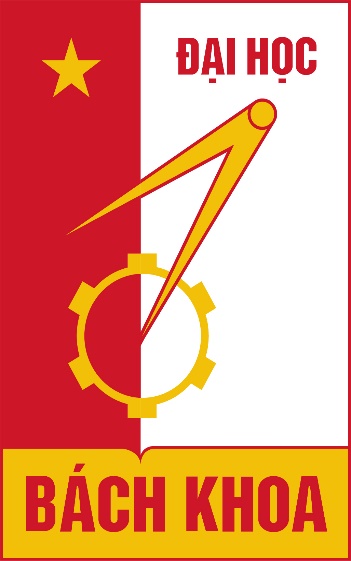
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

---o0o---



BÁO CÁO GIẢI TÍCH SỐ

ĐỀ TÀI:

TÌM MIN MAX CỦA HÀM MỘT BIẾN TRÊN KHOẢNG ĐÓNG

Giảng viên hướng dẫn: Cô Hà Thị Ngọc Yến

Nhóm sinh viên thực hiện:

Họ và tên MSSV

Trần Viết Tài 20185402

Nguyễn Hoàng Minh 20195902

Phạm Anh Đức 20195859

Phan Tiến Đạt 20195854

Nguyễn Thành Long 20195898

Hà Nội, Ngày 15 tháng 4 năm 2021

LỜI NÓI ĐẦU

Giải tích số là môn học nghiên cứu về các phương pháp, các thuật toán sử dụng sai số xấp xỉ cho các bài toán giải tích (phân biệt với bài toán rời rạc). Lớp bài toán đầu tiên được đề cập đến là các phương pháp để giải phương trình phi tuyến f(x) = 0 mà đặc biệt ở đây là tìm giá trị Min-Max của hàm một biến trong khoảng đóng cho trước.

Sau một thời gian tìm hiểu về đề tài “Tìm Min-Max của hàm một biến trên khoảng đóng” bọn em làm bản báo cáo này để trình bày về nội dung phương pháp, hệ thống ví dụ, cùng với thuật toán, chương trình cụ thể để giải quyết đề tài này. Trong quá trình làm báo cáo cũng như xây dựng chương trình, thuật toán…không tránh khỏi sai sót vậy chúng em mong nhận được những lời nhận xét bổ ích từ cô cũng như mọi người để đề tài này của chúng em hoàn thiện hơn. Bọn em cũng gửi lời cảm ơn cô Hà Thị Ngọc Yến đã giúp chúng em trong quá trình tìm hiểu đề tài này ạ. Hi vọng những phương pháp này sẽ là một công cụ hữu hiệu có thể sử dụng mỗi khi gặp bài toán tìm nghiệm của phương trình và ứng dụng vào một số bài toán khác.

MỤC LỤC

[1. Giới thiệu 4](#_Toc69373566)

[2. Xây dựng thuật toán Gradient Descent: 4](#_Toc69373567)

[Xây dựng công thức đạo hàm 6](#_Toc69373568)

[3. Xây dựng thuật toán tìm các cực trị trên [a, b] 7](#_Toc69373569)

[4. Sự hội tụ 7](#_Toc69373570)

[5. Lựa chọn learning rate (η) động 9](#_Toc69373571)

[6. Điều kiện dừng 9](#_Toc69373572)

[7. Thuật toán Gradient Descent 10](#_Toc69373573)

[Các bước của thuật toán: 10](#_Toc69373574)

[8. Sơ đồ khối 12](#_Toc69373575)

[9. Ví dụ: 16](#_Toc69373576)

[10. Code 18](#_Toc69373577)

[11. Đánh giá phương pháp 24](#_Toc69373578)

[12. Phương pháp duyệt thông thường 24](#_Toc69373579)

[• Ý tưởng và xây dựng phương pháp 24](#_Toc69373580)

[• Đánh giá 24](#_Toc69373581)

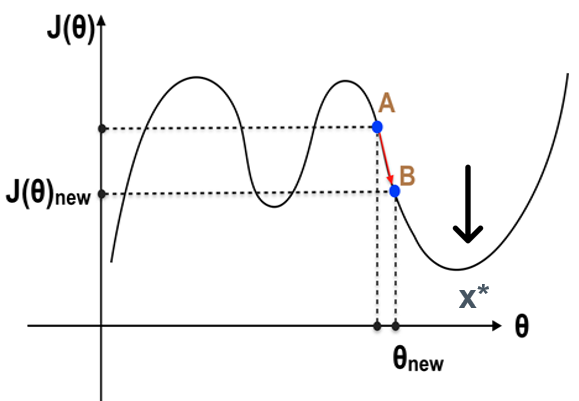
[13. Ứng dụng 25](#_Toc69373582)

# Giới thiệu

Trong kiến thức toán phổ thông chúng ta đã biết muốn tìm cực trị một hàm số y = (𝑥) chúng ta sẽ giải phương trình đạo hàm của hàm số:

Tuy nhiên phương trình trên không phải lúc nào cũng giải được dễ dàng vì có những trường hợp việc giải phương trình trên là bất khả thi. Vậy khi gặp những tình huống này chúng ta phải làm gì? May thay thuật toán Gradient Descent cho chúng ta cách thức tìm các điểm cực tiểu cục bộ này một cách xấp xỉ sau một số vòng lặp. Trong thực tế các giá trị dữ liệu không có đúng 100% mà đôi khi chúng ta chỉ cần những con số gần đúng. Bạn không thể nào cắt 1 chiếc bánh thành 2 phần đúng 50% mà chỉ áng chừng 2 nửa bằng nhau.

# Xây dựng thuật toán Gradient Descent:



Giả sử ta bắt đầu tìm cực trị của hàm số này từ điểm A.Vậy ta cần đưa điểm A gần về điểm cực trị 𝑥\* gần nhất. Như chúng ta đã biết, khi đạo hàm tại 1 điểm < 0 => hàm số này đang giảm và có khả năng tiến đến cực trị, nếu có cực trị thì chắc chắn sẽ là cực tiểu. Còn khi đạo hàm tại một điểm > 0 thì hàm số này đang tăng và có khả năng tiến tới cực trị, nếu có thì sẽ là cực đại.

Ta xét cụ thể trường hợp cực tiểu. Vậy nên ta sẽ di chuyển A một đoạn , nếu A nằm bên trái \* thì sẽ mang dấu dương còn A mà nằm bên phải \* thì sẽ mang dấu âm. Do đó ta có nhận xét rằng dấu của sẽ ngược dấu với .

Ngược lại, điểm A càng xa về phía bên phải thì giá trị f’(𝑥A) sẽ càng lớn hơn 0 và ngược lại. Vậy lượng dịch chuyển , một cách trực quan nhất, tỉ lệ nghịch với (𝑥A).

Với 2 nhận xét phía trên ta rút ra được công thức sau:

𝑥B = 𝑥A + **sign** ∗  𝜂  ∗  𝑓 ′ (𝑥A ) (\*).

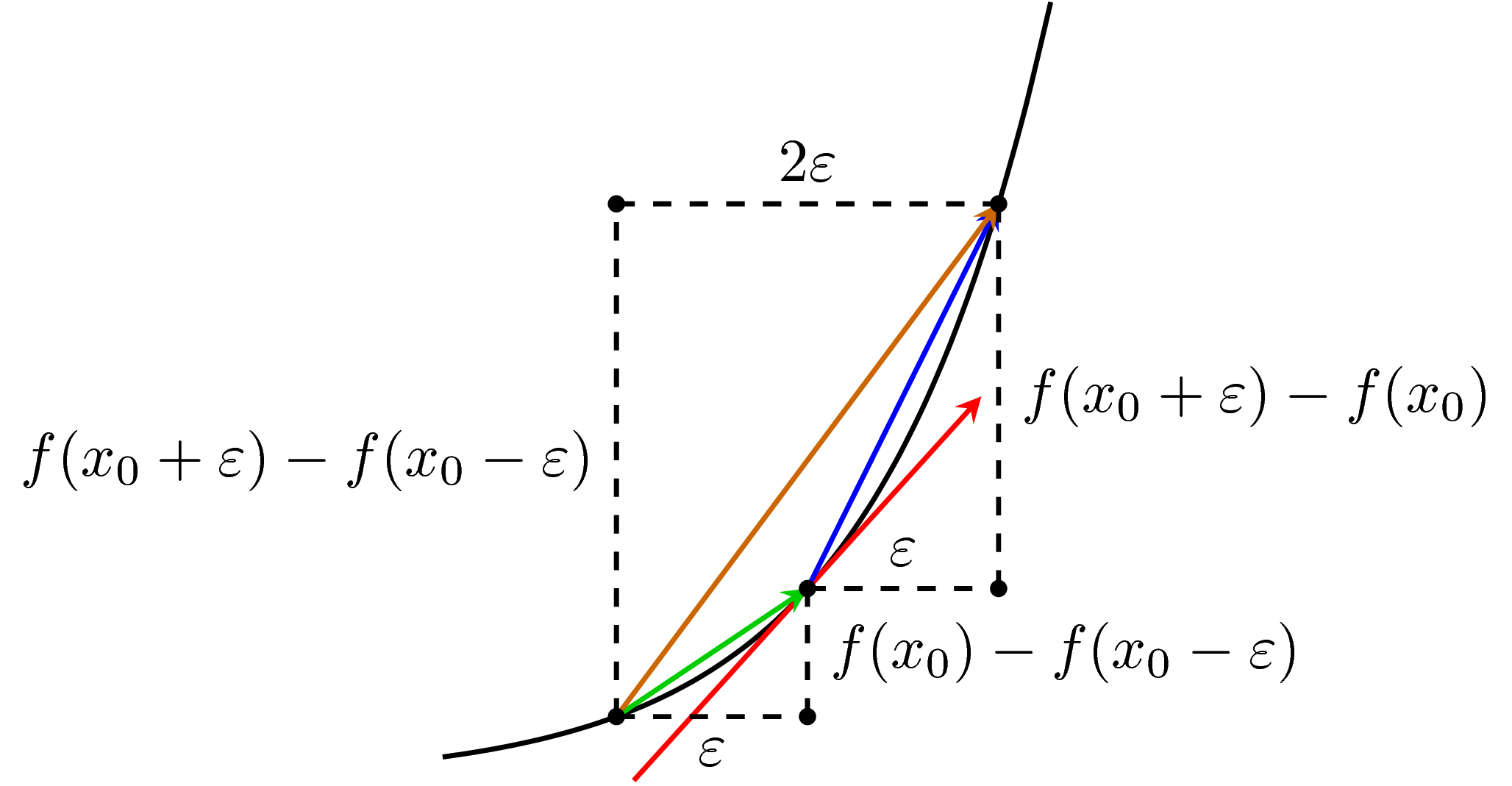
Ở đây biến **sign** thể hiện dấu (âm hoặc dương) của đạo hàm tại điểm 𝑥A. Nếu đạo hàm dương => ta đang đi tìm cực đại

Ngược lại, nếu đạo hàm âm, ta đang đi tìm cực tiểu.

Với **sign** = -1 khi lặp tìm cực tiểu và **sign** = +1 khi lặp tìm cực đại). Với là một số dương được gọi là **learning rate** (tốc độ học).

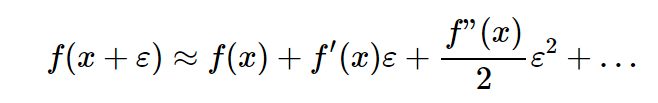
=>

## Xây dựng công thức đạo hàm

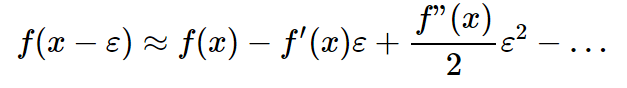


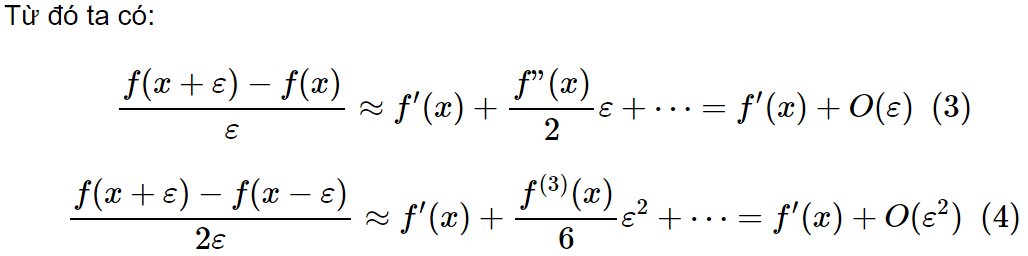
Chúng ta cùng quay lại một chút với Giải tích I: [Khai triển Taylor](http://mathworld.wolfram.com/TaylorSeries.html).

Với ε rất nhỏ, ta có hai xấp xỉ sau:



và:





Từ đó, nếu xấp xỉ đạo hàm bằng công thức (3) (xấp xỉ đạo hàm phải), sai số sẽ là O(). Trong khi đó, nếu xấp xỉ đạo hàm bằng công thức (4) (xấp xỉ đạo hàm hai phía), sai số sẽ là O() << O() với rất nhỏ.

# Xây dựng thuật toán tìm các cực trị trên [a, b]

Chart, line chart

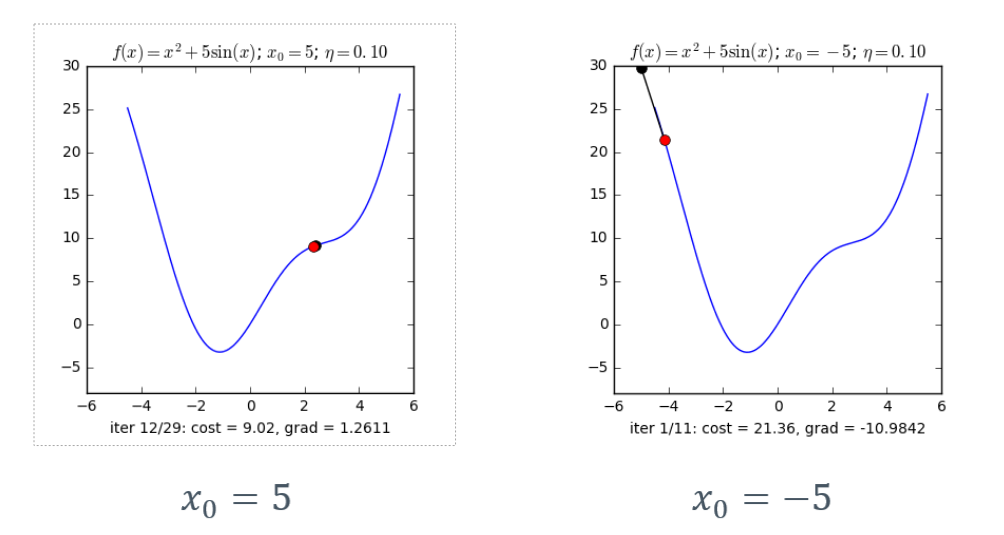
Description automatically generated

B1: 𝑓 ′(𝑥0 ) < 0 =>cực trị tiếp theo (nếu có) là cực tiểu.

B2: Dùng  (\*) với **sign** = lặp 𝑥0  𝑥 \*. Tăng 1 khoảng step  để 𝑥0 > 𝑥 \*

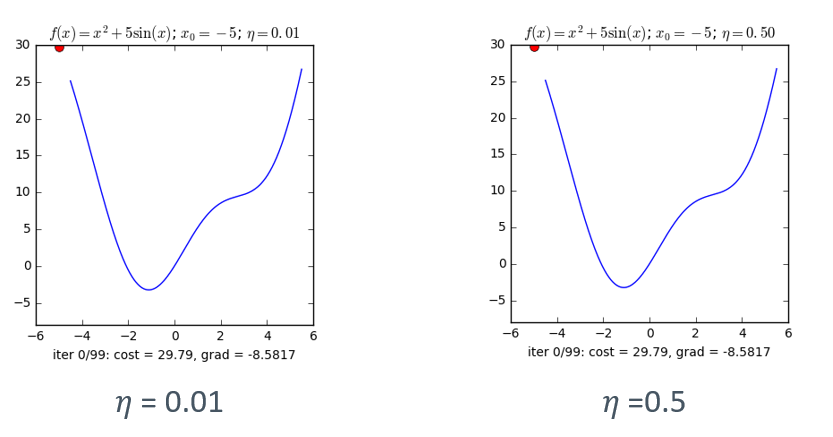
B3: Quay lại B1 với x0 mới. Lặp lại đến khi 𝑥0 = b.

# Sự hội tụ



Sự hội tụ của thuật toán phụ thuộc vào 2 yếu tố. Thứ nhất là điểm khởi tạo đầu. Với 𝑥0 = 5 (hình trái), đường đi của điểm A có chứa một khu vực có đạo hàm khá nhỏ gần điểm có hoành độ bằng 2, ở đoạn này các bạn có thể thấy độ dốc của đồ thị khá là nhỏ nên thuật toán duyệt ở đây khá là lâu, tưởng tượng như đi xe xuống dốc, dốc càng cao xe sẽ chạy càng nhanh, mà dốc thấp thì xe chạy chậm hơn vậy

ứng với 𝑥0 = −5 (hình phải), nghiệm hội tụ nhanh hơn vì điểm 𝑥0 = −5 gần với cực trị hơn.

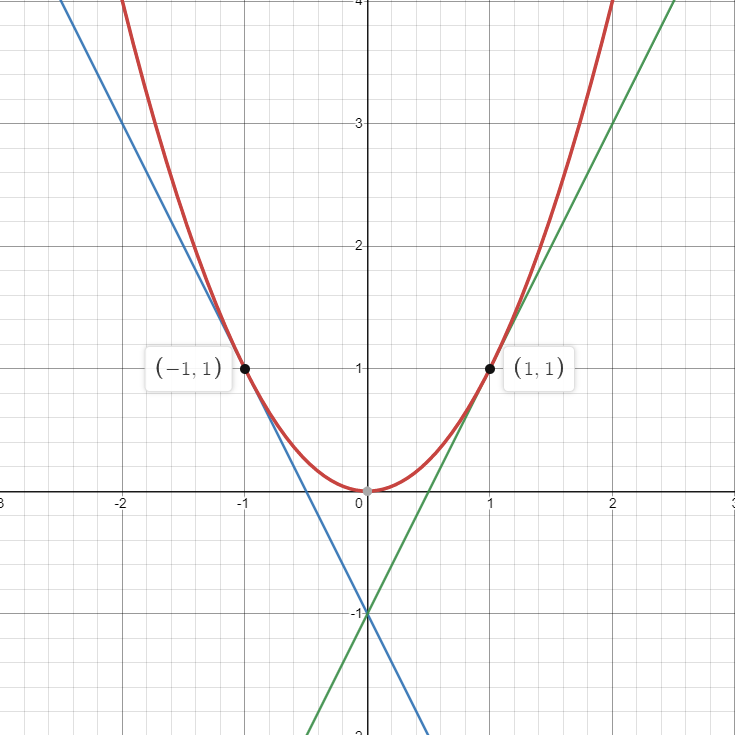
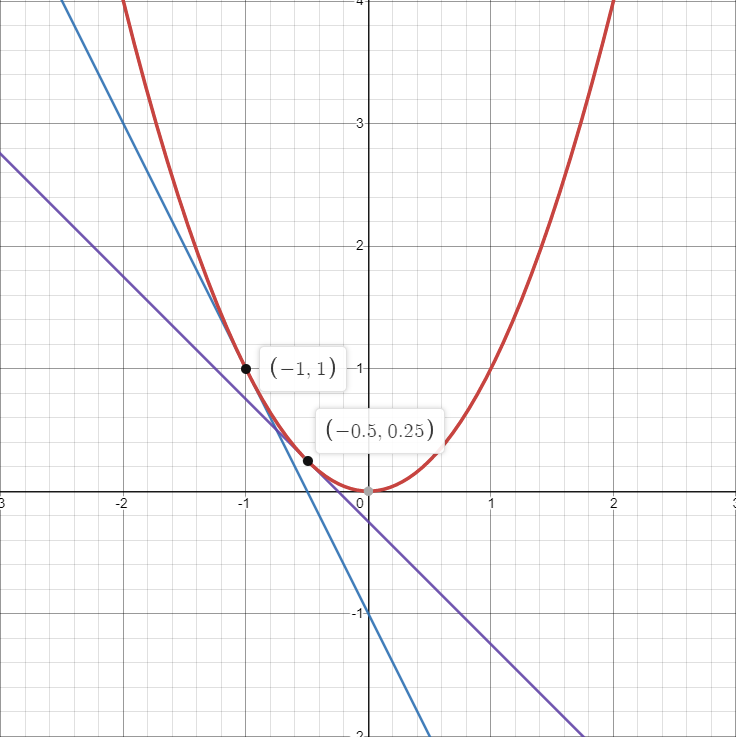


Sự hội tụ còn phụ thuộc vào yếu tố thứ 2 là Learning Rate hay còn gọi là eta. Với η = 0.01, tốc độ hội tụ rất chậm, dù đã được giới hạn tối đa 100 vòng lặp, thuật toán dừng lại trước khi tới đích, mặc dù đã rất gần với điểm cực trị. Do đó learning rate quá thấp sẽ ảnh hưởng tới tốc độ của thuật toán, thậm chí có thể không bao giờ tới được đích.

Với η = 0.5, thuật toán tiến rất nhanh tới gần đích. Ở đây có thể thấy, điểm xét nhảy liên tục từ dốc này sang dốc kia của đồ thị. Tuy nhiên, thuật toán không hội tụ được vì bước nhảy quá lớn.

Do đó, lựa chọn learning rate rất quan trọng. Ta có thể chọn learning rate khác nhau ở mỗi vòng lặp. Từ đó ta phát triển thêm thuật toán learning rate động.

# Lựa chọn learning rate (η) động

Eta động là giải pháp chọn eta qua mỗi vòng lặp để tránh việc đi chậm quá hoặc bỏ qua cực trị.

Khi đạo hàm của hai giá trị x liên tiếp có cùng dấu, tức là x chưa vượt qua điểm cực trị x\*, ta phải tăng eta lên (ở đây tăng eta lên 3 lần) nhằm giúp tăng tốc độ x đi đến x\*.

Khi đạo hàm của hai giá trị x liên tiếp trái dấu nhau, tức là chúng nằm ở hai phía so với điểm cực trị x\*, ta phải giảm eta đi (ở đây giảm eta đi một nửa) nhằm giúp hàm hội tụ tốt hơn, không bị phân kỳ.

# 

# Điều kiện dừng

Vì phụ thuộc vào hàm f(x), thuật toán có thể chạy mãi và rơi vào vòng lặp vô hạn, do đó, ta phải xây dựng điều kiện dừng cho thuật toán. Dưới đây là hai phương pháp:

- Giới hạn số vòng lặp: đảm bảo rằng chương trình không quá lâu và không làm đơ máy tính. Dù kết quả có thể chưa đạt đến giá trị sai số mong muốn nhưng vẫn tạm chấp nhận được.

-Kiểm tra giá trị đạo hàm: nếu gradient tại 𝑥0 sau n lần lặp nhỏ hơn một số cho trước thì dừng lại.

# Thuật toán Gradient Descent

Các bước của thuật toán:

B1:Nhập vào khoảng [a,b], f(x), ε nhỏ >0, step.​

B2:Tính đạo hàm f’(x):​

B2.1:dy=f(x+ ε)-f(x- ε)​

  B2.2:dx=2\* ε ​

  B2.3:f’(x)=dy/dx   ​

B3:So sánh f’(x) và 0​

  Nếu f’(x) = 0 => trả về x​

  Nếu f’(x) > 0, chọn **sign**= 1​

  Nếu f’(x) < 0, chọn **sign**= -1​

B4.1:TH xét 𝜂 tĩnh.Chọn 𝜂 nhỏ bất kì >0.Chuyển đến B5​

​B5:Tìm x new: ​

   x=xo+**sign**\* 𝜂\*f’(xo);​

   xo=x;​

B6:Sosánh|f’(x)| với ε​  
   Nếu |f’(x)| <ε, trả về x​

   Nếu |f’(x)| >ε, so sánh x với b.​

          Th1: x>b,thuật toán kết thúc, trả về x​

          Th2:x<b quay lại B5​  
B7:So sánh f(x\*), f(a), f(b) để tìm min, max      ​

   B7.1:Nếu f(a)>f(b), ta chọn tmax=a, tmin= b (ngược lại)​

   B7.2:Đặt xo=minmax​

         Th1:**Sign** <0. So sánh f(minmax) và f(tmin)​

                  Nếu f(minmax)<f(tmin), gán tmin= minmax​

                  Nếu f(minmax)>f(tmin), So sánh x0 với b. ​

                                                                      xo>b, kết thúc​

                                                                      xo<b, quay lại B5​

         Th2:**Sign** >0.​

                  Nếu f(minmax)>f(tmax) gán tmax= minmax​

                  Nếu f(minmax)<f(tmax) so sánh xo với b. ​

                                                                       xo>b,kết thúc​

                                                                       xo<b,quay lại B5​

B8:Hiển thị ra (x\*, f(x\*)), min, max, số bước lặp.​

​Chú ý: TH xét 𝜂 động​

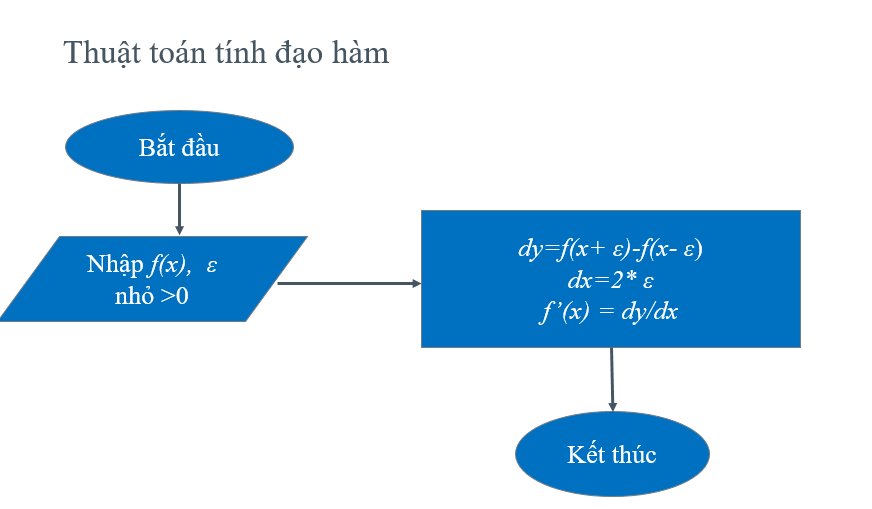
   B1:Nhập xo, 𝜂 = const​

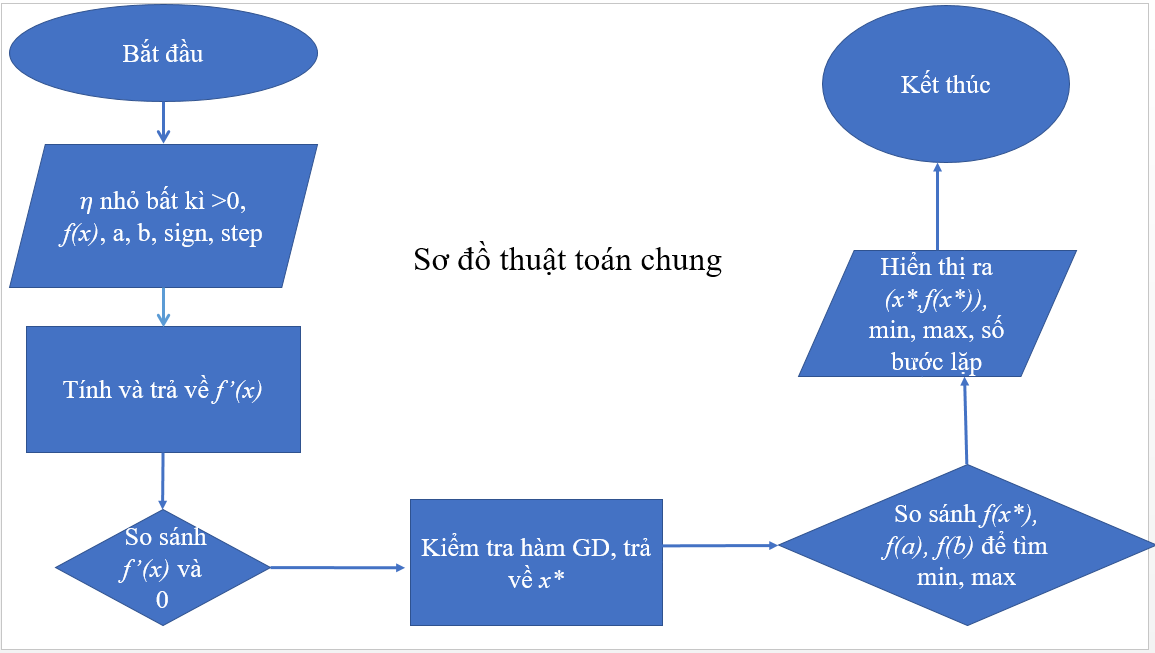
   B2:Xét dấu fp(xnew)\*fp(xo)​  
           Th1:fp(xnew)\*fp(xo) <0 , chọn  𝜂 =  𝜂/2​

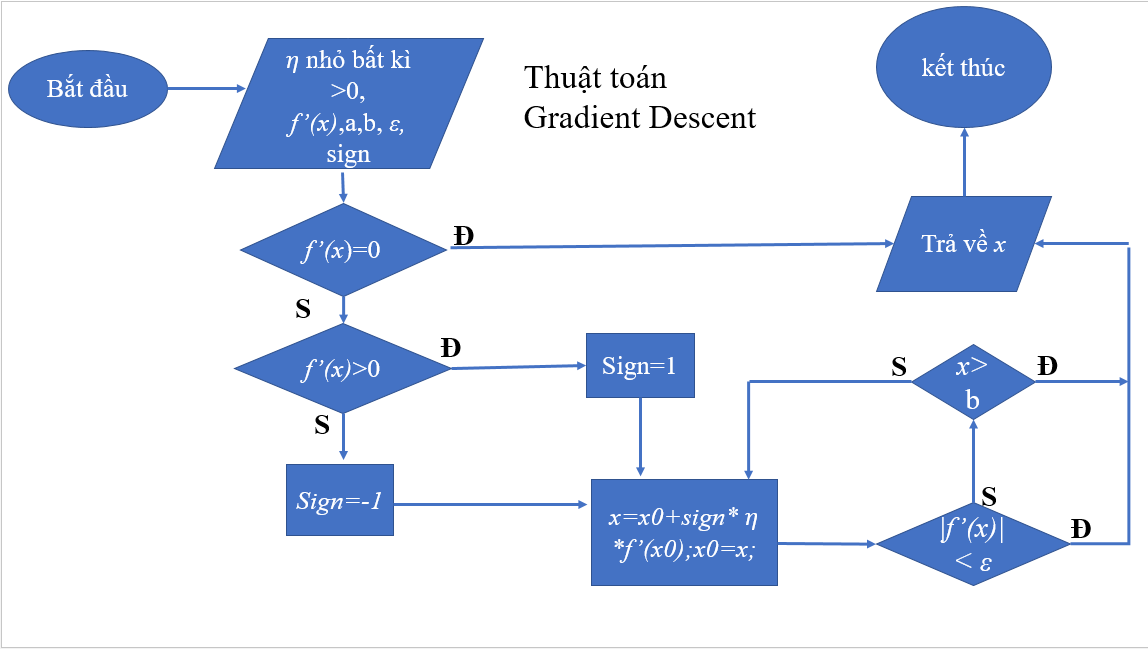
           Th2:fp(xnew)\*fp(xo) >0, nếu  𝜂 \* fp(xnew) < 1, 𝜂 =𝜂 \*1.5​

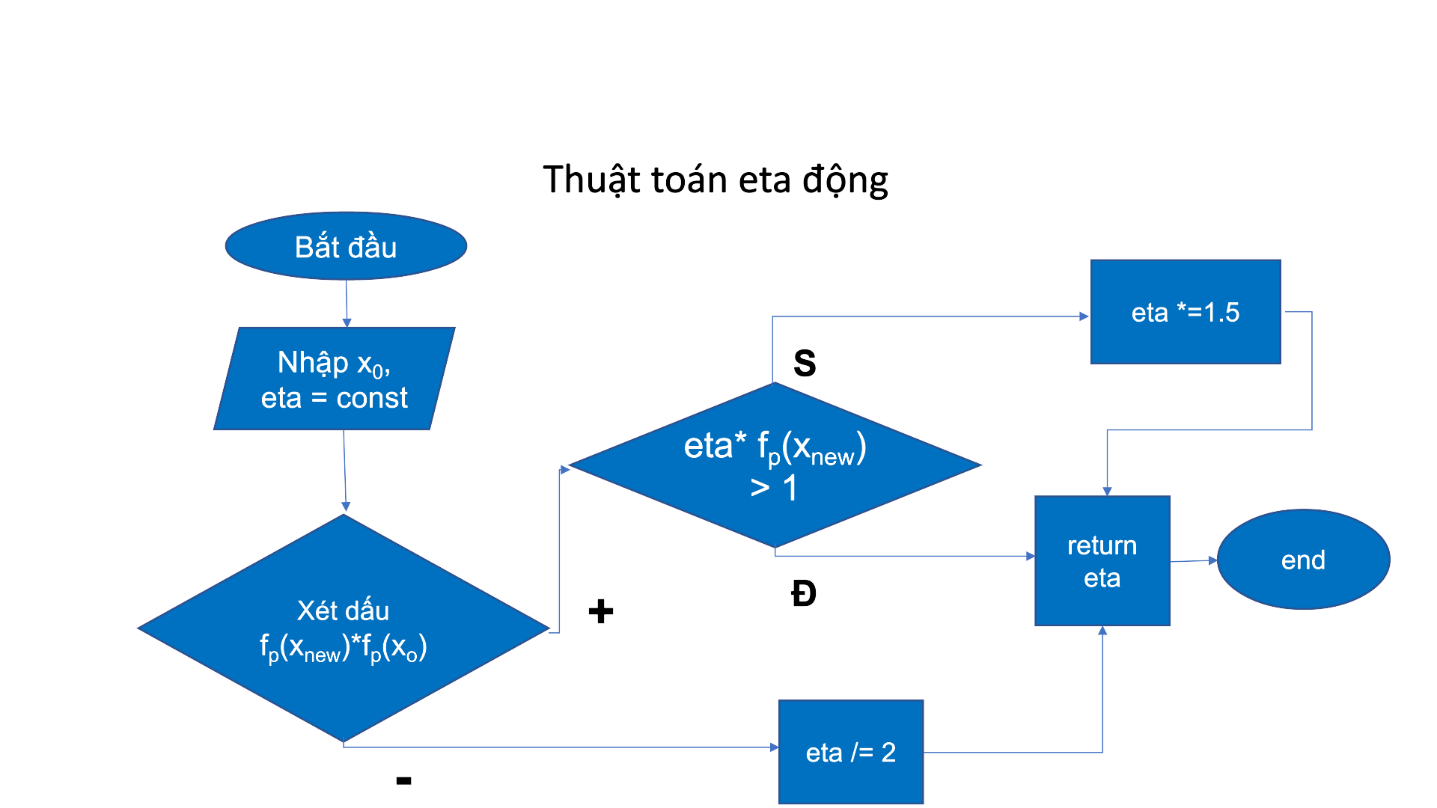
                                                   nếu  𝜂 \* fp(xnew) > 1, trả về  𝜂.Rồi thay 𝜂 mới vào thuật toán .

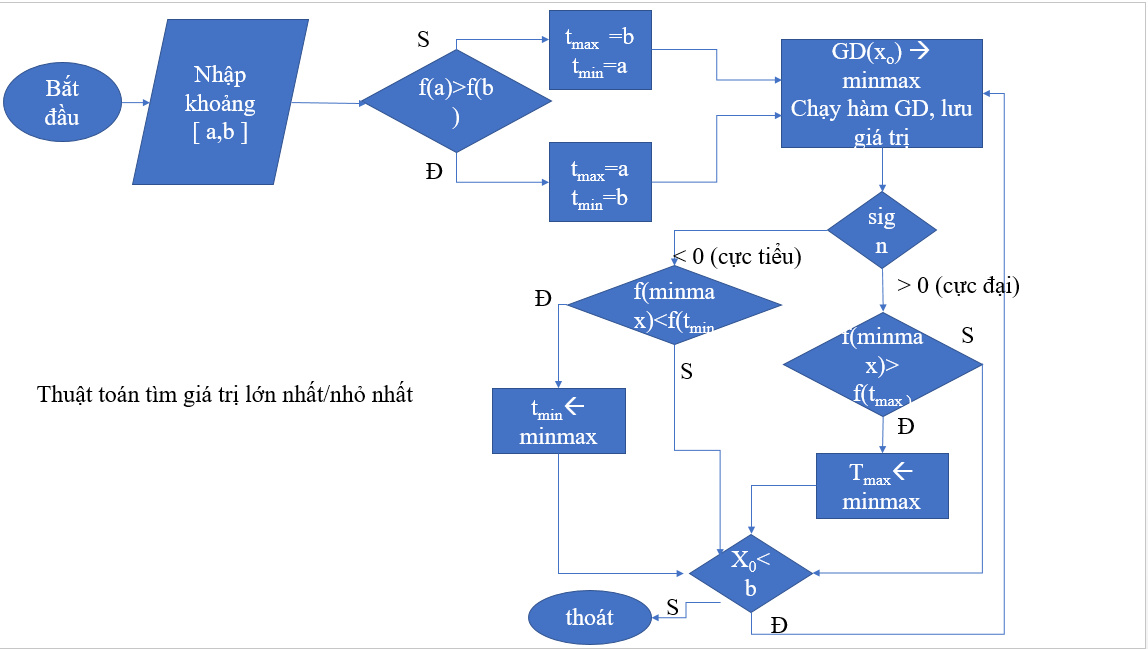
# Sơ đồ khối

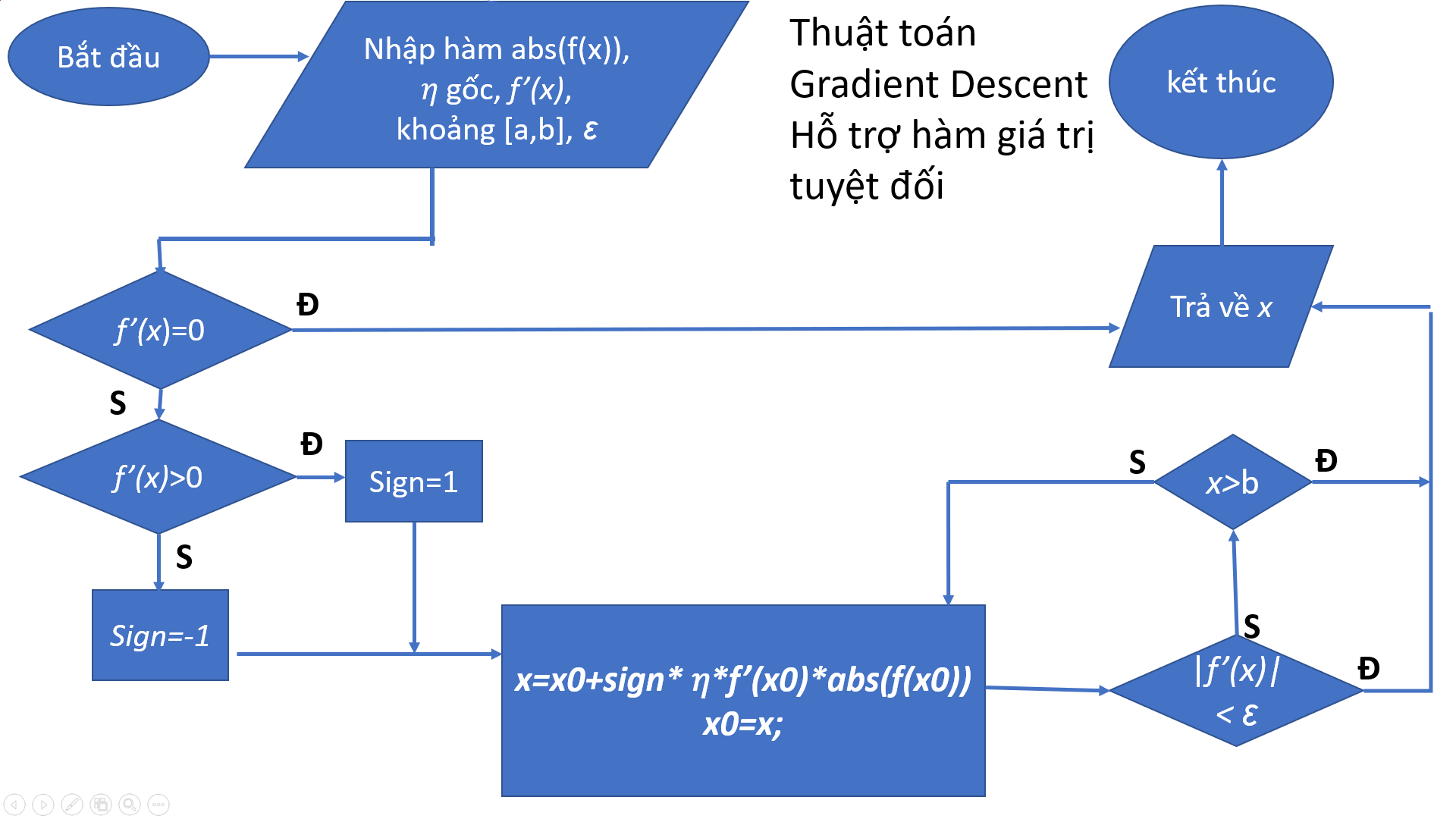










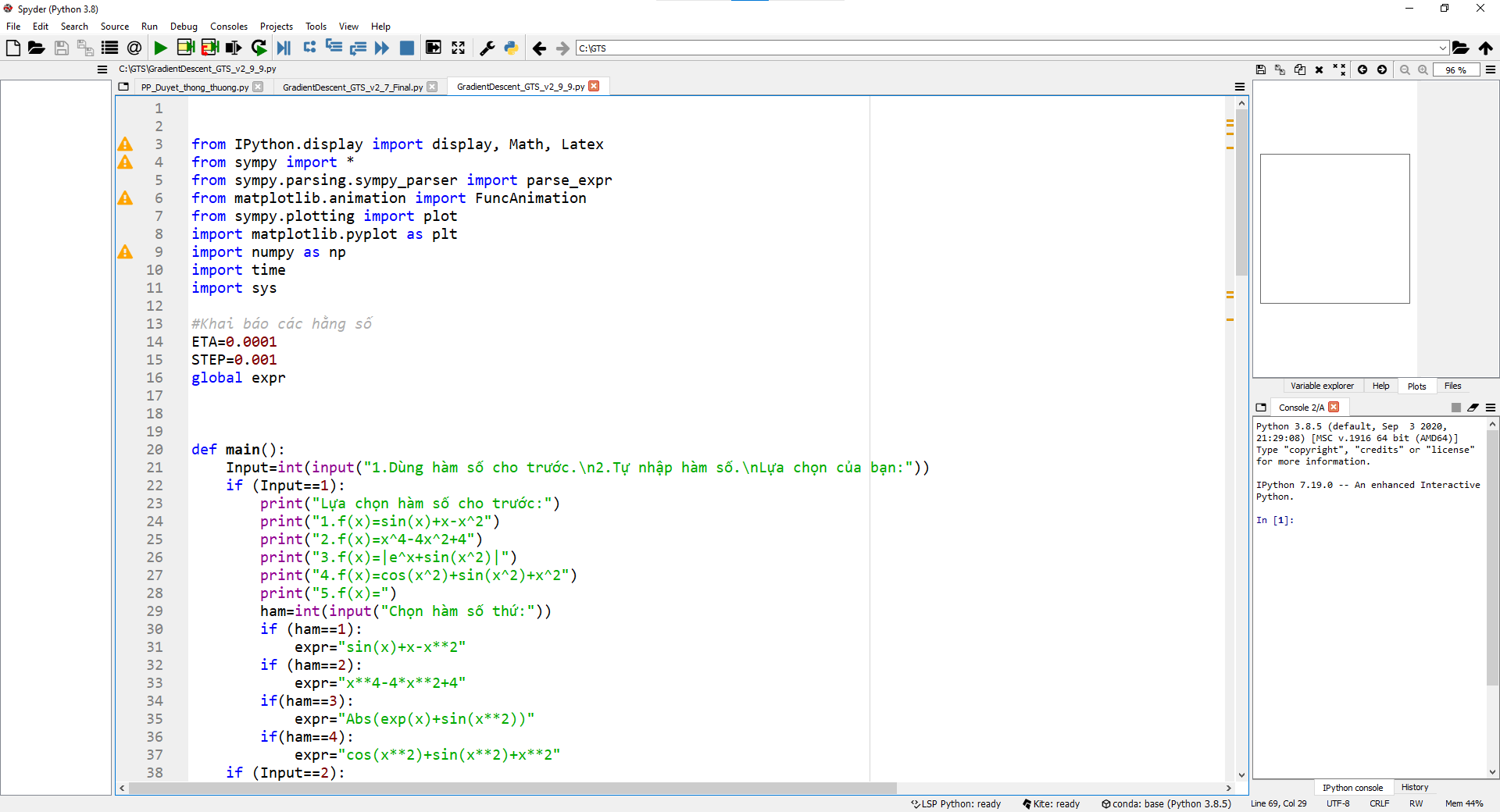


# Ví dụ:

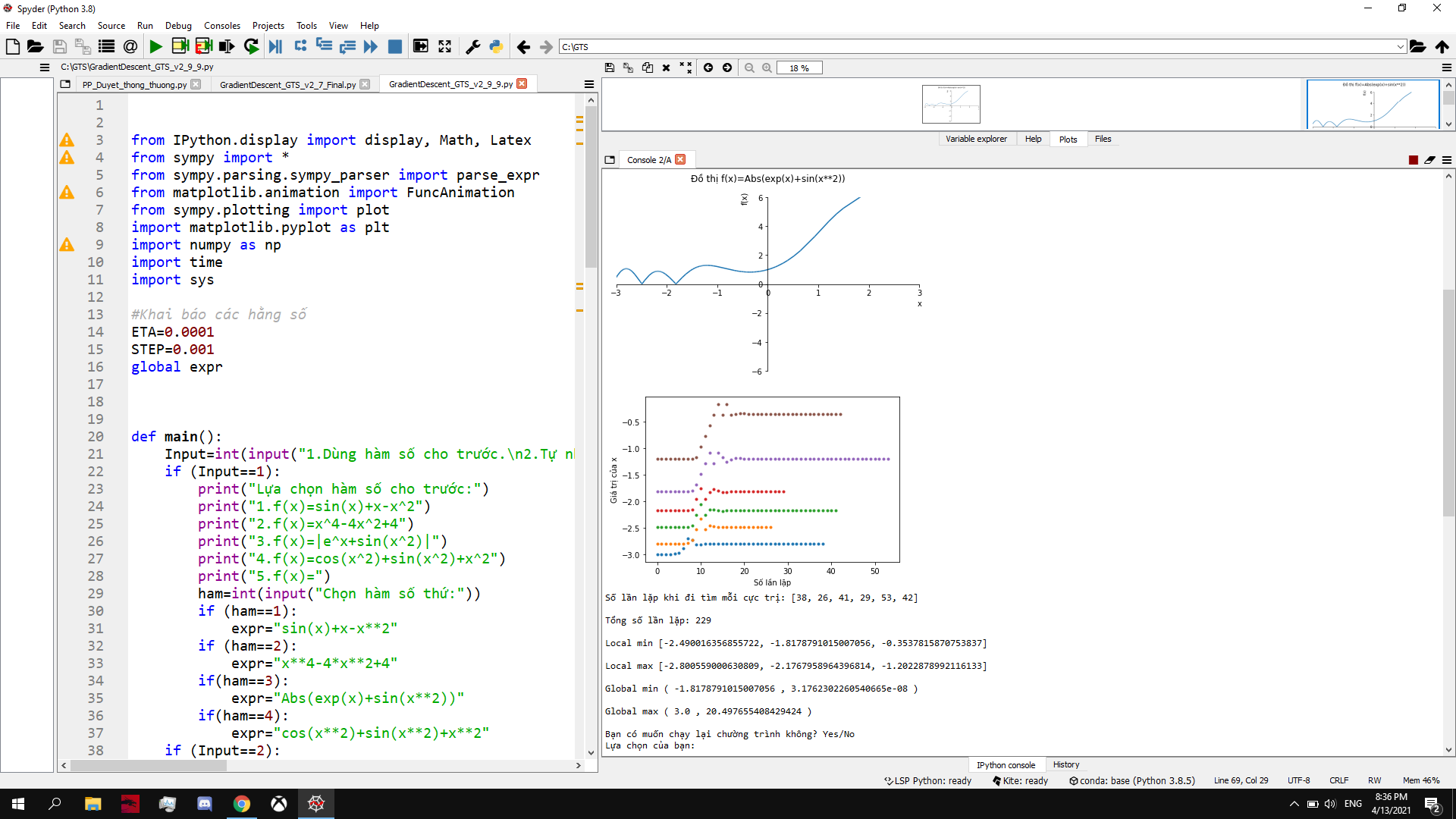
f(x)=|e^x + sin(x^2)| trong khoảng [-3, 3]

Code và kết quả chạy

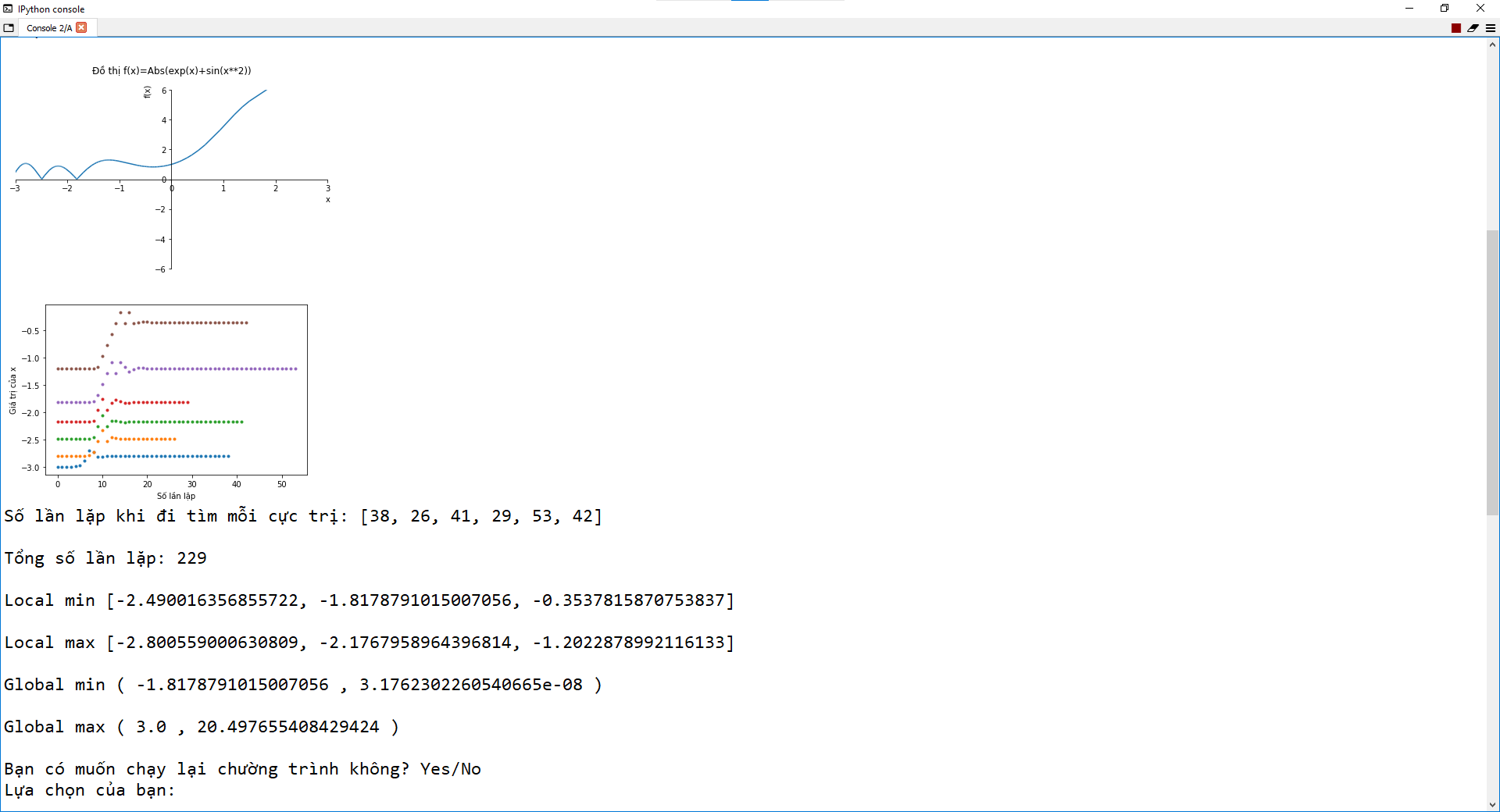
Hình 1:



Hình 2:



Hình 3:



# Code

#Một số kí hiệu toán học khi nhập hàm:

#Dấu mũ (a^b): Nhập a\*\*b

#Lấy trị tuyệt đối(|f(x)|,f(|x|)):Nhập Abs(f(x)),f(Abs(x))

#Lấy căn thức (sqrt()): Nhập sqrt()

#Hàm logarith: log(f(x))/log(a) với a là cơ số

from IPython.display import display, Math, Latex

from sympy import \*

from sympy.parsing.sympy\_parser import parse\_expr

from matplotlib.animation import FuncAnimation

from sympy.plotting import plot

import matplotlib.pyplot as plt

import numpy as np

import time

import sys

#Khai báo các hằng số

ETA=0.0001

STEP=0.001

global expr

def main():

Input=int(input("1.Dùng hàm số cho trước.\n2.Tự nhập hàm số.\nLựa chọn của bạn:"))

if (Input==1):

print("Lựa chọn hàm số cho trước:")

print("1.f(x)=sin(x)+x-x^2")

print("2.f(x)=x^4-4x^2+4")

print("3.f(x)=|e^x+sin(x^2)|")

print("4.f(x)=cos(x^2)+sin(x^2)+x^2")

print("5.f(x)=")

ham=int(input("Chọn hàm số thứ:"))

if (ham==1):

expr="sin(x)+x-x\*\*2"

if (ham==2):

expr="x\*\*4-4\*x\*\*2+4"

if(ham==3):

expr="Abs(exp(x)+sin(x\*\*2))"

if(ham==4):

expr="cos(x\*\*2)+sin(x\*\*2)+x\*\*2"

if (Input==2):

expr=input("Nhap ham so can tim\nf(x)=")

x= Symbol('x'),#sử dụng x làm biến toán học

f=lambdify(x, parse\_expr(expr))#tạo hàm số f(x)

a=float(input("Nhập khoảng [a,b]\nNhập a="))

b=float(input("Nhập b="))

if abs(a)>abs(b):Xlim=abs(a)

else:Xlim=abs(b)

plot(parse\_expr(expr),axis=True,xlim=(-Xlim,Xlim), ylim=(-2\*Xlim,2\*Xlim), autoscale = true,title="Đồ thị f(x)="+expr+"\n")

def fp(x):#Tính đạo hàm

dy= f(x+0.000001)-f(x-0.000001)

dx= 2\*0.000001

return dy/dx

def xetdau(x):

if fp(x)==0: return 0

if fp(x)>0: return +1

if fp(x)<0: return -1

def GD2(eta,x0):#Thuật toán gradient descent Eta động

**sign**=0

**sign**=xetdau(x0)

it=0

varX=[x0]

for it in range(300000):

deltaX=eta\*fp(x0)\*abs(f(x0)) # nhân thêm abs(f(x0)) khiến nó chạy chậm lại khi đến gần nghiệm của f(x)=0 (nếu hàm không có trị tuyệt đối thì sẽ làm GD chậm lại)

if deltaX>0.2 : deltaX=0.2 #giới hạn tốc độ dịch chuyển của x(phòng trường hợp x nhảy quá xa)

if deltaX<-0.2: deltaX=-0.2

x\_new = x0 + **sign**\*deltaX

if abs(fp(x\_new))<0.005:abslim=1e-15

else: abslim=1e-7

if abs(fp(x\_new)) < 1e-20 or abs(f(x\_new))<abslim or x\_new>b:

break

#eta động

if (fp(x\_new)\*fp(x0)>0 ):eta\*=3

if fp(x\_new)\*fp(x0)<0:eta/=5

x0=x\_new

varX.append(x\_new)

if x\_new<=b: #hiển thị các giá trị cực trị sau các lần lặp

plt.xlabel("Số lần lặp")

plt.ylabel("Giá trị của x")

plt.plot(varX,ls='None',marker='.')

return (x\_new, it, **sign**)

#cài đặt giá trị max, min ban đầu

if(f(a)>f(b)):

tmax=a; tmin=b

else:

tmin=a; tmax=b

Sum=0 #tổng số lần lặp

x\_new=a

localMin=[]# Tạo 1 list để lưu cực tiểu

localMax=[]# Tạop 1 list để lưu cực đại

iterationOfEachMinMax=[] #Tạo 1 list để đếm mỗi lần lặp

i=0

#chương trình chính

while(i<300):

(minMax,itera,**sign**)=GD2(ETA,x\_new)

x\_new=minMax

# Kiểm tra điểm tới hạn có phải cực trị hay không?

if fp(x\_new-0.1)\*fp(x\_new+0.1)<0 :

if( **sign**>0 and minMax <= b ):

localMax.append(minMax)

if f(minMax)>f(tmax): tmax=minMax

if(**sign**<0 and minMax <= b):

localMin.append(minMax)

if f(minMax)<f(tmin): tmin=minMax

iterationOfEachMinMax.append(itera)

Sum+=itera

while (abs(fp(x\_new))<1e-9 or abs(f(x\_new))<1e-3 and x\_new<b):

x\_new+= STEP

i+=1

if(x\_new>b): break

plt.show()

print("Số lần lặp khi đi tìm mỗi cực trị:",iterationOfEachMinMax)

print("\nTổng số lần lặp:",Sum)

print("\nLocal min",localMin)

print("\nLocal max",localMax)

print("\nGlobal min (",tmin,",",f(tmin),")" )

print("\nGlobal max (",tmax,",",f(tmax),")")

main()

luachon=str(input("Bạn có muốn chạy lại chường trình không? Yes/No\nLựa chọn của bạn:"))

Luachon=luachon.upper()

while (Luachon=='YES' or Luachon=='Y'):

main()

luachon=str(input("Bạn có muốn chạy lại chường trình không? Yes/No\nLựa chọn của bạn:"))

Luachon=luachon.upper()

if (Luachon=='NO' or Luachon=='N'):

text="Chương trình đang kết thúc . . .\n"

for char in text:

sys.stdout.write (char)

time.sleep(0.07)

print("Đã kết thúc!")

# Đánh giá phương pháp

## Ưu điểm:

* Thuật toán hội tụ nhanh với nhiều dạng hàm​.
* Gradient Descent có lợi thế khi việc khảo sát hàm số để tìm cực trị là quá hoặc quá phức tạp.

## Hạn chế:

* f(x) và f'(x) phải cùng liên tục trên [a, b]​.
* Gặp khó khăn với hàm có khoảng cách cực trị quá bé hoặc quá dốc.

# Phương pháp duyệt thông thường

## Ý tưởng và xây dựng phương pháp

* Chia [a, b] thành các đoạn rất nhỏ. Tính giá trị tại các điểm rồi so sánh giá trị để tìm ra f(x) nhỏ nhất và lớn nhất.
* Sai số  |xn − x\* |< step.​
* Đánh giá​
* Chương trình khá tốt, sai số có thể điều chỉnh khoảng chia.
* Thời gian chạy lâu, các bước lặp khá lớn.​

# Ứng dụng

Ở đây chúng em tập trung và ứng dụng của Gradient Descent.

Như phần đầu nhóm cũng đã đề cập đến, phương pháp này ứng dụng rất nhiều trong Machine Learning,Deep Learning để tìm điểm cực đại, cực tiểu của hàm một biến, và đặc biệt với hàm nhiều biến hay đi giải phương trình f’(x)=0 bằng các biến thể của Gradient decent

[1] Giáo trình Giải tích số của thầy Lê Trọng Vinh.

[2][http://tutorials.aiclub.cs.uit.edu.vn/index.php/2020/06/07/machine-learning-gradient-descent-la-gi-phan-1-5/#:~:text=%E2%80%9CGradient%20Descent%20l%C3%A0%20m%E1%BB%99t%20thu%E1%BA%ADt,m%E1%BA%A5t%20m%C3%A1t%20Loss%20function\*).%E2%80%9D](http://tutorials.aiclub.cs.uit.edu.vn/index.php/2020/06/07/machine-learning-gradient-descent-la-gi-phan-1-5/#:~:text=%E2%80%9CGradient%20Descent%20l%C3%A0%20m%E1%BB%99t%20thu%E1%BA%ADt,m%E1%BA%A5t%20m%C3%A1t%20Loss%20function*).%E2%80%9D)