**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC**

**---o0o---**


# BÁO CÁO MÔN: 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

| Họ và tên | MSSV | Mã lớp |
|---|---|---|
| 1.Lê Trung Kiên | 20195893 | 118210 |
| 2.Nguyễn Thị Hường | 20195885 | 118210 |

HÀ NỘI-12/2020

# MỤC LỤC

# PHẦN MỞ ĐẦU

Giải tích số là môn học về các nghiên cứu, phương pháp về 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 hợn. 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!

Nhóm sinh viên nhóm 7
Lê Trung Kiên
Nguyễn Thị Hường

# PHẦN NỘI DUNG

## I.Bài toán

Tìm giá trị Min, Max của hàm $f(x)$ trên [a;b]

Giả thiết (a,b) là một khoảng cho trước và $f(x)$ là hàm liên tục trên [a,b].

## II.Phương pháp

Đối với đề tài này nhóm bọn em sẽ sử dụng 2 phương pháp khác nhau qua đó phân tích so sánh để làm rõ ưu, nhược điểm của chúng.

### 2.1. Gradient decent

a, Giới thiệu

**Gradient Descent**

Trong Machine Learning nói riêng và Toán Tối Ưu nói chung, chúng ta thường xuyên phải tìm giá trị nhỏ nhất (hoặc đôi khi là lớn nhất) của một hàm số nào đó. Ví dụ như các hàm mất mát trong hai bài Linear Regression và K-means Clustering. Nhìn chung, việc tìm giá trị nhỏ nhất(Min) của các hàm mất mát trong Machine Learning là rất phức tạp, thậm chí là bất khả thi. Thay vào đó, người ta thường cố gắng tìm các điểm cực tiểu, và ở một mức độ nào đó, coi đó là nghiệm cần tìm của bài toán.

Các điểm cực tiểu $x^*$ là nghiệm của phương trình đạo hàm bằng 0. Nếu bằng một cách nào đó có thể tìm được toàn bộ (hữu hạn) các điểm cực tiểu, ta chỉ cần thay từng điểm $x^*$ đó vào hàm số rồi tìm điểm làm cho hàm có giá trị nhỏ nhất; các bước tương tự với việc tìm giá trị lớn nhất thông qua điểm cực đại (*đoạn này nghe rất quen thuộc, đúng không?*). Tuy nhiên, trong hầu hết các trường hợp, việc giải phương trình đạo hàm bằng 0 là bất khả thi. Nguyên nhân có thể đến từ sự phức tạp của dạng của đạo hàm, từ việc các điểm dữ liệu có số chiều lớn, hoặc từ việc có quá nhiều điểm dữ liệu.

Hướng tiếp cận phổ biến nhất là xuất phát từ một điểm mà chúng ta coi là *gần* với nghiệm của bài toán, sau đó dùng một phép toán lặp để *tiến dần* đến điểm cần tìm, tức đến khi đạo hàm gần với 0. Gradient Descent (viết gọn là GD) và các biến thể của nó là một trong những phương pháp được dùng nhiều nhất.

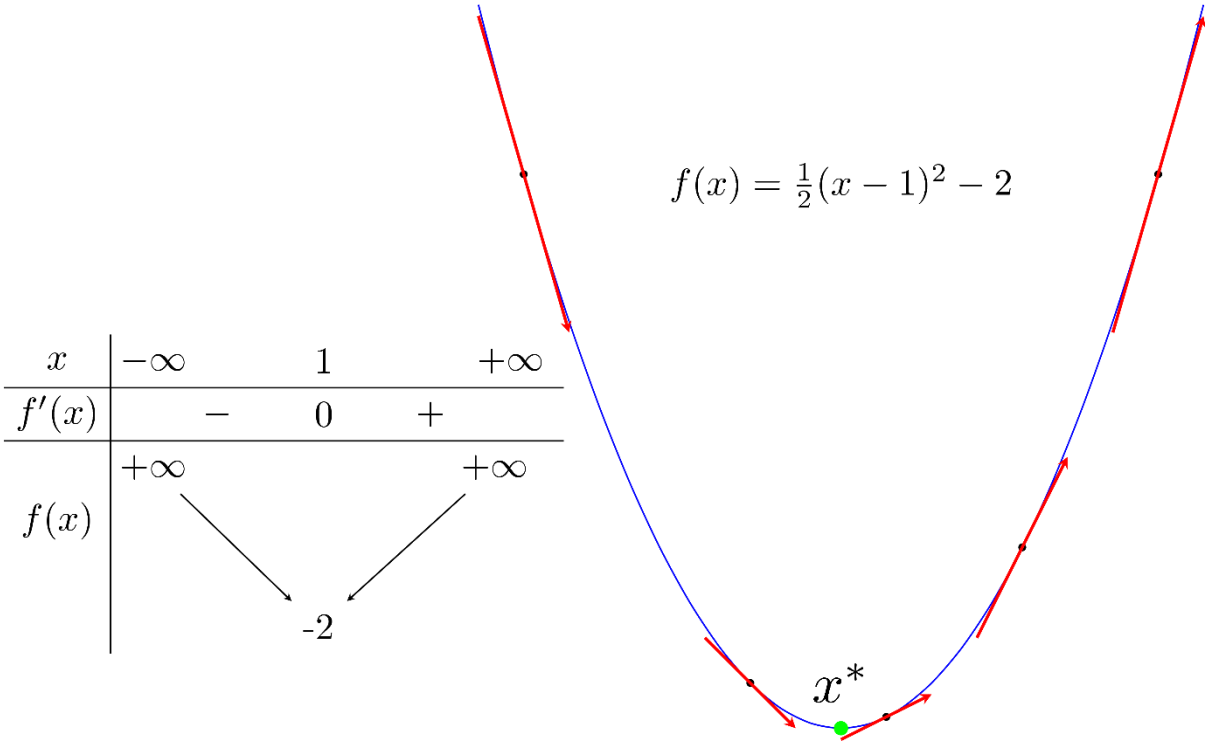
Vì kiến thức về GD khá rộng nên trong giới hạn bản báo cáo này chúng em chỉ xin phép để cấp đến phương pháp GD dành cho hàm một biến.

Trong kiến thức toán phổ thông chúng ta đã biết, muốn tìm cực trị một hàm số y=f(x) chúng ta sẽ giải phương trình đạo hàm của hàm số f(x) bằng 0.

$$f'(x)=0$$

Tuy nhiên phương trình trên không phải lúc nào cũng giải được dễ dàng, 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.

Có thể thấy hình vẽ dưới đây quen thuộc:




Điểm màu xanh lục là điểm cực tiểu, và cũng là điểm làm cho hàm số đạt giá trị nhỏ nhất.

Giả sử chúng ta đang quan tâm đến một hàm số một biến có đạo hàm mọi nơi. Bọn em sẽ nhắc lại vài điều đã quá quen thuộc:

1. Điểm điểm cực tiểu $x^*$ của hàm số là điểm có đạo hàm $f(x^*) = 0$. Hơn thế nữa, trong lân cận của nó, đạo hàm của các điểm phía bên trái $x^*$ là không dương, đạo hàm của các điểm phía bên phải $x^*$là không âm.

2. Đường tiếp tuyến với đồ thị hàm số đó tại 1 điểm bất kỳ có hệ số góc chính bằng đạo hàm của hàm số tại điểm đó.

Trong hình phía trên, các điểm bên trái của điểm cực tiểu $x^*$ màu xanh lục có đạo hàm âm, các điểm bên phải có đạo hàm dương. Và đối với hàm số này, càng xa về phía trái của điểm $x^*$ thì đạo hàm càng âm, càng xa về phía phải thì đạo hàm càng dương.

b, Xây dựng công thức:

Đối với cực tiểu

- $x^*$ với $f'(x^*) = 0$
- Tại vị trí $x_n$ bất kì
- Nếu đạo hàm của hàm số tại $x_n$ : $f'(x_n) > 0$ thì $x_n$ nằm về bên phải so

với $x*$ (và ngược lại). Để điểm tiếp theo $x_{n+1}$ gần với $x^*$ hơn, chúng ta cần di chuyển $x_{n+1}$ về phía bên trái, tức về phía *âm*. Nói các khác, chúng ta cần di chuyển ngược dấu với đạo hàm: $x_{n+1} = x_n + \Delta$. Trong đó $\Delta$ là một đại lượng ngược dấu với đạo hàm $f'(x_n)$

- $x_n$ càng xa $x^*$về phía bên phải thì $f'(x_n)$ càng lớn hơn 0 (và ngược lại).

Vậy, lượng di chuyển $\Delta$ , một cách trực quan nhất, là tỉ lệ thuận với $-f'(x_n)$. Tương tự với cực đại.

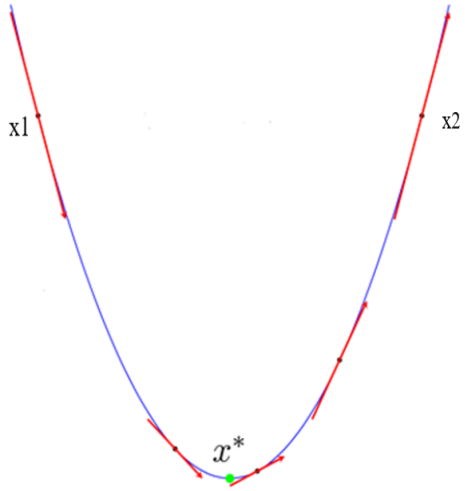
- Các nhận xét phía trên cho chúng ta có CTTQ:

  $x_n = x_{n-1} + \text{sign} * \eta * f'(x_{n-1})(*)$

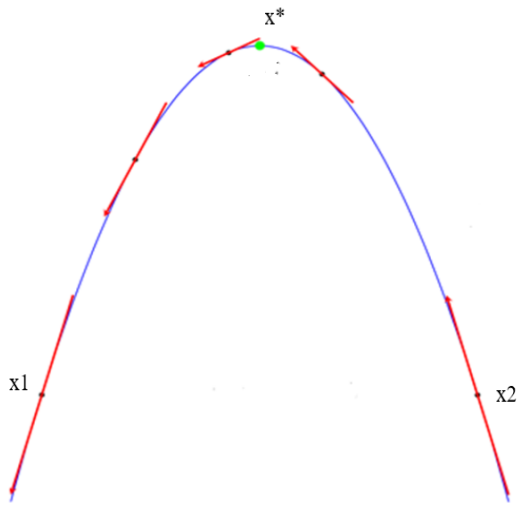
  *(Với sign=-1 khi lặp tìm cực tiểu và sign=1 khi lặp tìm cực đại)*
- Với $\boldsymbol{\eta}$ ($\boldsymbol{eta}$) là một số dương được gọi là *learning rate* (tốc độ học)
- => $x_n \approx x^*$

c, Áp dụng vào cho bài toán tìm giá trị lớn nhất và nhỏ nhất trên khoảng đóng [a,b]

Cực tiểu




Cực đại




Bắt đầu xuất phát từ $x_0$=a và di chuyển dần về điểm b.

- B1: Nếu $f'(x_0) < 0$ thì điểm cực trị tiếp theo nếu có là cực tiểu. (Tương tự nếu $f'(x_0) > 0$ thì điểm tiếp theo là cực đại.)

- B2: Ta sẽ dùng CTTQ (*) với sign=-1( hoặc 1) để lặp $x_0$ đến đủ gần $x^*$. Sau đó ta sẽ tăng $x_0$ thêm 1 khoảng ***step*** đủ nhỏ để $x_0$ vượt qua $x^*$ : $x_0 = x_0 + step$. Sau đó quay lại B1 với $x_0$ mới.
- Quá trình lặp lại đến khi $x_0$=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.

d, Điều kiện hội tụ và tốc độ hội tụ

Điều kiện $f(x)$ liên tục trên [a,b].

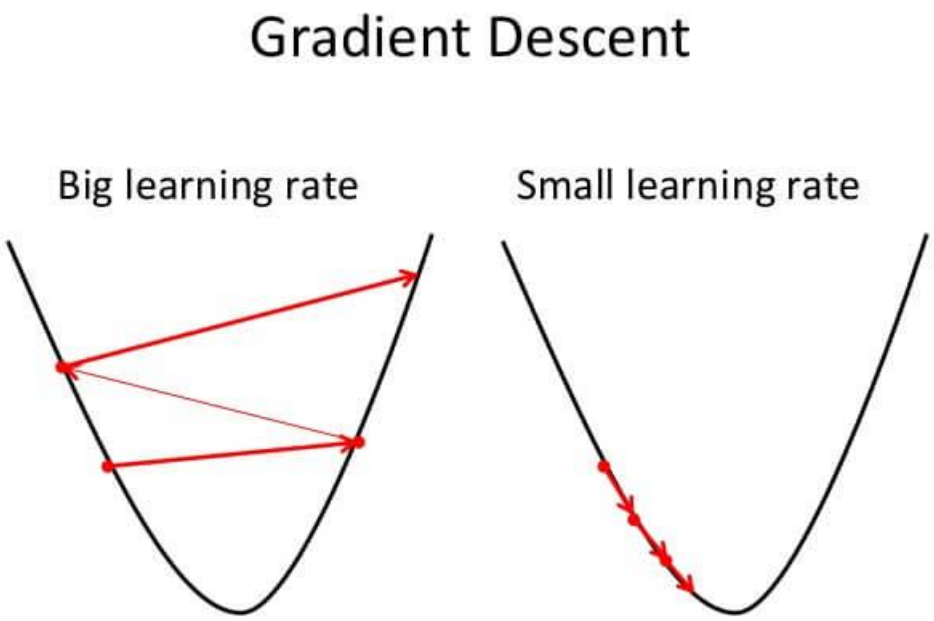
Xuất phát từ (*) chúng ta thấy tốc độ hội tụ của GD phụ thuộc vào:

- Điểm khởi tạo ban đầu
- Hệ số *learning rate* $\eta$

Giải pháp? Điều chỉnh các yếu tố trên

**Tìm $\eta$ hợp lý**

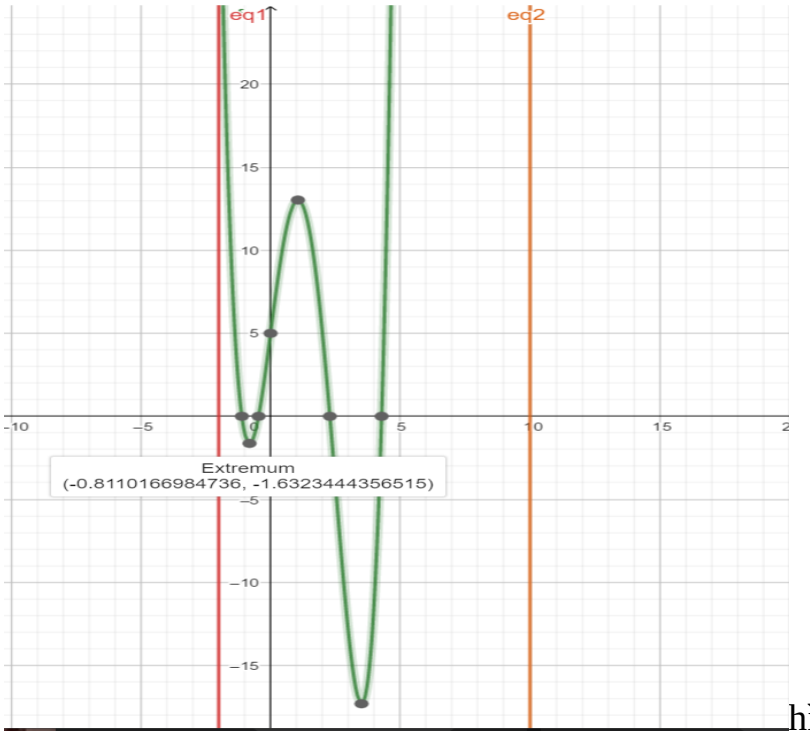
Như phần trên bọn em đã trình bày, tốc độ hội tụ của phương pháp phụ thuộc rất nhiều vào việc lấy $\eta$ *ban đầu*. Và việc tìm ra $\eta$ *hợp lý* rất khó.



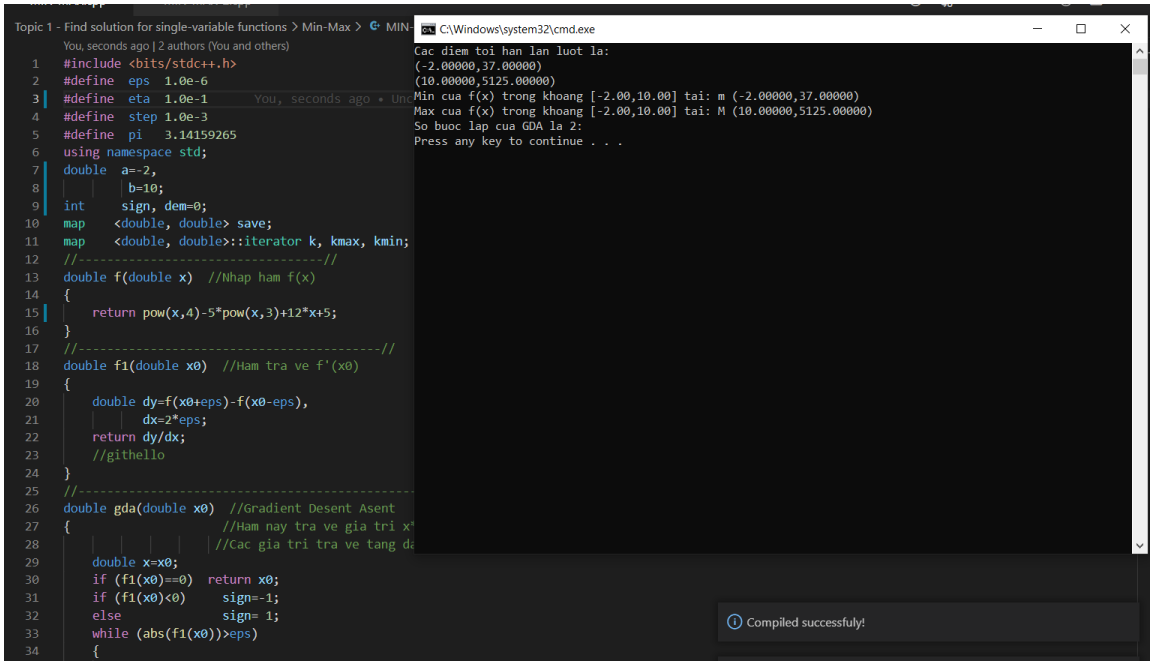
1. Với *learning rate* $\eta$ nhỏ , tốc độ hội tụ rất chậm sẽ ảnh hưởng tới tốc độ của thuật toán rất nhiều, thậm chí không bao giờ tới được đích nếu mình giới hạn số bước lặp.
2. Với *learning rate* $\eta$ lớn, thuật toán có tiến rất nhanh tới *gần đích* sau vài vòng lặp. Tuy nhiên, có trường hợp thuật toán không hội tụ được vì *bước nhảy* quá lớn, khiến nó cứ *quẩn quanh* ở đích.

Ta có so sánh rõ hơn ở ví dụ sau

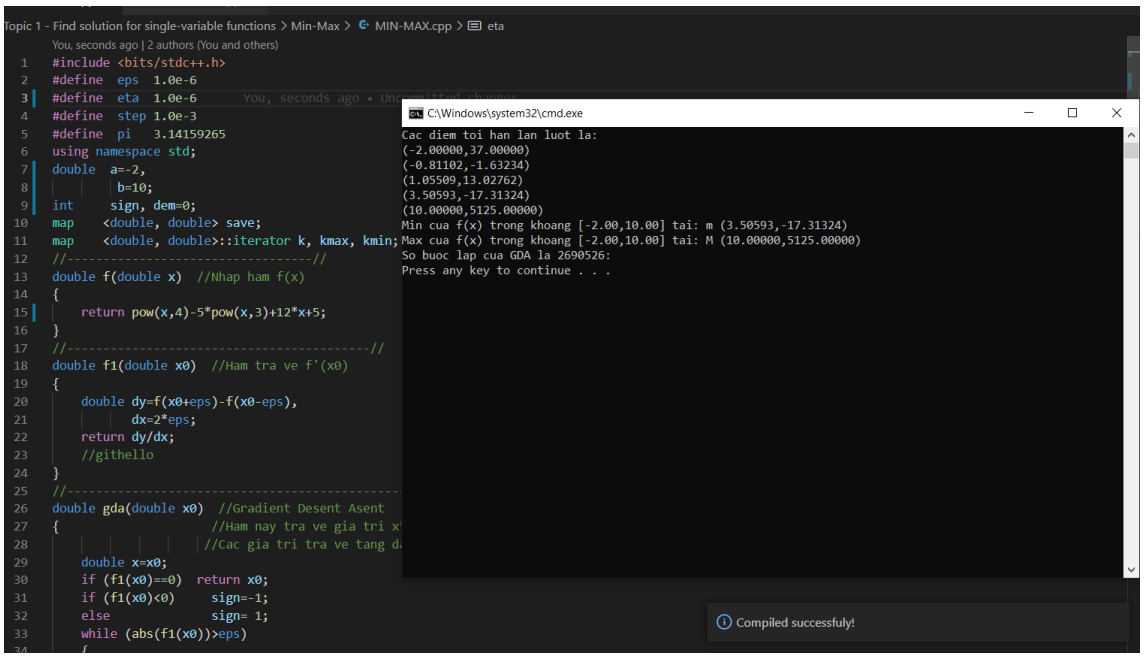
Xét hàm $f(x)= x^4-5x^3+12x+5$ trên [-2,10]



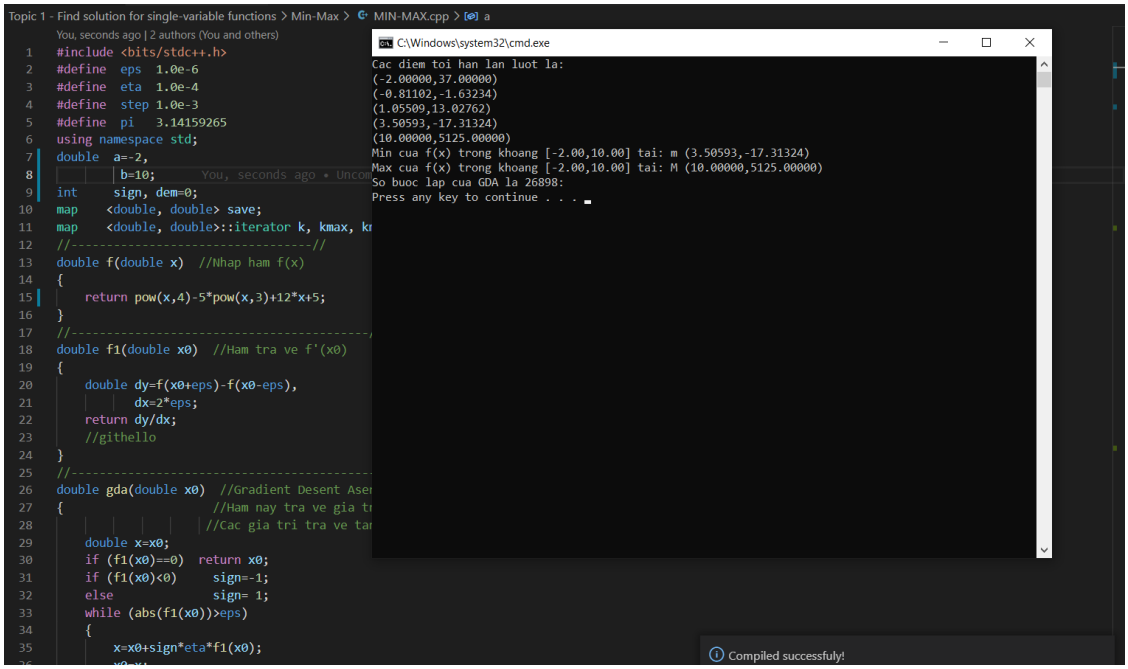
hình 4


Topic 1 - Find solution for single-variable functions > Min-Max > MIN-
You, seconds ago | 2 authors (You and others)
#include <bits/stdc++.h>
#define eps 1.0e-6
#define eta 1.0e-1
#define step 1.0e-3
#define pi 3.14159265
using namespace std;
double a=-2,
b=10;
int sign, dem=0;
map <double, double> save;
map <double, double>::iterator k, kmax, kmin;
//----------//
double f(double x) //Nhap ham f(x)
{
return pow(x,4)-5*pow(x,3)+12*x+5;
}
//----------//
double f1(double x0) //Ham tra ve f'(x0)
{
double dy=f(x0+eps)-f(x0-eps),
dx=2*eps;
return dy/dx;
//githello
}
//----------
double gda(double x0) //Gradient Desent Asent
{
//Ham nay tra ve gia tri x*
//Cac gia tri tra ve tang da
double x=x0;
if (f1(x0)==0) return x0;
if (f1(x0)<0) sign=-1;
else sign= 1;
while (abs(f1(x0))>eps)
{
C:\Windows\system32\cmd.exe
Cac diem toi han lan luot la:
(-2.00000,37.00000)
(10.00000,5125.00000)
Min cua f(x) trong khoang [-2.00,10.00] tai: m (-2.00000,37.00000)
Max cua f(x) trong khoang [-2.00,10.00] tai: M (10.00000,5125.00000)
So buoc lap cua GDA la 2:
Press any key to continue . . .
Compiled successfuly!

hình5


Topic 1 - Find solution for single-variable functions > Min-Max > MIN-MAX.cpp > eta
You, seconds ago | 2 authors (You and others)
#include <bits/stdc++.h>
#define eps 1.0e-6
#define eta 1.0e-6
#define step 1.0e-3
#define pi 3.14159265
using namespace std;
double a=-2,
b=10;
int sign, dem=0;
map <double, double> save;
map <double, double>::iterator k, kmax, kmin;
//----------//
double f(double x) //Nhap ham f(x)
{
return pow(x,4)-5*pow(x,3)+12*x+5;
}
//----------//
double f1(double x0) //Ham tra ve f'(x0)
{
double dy=f(x0+eps)-f(x0-eps),
dx=2*eps;
return dy/dx;
//githello
}
//----------
double gda(double x0) //Gradient Desent Asent
{
//Ham nay tra ve gia tri x*
//Cac gia tri tra ve tang da
double x=x0;
if (f1(x0)==0) return x0;
if (f1(x0)<0) sign=-1;
else sign= 1;
while (abs(f1(x0))>eps)
C:\Windows\system32\cmd.exe
Cac diem toi han lan luot la:
(-2.00000,37.00000)
(-0.81102,-1.63234)
(1.05509,13.02762)
(3.50593,-17.31324)
(10.00000,5125.00000)
Min cua f(x) trong khoang [-2.00,10.00] tai: m (3.50593,-17.31324)
Max cua f(x) trong khoang [-2.00,10.00] tai: M (10.00000,5125.00000)
So buoc lap cua GDA la 2690526:
Press any key to continue . . .
Compiled successfuly!

hình 6



hình7

Có thể thấy với [a,b]=[-2,10] cố định thì

- η (ở đây chúng em đặt là eta)= 0.1 thì hàm GDA chỉ có bước lặp là 2 và không thể in ra các điểm cực trị dẫn đến kết quả sai.(hình 5)
- $\eta=10^{-6}$ chương trình vẫn chạy ra kết quả đúng như hình 4 (xem hình 6) nhưng số lần lặp của hàm GDA rất lớn : 2690526
- $\eta=10^{-4}$ chương trình chạy đúng và số bước lặp nhỏ hơn so với $\eta=10^{-6}$), chỉ có 26898 bước.

Như vậy việc lựa chọn *learning rate* $\eta$ rất quan trọng trong các bài toán thực tế. Việc lựa chọn giá trị này phụ thuộc nhiều vào từng bài toán và phải làm một vài thí nghiệm để chọn ra giá trị tốt nhất. Và $\eta$ phải thay đổi qua từng vòng lặp nhưng vì còn hạn chế nên chúng em dừng lại ở việc để $\eta$ cố định

Qua tìm hiểu ở các tài liệu cũng như bài báo liên quan, bọn em tìm thấy một bài báo có đề xuất việc tối ưu hệ số này. Bài báo đề cập đến 2 trường hợp chính:

Trường hợp 1: Với $\eta$ nhỏ không đủ để bước nhảy đưa $x_1$ vượt qua vị trí của $x^*$ thì thì 2 điểm liên tiếp sẽ vẫn cùng phía so với $x^*$ , đạo hàm của 2 điểm này sẽ cùng dấu, suy ra $f'(x_1)\,.\,f'(x_2)>0$. Vậy để điều chỉnh ta nên lấy $\eta=\eta*2$.

Trường hợp 2: Nếu $\eta$ lớn có thể khiến $x_1$ vượt qua vị trí của $x^*$ dẫn đến bỏ sót điểm cực trị cần tìm (có thể ảnh hưởng đến kết quả cuối cùng). Lúc này $f'(x_1)$ . $f'(x_2)$ <0. Để điều chỉnh lấy $\eta = \eta/2$.

- Đối với thuật toán và chương trình của phần này bọn em xin phép không đưa vào trong chương trình code, vì chúng em vẫn còn gặp một số khó khăn khi sử dụng giải pháp này nên chỉ giới thiệu qua ở bản báo cáo.

Chi tiết thì cô và mọi người có thể xem ở đây:

https://drive.google.com/file/d/1bKLCCdP7i1RUfAntfg4xOnTtKRM8ZEz0/view

**Các điều kiện dừng khác:**

Vì phụ thuộc vào hàm f(x) nên có thể sẽ có những điều kiện dừng mãi không thể xảy ra và vòng lặp sẽ rơi vào vô hạn nên chúng em đề xuất một số điều kiện dừng khác

- **Giới hạn số vòng lặp:** đây là phương pháp phổ biến nhất và cũng để đảm bảo rằng chương trình chạy không quá lâu. Tuy nhiên, một nhược điểm của cách làm này là có thể thuật toán dừng lại trước khi đủ gần với nghiệm.
- **Kiểm tra giá trị đạo hàm:** So sánh gradient của nghiệm tại hai lần cập nhật liên tiếp, khi nào giá trị này đủ nhỏ thì dừng lại.

### 2.2. Phương pháp tìm duyệt thông thường

Ý tưởng và xây dựng phương pháp

Đây là cách tìm min max cơ bản nhất, không cần quan tâm đến dấu của f'(x) mà chỉ cần f(x) liên tục trên [a,b].

Đúng như tên gọi chúng ta sẽ chia đoạn [a;b] thành các đoạn nhỏ và tính giá trị tại các điểm liên tiếp nhau cách nhau một đoạn ***step*** rất nhỏ và so sánh các giá trị. Sau đó chọn ra điểm làm giá trị f(x) nhỏ nhất và lớn nhất.

Sai số là $|x_n - x_*|$< ***step*** nhỏ >0. ***Step*** càng nhỏ thì độ chính xác càng cao nhưng thời gian chạy cũng lâu hơn.

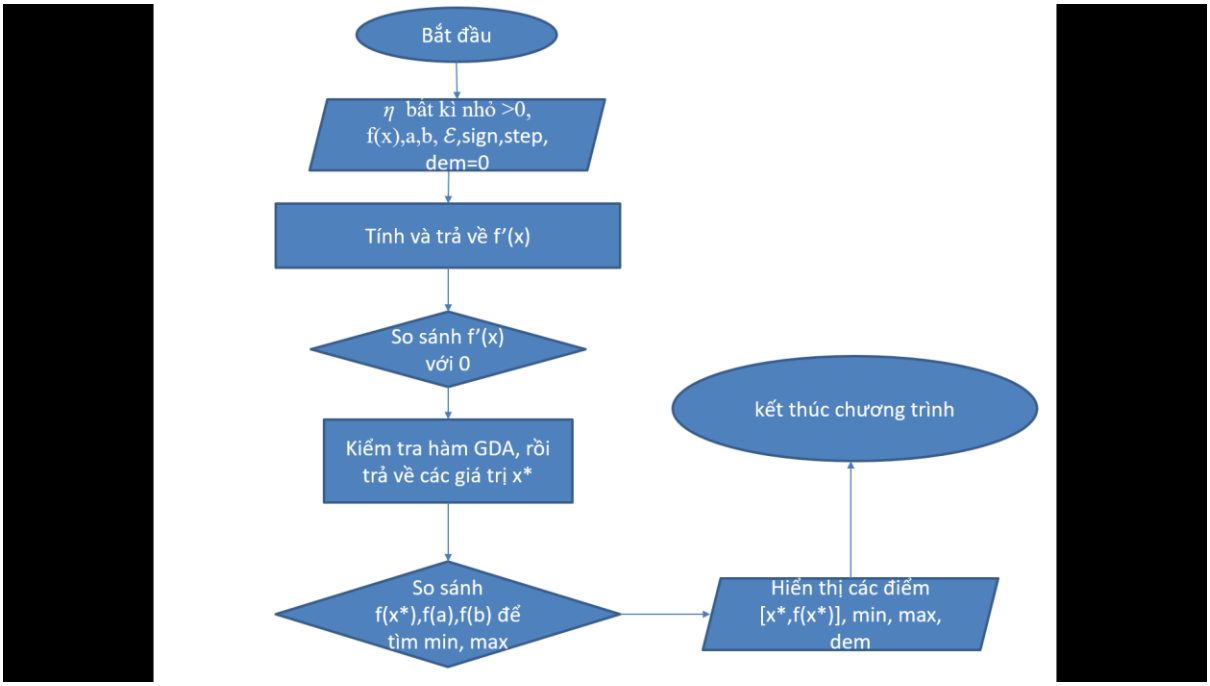
Rồi so sánh các f(x) và xuất ra min, max

## III. Thuật toán, chương trình và ví dụ minh họa

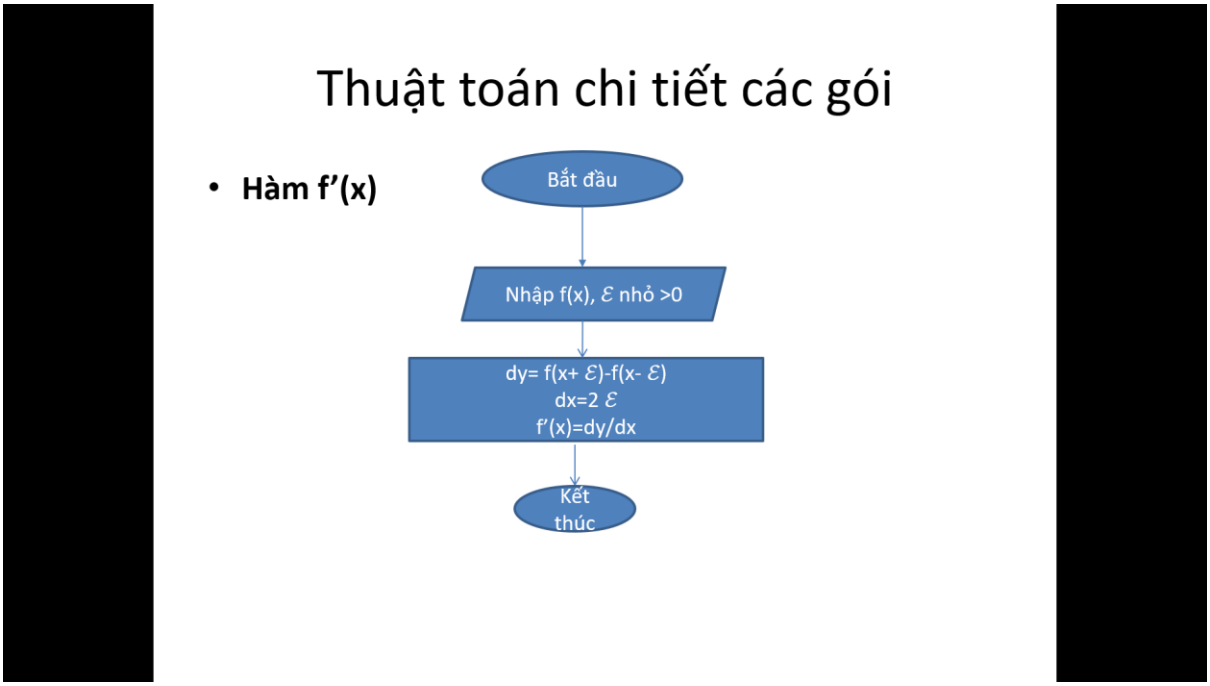
## 3.1.Phương pháp Gradient decent

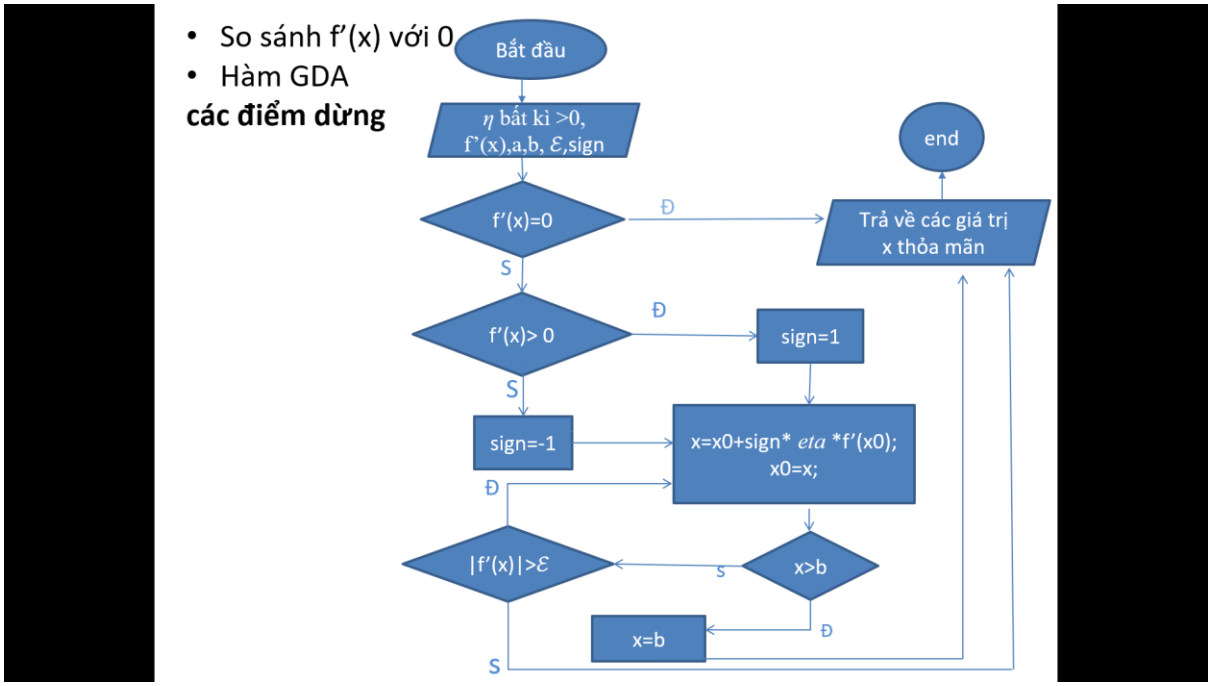
### 3.1.1.Thuật toán

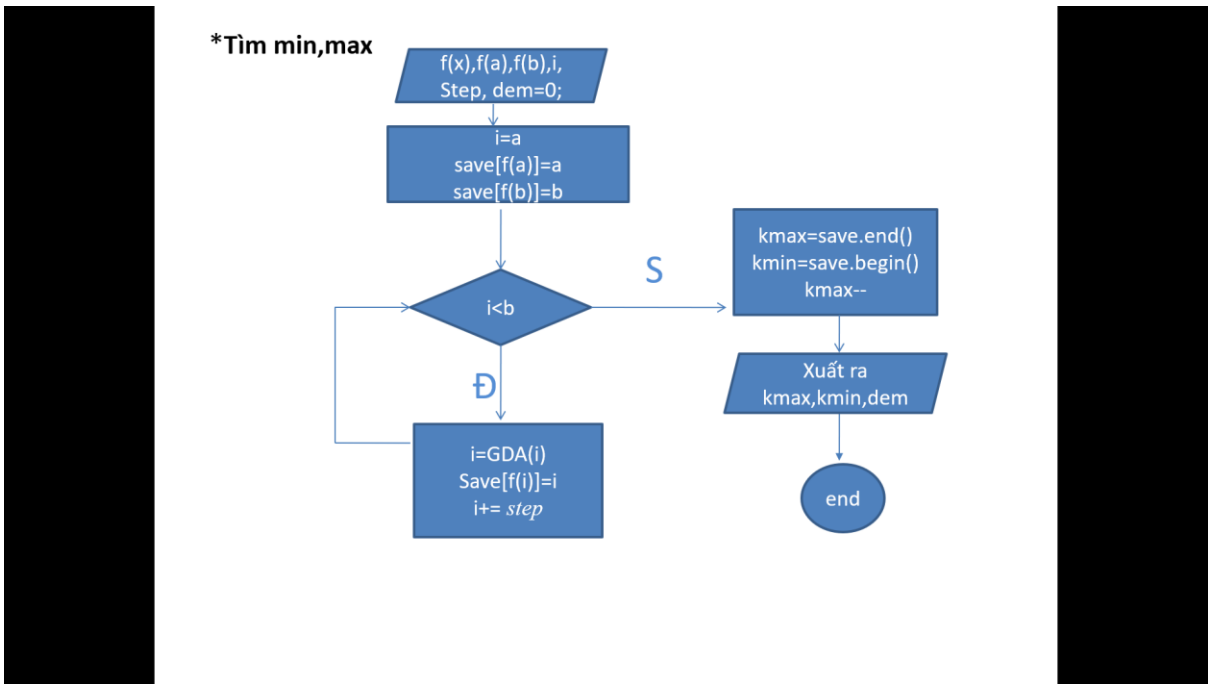
Sơ đồ khối chung




Sơ đồ khối chi tiết các gói.






Có sử dụng " Map" trong c++

3.1.2.Chương trình

Chương trình được code bằng ngôn ngữ C++

```
#include <bits/stdc++.h>
#define  eps  1.0e-6
#define  eta  1.0e-3
#define  step 1.0e-3
```

```
#define  pi   3.14159265
#define e   2.718281828459
using namespace std;
double  a=0,
        b=4;
int     sign, dem=0;
map    <double, double> save;
map    <double, double>::iterator k, kmax, kmin;
//--------------------------------//
double f(double x)  //Nhap ham f(x)
{
   return pow(x,2)+12*x+5;
}
//---------------------------------------//
double f1(double x0)  //Ham tra ve f'(x0)
{
  double dy=f(x0+eps)-f(x0-eps),
        dx=2*eps;
  return dy/dx;

  //githello
}
//-----------------------------------------------------------------------------------------//
double gda(double x0)  //Gradient Desent Asent
{                   //Ham nay tra ve gia tri x*>x0 thoa man f'(x*)=0
                   //Cac gia tri tra ve tang dan vi x0 tang dan (i:=a->b)
   double x=x0;
   if (f1(x0)==0)  return x0;
  if (f1(x0)<0)     sign=-1;
  else              sign= 1;
  while (abs(f1(x0))>eps)
  {
     x=x0+sign*eta*f1(x0);
    x0=x;
    if (x0>b)   return b;
    dem++;
  }
  return x;
}
//-------------------------------------------------------------------------------//
void luutru()   //Luu cac x* f(x*), a f(a), b f(b) vao map
{
  double i=a;
```

```
   save[a]=f(a),
   save[b]=f(b);
    while (i<b)
    {
       i=gda(i);
       save[i]=f(i);
       do
       {
          i+=step;
          if (i>=b) break;
       }
       while ((f1(i)*sign>0) && (abs(f1(i) > f1(i+step))));
    }
}
//-----------------------------------------------//
void xuat1()  //Xuat cac diem toi han
{
   cout<<"Cac diem toi han lan luot la: "<<endl;
   for (k=save.begin(); k!=save.end(); k++)
   {
      printf("(%5.5f,%5.5f)\n",k->first,k->second);
   }
}
//-----------------------------------------------//
void xuat2() //Tim va xuat max min
{
    kmax=kmin=save.begin();
   for (k=save.begin(); k!=save.end(); k++)
   {
      if (k->second > kmax->second) kmax=k;  // tim f(x) max
      if (k->second < kmin->second) kmin=k; //  tim f(x) min
   }
   printf("Min cua f(x) trong khoang [%5.2f,%5.2f] tai: m
(%5.5f,%5.5f)\n",a,b,kmin->first,kmin->second);
   printf("Max cua f(x) trong khoang [%5.2f,%5.2f] tai: M
(%5.5f,%5.5f)\n",a,b,kmax->first,kmax->second);
   printf("So buoc lap cua GDA la %d:",dem);
}
//--------------------------//
int main()
{
    luutru();
   xuat1();
```

```
  xuat2();
}
```

3.1.3 Ví dụ chi tiết cho thuật toán

Ví dụ 1 Xét hàm f(x)= $x^2 + 12x + 5$ *với* $\boldsymbol{eta} = \boldsymbol{\eta} = \boldsymbol{10^{-4}}$

[a,b]=[-10,0] thỏa mãn điều kiện liên tục.

- Tìm f'(x)

  Tính $f'(x) = \lim_{\varepsilon \to 0} \left[\frac{f(x+\varepsilon)-f(x-\varepsilon)}{2\varepsilon}\right]$

$= \lim_{\varepsilon \to 0} \frac{(x+\varepsilon)^2+12(x+\varepsilon)+5-[(x-\varepsilon)^2+12(x-\varepsilon)+5]}{2\varepsilon}$

$= \lim_{\varepsilon \to 0} \frac{4x\varepsilon+24\varepsilon}{2\varepsilon}$

$= \lim_{\varepsilon \to 0} (2x + 12)$

$= 2x + 12$

Vậy $f'(x) = 2x + 12$

- So sánh f'(x) với 0 và xuất các điểm dừng(f'(x)=0)

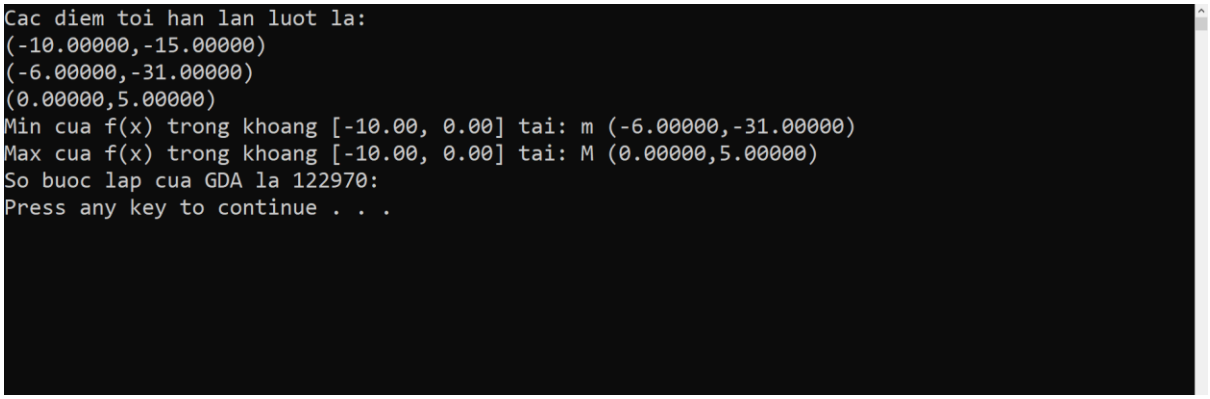
$$x_0 = a = -10$$
$$f'(x_o) = -8 < 0 \Rightarrow sign = -1$$
$$x = x0 - eta.8 = -10 - 0.0001.(-8) = -9.992$$

Rồi kiểm tra f'(x)

+ Nếu |f'(x)| < eps thì xuất x vào mảng

+ Nếu |f'(x)| > eps thì tiếp tục quá trình trên.

- Kết quả



Số bước lặp 122970

***Kiểm tra lại***

Theo trên $f'(x) = 2x + 12$

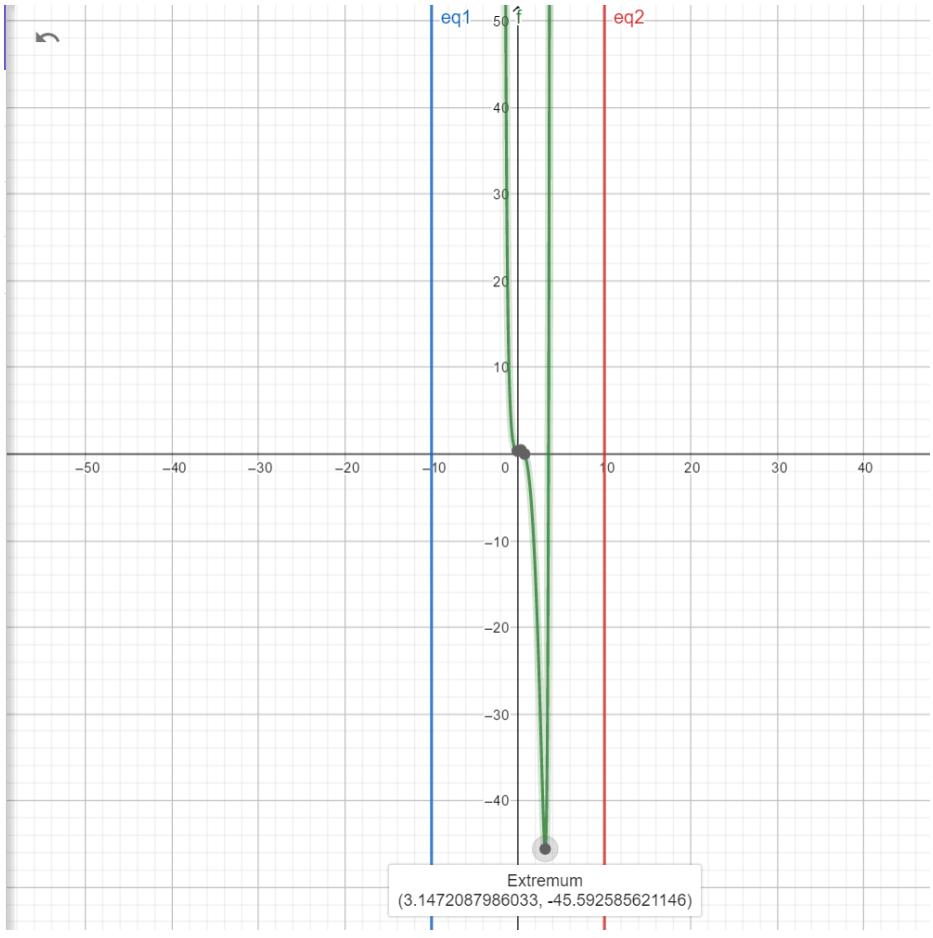
⇨ $f'(x) = 0 <=> 2x + 12 = 0 => x = -6$ là điểm cần tìm.

$$rồi\ tính\ f(-6) = -31, f(-10) = -15,$$
$$f(0) = 12$$
$$Vậy\ minf(x) = -31\ tại x = -6;$$
$$maxf(x) = 5\ tại\ x = 0$$

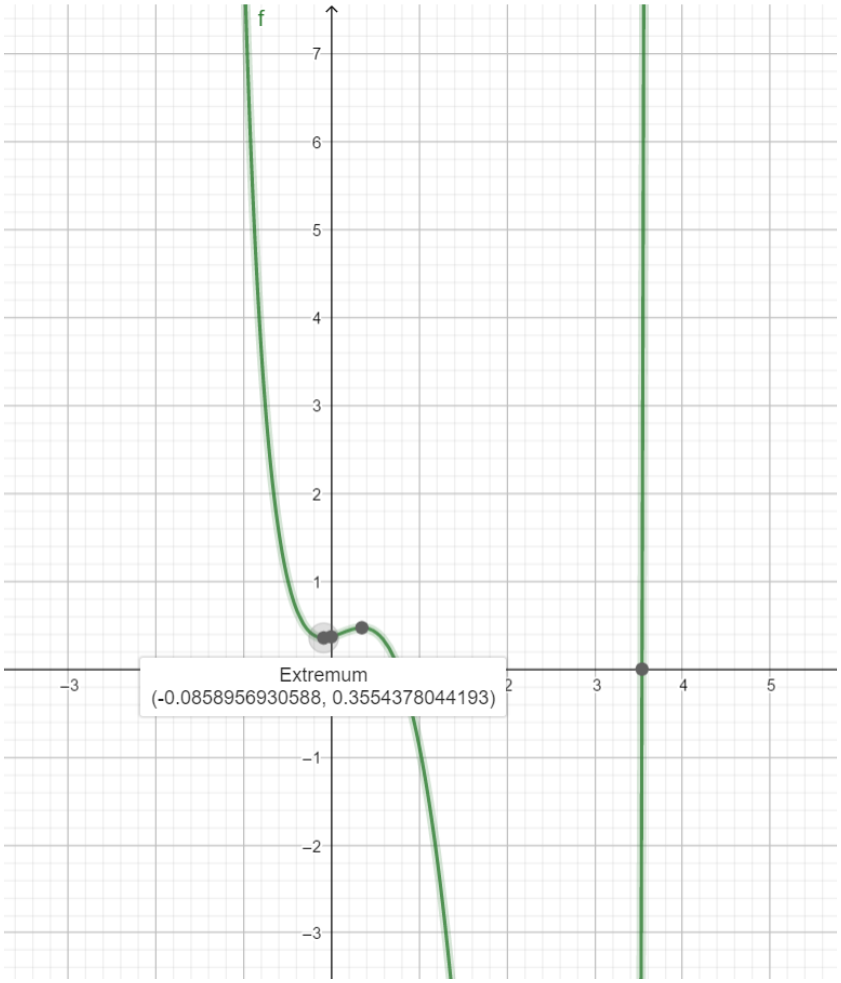
*Như vậy chương trình đã chạy ra kết quả chính xác.

Ví dụ 2: f(x)=$e^{x^2-2x-1} + x - 2x^3$, trên [-10,10] và trên [-2,10], $\boldsymbol{\eta = 10^{-4}}$

Ta có đồ thị hàm số



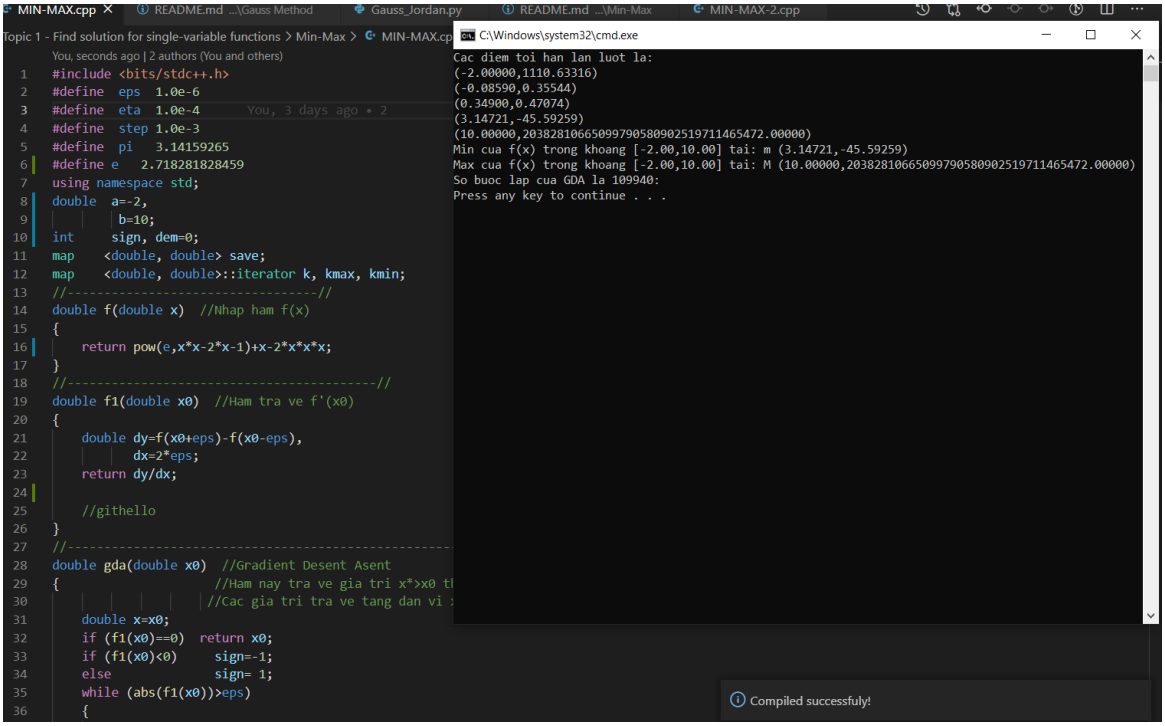

Đồ thị được phóng to như sau




```
Topic 1 - Find solution for single-variable functions > Min-Max > MIN
You, seconds ago | 2 authors (You and others)
#include <bits/stdc++.h>
#define  eps  1.0e-6
#define  eta  1.0e-4        You, 3 days ago • 2
#define  step 1.0e-3
#define  pi   3.14159265
#define e   2.718281828459
using namespace std;
double  a=-10,
        b=10;
int     sign, dem=0;
map    <double, double> save;
map    <double, double>::iterator k, kmax, kmin;
//--------------------------------//
double f(double x)  //Nhap ham f(x)
{
    return pow(e,x*x-2*x-1)+x-2*x*x*x;
}
//----------------------------------------//
double f1(double x0)  //Ham tra ve f'(x0)
{
    double dy=f(x0+eps)-f(x0-eps),
           dx=2*eps;
    return dy/dx;

    //githello
}
//-------------------------------------------------------------------------------------//
double gda(double x0)  //Gradient Desent Asent
{                      //Ham nay tra ve gia tri x*>x0 thoa man f'(x*)=0
                     //Cac gia tri tra ve tang dan vi x0 tang dan (i:=a->b)
    double x=x0;
```

```
C:\Windows\system32\cmd.exe
(-10.00000,47978133272897650234590297974011104620927872516751 36.00000)
(10.00000,203828106650997905809025197114654 72.00000)
Min cua f(x) trong khoang [-10.00,10.00] tai: m (10.00000,203828106650997905809025197114654 72.00000)
Max cua f(x) trong khoang [-10.00,10.00] tai: M (-10.00000,47978133272897650234590297974011104620927872516751
5136.00000)
So buoc lap cua GDA la 0:
Press any key to continue . . .
```



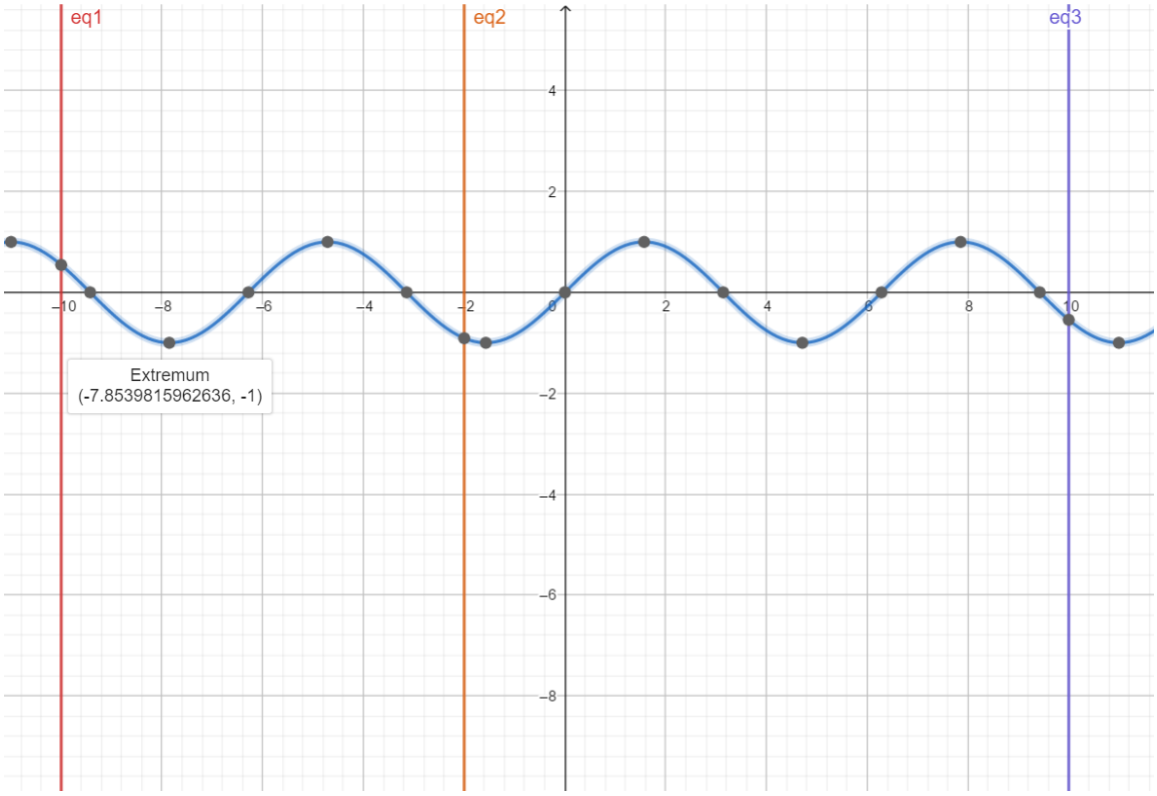

Với hàm f(x) này với [a,b]=[10,10] hàm GDA không thực hiện được vì vậy đã đưa ra kết quả sai.

Nhưng với [a,b]=[-2,10] chương trình chạy ra kết quả đúng như hình trên.

**Nguyên nhân là do**:

- Trên [-10;10] thì |f'(x)| rất lớn. Ví dụ tại $x_0 = a = -10$: $|f'(x_0)| \approx 10^{54}$ dẫn đến tại vòng lặp đầu tiên: $x = x_0 - \eta * f'(x_0) > b$ và vòng lặp kết thúc (bước nhảy quá lớn, hệ số $\eta$ không đủ nhỏ để hãm $f'(x_0)$ lại). Lúc đó sẽ không có điểm dừng và Min Max chính là giá trị tại 2 đầu mút [-10;10]
- Còn trên [2;10] cũng tương tự nhưng $\eta$ đủ nhỏ để làm cho bước nhảy không quá lớn

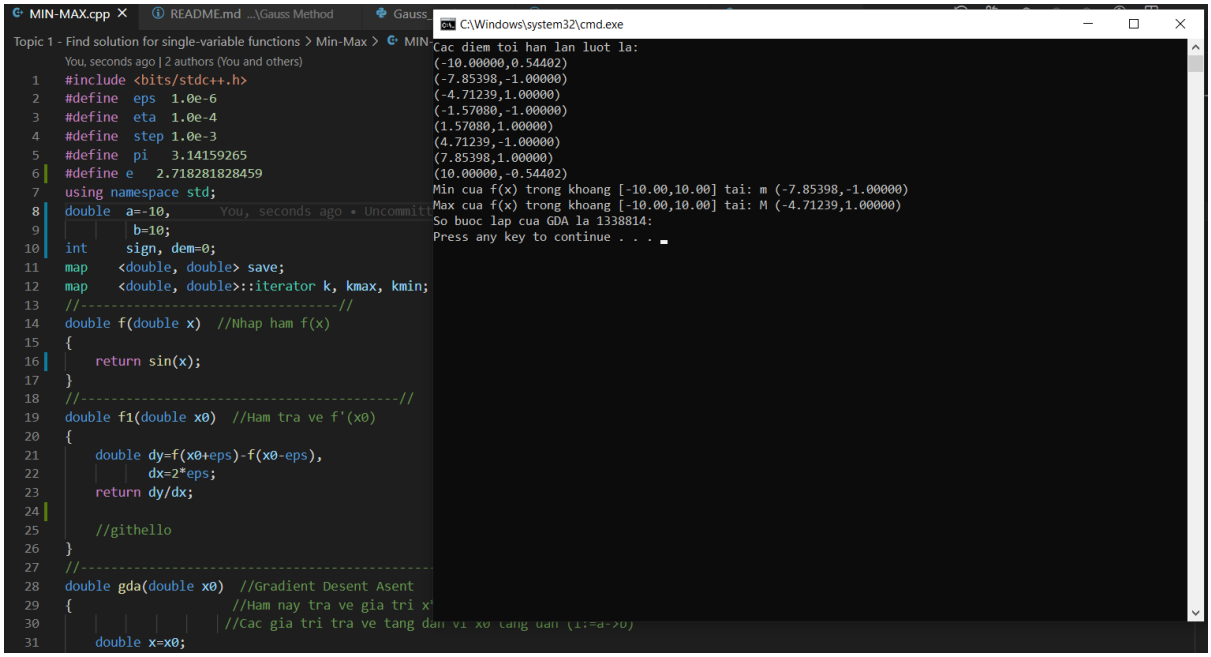
*Nhưng đối với những hàm có tính chất đặc biệt như sin(x), cos(x) kết quả lại không bị ảnh hưởng bởi điều này bởi $|f'(x)| \leq 1$ điều đó làm cho bước nhảy ở các lần lặp không quá lớn và các điểm dừng sẽ không bị bỏ sót, kết quả đưa ra sẽ chính xác hơn. Điều này tương tự đối với các hàm f(x) mà có |f'(x)| trên [a;b] không quá lớn.




```
Topic 1 - Find solution for single-variable functions > Min-Max > MIN-MAX.cpp
You, seconds ago | 2 authors (You and others)
#include <bits/stdc++.h>
#define  eps  1.0e-6
#define  eta  1.0e-4
#define  step 1.0e-3
#define  pi   3.14159265
#define e   2.718281828459
using namespace std;
double  a=-2,
        b=10;
int     sign, dem=0;
map    <double, double> save;
map    <double, double>::iterator k, kmax, kmin;
//---------------------------------//
double f(double x)  //Nhap ham f(x)
{
    return sin(x);      You, seconds ago • Uncommitted
}
//-----------------------------------------//
double f1(double x0)  //Ham tra ve f'(x0)
{
    double dy=f(x0+eps)-f(x0-eps),
           dx=2*eps;
    return dy/dx;

    //githello
}
//-----------------------------------------------------
double gda(double x0)  //Gradient Desent Asent
{                     //Ham nay tra ve gia tri x*>x0 thoa man f'(x*)=0
                     //Cac gia tri tra ve tang dan vi x0 tang dan (i:=a->b)
```

```
C:\Windows\system32\cmd.exe
Cac diem toi han lan luot la:
(-2.00000,-0.90930)
(-1.57080,-1.00000)
(1.57080,1.00000)
(4.71239,-1.00000)
(7.85398,1.00000)
(10.00000,-0.54402)
Min cua f(x) trong khoang [-2.00,10.00] tai: m (-1.57080,-1.00000)
Max cua f(x) trong khoang [-2.00,10.00] tai: M (1.57080,1.00000)
So buoc lap cua GDA la 875275:
Press any key to continue . . .
```




## 3.2.Phương pháp tìm Min, Max thông thường.

### 3.2.1.Chương trình

Chương trình được code bằng ngôn ngữ C++

```
#include <bits/stdc++.h>
#define  step 1.0e-5
using namespace std;
//-------------------------//
int dem=0;
double f( double x)
{
   return  pow(x,2)+12*x+5;
}
//-----------------------------------------------//
double x( double a, double b, string s)
{
   double i, xmax, xmin;
   xmax=a; xmin=a; i=a;
   do
   {
      if (f(i) < f(xmin))
         {
            xmin=i;
         }
      if (f(i) > f(xmax))
         {
```

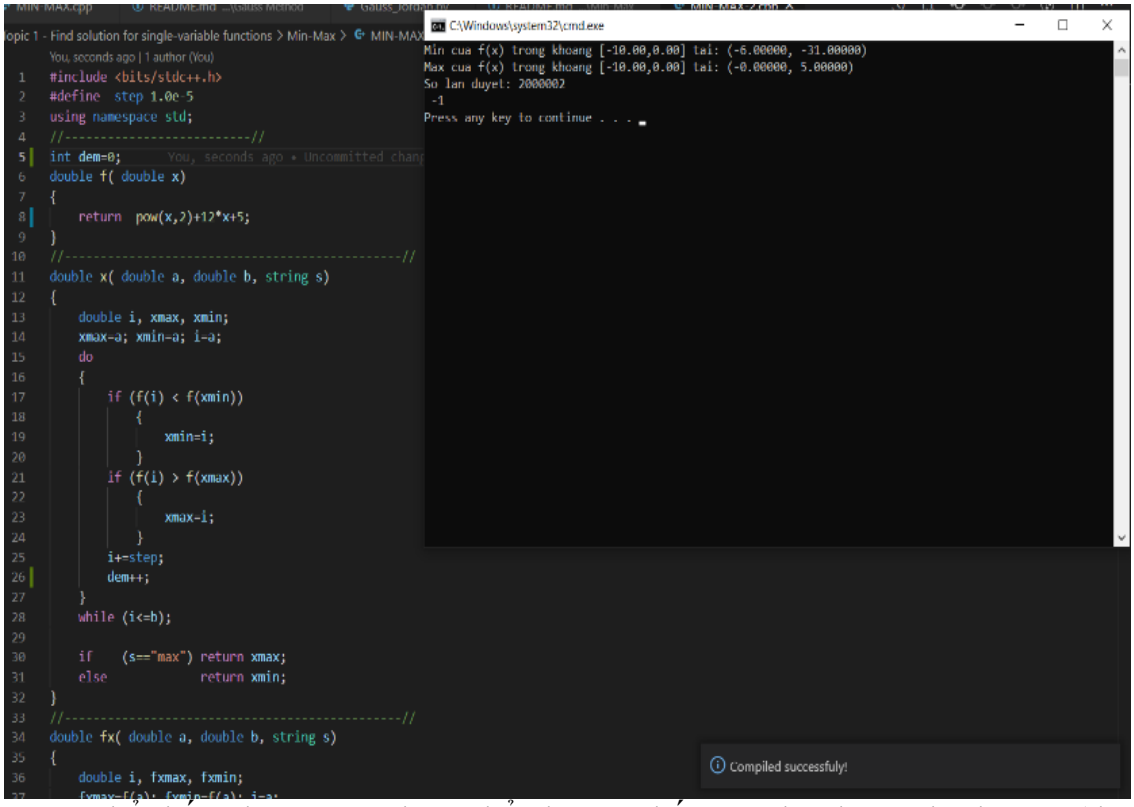
```
            xmax=i;
          }
       i+=step;
       dem++;
    }
    while (i<=b);

    if    (s=="max") return xmax;
    else           return xmin;
}
//-----------------------------------------//
double fx( double a, double b, string s)
{
    double i, fxmax, fxmin;
    fxmax=f(a); fxmin=f(a); i=a;
    do
    {
       if (f(i) < fxmin)
          {
             fxmin=f(i);
          }
       if (f(i) > fxmax)
          {
             fxmax=f(i);
          }
       i+=step;
    }
    while (i<=b);

    if    (s=="max") return fxmax;
    else           return fxmin;
}
//-----------------------------------------//
int main()
{
    double a=-10, b=0;
    printf("Min cua f(x) trong khoang [%3.2f,%2.2f] tai: (%2.5f,
%2.5f)\n",a,b,x(a,b,"min"),fx(a,b,"min"));
    printf("Max cua f(x) trong khoang [%3.2f,%2.2f] tai: (%2.5f,
%2.5f)\n",a,b,x(a,b,"max"),fx(a,b,"max"));
    printf("So lan duyet: %d",dem);
}
```

3.2.3.Ví dụ minh họa thuật toán

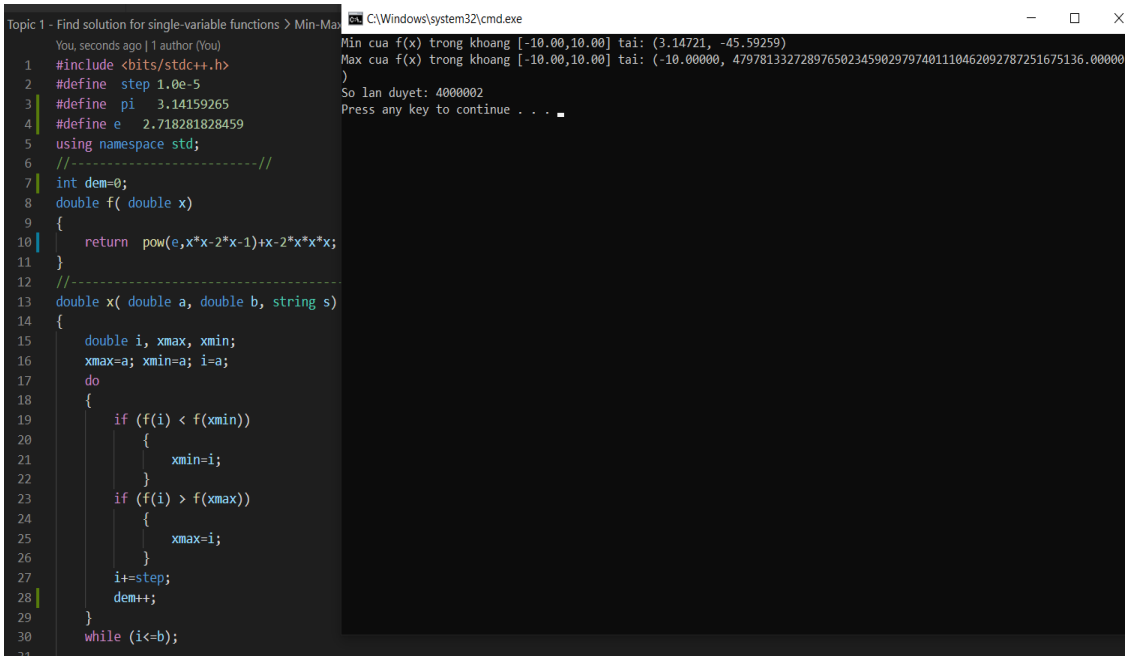
Ví dụ 1: Xét hàm số f(x)= $x^2 + 12x + 5$, trên [-10,0], với ***step = 10$^{-5}$***
Kết quả chương trình:

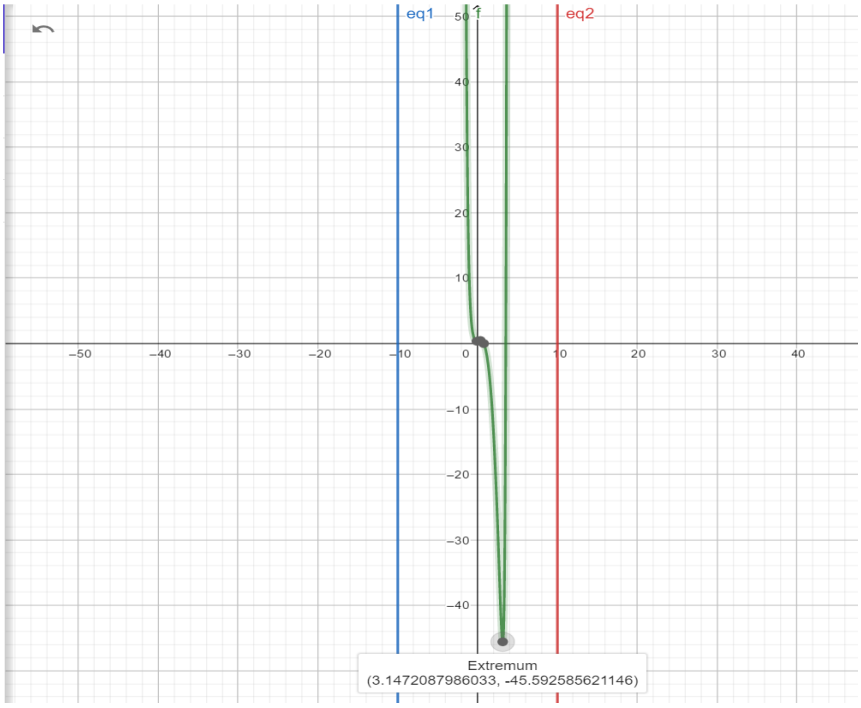



Ta có thể thấy chương trình có thể chạy và kết quả thu được chính xác. (đúng với kết quả thu được ở phương pháp Gradient Decent)

Ví dụ 2: f(x)=$e^{x^2-2x-1} + x - 2x^3$, trên [-10,10], với ***step = 10$^{-5}$***

Chương trình chạy ra kết quả sau:





Như vậy ở ví dụ này phương pháp duyệt thường này đã giải quyết nhược điểm tồn tại trên của Gradient decent mặc dù chương trình chạy khá lâu.

## IV. Đánh giá phương pháp

### 4.1 Gradient decent

Mặc dù hiện tại phương pháp này được dùng khá phổ biến để tìm Min, Max và cũng khá hiệu quả thay vì đi tính đạo hàm sẽ gây khó khăn vì các hàm phức tạp nhưng cũng vẫn còn tồn tại những hạn chế nhất định( có thể do ý kiến chủ quan của chúng em ).

### Hạn chế

- Chỉ làm được với các hàm f(x) liên tục trên [a;b]
- Chỉ làm chính xác với các hàm có f(x) và f'(x) cùng liên tục trên [a;b]
    - Do thuật toán này đi tìm các điểm x* thỏa mãn f'(x*)=0, nhưng các điểm cực trị thì không nhất thiết f(x*)=0.
    - Và f’(x) có thể không xác định, thì vẫn thỏa mãn miễn là đổi dấu khi đi qua x*, vì thế nếu f'(x) không liên tục thì nó có thể bỏ sót hoặc không chạy được.
- Có thể in ra nhiều giá trị xấp xỉ nhau xung quanh điểm x*

- Đối với các hàm f(x) có khoảng cách các cực trị quá bé, nhỏ hơn ***step*** cũng không thể chính xác. Bởi:
  - Các giá trị x* chỉ tìm được xấp xỉ chứ không chính xác
  - Sau khi tìm được x* tạm chấp nhận được, ta sẽ tăng i lên 1 đoạn ***step*** để nó vượt qua x*, do đó sẽ bị bỏ sót các điểm tới hạn trong khoảng (x*, x* + step)
- Đối với các trường hợp độ lớn của f'(x) lớn hơn nhiều so với khoảng cách các cực trị sẽ in thiếu điểm dừng vì độ lớn các bước nhảy tại các vị trí phụ thuộc vào f'(x) tại đó , gây ra kết quả tìm Min, Max cuối cùng sai

### 4.2.Phương pháp tìm duyệt thông thường

Do phương pháp này duyệt tất cả các giá trị trong [a,b] hơn kém nhau một lượng ***step*** nhỏ >0, mà không cần quan tâm f'(x) hay bị ràng buộc bởi các yếu tố khác như hệ số $\boldsymbol{\eta}$ nên chương trình rất tốt, và sai số có thể điều chỉnh theo ***step*** dễ dàng ($|\boldsymbol{x_n - x_*}|$<***step***).

Nhưng có một điều không trách khỏi là thời gian chạy chương trình khá lâu hay các bước lặp khá lớn, lớn hơn rất nhiều của Gradient decent.

## V.Ứng dụng

Ở đây chúng em tập chung và ứng dụng của Gradient decent.

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 như: **Newton's method, Mini-batch Gradient Descent, Batch Gradient Descent, Stochastic Gradient Descent.**

## VI.Tài liệu tham khảo


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

[2] https://machinelearningcoban.com/ bài 7+8

[3] "MỘT PHƯƠNG PHÁP TĂNG TỐC KHẢ NĂNG HỘI TỤ ĐỐI VỚI GRADIENT DESCENT" đăng trên Tạp chí Nghiên cứu KH&CN quân sự, Số 68, 8 – 2020

[4] https://ruder.io/optimizing-gradient-descent/