**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC**

**\*\*\*\*\*\*\*\*\*\***

****

**BÁO CÁO MÔN GIẢI TÍCH SỐ**

**ĐỀ TÀI:**

**“Phương pháp lặp đơn tìm nghiệm của phương trình “**

**Giáo viên hướng dẫn : TS. Hà Thị Ngọc Yến**

**Sinh viên : Lê Hải Phong**

**Mã số sinh viên : 20200460**

***Hà Nam , 1 – 2022***

**LỜI NÓI ĐẦU**

Thuật toán nằm trong bộ môn giải tích số. Trong [toán học](https://vi.wikipedia.org/wiki/To%C3%A1n_h%E1%BB%8Dc) và [khoa học máy tính](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh), một thuật toán, còn gọi là giải thuật, là một [tập hợp hữu hạn](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_h%E1%BB%A3p#L%E1%BB%B1c_l%C6%B0%E1%BB%A3ng_c%E1%BB%A7a_t%E1%BA%ADp_h%E1%BB%A3p_-_H%E1%BB%AFu_h%E1%BA%A1n_v%C3%A0_v%C3%B4_h%E1%BA%A1n) các hướng dẫn [được xác định rõ ràng](https://vi.wikipedia.org/w/index.php?title=%C4%90%C6%B0%E1%BB%A3c_x%C3%A1c_%C4%91%E1%BB%8Bnh_r%C3%B5&action=edit&redlink=1), có thể thực hiện được bằng máy tính, thường để giải quyết một lớp vấn đề hoặc để thực hiện một phép tính. Nếu người học nắm chắc bộ môn giải tích số nói chung và thuật toán nói riêng sẽ dần tiếp cận với nền tri thức mới theo kịp với thời đại 4.0 làm cho con người phát triển toàn diện trong mọi lĩnh vực . Sau khi học và tìm hiểu trong một thời gian bài cáo của em với chủ đề “ Phương pháp lặp đơn tìm nghiệm của phương trình” đã phần nào giải đáp và khắc sâu cho em trong việc ứng dụng vào thực tế.

**I.GIỚI THIỆU PHƯƠNG PHÁP**

Trong toán học ; bài toán tìm nghiệm của phương trình (giải phương trình) là một bài toán nổi tiếng được nghiên cứu từ xưa đến nay. Cụ thể hơn đó là bài toán tìm nghiệm của phương trình đa thức

đặc biệt là đa thức có bậc 3 hoặc bậc lớn hơn hay là các loại phương trình phức tạp thì việc đi tìm nghiệm đúng của phương trình đó rất khó khăn. Đặc biệt khi chúng ta làm việc với các phương trình có nghiệm là các số thập phân vô hạn thì chúng ta phải làm tròn số ; lúc đó chúng ta gọi nghiệm đó là nghiệm gần đúng.

Thực tế có rất nhiều phương pháp để tìm được gần đúng của phương trình :

+) Phương pháp chia đôi

+) Phương pháp dây cung

+) Phương pháp tiếp tuyến

+) Phương pháp lặp đơn

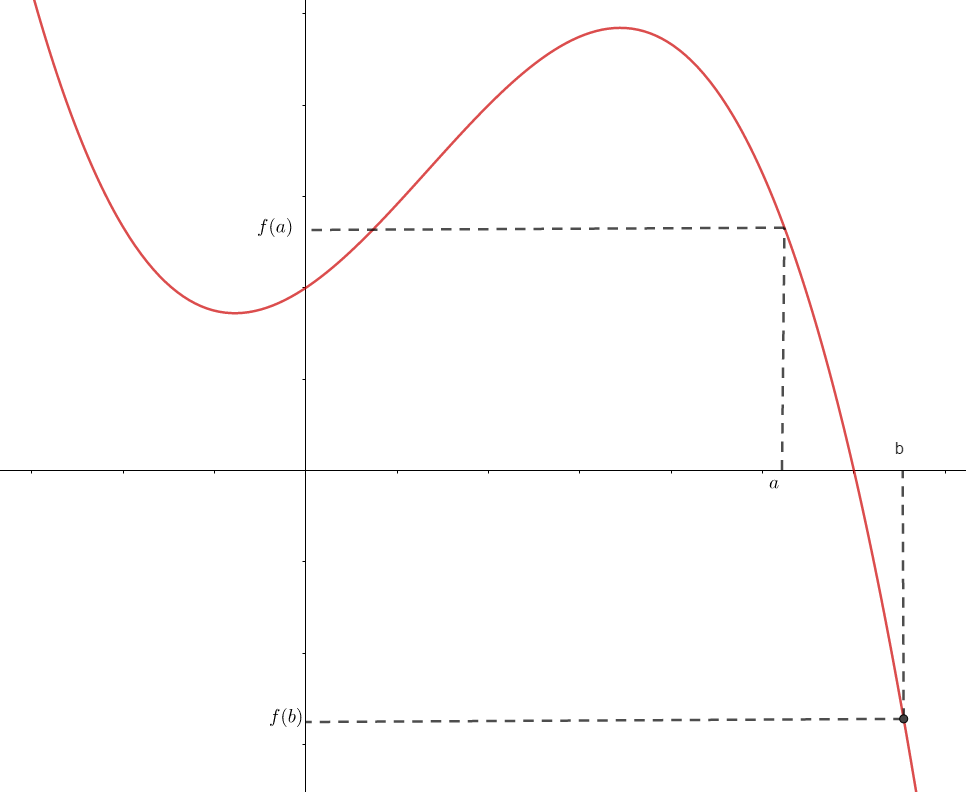
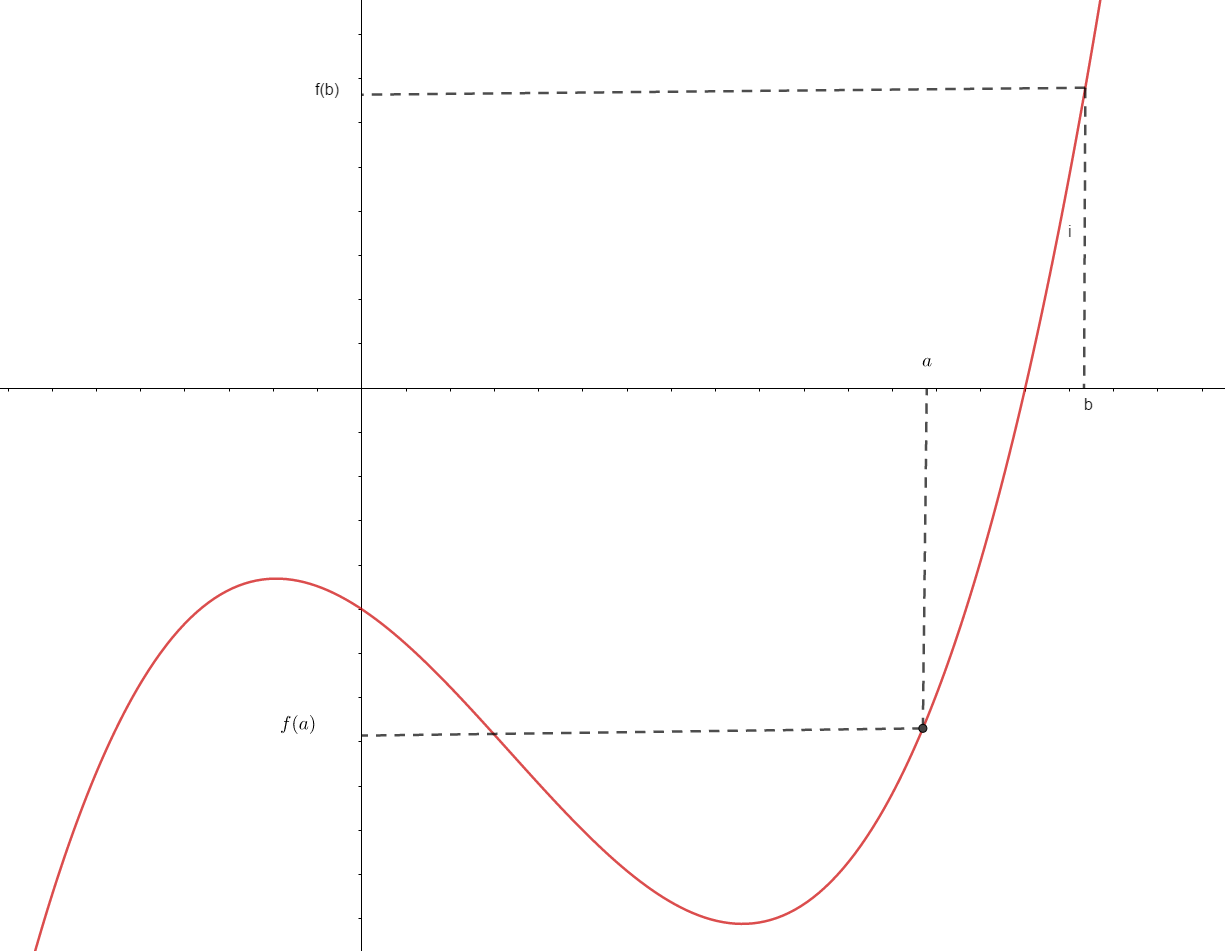
Mỗi phương pháp đều có ưu nhược điểm khác nhau nhưng đều có mục đích chung là tìm nghiệm gần đúng của phương trình ; ở bài báo này em sẽ trình bày bản chất; kĩ thuật , thuật toán, ví dụ của phương pháp lặp đơn( lặp cổ điển). Chúng ta sẽ điều chỉnh nghiệm dần dần theo kỹ thuật của phương pháp lặp đơn . Vậy điều chỉnh nghiệm như thế nào ? Chúng ta hãy đi tìm hiểu qua báo cáo lần này.

**II) NHỮNG LÝ THUYẾT CƠ BẢN**

1. **Khoảng cách li nghiệm của  phương trình**

a) Định Nghĩa : Khoảng đóng hay khoảng mở trên đó tồn tại duy nhất 1 nghiệm của phương trình thì khoảng đó gọi là khoảng cách li nghiệm

b) Điều kiện:  là khoảng cách li nghiệm của phương trình  khi và liên tục trên 







**Ví dụ 1:** Cho phương trình 



Đặt 

luôn liên tục trên R

Xét trên , ta thấy 

là khoảng li nghiệm của phương trình 

Ta cũng thấy : Trên tồn tại duy nhất 1 nghiệm là nghiệm của phương trình

**Nhận xét:** +) Nếu hàm số liên tục trong và , tồn tại và không đổi dấu trong  thì trong phương trình luôn có nghiệm duy nhất

+) Nếu hàm số  luôn tăng hoặc luôn giảm trên miền xác định D và có nghiệm thực trên D thì đây là nghiệm duy nhất của phương trình

**2) Hàm số co trên **

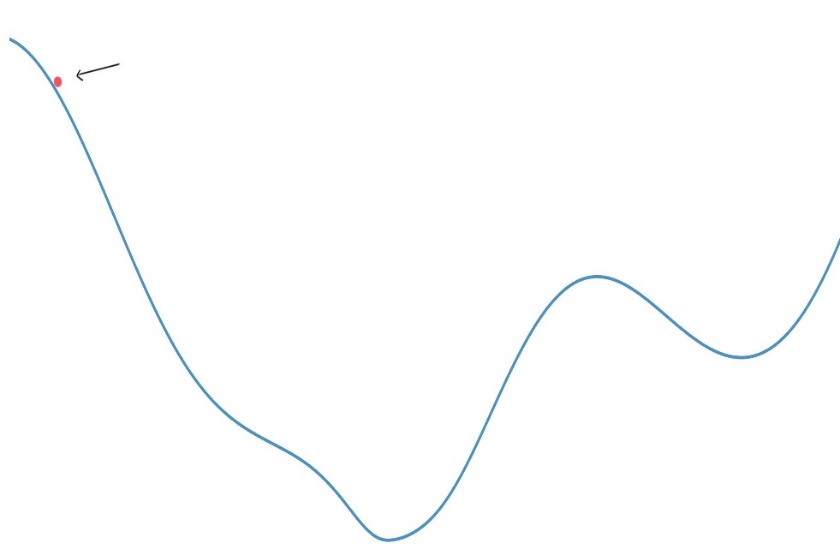
**Định nghĩa :** Hàm số được gọi là hàm co trên đoạn **** nếu tồn tại số thoả mãn  sao cho 

**Định lý :** Nếu hàm số  liên tục trên **** và có đạo hàm liên tục trên **** và tồn tại số q thoả mãn  sao cho  thì hàm số được gọi là hàm số co trên ****

**3) Phương pháp Gradient Descent tìm giá trị lớn nhất và giá trị nhỏ nhất của hàm số** 

**a) Ý tưởng phương pháp**

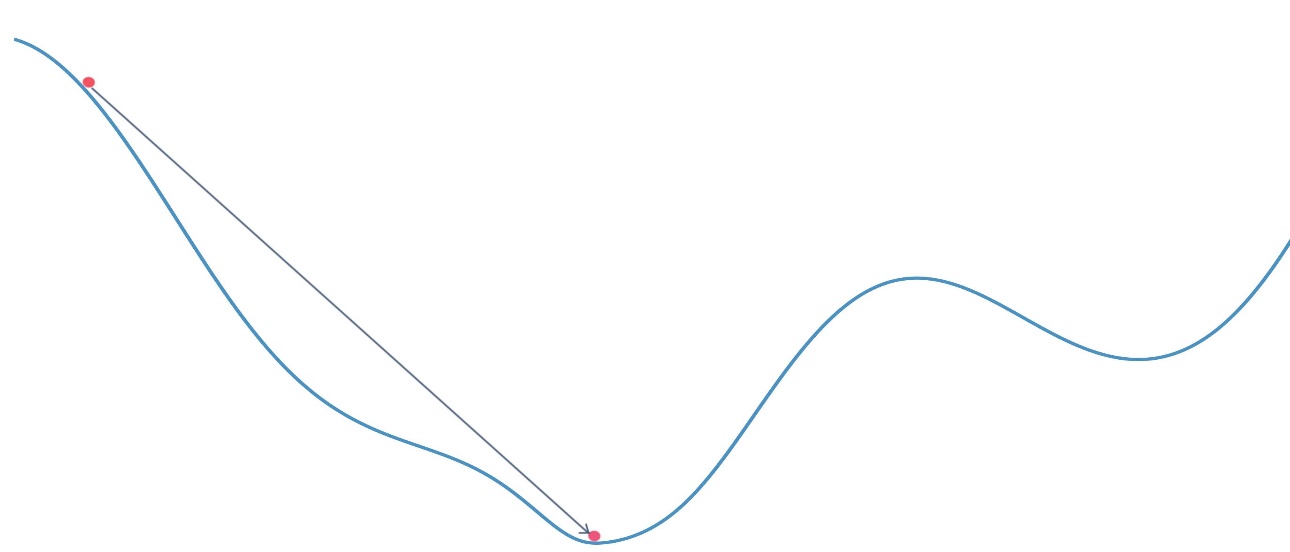
Giả sử một người đang lạc ở trên đỉnh núi, trời sắp tối và đỉnh núi rất lạnh khi về đêm nên cần phải đi xuống thung lũng để dựng trại, xung quanh toàn là sương mù nên anh ta không thể xác định chính xác thung lũng ở đâu. Làm thế nào để đi xuống thung lũng? Vì đam mê toán học, trong đầu anh ấy lóe lên ý tưởng rằng sẽ cố gắng xác định đồ thị hàm số biểu diễn bề mặt của dãy núi, không chần chừ,anh ta bắt tay vào thực hiện. Bằng một cách nào đó anh ấy đã tìm ra được hình dạng của dãy núi như sau:



Bằng một phép nội suy, anh ta suy ra được phương trình của đồ thị hàm số trên:

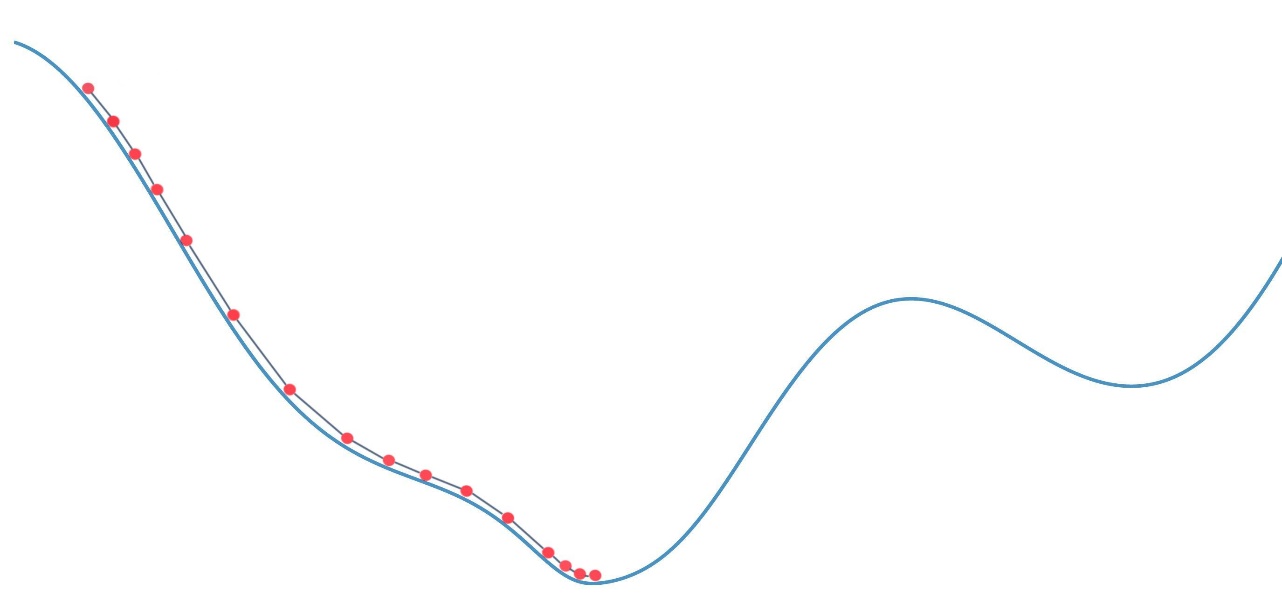
Đạo hàm nó:

Giải phương trình đạo hàm bằng 0 và tìm được thung lũng gần đó:



**Anh ta nói xạo đó, dễ gì mà tìm được hàm số, nói gì đến tính toán!** **Trong thực tế không ai làm như vậy cả.**

Thực chất anh ta làm theo cách bên dưới:  
–*Bước 1*: Khảo sát vị trí đang đứng và tất cả những vị trí xung quanh có thể nhìn thấy, sau đó xác định vị trí nào hướng xuống dốc nhiều nhất.  
–*Bước2*: Di chuyển đến vị trí đã xác định ở bước 1 .  
–*Bước 3*: Lặp lại hai các bước trên đến khi nào tới được thung lũng.



Hay nói cách khác là đi men theo mặt dốc, đây chính là ý tưởng của thuật toán Gradient Descent.

**b) Xây dựng công thức**

Đối với cực tiểu

* 𝑥\* với 𝑓 ' (𝑥\*) = 0
* Tại vị trí 𝑥n bất kì

Nếu đạo hàm của hàm số tại 𝑥n : 𝑓 ′ (𝑥n ) > 0 thì 𝑥n nằm về bên phải so với 𝑥\* (và ngược lại). Để điểm tiếp theo 𝑥n+1 gần với 𝑥\* hơn, chúng ta cần di chuyển 𝑥n+1 về phía bên trái, tức về phía âm. Nói cách khác, chúng ta cần di chuyển ngược dấu với đạo hàm:

𝑥n+1 = 𝑥n + Δ. Trong đó Δ là một đại lượng ngược dấu với𝑓 ′ (𝑥n )

* 𝑥n càng xa 𝑥\* về phía bên phải thì 𝑓 ′ (𝑥n ) càng lớn hơn 0 (và ngược lại). Vậy, lượng di chuyển Δ , một cách trực quan nhất, là tỉ lệ thuận với −𝑓 ′ (𝑥n ). Tương tự với cực đại.
* Các nhận xét phía trên cho chúng ta có CTTQ:

𝑥n+1 = 𝑥n + sign ∗ 𝜂 ∗ 𝑓 ′ (𝑥n ) (\*)

(Với sign = -1 khi lặp tìm cực tiểu và sign = 1 khi lặp tìm cực đại)

Với 𝜼 (𝒆𝒕𝒂) là một số dương được gọi là learning rate (tốc độ học)

=> 𝑥n ≈ 𝑥\*

***\*Áp dụng vào bài toán tìm GTLN, GTNN của hàm f (x)=0 trên [a,b]***

Bắt đầu xuất phát từ 𝑥0 = a và di chuyển dần về điểm b.

* B1: Nếu 𝑓 ′ (𝑥0 ) < 0 thì điểm cực trị tiếp theo nếu có là cực tiểu. (Tương tự nếu 𝑓 ′ (𝑥0 ) > 0 thì điểm tiếp theo là cực đại).
* B2: Ta sẽ dùng CTTQ (\*) với sign = -1( hoặc 1) để lặp *x0* đến đủ gần x\*. Sau đó ta sẽ tăng 𝑥0 thêm 1 khoảng step đủ nhỏ để *x0* vượt qua 𝑥\* : *x0 = x0 +step*. Sau đó quay lại B1 với *x0* mới.
* Quá trình lặp lại đến khi 𝑥0 = b

Sau khi tìm được các điểm cực đại, cực tiểu chúng ta tiến hành so sánh 𝑓(𝑥\* ), 𝑓(𝑎), 𝑓(𝑏) từ đó tìm giác các giá trị lớn nhất (Max) và nhỏ nhất (Min) của hàm đã cho.

**c) Thuật toán sử dụng cho bài toán lặp đơn**

**Gói findMax(f,a,b):**

*Input:* hàm f, đoạn [a,b]

*Output:* GTLN của hàm f trên đoạn [a,b]

B1: Gán cuctri = Gradient\_descent(f,a,b) là danh sách các điểm cực trị và bao gồm a,b

B2: Tính các giá trị f(x) với x là các điểm trong cuctri

B3: Trả về giá trị lớn nhất của các giá trị tìm được trong bước 2

**Gói approxive\_derivative (f, x0):**

*Input:* hàm f, điểm x0

*Output:* xấp xỉ đạo hàm của hàm f tại điểm x0

B1: Trả về (f(x0+1e-6)-f(x0-1e-6))/2e-6. Kết thúc gói

**Gói fixeta(f,sgn,dfx0,x0):**

*Input:* hàm f, sgn là dấu của đạo hàm hiện tại, điểm x0, df là xấp xỉ đạo hàm tại x0

*Output:* Giá trị eta hợp lý

B1: Gán eta = 1e-14

B2: Nếu sgn\* approxive\_derivative(f,x0+sgn\*eta\*dfx0) >= 0 và |eta\*dfx0| < 1 thì sang B3, ngược lại sang B4

B3: Gán eta = 2\*eta

B4: Nếu sgn\* approxive\_derivative(f,x0+sgn\*eta\*dfx0) <= 0 thì sang B5, ngược lại sang B6

B5: Gán eta = eta/2

B6: Trả về eta

**Gói Gradient\_descent(f,a,b):**

*Input:* hàm f, đoạn [a,b]

*Output:* Các điểm cực trị trên đoạn [a,b] và cả 2 điểm a,b

B1: Gán sgn=1, xr là mảng chứa a, x0 = a, eta = 1e-14

B2: Nếu x0 < b thì sang B3, ngược lại đến B8

B3: Gán x0 = x0 + 5e-7, df = approxive\_derivative (f, x0)

B4: Nếu thì sang B5, ngược lại sang B7

B5: Gán df = approxive\_derivative(f,x0), eta = fixeta(f,sgn,df,x0), x0=x0+sgn\*eta\*df

B6: Nếu x0 > b chuyển sang B8

B7: Gán sgn = -sgn, nếu a < x0 < b thì thêm x0 vào mảng xr, quay lại B2

B8: Thêm b vào mảng xr, sau đó trả về xr, kết thúc gói.

ETA = 1e-15

EPSILON = 1e-6

def f\_(f, x,eps=EPSILON):

    return (f(x+eps)-f(x-eps))/(2\*eps)

def gradient\_descent(f, a, b):

    def fixeta(dfx0,x0):

        eta = ETA

        while sgn\*f\_(f,x0+sgn\*eta\*dfx0)>=0 and eta<1:

            eta \*= 2

        while sgn\*f\_(f,x0+sgn\*eta\*dfx0)<=0:

            eta /= 2

        return eta

    sgn, count = 1 if f(a+EPSILON) > f(a) else -1, 0

    xr = [a]

    x0 = a + 0.5\*EPSILON

    eta = ETA

    while x0 < b:

        dx = f\_(f, x0)

        while abs(dx) > 0.001\*EPSILON:

            dx = f\_(f, x0)

            eta = fixeta(dx,x0)

            x0 += sgn\*eta\*dx

            count += 1

            if x0 > b:

                break

        sgn = -sgn

        if x0 > a and x0 < b:

            xr.append(x0)

        x0 += 0.5\*EPSILON

    xr.append(b)

    return xr

def findMax(pof, a, b):

    if a>b:

        a,b = b,a

    lastPoints = gradient\_descent(f, a, b)

    result = max(map(f, lastPoints))

    return result

def findMin(f, a, b):

    if a>b:

        a,b = b,a

    lastPoints = gradient\_descent(f, a, b)

    result = min(map(f, lastPoints))

    return result

**III ) NỘI DUNG PHƯƠNG PHÁP LẶP ĐƠN**

1. **Input và output bài toán**
2. **Input bài toán**

**+**) 2 số a;b , 1 hàm số 1 giá trị ban đầu ,số chữ số làm tròn :e

1. **Output bài toán**

Nghiệm của phương trình:

**2) Ý tưởng bài toán**

Mục đích của bài toán là chúng ta phải đi tìm nghiệm đúng của phương trình .

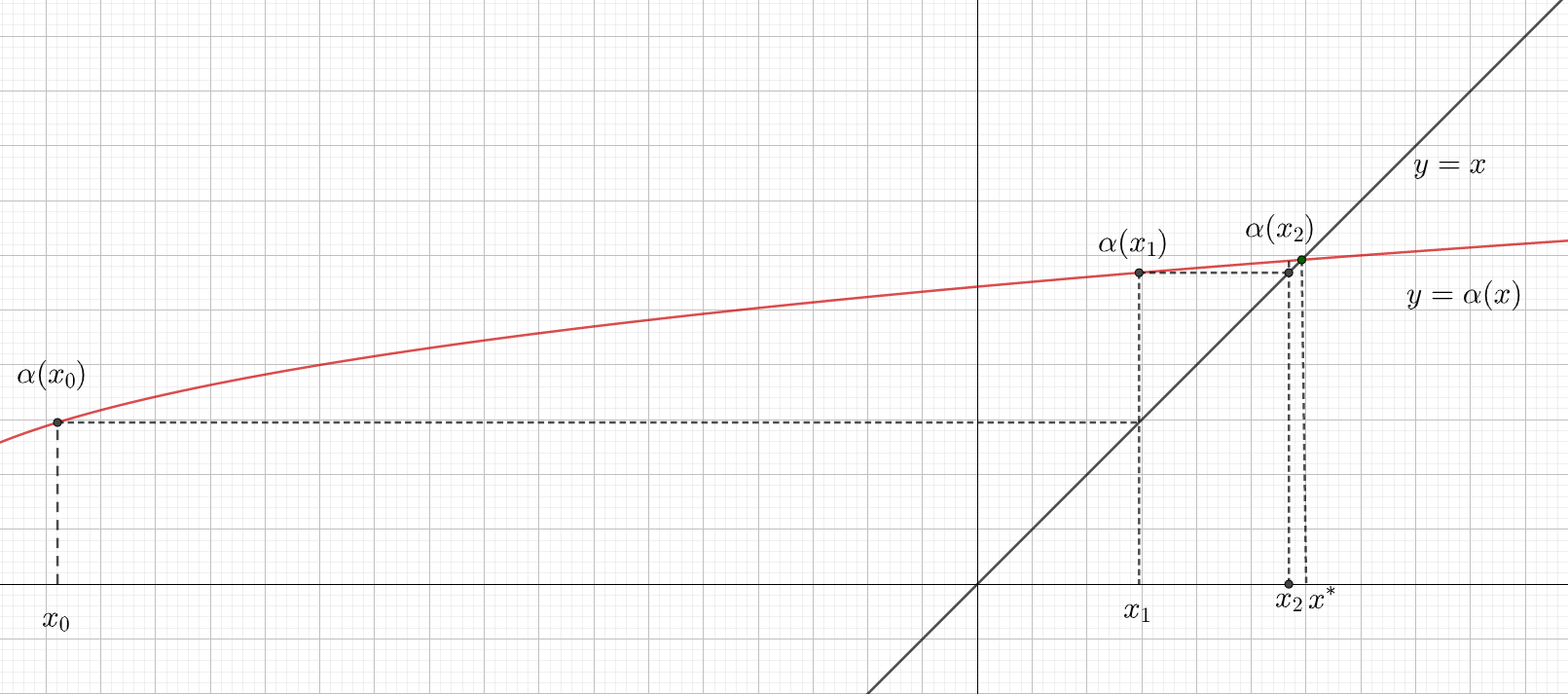
Từ phương trình  ta biến đổi tương đương về phương trình 

=>Nghiệm của phương trình cũng chính là nghiệm của phương trình hay chính là hoành độ giao điểm của 2 đồ thị hàm số và 

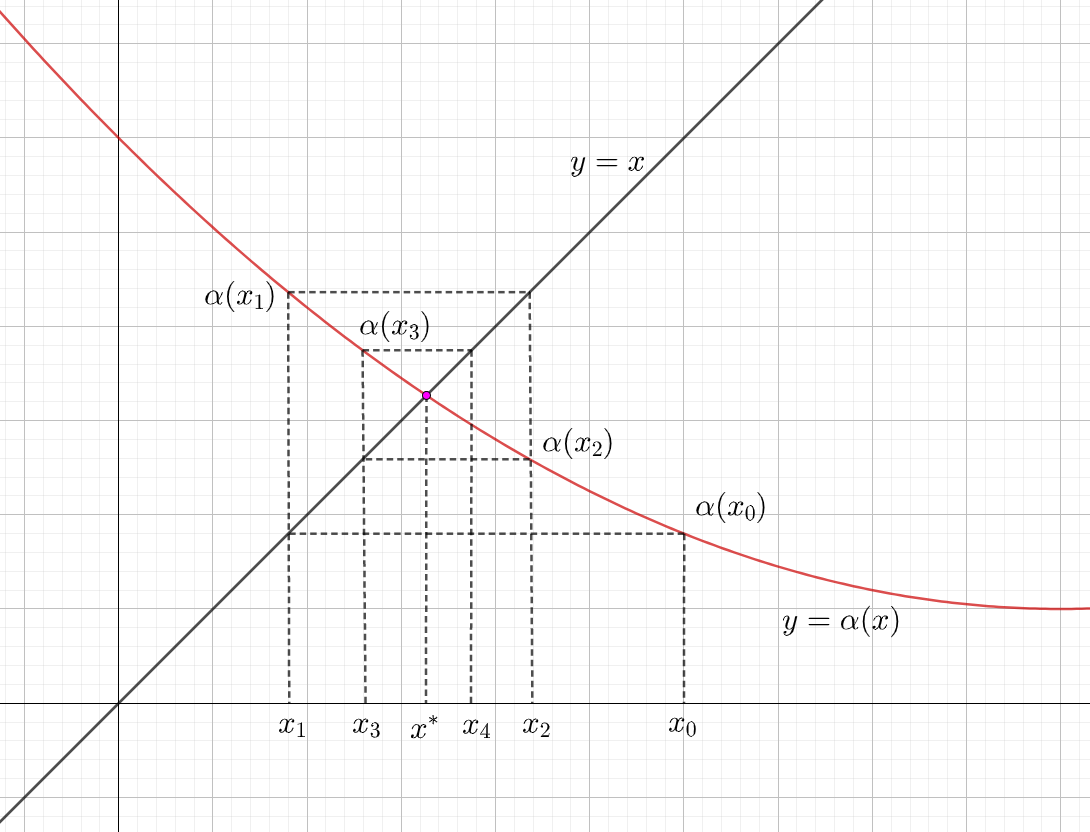
Với giả thiết có phương trình và giá trị ban đầu với  là khoảng cách li nghiệm , ta phải đi tìm nghiệm gần đúng của phương trình 

Với ý tưởng của phương pháp trên ta biến đổi từ phương trình  về phương trình và đi tìm hoành độ giao điểm của 2 đồ thị hàm số và . Ta sẽ điều chỉnh giá trị ban đầu tiến sát với giá trị nghiệm của phương trình bằng 1 kỹ thuật duy nhất nhưng làm lại nhiều lần . Kỹ thuật đó được gọi là lặp đơn.

Ta sẽ xét các đồ thị sau :



Đồ thị 1



Đồ thị 2

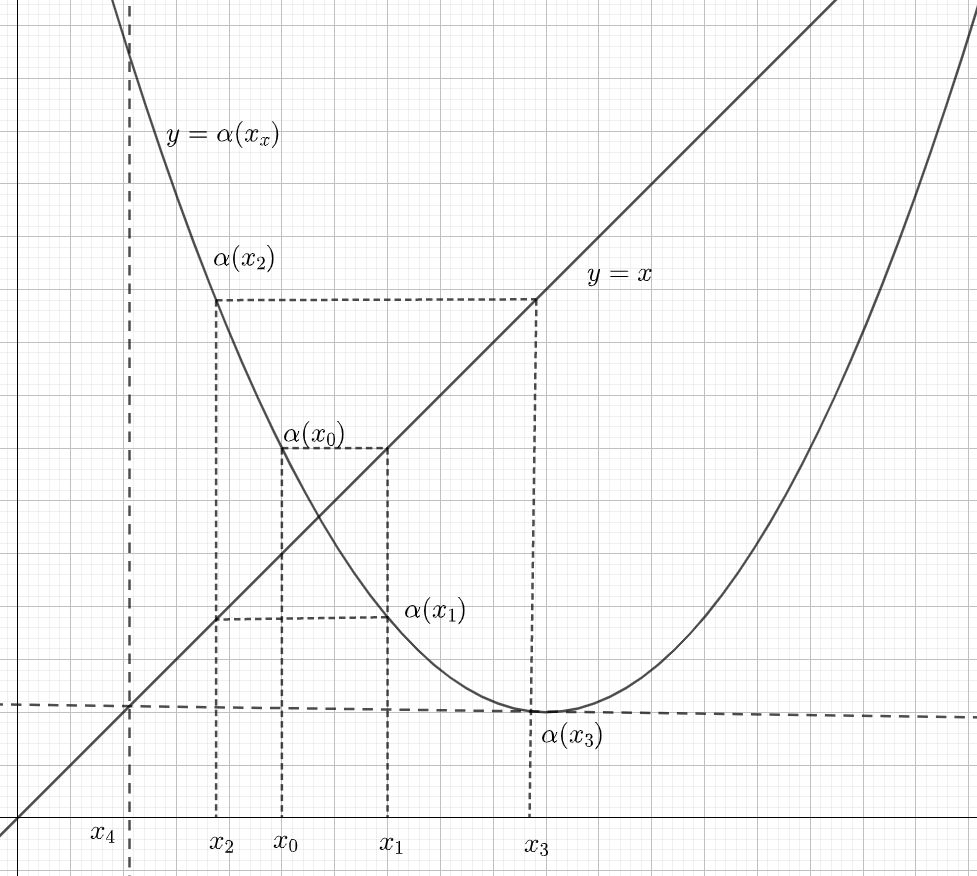
Ta lấy giá trị bất kì rồi ta tính ; tiếp theo ta tính , tiếp theo tính giá trị , ta làm lại nhiều như thế , dựa vào đồ thị ta thấy : Các giá trị  đang tiến dần đến giá trị là hoành độ giao điểm của 2 đồ thị hàm số và .

**Nhận xét:** Ở 2 đồ thị thì các giá trị  tiến đến giá trị theo các con đường khác nhau :

**+)** Ở đồ thị thứ nhất các giá trị tiến đến giá trị  theo 1 hương duy nhất **+)** Ở đồ thị thứ hai các giá trị  tiến đến giá trị  theo hai hướng khác nhau

****Tuỳ vào tính chất của hàm số  mà các giá trị  tiến về giá trị  theo các hướng khác nhau. Nhưng không phải lúc nào thì các giá trị  cũng tiến đến giá trị .

Ta xét đồ thị sau



Đồ thị 3

Ta cũng lặp lại phương pháp tính như trên nhưng ta thấy các giá trị  đang tiến ra xa giá trị .

**3) Xây dựng công thức và sự hội tụ của phương pháp**

**a)Xây dựng công thức**

Giả thiết : Cho phương trình có khoảng cách li nghiệm là .

Chọn bất kì thuộc  ; ta tính  rồi đặt 

Tiếp theo ta lại tính và đặt 

Cứ lặp lại các bước trên ta sẽ tính được 

Quá trình trên gọi là quá trình lặp và n là thứ của phép lặp

* Xây dựng được dãy số tổng quát : 

**b) Sự hội tụ của phương pháp**

Định nghĩa về sự hội tụ của phương pháp : Quá trình lặp trên gọi là hội tụ khi dãy  hội tụ khi 

Nếu dãy hội tụ thì 

Do  liên tục trên  và  nên :



=> là nghiệm của phương trình hay phương trình 

Vậy quá trình trên sẽ hội tụ nếu dãy  :  hội tụ

**4. Cơ sở tìm nghiệm của phương pháp lặp đơn**

Xét phương trình với là khoảng cách li nghiệm của phương trình ;  là 1 nghiệm của phương trình  với . Ta xây dựng 1 dãy số tổng quát:

Ta sẽ chứng minh nếu hàm số là hàm co thì dãy số  sẽ hội tụ về 

Hàm số liên tục và có đạo hàm liên tục trong .

Ta có : 

Đặt 

Bây giờ ta chứng mình với là hàm co tức là  thì 

Theo định lý Lagrange cho hàm số , ta có :

với c là giá trị trung gian nằm giữa và 

Do : =>



Chứng minh tương tự, ta có : 

Nhân vế với vế của các bất đẳng thức trên ; ta có : 

Vì  

Vậy với  thì 

**5.Gói thuật toán kiểm tra điều kiện co của hàm số**

**Bước 1**: Nhập :

+)Cận dưới :a

+)Cận trên :b

+)Làm tròn đến chữ số thứ e

Gói 1

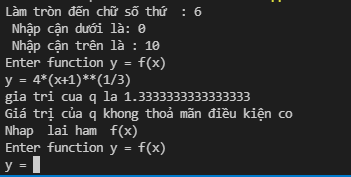
**Bước 2**:Nhập +)Hàm số

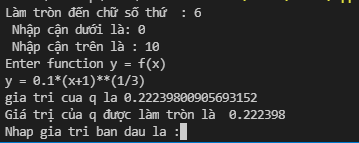
+)Giá trị ban đầu

**Bước 3**: Tìm q bằng phương pháp grd : q=

**Bước 4**: Nếu => quay lại bước 2

**Bước 5**: Thực hiện các chương trình tiếp theo





**6) 1 cách xây dựng hàm số **

Giả thiết : Cho hàm số và khoảng cách li nghiệm . Ta thường xây dựng hàm số như sau:

Với khoảng cách li nghiệm  thì  



Ta có : 

Gọi  là nghiệm của phương trình =>=>

Do hàm số  liên tục và nên tồn tại 1 số đủ nhỏ sao cho :

 ; ta luôn có 

**7) Hai công thức đánh giá sai số**

**a) Sai số hậu nghiệm**

Theo chứng mình trên ta có:

****

Để nghiệm đúng đến sai số thì



Vậy công thức sai số hậu là:

**b)Sai số tiên nghiệm**

Theo định lý Lagrange với c là giá trị trung gian giữa  và ; ta có :



Theo sai số hậu nghiệm , ta có 



Để nghiệm đúng đến sai số thì 

Từ đây , ta có : 



Số lần lặp cho công thức tiên nghiệm là :

**8) Gói thuật toán tìm nghiệm của phương trình dựa theo công thức sai số hậu nghiệm và công thức sai số tiên nghiệm**

**a) Gói thuật toán dựa theo công thức sai số hậu nghiệm**

Bước 1: Nhập :

Cận dưới : a

Cận trên: b

Số chữ số làm tròn : e

Hàm số  và giá trị ban đầu 

Bước 2 : Dùng **Gói 1** kiểm tra điều kiện co

Bước 3 : Tính ;nếu là nghiệm thì nghiệm cần tìm là

Bước 3 : Gán biến đêm count=0; Nghiem\_dung=

Bước 4: Khởi tạo vòng lặp While kiểm tra điều kiện :

|Nghiem\_dung-| <

In ra các giá trị từng vòng lặp

Mỗi lần lặp tăng count lên 1

Bước 5: In ra nghiệm của phương trình là Nghiem\_dung và số lần lặp cần thiết cho công thưc hậu nghiệm là count

**b) Gói thuật toán dựa theo sai số tiên nghiệm**

Bước 1: Nhập a,b,,

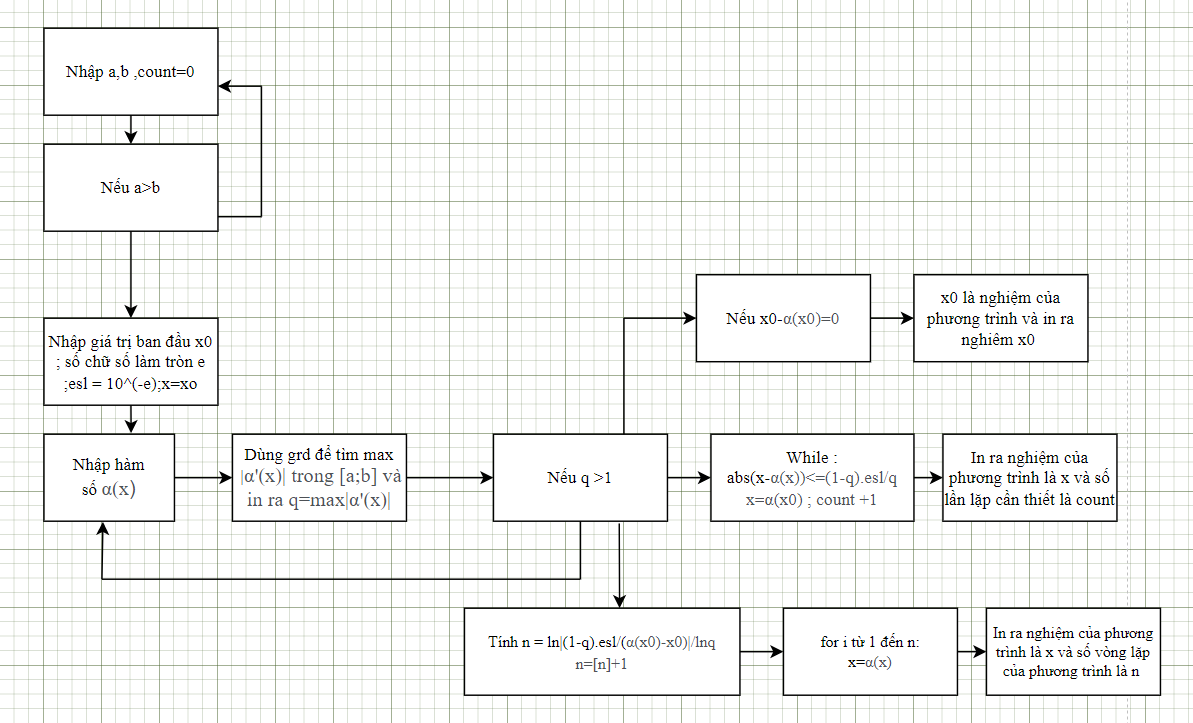
Bước 2: Tính

Bước 3: Nếu là nghiệm thì nghiệm cần tìm là

Bước 4:Gán n=sau đó n=[n]+1

Bước 5 : Dùng for chạy i từ 2 đến n với

Bước 6: In ra đáp án nghiệm và số lần lặp cần thiết

**9) Sơ đồ thuật toán** 

**10. Bình luận**

a) Xấp xỉ ban đầu  không nhất thiết phải thật gần nghiệm đúng 

b) Phép lặp đơn có khả năng tự sửa sai . Nếu xấp xỉ thứ k, mắc sai số thì có thể coi như xấp xỉ ban đầu mới

c) Có các tính chất sai số ( tiên nghiệm , hậu nghiệm )

d)Thuật toán đơn giản dễ lập trình

e) Dễ song song hoá trên máy tính ( supper computer)

f) Tốc độ hội tụ phụ thuộc vào hệ số q : nếu q càng nhỏ thì tốc độ hội tụ nhanh ; nếu q càng lớn thì tốc độ hội tụ rất chậm

**11.Chương trình chính**

# Tìm nghiệm đúng theo sai số hậu nghiệm

print("\*\*\*Chương trình tìm nghiệm theo công thức hậu nghiệm\*\*\*")

x = t

k = 0

count = 0

w = 1

Nghiem\_dung1 = x

print(f"Giá trị khởi tạo ban đầu là : {x} ")

while(abs(x-k) >= ((1-q)\*(10\*\*(-e)))/q):

    y = f(x)

    print(f"Giá trị thứ  {w} là :{y}")

    k = x

    x = y

    count += 1

    w += 1

Nghiem\_dung1 = y

X1 = round(Nghiem\_dung1, e)

print(f"Nghiem theo công thức hậu nghiệm chưa được làm tròn  là {Nghiem\_dung1} ")

print(f"NNghiệm của phương trình theo công thức hậu nghiệm làm tròn là Xo= {X1}")

print(f"Số lần lặp theo công thức hậu nghiệm là {count}")

print("-------------------------------------")

# Tìm nghiệm đúng theo công thức tiên nghiệm

x = t

qe = f(t)

k = 0

count = 1

Nghiem\_dung2 = x

if(Nghiem\_dung2-f(Nghiem\_dung2) == 0):

    Nghiem\_dung2 = x

n = log(abs(((1-q)\*(10\*\*(-e)))/(x-qe))) / log(q)

n = int(n)+1

for i in range(1, n+1, 1):

    x = f(x)

Nghiem\_dung2 = x

X3 = round(y, e)

print(f"Nghiệm theo công thức tiên nghiệm chưa được làm tròn là {Nghiem\_dung2}")

print(f"Nghiem dung theo công thức tiên nghiệm là {X3}")

print(f"Số lần lặp cần thiết của công thức tiên nghiệm là {n}")

Ta thấy chương trình chính rất đơn giản chỉ dùng vòng lặp while và for để in ra nghiệm và số lần lặp

**12. Hướng dẫn sử dụng chương trình**

**-**Chương trình được viết bằng ngôn ngữ python trên nền tảng Vs.Code . Để chạy chương trình , các bạn có thể chạy bằng vscode; pycharm, jupiter ; …..

-Chương trình dùng 2 thư viện chính đó là sympy và math trong python:

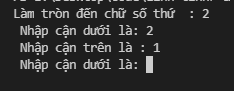
+)Sympy để nhập hàm và tính đạo hàm cấp 1 và cấp 2 của hàm 

+) Math để tính các giá trị ln; max; round;….

Khi bắt đầu chạy chương trình : 

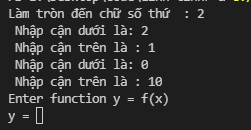
Các bạn vui lòng nhập số chữ số làm tròn nghiệm của phương trình

Tiếp theo các bạn nhập a,b là các khoảng cách li nghiệm



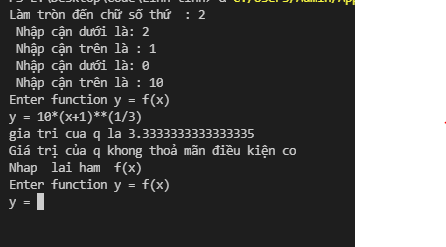
Nếu nhập a>b , chương trình sẽ yêu cầu nhập lại

Tiếp theo là nhập hàm 

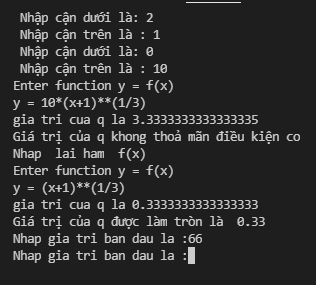


Sau đó chương trình sẽ in ra giá trị của q là giá trị lớn nhất của hàm số 

Nếu q >1 thì chương trình sẽ bắt nhập lại hàm 



Khi giá trị của q thoả mãn chương trình sẽ bắt nhập giá trị ban đầu. Khi giá trị ban đầu không nằm trong khoảng cách li nghiệm ; chương trình sẽ bắt nhập lại giá trị ban đầu

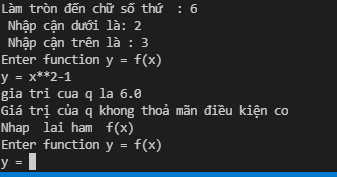


Khi nhập giá trị ban đầu thoả mãn ; chương trình sẽ ra output cần tìm của bài toán

**III) HỆ THỐNG VÍ DỤ**

Ví dụ 1: Xét hàm số sau có thoả mãn điều kiện co hay không :  trên 

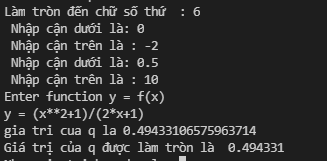
Ta có : 

=> hàm số không thoả mãn điều kiện co 

**Ví dụ 2 :** Xây dựng hàm  từ  với khoảng cách li nghiệm là 

Ta có: Xây dựng hàm dựa vào công thức :

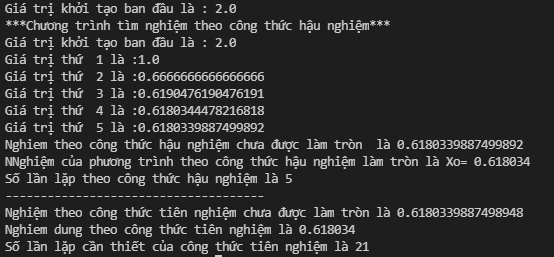


Theo chứng minh trên thì hàm số  thoả mãn điều kiện có 

Xây dựng 1 dãy sao cho bất kì thuộc [a,b] và mà dãy sẽ hội tụ về nghiệm của phương trình

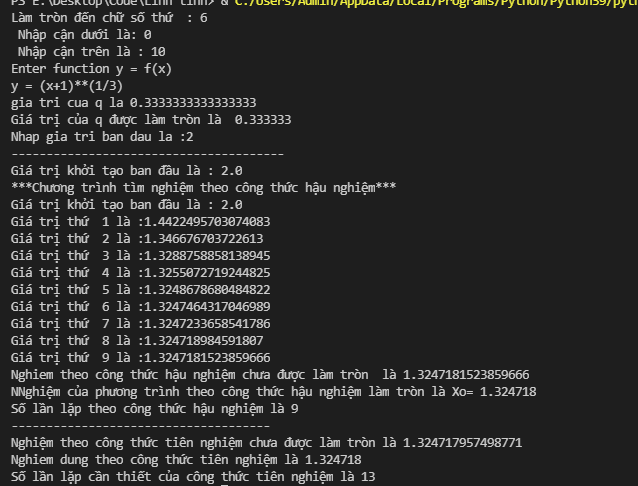
Bảng giá trị của dãy số

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n |  |  |  |  |  |
| 0 | 2 | 3 | 4 | 6 | 8 |
| 1 | 1 | 1.42857143 | 1.88888889 | 2.84615385 | 3.82352941 |
| 2 | 0.66666666 | 0.78835979 | 0.95607235 | 1.35985853 | 1.80632253 |
| 3 | 0.61904762 | 0.62929283 | 0.65727309 | 0.76597633 | 0.92415545 |
| 4 | 0.61803445 | 0.61809011 | 0.61869922 | 0.62667828 | 0.65093431 |
| 5 | 0.61803398 | 0.61803399 | 0.61803419 | 0.61806715 | 0.61850423 |
| 6 | 0.61803398 | 0.61803399 | 0.61803399 | 0.61803399 | 0.61803408 |
| 7 |  |  | 0.61803399 | 0.61803399 | 0.61803399 |
| 8 |  |  |  |  | 0.61803399 |

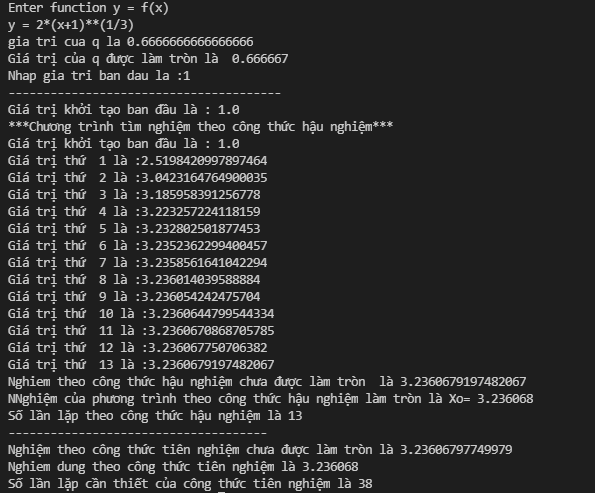


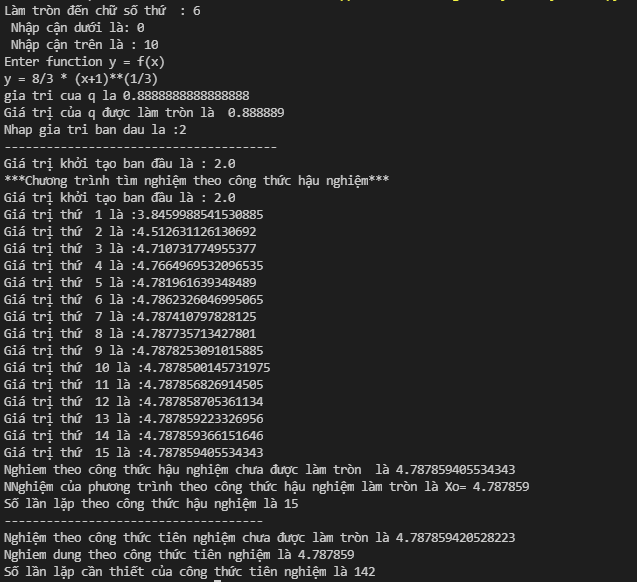
**Ví dụ 3: Tìm nghiệm của phương trình trong các trường hợp sau :**

a) với khoảng li nghiệm là



b) với khoảng li nghiệm là



c) với khoảng cách li nghiệm là 

Qua 3 ví dụ trên ta thấy tốc độ hội tụ và số lần lặp phụ thuộc vào giá trị của q

**LỜI KẾT**

Như vậy trong quá trình tìm hiểu, học tập bộ môn giải tích số nói chung và thuật toán nói riêng đã giúp em một người sinh viên trường đại học bách khoa dần sẽ tự tin hơn trong việc lĩnh ngộ những kiến thức, những ngành công nghệ mới ở Việt Nam cũng như trên thế giới.

Em xin trân thành cảm ơn !