

-0.8646  0.7869  -0.2236
0.0000  0.0000  0.0000
0.0000  0.0000  0.0000
         1.0000            1.0000           1.0000
        -0.3013            0.0000           0.0000
         0.2605            0.0000           0.0000
m=2
Eigenvalue 3: -0.8646
Eigenvector: -0.4311  1.0000  0.6326
```


Ví dụ nghiệm bội

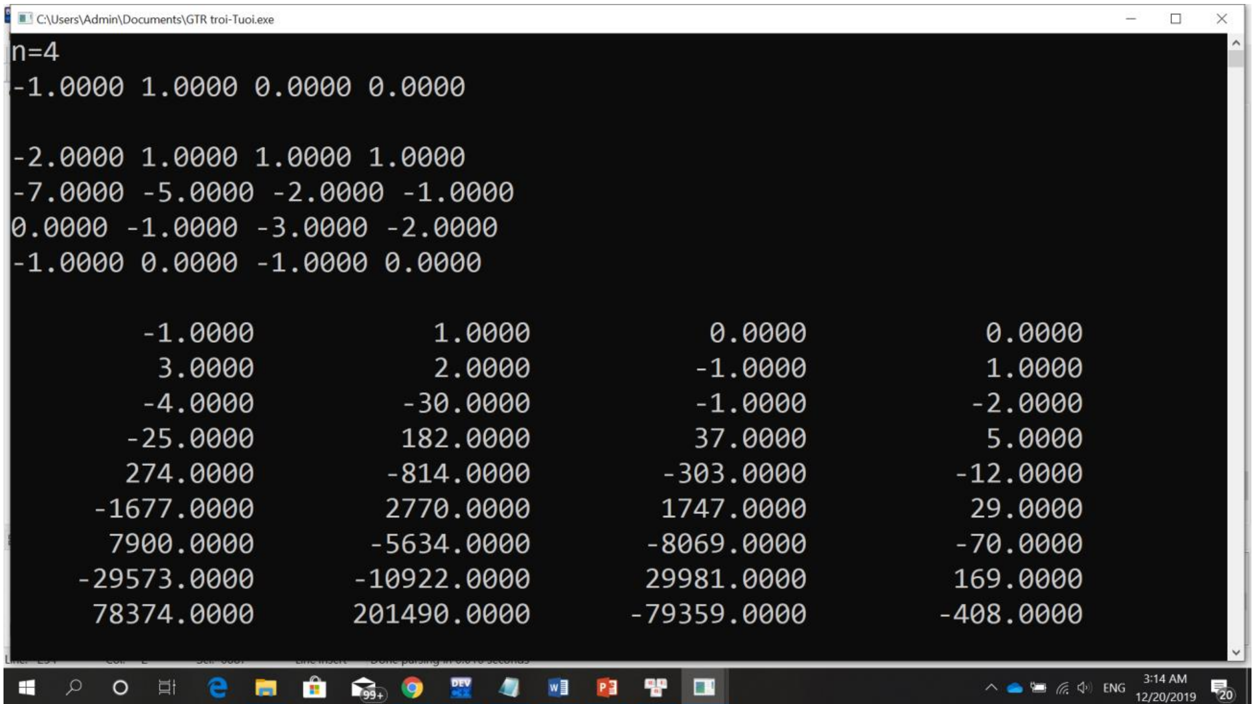


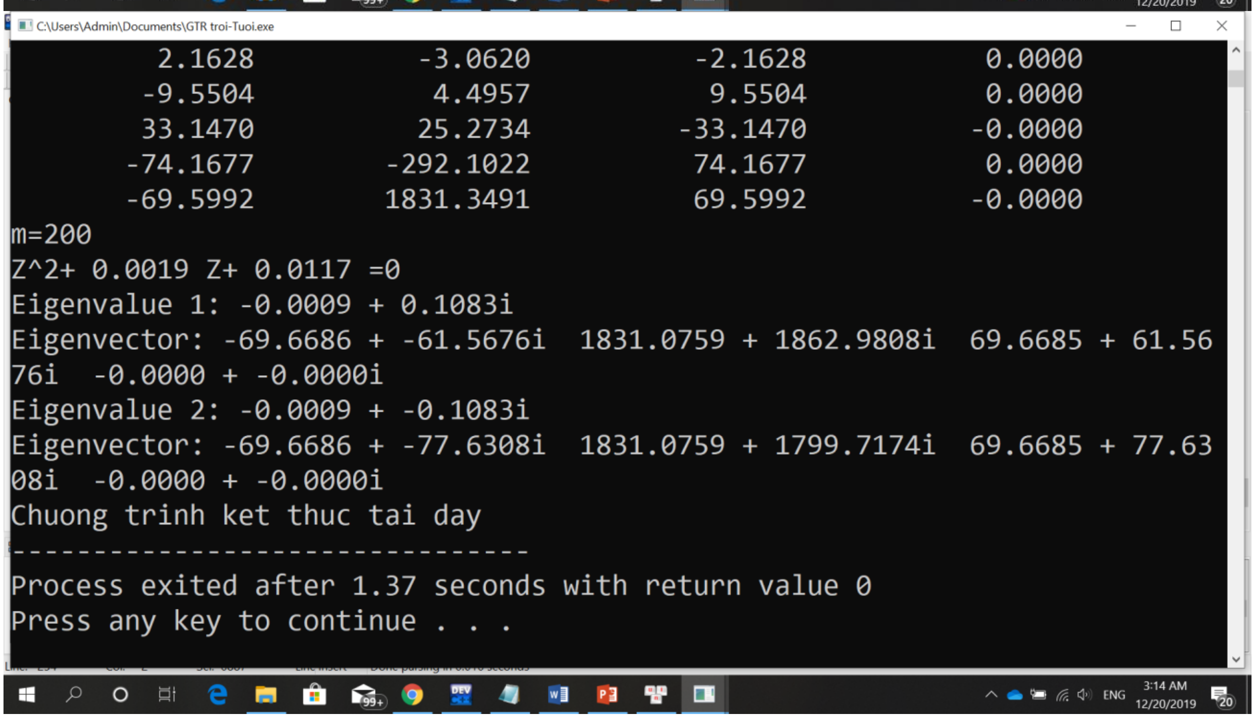
```
C:\Users\Admin\Documents\GTR troi-Tuoi.exe
n=3
1.0000 1.0000 1.0000

2.0000 0.0000 3.0000
0.0000 2.0000 1.0000
0.0000 0.0000 3.0000

         1.0000            1.0000            1.0000
         5.0000            3.0000            3.0000
        19.0000            9.0000            9.0000
        65.0000           27.0000           27.0000
       211.0000           81.0000           81.0000
       665.0000          243.0000          243.0000
      2059.0000          729.0000          729.0000
      6305.0000         2187.0000         2187.0000
     19171.0000         6561.0000         6561.0000
     58025.0000        19683.0000        19683.0000
```





Ví dụ TH nghiệm phức





4.2. Thuật toán và chương trình 2

4.2.1. Tổng quan

- B1: Nhập ma trận vuông cấp n
- B2:
    - Khởi tạo vector $Y = (1,1,\dots,1)$

- Tính các vector $A^m Y$ và kiểm tra:
  - Nếu các vector kề nhau hội tụ, đánh dấu là trường hợp 1.
  - Nếu các vector có bậc luỹ thừa cùng chẵn hoặc lẻ hội tụ, đánh dấu là trường hợp 3.
  - Nếu từ vector thứ 10 trở đi, 2 trường hợp trên không có dấu hiệu thoả mãn, đánh dấu là trường hợp 4.

- B3: Xử lý các trường hợp:
  - TH1&2: Đưa ra trị riêng trội và vector tương ứng. Lặp/Xuống thang n lần để tính ma trận mới và tìm trị riêng tiếp theo.
  - TH3&4: Đưa ra các trị riêng trội và vector riêng tương ứng. Thông báo kết thúc chương trình.

4.2.2. Giả mã

** Giải thuật chung:*

start

1. Enter N, A
2. Initialize $B[i][0] = 1$
3. Determining the first 3 vectors by calculating $B = A^j Y$ ($j = 1 \dots 3$)

From the 4th simplifying vector form:

4. Check if:

$diff\_convergence(B, j, j-1, n) \leq e$ ➔ case 1 or 2
$diff\_convergence(B, j, j-2, n) \leq e$ ➔ case 3
others: case 4

5. Switch case

a) Case 1-2:
while (repeat<n)
    finite descent;
    <print eigenvalue and corresponding eigenvector>;
    repeat++;
b) Case 3:
<print eigenvalue and corresponding eigenvector>; break;
c) Case 4:
<find coefficients for $z^2 + pz + q = 0$ >;
<print eigenvalue and corresponding eigenvector>; break;

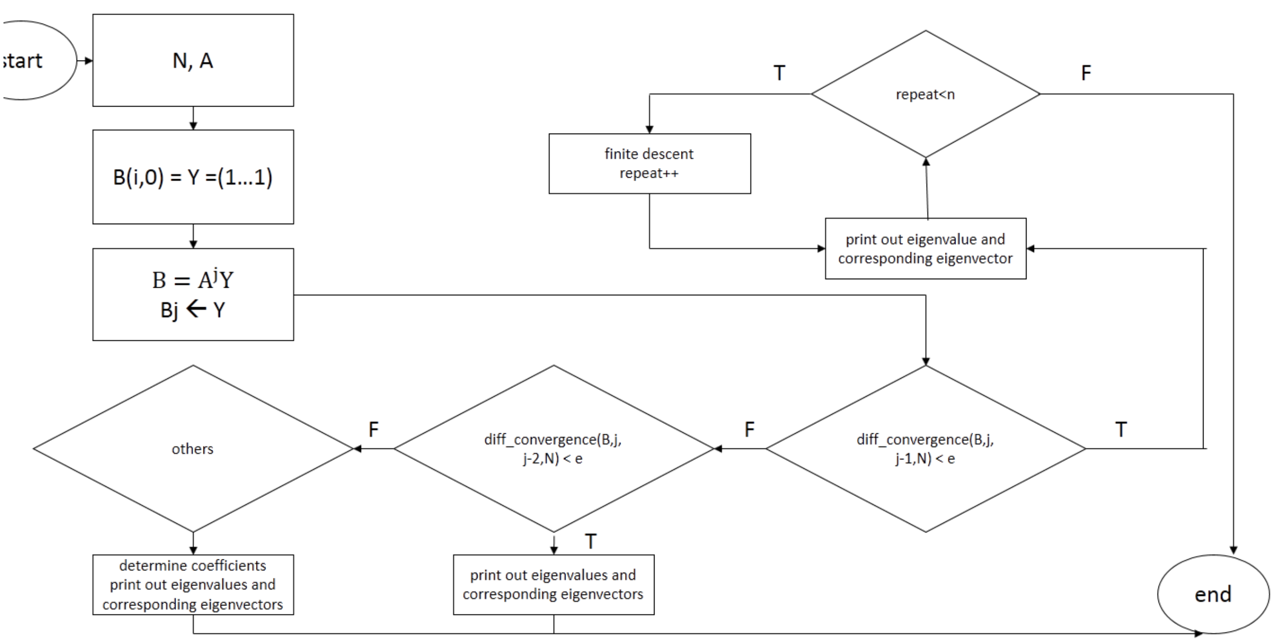
end.

** Giải thuật xuống thang:*

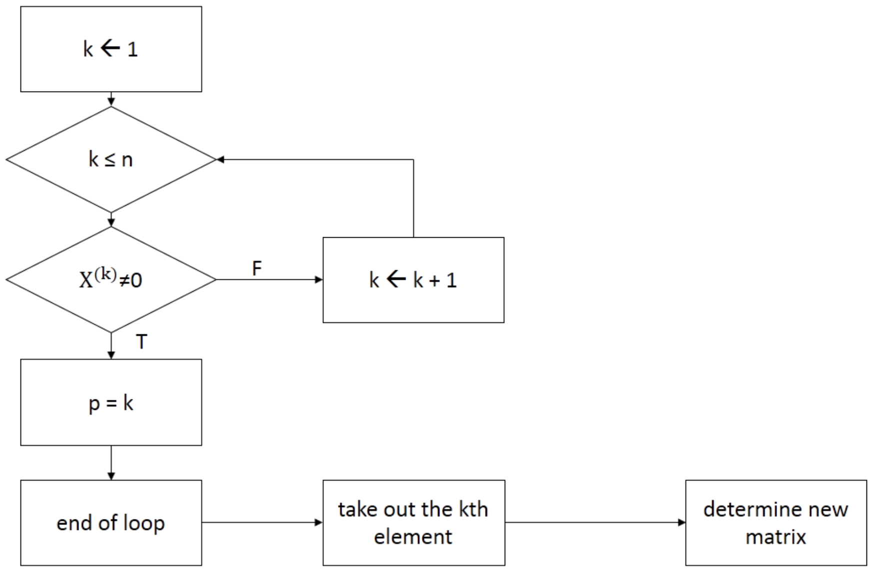
1. $k = 1$
2. while $(k \le N)$
if $X^{(k)} \neq 0 \quad p = k$
$k++$
3. <take the kth element and determine the new matrix>

4.2.3. Sơ đồ thuật toán

** Giải thuật chung:*



** Giải thuật xuống thang:*



4.2.4. Thuật toán chi tiết

Trong chương trình, ta sẽ lưu các vector $A^m Y$ vào các cột của ma trận B. Ta có thể thay vì lưu toàn bộ các vector tính được thì chỉ lưu 1 số lượng vector nhất định vào các mảng 1 chiều.

Đối với phương pháp luỹ thừa, quan trọng nhất của thuật toán là điều kiện dừng vòng lặp tính toán vì việc tính toán chỉ gồm các phép nhân ma trận và các công thức có sẵn. Sau đây là thuật toán của hàm $diff\_convergence()$ được dùng để xác định sai lệch giữa 2 vector được lưu trong ma trận B:

- Input: mảng B, cột c1, cột c2, số lần lặp k.
- Biến $temp \coloneqq fabs(B[1][c1] - B[1][c2]$
- For $i = 2 \to k$
    - Nếu $(fabs(B[i][c1] - B[i][c2]) > temp$ thì
      $temp = fabs(B[i][c1] - B[i][c2])$
- $return\ temp$

Sau mỗi lần tính thêm được 1 vector $A^m Y$, hàm $diff\_convergence()$ sẽ được sử dụng như sau:

- Nếu $diff\_convergence(B, j, j-1, n) \leq e$, thì $testcase = 1$.
- Nếu $diff\_convergence(B, j, j-2, n) \leq e$, thì $testcase = 3$.
- Nếu $diff\_convergence(B, j, j-1, n) - diff\_convergence(B, j-1, j-2, n) > 0$ và $diff\_convergence(B, j, j-2, n) - diff\_convergence(B, j-1, j-3, n) > 0$ và $j > 10$ thì $testcase = 4$.
  (Ở đây ta có thể dùng điều kiện đơn giản hơn rằng nếu không xảy ra $testcase =$

1 *và* $testcase = 3$ *thì* $testcase = 4$ *bởi nếu trị riêng không rơi vào 3 trường hợp đầu thì sẽ rơi và trường hợp phức)*

- Trường hợp 1 và 2:
  - Tính thêm 1 vector $A^{m+1}Y$. Tính được giá trị riêng trội bằng toạ độ lớn nhất của $A^{m+1}Y$.
  - In ra vector riêng là $A^{m+1}Y$.
  - Tính ma trận A mới để tìm trị riêng tiếp theo.
- Trường hợp 3:
  - Tính thêm 2 vector $A^{m+1}Y$ và $A^{m+2}Y$.
  - Tìm toạ độ lớn nhất của $A^{m+2}Y$ và $A^{m}Y$.
  - Tính được 2 trị riêng trái dấu là 2 căn của tỷ số giữa 2 toạ độ trên.
  - Tính được 2 vector riêng tương ứng với các trị riêng theo công thức.
  - Kết thúc chương trình.
- Trường hợp 4:
  - Tính thêm 2 vector $A^{m+1}Y$ và $A^{m+2}Y$.
  - Thiết lập và giải phương trình (8) để tìm được 2 trị riêng phức.
  - Tính vector riêng theo công thức.
  - Kết thúc chương trình.

4.2.5. Chương trình

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef long long int lli;
typedef long double ld;
#define mp make_pair
#define pb push_back
#define fst first
#define snd second
#define null NULL
#define ms(x) memset(x,0,sizeof(x))
#define _read(x) freopen(x,"r",stdin)
#define _write(x) freopen(x,"w",stdout)
#define files(x) _read(x ".in"); _write(x ".out")
#define filesdatsol(x) _read(x ".DAT"); _write(x ".SOL")
#define pll pair<ll,ll>
#define vll vector<ll>
#define vpll vector<pll>
#define FOR(i,n) for(i=0; i<(n); i++)
#define REP(i,a,b) for(i=a; i<(b); i++)
#define PER(i,b,a) for(i=b; i>(a); i--)
#define FS(i,a) for(int i=0;a[i];++i)
```

```
#define TRAV(i,n) for(auto &i:n)
#define SZ(x) (int)(x).size()
#define ALL(t) t.begin(),t.end()
#define e 0.0001
const lli MAX=100001;
const lli mod=1000000007;
const int N=100;
template <typename T>  T sqr(T x){
    return x*x;
}

namespace _buff{
    const size_t BUFF=1<<19;
    char ibuf[BUFF], *ib=ibuf, *ie=ibuf;
    char getc(){
        if(ib==ie){
            ib=ibuf;
            ie=ibuf+fread(ibuf,1,BUFF,stdin);
        }
        return ib==ie ? -1 : *ib++;
    }
}

ll read(){
    using namespace _buff;
    ll ret=0;
    bool pos=true;
    char c=getc();
    for(; (c<'0' || c>'9') && c!='-'; c=getc()) {assert(~c);}
    if(c=='-'){
        pos=false;
        c=getc();
    }
    for(; c>='0' && c<='9'; c=getc()){
        ret=(ret<<3)+(ret<<1)+(c^48);
    }
    return pos ? ret : -ret;
}

int i, j, k, n, testcase, _temp, repeat;
float A[N][N], B[N][N], S, x, y, z, temp;

float diff_convergence(float B[N][N],int c1,int c2,int k){
    int x;
    float temp=fabs(B[1][c1]-B[1][c2]);
    REP(x,1,k+1){
        if(temp<fabs(B[x][c1]-B[x][c2])) temp=fabs(B[x][c1]-B[x][c2]);
    }
    return temp;
}

int main(){
```

```
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    srand(time(NULL));
    cout.precision(6); cout << fixed;

    FILE *f1, *f2;
    f1=fopen("input.txt","r"); fscanf(f1,"%d",&n); printf("%d\n",n);
    REP(i,1,n+1) REP(j,1,n+1) fscanf(f1,"%f ",&A[i][j]);
    fclose(f1);
    REP(i,1,n+1){
        REP(j,1,n+1) printf("%3.2f  ",A[i][j]);
        printf("\n");
    }
    printf("\n");
    f2=fopen("output.txt","w");
    REP(repeat,1,n+1){
        REP(i,1,n+1) B[i][0]=1; // Initial value of vector Y
        REP(k,1,4) //Determining the first 3 vectors
            REP(i,1,n+1){
                S=0;
                REP(j,1,n+1)
                    S+=A[i][j]*B[j][k-1];
                B[i][k]=S;
            }
        j=3;
        for(;;){
            j++;
            REP(i,1,n+1){ //Determining next vectors
                S=0;
                REP(k,1,n+1)
                    S+=A[i][k]*B[k][j-1]; B[i][j]=S;
            }
            temp=B[1][j];
            REP(i,1,n+1) if(fabs(B[i][j])>fabs(temp)) temp=B[i][j];
            REP(i,1,n+1) B[i][j]/=temp;
            if(diff_convergence(B,j,j-1,n)<=e) {testcase=1; break;}
            if(diff_convergence(B,j,j-2,n)<=e&&(B,j,j-1,n)>e) {testcase=3; break;}
            if(diff_convergence(B,j,j-1,n)-diff_convergence(B,j-1,j-2,n)>0 && (diff_convergence(B,j,j-2,n)-diff_convergence(B,j-1,j-3,n)>0) && (j>7)) {testcase=4; break;}
            if(j>=100) {testcase=4; break;}
        }
        printf("\nChains of determined vector: \n");
        REP(i,1,n+1){
            REP(k,1,j+1) printf("%10f ",B[i][k]);
            printf("\n");
        }
        printf("\nTimes of iteration: %d \n",j);
        switch(testcase){
            case 1:{
                REP(i,1,n+1){
```

```
                        S=0;
                        REP(k,1,n+1)
                            S+=A[i][k]*B[k][j-1];
                            B[i][j]=S;
                    }
                    temp=B[1][j]; _temp=1;
                    REP(i,1,n+1) if(B[i][j]>temp) {temp=B[i][j]; _temp=i;}
                    printf("The %dth eigenvalue is: %f\n",repeat,B[_temp][j]/B[_temp][j-1]);
                    printf("The corresponding eigenvector is: ");
                    REP(i,1,n+1)
                        printf("%f ",B[i][j]/temp);
                    S=0; printf("\n");
                    REP(i,1,n+1) S+=pow(B[i][j],2);
                    REP(i,1,n+1)// Determining new matrix to find new eigenvalue
                        REP(k,1,n+1)
                            A[i][k]-=B[_temp][j]/B[_temp][j-1]/S*B[i][j]*B[k][j];
                    break;
                }
                case 3:{
                    j++;
                    REP(i,1,n+1){
                        S=0;
                        REP(k,1,n+1)
                            S+=A[i][k]*B[k][j-1];
                        B[i][j]=S;
                    }
                    j++;
                    REP(i,1,n+1){
                        S=0;
                        REP(k,1,n+1)
                            S+=A[i][k]*B[k][j-1];
                        B[i][j]=S;
                    }
                    printf("\nThe %dth eigenvalue is: %f ",repeat,sqrt(B[1][j]/B[1][j-2]));
                    printf("The corresponding vector is: ");
                    REP(i,1,n+1) printf("%f ",B[i][j]+sqrt(B[1][j]/B[1][j-2])*B[i][j-1]);
                    printf("\nThe %dth eigenvalue is: %f ",repeat+1,-sqrt(B[1][j]/B[1][j-2]));
                    printf("The corresponding vector is: ");
                    REP(i,1,n+1) printf("%f ",B[i][j]-sqrt(B[1][j]/B[1][j-2])*B[i][j-1]);
                    printf("\nIf signs are different, the program will stop here");
                    break;
                }
                case 4:{
                    j++;
```

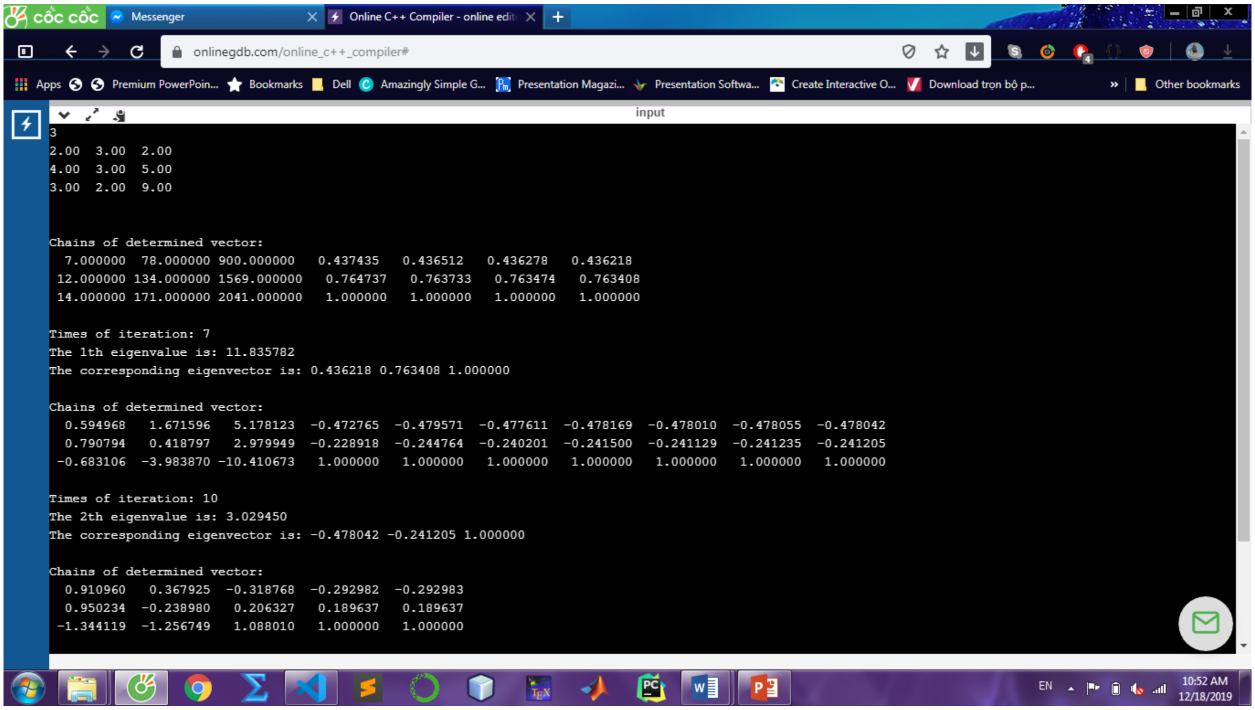
```
                    REP(i,1,n+1){
                        S=0;
                        REP(k,1,n+1)
                            S+=A[i][k]*B[k][j-1];
                        B[i][j]=S;
                    }
                    j++;
                    REP(i,1,n+1){
                        S=0;
                        REP(k,1,n+1)
                            S+=A[i][k]*B[k][j-1];
                        B[i][j]=S;
                    }
                    x=1; // Determining the coefficients of the equation
                    y=(B[1][j]*B[2][j-2]-B[1][j-2]*B[2][j])/(B[1][j-2]*B[2][j-1]-B[1][j-1]*B[2][j-2]);
                    z=(B[1][j-1]*B[2][j]-B[1][j]*B[2][j-1])/(B[1][j-2]*B[2][j-1]-B[1][j-1]*B[2][j-2]);
                    float delta=y*y-4*x*z;
                    printf("%f %f %f %f",x,y,z,delta);
                    printf("\nThe %dth eigenvalue is: %.2f+%.2fi\n",repeat,-y/2/x,sqrt(-delta)/2/x);
                    printf("The corresponding vector is: "); temp=B[1][j];
                    REP(i,1,n+1) printf("%.2f-(%.2f)i ",(B[i][j+1]-delta*B[i][j])/temp,(sqrt(-delta)*B[i][j])/temp); printf("\n");
                    printf("\nThe %dth eigenvalue is: %.2f-%.2fi\n",repeat+1,-y/2/x,sqrt(-delta)/2/x);
                    printf("The corresponding eigenvector is: ");
                    REP(i,1,n+1) printf("%.2f+(%.2f)i ",(B[i][j+1]-delta*B[i][j])/temp,(sqrt(-delta)*B[i][j])/temp); printf("\n");
                    printf("If complex eigenvalues are met, the program will stop here");
                    break;
                }
            }
            if (testcase==3 || testcase==4) break;
        }
    }
```

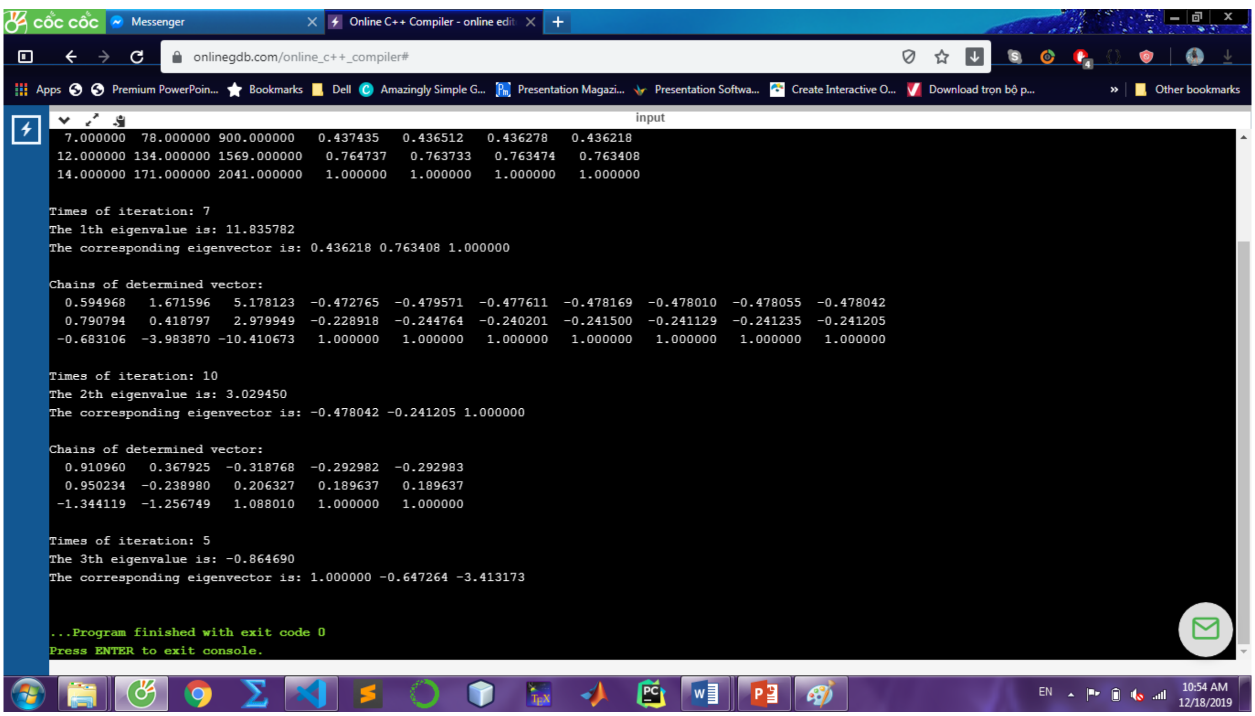
3.4. Ví dụ

Ví dụ 1: Tìm các giá trị riêng của ma trận:

2 3 2 4 3 5 3 2 9

* Đây là trường hợp 1 (thực, đơn bội 1)

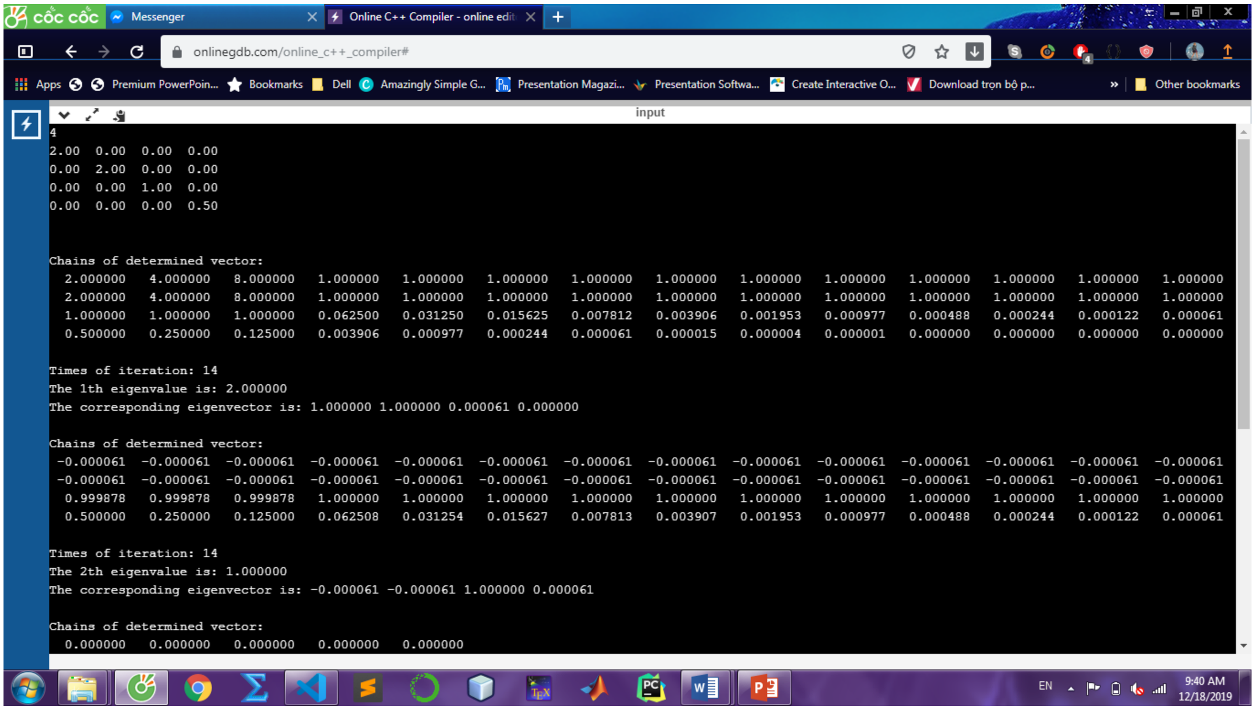


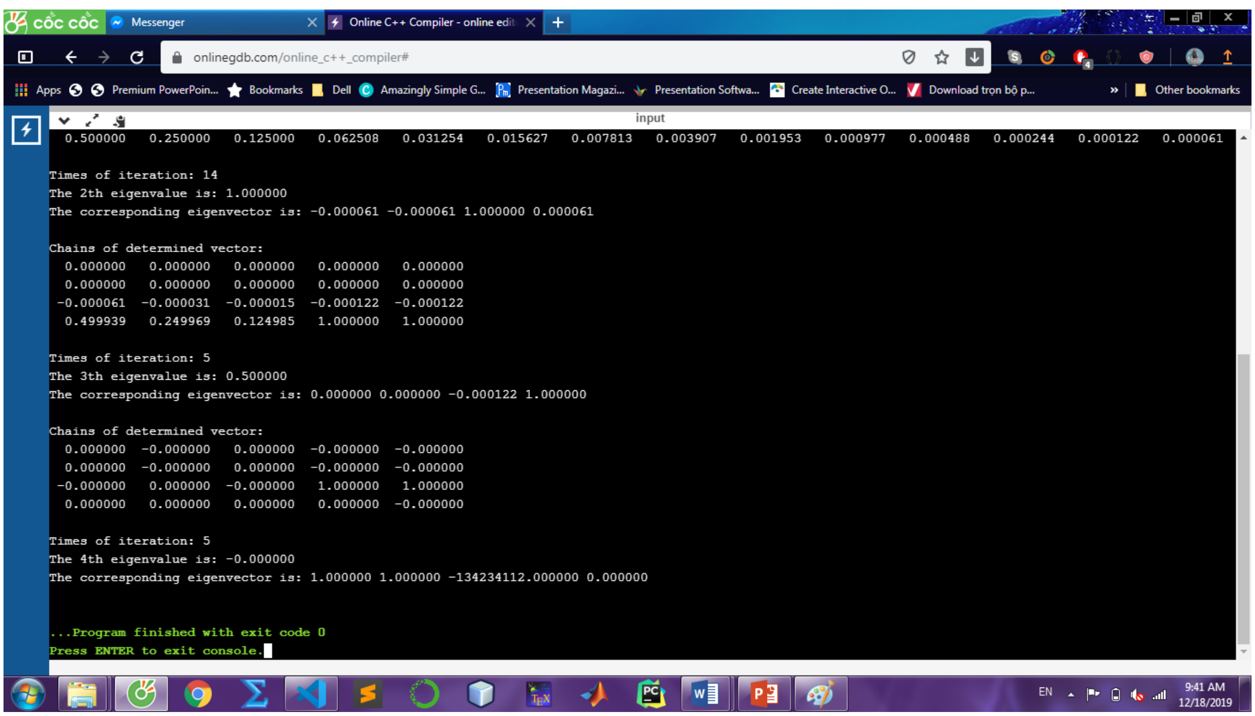


Ví dụ 2: Tìm các giá trị riêng của ma trận:

$$\left(\frac{2\ 0\ 0\ 0 0\ 2\ 0\ \ 0 0\ \ 0\ 1\ 0}{0\ \ 0\ \ 0\ \ 0.5}\right)$$

* Đây là trường hợp 2 (thực, bội 2)

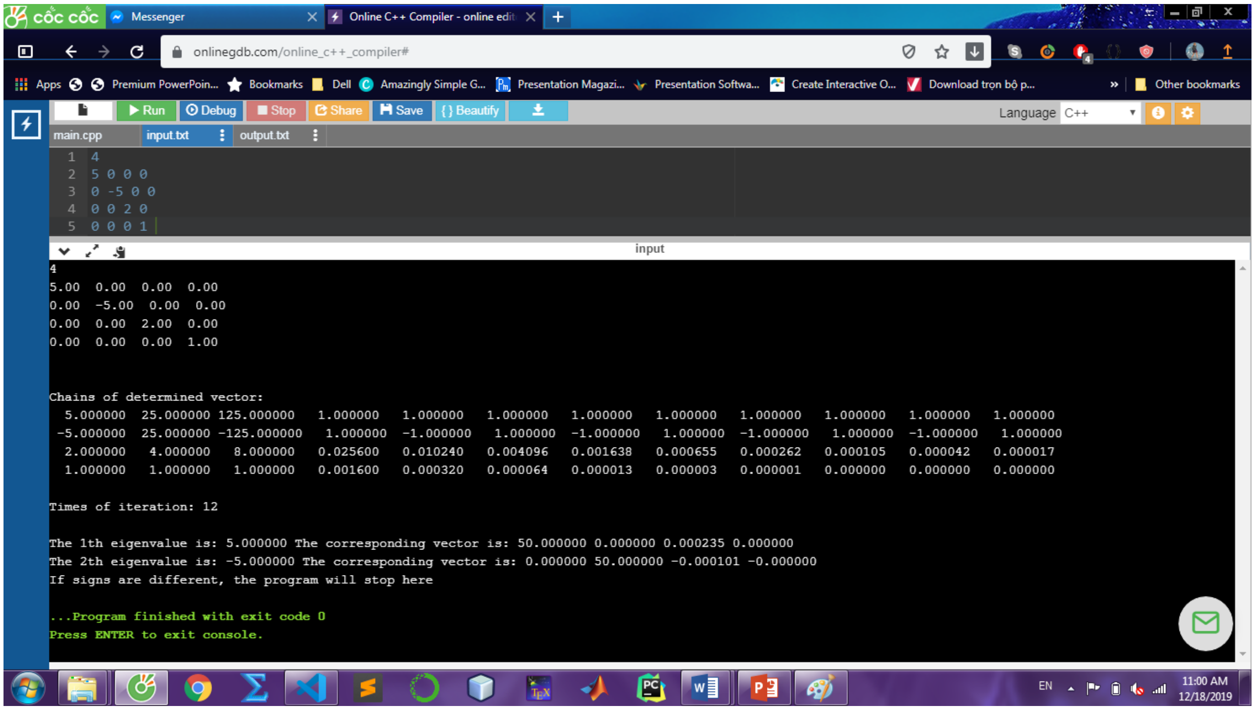




Ví dụ 3: Tìm các giá trị riêng của ma trận:

$$(5\ \ 0\ \ 0\ \ 00-5\ \ 0\ \ 00\ \ 0\ \ 2\ \ 00\ \ 0\ \ 0\ \ 1)$$

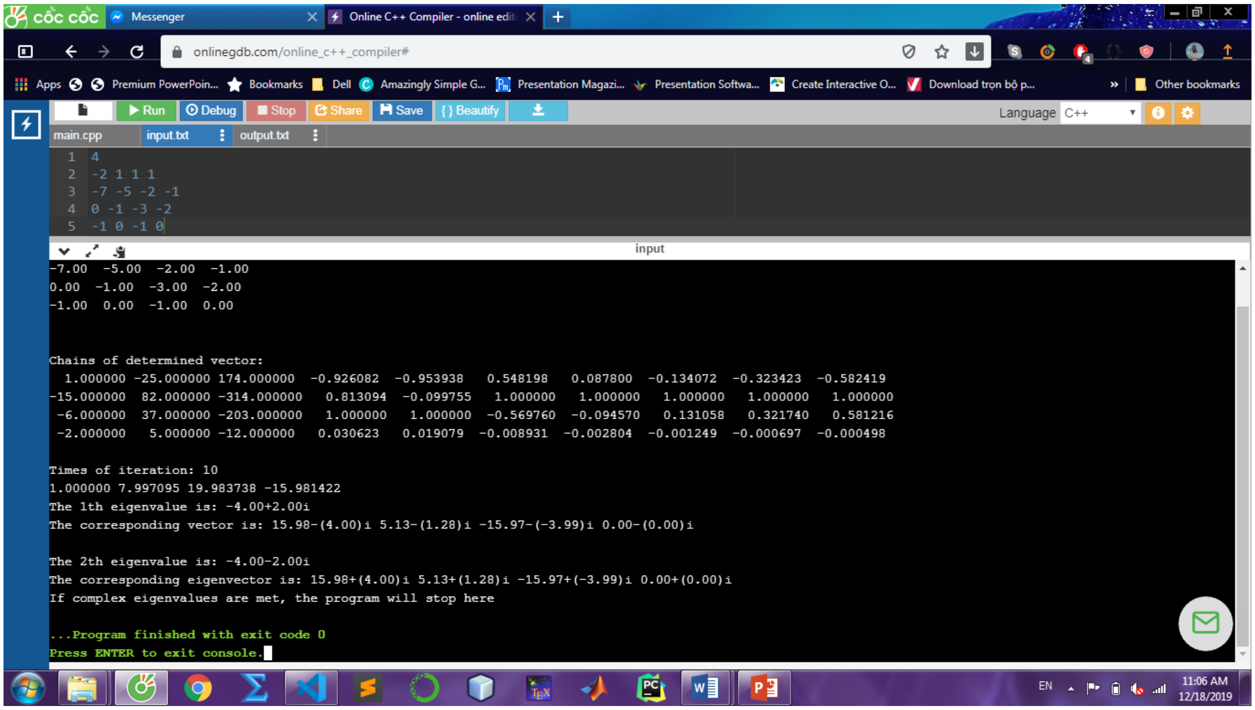
* Đây là trường hợp 3 (2 trị riêng trái dấu)



Ví dụ 4: Tìm các giá trị riêng của ma trận:

$$\left(\frac{-2 \quad 1 \quad 1 \quad 1}{-7-5-2-1 \quad 0 \;-1-3 \;-2 \;-1 \quad 0 \quad -1 \quad 0}\right)$$

* Đây là trường hợp 4 (trị riêng phức liên hợp)



Online C++ Compiler - online edit
onlinegdb.com/online_c++_compiler#
Run
Debug
Stop
Share
Save
{ } Beautify
Language C++
main.cpp
input.txt
output.txt
4
-2 1 1 1
-7 -5 -2 -1
0 -1 -3 -2
-1 0 -1 0
input
-7.00 -5.00 -2.00 -1.00
0.00 -1.00 -3.00 -2.00
-1.00 0.00 -1.00 0.00
Chains of determined vector:
1.000000 -25.000000 174.000000 -0.926082 -0.953938   0.548198   0.087800 -0.134072 -0.323423 -0.582419
-15.000000 82.000000 -314.000000   0.813094 -0.099755   1.000000   1.000000   1.000000   1.000000   1.000000
-6.000000 37.000000 -203.000000   1.000000   1.000000 -0.569760 -0.094570   0.131058   0.321740   0.581216
-2.000000   5.000000 -12.000000   0.030623   0.019079 -0.008931 -0.002804 -0.001249 -0.000697 -0.000498
Times of iteration: 10
1.000000 7.997095 19.983738 -15.981422
The 1th eigenvalue is: -4.00+2.00i
The corresponding vector is: 15.98-(4.00)i 5.13-(1.28)i -15.97-(-3.99)i 0.00-(0.00)i
The 2th eigenvalue is: -4.00-2.00i
The corresponding eigenvector is: 15.98+(4.00)i 5.13+(1.28)i -15.97+(-3.99)i 0.00+(0.00)i
If complex eigenvalues are met, the program will stop here
...Program finished with exit code 0
Press ENTER to exit console.
11:06 AM
12/18/2019

KẾT LUẬN

Các giá trị riêng của ma trận và các vector riêng tương ứng có rất nhiều ứng dụng trong cuộc sống. Tuy nhiên, trên thực tế, không phải lúc nào người ta cũng cần đến tất cả trị riêng và vector riêng và nhu cầu chỉ cần tính gần đúng trị riêng, vector riêng. Phương pháp lũy thừa để tìm giá trị riêng trội và phương pháp xuống thang để tìm giá trị riêng trội tiếp theo là những phương pháp được tìm ra để giải quyết vấn đề đó.

Mỗi phương pháp đều có những ưu, nhược điểm riêng. Đối với hai phương pháp mà nhóm trình bày trong báo cáo cũng như vậy. Phương pháp lũy thừa được áp dụng rộng rãi trong tin học và vật lý. Tuy nhiên, nhược điểm của nó là tốc độ hội tụ không ổn định, và với riêng bài toán tìm trị riêng trội, một số ma trận không thể áp dụng phương pháp này do khi xấp xỉ nhiều lần, tính hội tụ không còn đúng.

Trong quá trình áp dụng hai phương pháp này, cũng cần chú ý khống chế để tránh tình trạng tràn số.

Một lần nữa, xin chân thành cảm ơn cô và các bạn đã dành thời gian đọc báo cáo này.

DANH MỤC TÀI LIỆU THAM KHẢO

1. Giáo trình Giải tích số - Lê Trọng Vinh, NXB Khoa học và Kĩ thuật
2. Giải tích số - Phạm Kỳ Anh, NXB Đại học Quốc gia Hà Nội