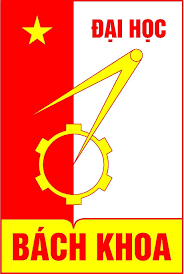
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC**



**BÁO CÁO GIẢI TÍCH SỐ**

***CHỦ ĐỀ 24:* TÌM GIÁ TRỊ RIÊNG TRỘI VÀ GIÁ TRỊ RIÊNG TRỘI TIẾP THEO**

Nhóm sinh viên: Vũ Thị Kim Dung (MSSV: 20195959)

Lê Đức Tài (MSSV: 20195163)

Lớp: CTTN Toán Tin K64

Giảng viên hướng dẫn: TS. Hà Thị Ngọc Yến

# **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 chương trình được viết trên ngôn ngữ Python.

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

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**

[LỜI NÓI ĐẦU 2](#_heading=h.gjdgxs)

[PHẦN I. CƠ SỞ TOÁN HỌC CỦA PHƯƠNG PHÁP LŨY THỪA 5](#_heading=h.1fob9te)

[PHẦN II. PHƯƠNG PHÁP LŨY THỪA](#_heading=h.3znysh7) [TÌM GIÁ TRỊ RIÊNG TRỘI](#_heading=h.2et92p0) [6](#_heading=h.3znysh7)

PHẦN III. THUẬT TOÁN VÀ CHƯƠNG TRÌNH 15

[PHẦN IV. KẾT LUẬN 26](#_heading=h.4d34og8)

**PHẦN I: CƠ SỞ TOÁN HỌC CỦA PHƯƠNG PHÁP LŨY THỪA**

1. **Khái niệm giá trị riêng và vector riêng**

Cho ma trận là ma trận vuông cấp n trên trường số . Số được gọi là giá trị riêng của ma trận nếu tồn tại một véctơ sao cho . Khi đó vector được gọi là vector riêng của ma trận ứng với trị riêng .

1. **Tính chất của giá trị riêng và vector riêng**

* Giá trị riêng chính là nghiệm của phương trình . Hay còn được gọi là phương trình đặc trưng của ma trận
* 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 có giá trị riêng là 0 thì không khả nghịch. Ngược lại, nếu mọi giá trị riêng của khác 0 thì khả nghịch.

1. **Một số cách tìm giá trị riêng và vector riêng**

* Giải phương trình tìm các giá trị riêng. Ứng với mỗi giá trị riêng ta giải hệ phương trình tuyến tính thuần nhất .
* Sử dụng phương pháp Danhilepski đưa ma trận 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.

Phương pháp đầu chỉ mang tính chất lý thuyết. Phương pháp Danhilepski tuy đơn giản và nhanh nhưng có một nhược điểm lớn, đó là phải giải một phương trình đa thức bậc n. Đều này khiến cho việc tính toán các giá trị riêng của ma trận có cấp lớn là điều vô cùng khó khăn. Còn phương pháp luỹ thừa tuy tốc độ hội tụ chậm, nhưng lại có thể tính được giá trị riêng của mọi ma trận.

**PHẦN II: PHƯƠNG PHÁP LŨY THỪA ĐỂ TÍNH GẦN ĐÚNG GIÁ TRỊ RIÊNG TRỘI**

**Bài toán: Trong kỹ thuật, ta thường xuyên làm việc với các ma trận lớn và việc tính toán chính xác các trị riêng theo các phương pháp thông thường đã biết rất khó khăn. Điều này đòi hỏi ta phải tìm ra 1 phương pháp khác giúp ta giải quyết các ma trận lớn. Phương pháp tính gần đúng trị riêng của ma trận tiêu biểu nhất là phương pháp luỹ thừa.**

1. **Khái niệm giá trị riêng trội.**

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

Khi đó được gọi là giá trị riêng trội của ma trận . Vector ứng với được gọi là vector riêng trội của ma trận .

Không phải ma trận nào cũng có giá trị riêng trội. Ví dụ ma trận

không có giá trị riêng trội vì 2 giá trị riêng của nó có trị tuyệt đối bằng nhau.

Tương tự, dựa vào các trường hợp nghiệm của phương trình đặc trưng của ma trận ta có thể có nhiều trường hợp khác nhau của các giá trị riêng.

1. **Tìm trị riêng trội đơn, thực.**

Giả sử ma trận cấp n có đủ n trị riêng thực hoặc phức, được sắp xếp theo thứ tự môđun giảm dần sao cho (1) tức là ma trận có trị riêng trội.

Các vector riêng tương ứng là là hệ vector độc lập tuyến tính

Khi đó, ta có vector Y bất kỳ đều là tổ hợp tuyến tính của hệ các vector riêng của .

Ta thực hiện việc tính dãy:

Do ta đã có . Tiếp tục tính

Với ta có

Do điều kiện (1) nên khi thì . Và khi đó, ta có

Hay với m đủ lớn thì

Và

Vậy, đẳng thức trên cho thấy là vector riêng của ma trận .

* Ta có thể và nên thu nhỏ các vector sau mỗi lần tính để giảm khối lượng tính toán vì nếu như lớn thì các hệ số của vector cũng sẽ rất lớn và việc xảy ra tràn số là khó tránh khỏi.

Để tính toán trị riêng của ma trận từ dãy các vector , trong sách giáo trình giải tích số của thầy Lê Trọng Vinh có đưa ra công thức:

Là tỉ số giữa các thành phần thứ của các vector và . Và việc tính toán các vector sẽ kết thúc khi các tỉ số xấp xỉ bằng nhau. Tuy nhiên công thức này không hoàn toàn đúng trong mọi trường hợp. Ta sẽ xét ví dụ sau:

Ví dụ 1: cho ma trận ma trận này có 2 giá trị riêng và thoả mãn điều kiện (1). Ta thực hiện phương pháp luỹ thừa và có được kết quả sau

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 |  | 1 |
| 1 | 2 | 4 | 8 | 16 |  |  |

Có thể thấy ở ví dụ trên, dãy có hội tụ về họ vector chính là vector riêng ứng với trị riêng trội , thế nhưng các tỉ số lại không đổi trong suốt quá trình tính toán. Vậy nếu sử dụng công thức , quá trình lặp sẽ diễn ra mãi mãi. Thay vào đó, người báo cáo xin đưa ra cách làm khác như sau:

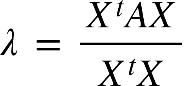
* Sau mỗi lần tính được 1 vector , ta sẽ thu nhỏ vector sao cho hệ số lớn nhất bằng 1
* So sánh các hệ số tương ứng giữa và , nếu chúng gần xấp xỉ nhau hoặc sai lệch không đáng kể thì có thể dừng quá trình tính toán
* Để tính trị riêng, ta chỉ việc lấy vector cuối cùng vừa tính được, tính tiếp mà không thu nhỏ, lấy tỷ số toạ độ lớn nhất của 2 vector và làm trị riêng trội.

Theo cách làm này, ví dụ 1 sẽ cho dãy vector như sau:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 0.5 | 0.25 | 0.125 |  | 0.008 | 0.004 |
| 1 | 1 | 1 | 1 |  | 1 | 1 |

Theo bảng trên, sai lệch giữa 2 vector và là 0.004 tương đối nhỏ và có thể chấp nhận được (nếu cần chính xác hơn có thể tiếp tục tính cho đến khi đạt được sai số mong muốn).

Từ đây ta tìm được vector riêng tương ứng là X



Ta tinh vector riêng theo công thức:

**Tốc độ hội tụ**: ta sẽ xét 2 ví dụ sau:

Xét ma trận quá trình tính trị riêng trội chính xác đến chữ số thứ 3 sau dấu phẩy được thể hiện qua bảng sau:

| Y | AY | A2Y | A3Y | A4Y |
| --- | --- | --- | --- | --- |
| 1 | 0.789 | 0.801 | 0.800 | 0.800 |
| 1 | 1 | 1 | 1 | 1 |

Tương tự với ma trận ta cung có bảng kết quả sau:

| Y | BY | B2Y | … | B66Y | B67Y | B68Y |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 0.5 | 0.941 | … | 0.715 | 0.714 | 0.714 |
| 1 | 1 | 1 | … | 1 | 1 | 1 |

Nhận xét: Với ma trận A, chỉ cần 4 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 68 lần mới đạt được điều tương tự. Lý do là vì ma trận A có 2 giá trị riêng là và . Còn ma trận B có 2 giá trị riêng là và . Ở trên ta đã rút ra kết luận khi thì . Và khi đó, ta có

Với m tương đương với số phép nhân mà ta phải thực hiện. Vậy nếu m tối thiểu nhỏ thì tốc độ hội tụ nhanh. Tức là tỷ số càng nhỏ thì 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.

1. **Tìm trị riêng trội, thực bội r.**

Giả sử thực và bội r, tức là:

(3)

Khi đó, biểu thức (2) sẽ có dạng:

Tương tự như trường hợp trước, khi m đủ lớn, ta có:

Với cách làm tương tự với trường hợp thực, đơn ta sẽ tìm được giá trị riêng và vector riêng tương ứng.

Tuy nhiên, trong trường hợp bội , sẽ có vector riêng độc lập tuyến tính ứng với nhưng ta chỉ có thể tìm được 1 vector. Và trong quá trình tính toán, dựa vào các vector , sẽ không thể xác định được liệu là đơn hay bội 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 trị riêng đơn hay trị riêng có bội.

1. **Trường hợp và thực trái dấu.**

Giả sử , khi đó ta có:

(4)

Khi đó:

Theo giả thiết (4), khi đủ lớn:

Ta nhận thấy rằng, khi các vector liền nhau trong dãy luỹ thừa không có dấu hiệu hội tụ, thế nhưng các vector ở các bước luỹ thừa cùng chẵn hoặc cùng lẻ thì lại cùng nhau tạo thành dãy vector hội tụ. Và đồng thời:

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

Đến đây ta đã có thể kết luận và lần lượt là các vector riêng ứng với và .

1. **Trường hợp và là phức liên hợp.**

Giả sử ma trận A có (phức liên hợp) và:

Biểu thức (2) được viết lại thành:

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

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 , . Khi đó, và là nghiệm của phương trình:

Viết lại phương trình (7): (9)

Công việc cần làm là tìm và , 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 và ), kết hợp với phương trình (8) ta có hệ 3 phương trình:

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:

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

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

Nhân A vào các vế:

Vậy là vector riêng ứng với ,

là vector riêng ứng với .

Ví dụ: Cho ma trận

có 2 trị riêng là và . Quá trình thực hiện phương pháp luỹ thừa được thể hiện qua bảng sau:

|  | 1 | 1 | 1 | 1 |
| --- | --- | --- | --- | --- |
|  | -0.0667 | 1 | 0.4 | 0.1333 |
|  | -0.3049 | 1 | 0.4512 | 0.0609 |
|  | -0.5541 | 1 | 0.6464 | 0.0328 |
|  | -0.9261 | 0.8131 | 1 | 0.0306 |
|  | -0.9539 | -0.0997 | 1 | 0.019 |
|  | 0.5481 | 1 | -0,5697 | -0.009 |
|  | 0.0878 | 1 | -0.0945 | -0.0028 |
|  | -0.134 | 1 | 0.131 | -0.001 |

Ta có thể thấy các vector (kế tiếp nhau hay có số bậc luỹ thừa cùng chẵn hoặc lẻ) không có dấu hiệu hội tụ, có thể kết luận đây là trường hợp trị riêng phức liên hợp.

Ta tiếp tục tính và . Lý do ta phải tính thêm 2 vector vì ta đã thu nhỏ các vector , và ta cần 3 vector liên tiếp nhau nên phải tính tiếp 2 vector kết hợp với tạo thành hệ (10) Chọn 2 toạ độ đầu của và ta lập được phương trình (8):

Giải ta được đây chính là 2 giá trị riêng mà ta cần tìm. Việc tìm các vector riêng tương ứng là khá đơn giản với công thức đã tìm ra.

1. **Tính trị riêng tiếp theo.**

**PHƯƠNG PHÁP XUỐNG THANG TÌM GIÁ TRỊ TRỘI RIÊNG TIẾP THEO.**

***\*Ý tưởng phương pháp:***

- Đối với bài toán tìm nghiệm của đa thức, sau khi tìm được 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:

* 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***

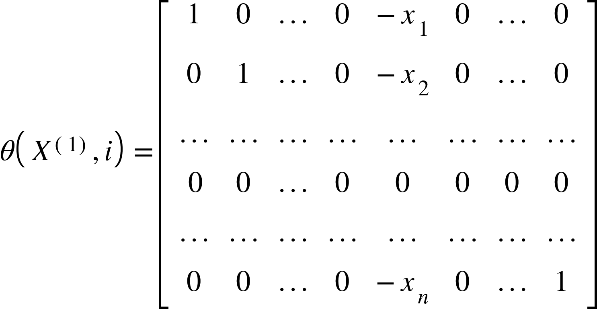
Xét ma trận là ma trận vuông cấp n có đủ n giá trị riêng là và các véc-tơ riêng tương ứng là

Qua phương pháp lũy thừa, ta đã tìm được giá trị riêng trội của A là và

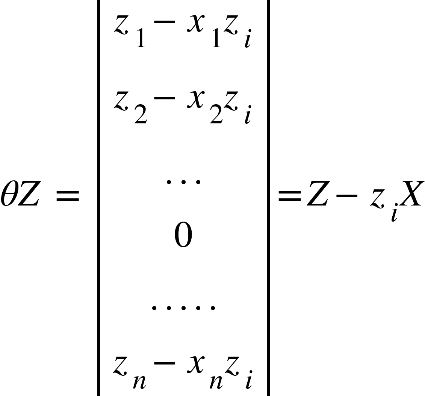
véc-tơ riêng tương ứng là

Chuẩn hóa ,ta được véc-tơ có thành phần thứ i bằng 1.Ta vẫn gọi véc-tơ mới là

Xét ma trận θ phụ thuộc vào véc-tơ và chỉ số i nên kí hiệu là



Với mọi véc-tơ ta đều có:



Do đó, với véc-tơ ta có

*Nhận xét:* với là hàng thứ i của ma trậnA

Ta sẽ chứng minh ma trận có các giá trị riêng và 0; các

véc-tơ riêng tương ứng là và .

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

Thật vậy, ta có:

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

Tiếp theo, với k =2,3,…,n, gọi là phần tử thứ i của . Khi đó ta có:

Suy ra:

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

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

Để tìm véc-tơ riêng tương ứng với của ma trận A, ta xuất phát từ véc-tơ riêng của ma trận . Điều cần làm ở đây là tìm hệ số

Ta có:

Suy ra

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

Suy ra:

=>

+ Nếu ta có , suy ra chính là véc-tơ riêng tương ứng với của ma trận A. Hay nói cách khác, trong trường hợp này

+ Nếu . Vì có phần tử thứ i bằng 1, ta có thể tính bằng cách chia phần tử thứ i của cho , tức là ta có:

**Tổng kết:** Bằng phương pháp lũy thừa, ta tìm được giá trị riêng trội và véc-tơ riêng tương ứng của ma trận A. Áp dụng phương pháp xuống thang, ta biến đổi ma trận A thành ma trận . Lại áp dụng phương phương pháp lũy thừa đối với ma trận ta tìm được là giá trị riêng trội của , đồng thời là giá trị riêng trội tiếp theo của A. Từ véc-tơ là véc-tơ riêng ứng với của ma trận , ta tìm được véc-tơ ứng với 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.

**PHẦN III: THUẬT TOÁN VÀ CHƯƠNG TRÌNH**

* **Thuật toán tổng quan:**

Tổng quan:

-        B1: Nhập ma trận vuông cấp n và vector Y

-        B2:  Tính các vector  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 tính đến 200 lần mà ko có dấu hiệu thõa mãn hai trường hợp trên thì 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. Kết thúc chương trình.

**1. Gói xử lý dữ liệu**

**- Gói Tính**<math xmlns="http://www.w3.org/1998/Math/MathML"><msup><mi>A</mi><mi>m</mi></msup><mi>Y</mi></math> **:**

Lưu các cột  <math xmlns="http://www.w3.org/1998/Math/MathML"><msup><mi>A</mi><mi>m</mi></msup><mi>Y</mi></math> liên tiếp thành một mảng B

Ngay từ đầu, véc-tơ Y được lưu tại vị trí B[0]

Đầu vào của gói này là ma trận A, 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.

* **Gói Chuẩn Hóa Vector:**

Đầu vào là một vector M

Tìm số có giá trị tuyết đối lớn nhất trong vector M là maxi

Vector chuẩn hóa là M’ = M/maxi

**Gói Kiểm Tra**

Đầu vào là mảng B và số m, n

Tính F = B[m] - B[n]

Tìm số có trị tuyệt đối lớn nhất trong F

Trả về trị tuyết đối của số đó

Đầu vào là ma trận A và vector Y:

Tạo một mảng B, lưu ví trí B[0] bằng vector Y

Tính 3 vector đầu tiên là AY, A^2Y, A^3Y lưu 3 vector này vào các vị trí tiếp theo của mảng B

Nếu Kiem\_tra(B, m - 1, m - 2) <= E: là TH1, kết thúc vòng lặp

Nếu Kiem\_tra(B, m  - 1, m - 3) <= E: là TH3, kết thúc vòng lặp

Nếu lặp đến 200 lần mà ko có dấu hiệu hội tụ của hai trường hợp 1 và 3 thì là TH4, kết thúc vòng lặp

( E là sai số mong muốn giữa các vector)

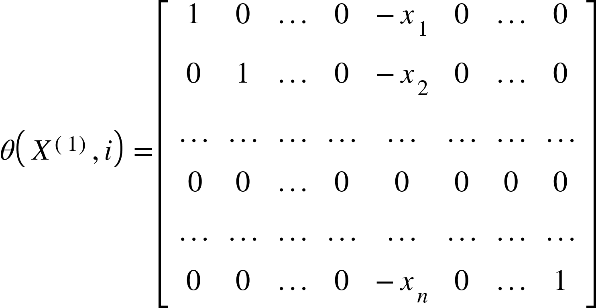
Sau mỗi lần kiểm tra điều kiện xong thì ta tính A^mY tiếp theo và chuẩn hóa nó.

Lưu vector vừa chuẩn hóa vào vào các vị trí tiếp theo của mảng B

Gói này trả về số lần tính m, TH và mảng B

2 **Gói chuyển ma trận:**

Đầu vào là ma trận A và vector riêng của A

Tạo ma trận 



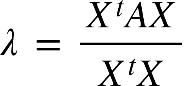
Tính A’ = A

Trả về A’

3 Tính toán các trường hợp:

Trường hợp 1:

* Tìm được vector riêng: X



  Tìm giá trị riêng:

* Tính ma trận A mới  để tìm trị riêng tiếp theo.
* Thay đổi vector Y

Trường hợp 3:

* Tính thêm   và 
* Tìm tọa lớn nhất của   và 
* Tính được trị riêng trái dấu là 2 căn của tỷ số giữa 2 toạ độ trên
* Tính vector riêng theo công thức.
* Tính ma trận A mới để tìm trị riêng tiếp theo.

Trường hợp 4:

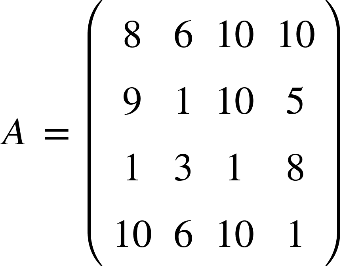
* Tính thêm 
* Thiết lập và giải phương trình (10) để tìm 2 giá trị riêng phức
* Tính vector riêng theo công thức đã tìm
* Kết thúc chương trình

**4 Chương trình:**

#Khai báo thư viện  
import numpy as np  
import math  
#Load file  
f = open("input.txt", "r")  
a = f.readline()  
n = int(a)  
print('Ma trận cỡ: {} x {}'.format(n, n))  
b = f.readline()  
b = b.split()  
X = []  
for i in (b):  
 X.append(float(i))  
X = np.array(X)  
Y = X.reshape(n, 1)  
#print('Vector Y là: ',X)  
# Y = np.random.rand(n, 1)  
c = f.read()  
c = c.split()  
A = []  
for i in c:  
 A.append(float(i))  
A = np.array(A).reshape(n, n)  
print('Ma trận A là: ')  
print(A)  
print('\_'\*100)  
  
E = 0.1  
#Chuẩn hóa  
def Chuan\_Hoa(M):  
 maxi = abs(M[0][0])  
 c = M[0][0]  
 for i in range (len(M)):  
 if maxi < abs(M[i][0]):  
 maxi = abs(M[i][0])  
 c = M[i][0]  
 matrix = M / c  
 return matrix  
#Kiểm tra  
def Kiem\_tra(B, m, n):  
 A = B[m] - B[n]  
 return max(abs(A))  
#Lũy thừa  
def Luy\_thua(A, B, m):  
 M = A.dot(B[m])  
 return M  
  
#Xử lí  
def Xu\_li(A, Y):  
 B = []  
 B.append(Y)  
 Z = np.zeros((n, 1))  
  
 B.append(Luy\_thua(A, B, 0))  
 B.append(Luy\_thua(A, B, 1))  
 B.append(Luy\_thua(A, B, 2))  
 m = 3  
 TH = 4  
 while True:  
 M = []  
 F = B[-1]  
 for i in range (len(F)):  
 x = round(F[i][0], 4)  
 M.append(x)  
 M = np.array(M).reshape(n, 1)  
 if sum(F == Z) == n:  
 sys.exit()  
 if m == 50:  
 break  
 if Kiem\_tra(B, m - 1, m - 2) <= E:  
 TH = 1  
 break  
 if Kiem\_tra(B, m - 1, m - 3) <= E:  
 TH = 3  
 break  
 m += 1  
 B.append(Chuan\_Hoa(Luy\_thua(A, B, m - 1)))  
  
  
 return m, TH, B  
#Chuyển ma trận xuống thang  
def Chuyen\_MT(A, VTR):  
 B = A  
 V = VTR.reshape(n,)  
 M = np.eye(n, dtype=float)  
 maxi\_1 = V.max()  
 index = 0  
 for i in range (n):  
 if V[i] == maxi\_1:  
 index += i  
 break  
 E = M[: , index: index + 1] - VTR.reshape(n, 1)  
 E[index][0] =0  
 M[:, index: index + 1] = E  
 A = M.dot(A)  
 return A  
for i in range(n):  
 m, TH, B = Xu\_li(A, Y)  
  
 if TH == 1:  
 VTR = B[-1]  
 VTR\_T = VTR.reshape(1, n)  
 AX = A.dot(VTR)  
  
 XTAX = (VTR\_T.dot(AX))  
 XTX = (VTR\_T.dot(VTR))  
 lamda = XTAX[0] / XTX[0]  
 lamda = round(lamda[0], 4)  
 print('Sau {} lần lặp'.format(m))  
 print('Giá trị riêng là: ', lamda)  
 print('Vector riêng tương ứng là: ')  
 print(VTR)  
 A = Chuyen\_MT(A, VTR)  
 Y = np.random.rand(n,1)  
 print('\_' \* 100)  
 elif TH == 3:  
 AmY = A.dot(B[-1])  
 Am1Y = A.dot(AmY)  
 Am2Y = A.dot(Am1Y)  
 lamda\_N = max(Am2Y)/max(AmY)  
 lamda\_N = math.sqrt(lamda\_N[0])  
  
 VTR\_N = Am1Y + lamda\_N \* AmY  
 VTR\_N = Chuan\_Hoa(VTR\_N)  
 print('Sau {} lần lặp'.format(m))  
 print('Giá trị riêng là: ', round(lamda\_N, 4))  
 print('Vector tương ứng là: ')  
 print(VTR\_N)  
 A = Chuyen\_MT(A, VTR\_N)  
 Y = np.random.rand(n, 1)  
 print('\_' \* 100)  
  
 elif TH == 4:  
 M = B[-1]  
 AmY = A.dot(M)  
 Am1Y = A.dot(AmY)  
 a\_1 = Am1Y[0][0]  
 a\_2 = Am1Y[1][0]  
 b\_1 = AmY[0][0]  
 b\_2 = AmY[1][0]  
 c\_1 = M[0][0]  
 c\_2 = M[1][0]  
 a = 1  
 b = (a\_1 \* c\_2 - c\_1 \* a\_2) / (c\_1 \* b\_2 - b\_1 \* c\_2)  
 c = (b\_1 \* a\_2 - a\_1 \* b\_2) / (c\_1 \* b\_2 - b\_1 \* c\_2)  
 print('Phươnng trình tìm đc: Z^2 + {}Z + {} = 0 '.format(b, c))  
 print()  
 denta = b \*\* 2 - 4 \* a \* c  
 if denta >= 0:  
 print('Phương trình có denta >= 0')  
 else:  
 lamda\_1 = complex(-b / (2 \* a), -math.sqrt(abs(denta)) / (2 \* a))  
 lamda\_2 = complex(-b / (2 \* a), math.sqrt(abs(denta)) / (2 \* a))  
 VTR\_1 = Am1Y - lamda\_1 \* AmY  
 VTR\_2 = Am1Y - lamda\_2 \* AmY  
 print('Giá trị riêng là: {}'.format(lamda\_1))  
 print('Vector riêng tương ứng là: ')  
 print(VTR\_1)  
 print('\*' \* 100)  
 print('Gía trị riêng là: {}'.format(lamda\_2))  
 print('Vector riêng tương ứng là: ')  
 print(VTR\_2)  
 break

**Ví dụ:**

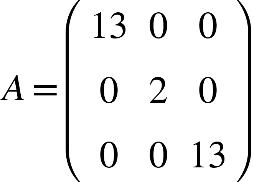
Sau đây sẽ là một số ma trận được chạy chương trình trên trình trên

Trường hợp 1: Trường hợp trị riêng trội thực, đơn (bội một):

Text

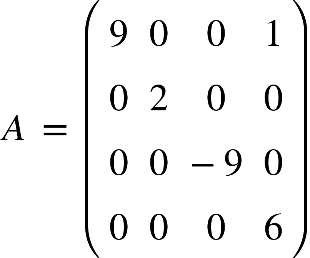
Description automatically generated

TH2: Trường hợp trị riêng trội thực, bội:



Graphical user interface, text, application

Description automatically generated

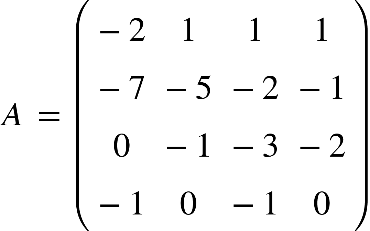
Trường hợp 3: Giá trị riêng trội đối nhau

 \*

Text

Description automatically generated

TH4 : Giá trị riêng phức:



Text

Description automatically generated

**Phần IV: Kết luận**

1. Ưu và nhược điểm của phương pháp luỹ thừa:
   1. Ưu điểm:
      1. Có thể tính trị riêng của các ma trận cấp lớn
      2. Dễ hiểu, dễ nhớ
      3. Linh hoạt, có thể thay đổi để tìm trị riêng nhỏ nhất
   2. Nhược điểm:
      1. Tốc độ hội tụ không ổn định
      2. Một số ma trận không thể áp dụng phương pháp này do khi xấp xỉ nhiều lần, tính hội tụ không còn đúng.
      3. Dễ bị tràn số nếu không thu nhỏ vector

Phương pháp luỹ thừa có ứng dụng quan trọng trong nhiều lĩnh vực khoa học kỹ thuật, đặc biệt là tin học.