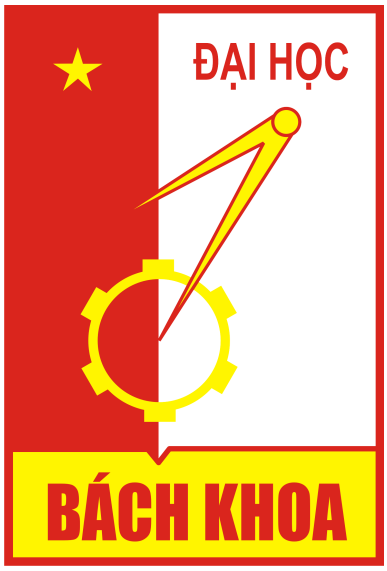
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



BÁO CÁO
GIẢI TÍCH SỐ

ĐỀ TÀI:

TÌM NGHIỆM CỦA $f(x) = 0$
BẰNG PHƯƠNG PHÁP DÂY CUNG

GV hướng dẫn: TS. HÀ THỊ NGỌC YẾN

Nhóm sinh viên thực hiện:

Họ và tên	*MSSV*
Hà Trọng Đạt	**20195851**
Hoàng Mỹ Gia Huy	**20195887**
Trịnh Xuân Đăng	**20195849**
Vũ Quang Minh	**20195903**
Đinh Thị Kim Liên	**20195895**

Hà Nội, Ngày 19 tháng 3 năm 2021

LỜI NÓI ĐẦU

Bài toán *Tìm nghiệm thực gần đúng của phương trình* $f(x) = 0$ *(đại số hay siêu việt)* là một trong những bài toán mà ta hay gặp phải trong quá trình tính toán kỹ thuật.

Sau khi tìm hiểu và nghiên cứu các phương giải bài toán, chúng em xin được trình bày ***Phương pháp dây cung***. Bài báo cáo dưới đây có hai phần: Lý thuyết và Thuật toán cho việc áp dụng phương pháp dây cung để giải bài toán.

Đề tài được thực hiện trong khoảng thời gian hạn chế nên không thể tránh khỏi sai sót nên rất mong nhận được sự đóng góp của cô và các bạn để toàn nhóm có thể hoàn thiện bài báo cáo một cách hoàn chỉnh nhất.

Chúng em xin cảm ơn cô Hà Thị Ngọc Yến đã có những hướng dẫn, góp ý trong quá trình tiến hành bài nghiên cứu này.

Nhóm sinh viên thực hiện

Mục lục

I Lý thuyết

1 Giới thiệu

Trong chương trình Toán học phổ thông ta đã biết: Nếu phương trình có dạng:

$$f(x) = ax + b = 0\,(a \neq 0) \text{ thì } x = \frac{-b}{a}$$

$$f(x) = ax^2 + bx + c = 0\,(a \neq 0) \text{ thì } x_{1,2} = \frac{-b \pm \sqrt{\Delta}}{2a}$$

Đối với đa thức bậc lớn hơn hoặc bằng 3 hay các loại phương trình khác, hầu như không có khả năng tìm được nghiệm đúng mà chỉ có gần đúng.

Hơn thế nữa, trong toán học tính toán, ta phải làm việc với số trong dạng số thập phân. Nên dù tìm được nghiệm đúng trong dạng số thập phân vô hạn ta phải quy tròn số và như vậy cũng vẫn là nghiệm gần đúng. Để đạt được mục tiêu trên thì quá trình tìm nghiệm gần đúng thực của phương trình ta phải tiến hành theo các bước sau:

1. Chọn nghiệm gần đúng đầu tiên gọi là xấp xỉ đầu x_0
2. Từ xấp xỉ đầu x_0, tìm thuật toán sửa dần nghiệm, thu được các xấp xỉ mới gần với nghiệm hơn.
3. Đánh giá sai số của nghiệm gần đúng tìm được với nghiệm đúng của phương trình.

Từ đó có rất nhiều phương pháp, thuật giải ra đời mà ta có thể kể đến như phương pháp *Chia đôi - Bisection Method, Tiếp tuyến - Newton's Method,...* và ngoài ra, ta còn nghiên cứu cả cách để tăng hiệu năng cho các phương pháp.

Sau đây chúng ta sẽ cùng nhau phân tích và nghiên cứu **Phương pháp Dây cung - *False Position Method***.

2 Tổng quan phương pháp

2.1 Nền tảng của phương pháp Dây cung

Yêu cầu bài toán:

Tìm nghiệm của phương trình $f(x) = 0$ *với sai số* ϵ.

Nhắc lại về khoảng phân ly nghiệm: Khoảng (a, b) được gọi là *khoảng phân ly nghiệm* của phương trình $f(x) = 0$ nếu trong khoảng (a, b) có đúng một nghiệm của phương trình.

Phương pháp tìm khoảng phân ly nghiệm

- Phương pháp khảo sát hàm số (lập bảng biến thiên)
- Phương pháp vẽ đồ thị hàm số

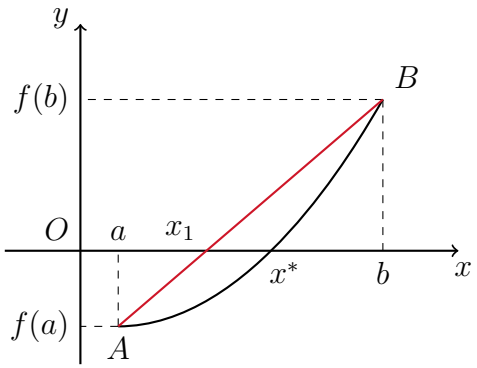
Ý tưởng cho phương pháp Dây cung:

Giả sử (a, b) là khoảng phân ly nghiệm x^* của phương trình thì $f(x)$ là hàm số liên tục và có đạo hàm liên tục trên $[a, b]$. Sau đó ta sẽ khảo sát đồ thị hàm số $y = f(x)$ trên đoạn [a,b]. Ta sẽ tiến hành Khảo sát nhanh đường cong $y = f(x)$ trên đoạn $[a, b]$ và tiến hành các bước như sau:

Bước 1: Trên mặt phẳng tọa độ, ta có 2 điểm $A(a, f(a))$ và $B(b, f(b))$

Bước 2: Vẽ phương trình đường thẳng đi qua 2 điểm A và B

Bước 3: Tìm giao điểm của dây cung với trục hoành. Khi đó điểm này sẽ coi như là **nghiệm gần đúng của phương trình**.



2.2 Xây dựng công thức

- Đầu tiên, ta thiết lập phương trình dây cung AB với $A(a,\ f(a))$ và $(B,\ f(b))$ có dạng

$$y - f(a) = \frac{f(b) - f(a)}{b - a}(x - a)$$

- Cho $y = 0$, ta giải ra được nghiệm x_1:

$$x_1 = a - \frac{b - a}{f(b) - f(a)}.f(a)$$

- Giả sử $x_0 = a,\ d = b$ (B là **điểm Fourier**) ta có:

$$x_1 = x_0 - \frac{d - x_0}{f(d) - f(x_0)}.f(x_0) \text{ với } x_1 \in (a, b)$$

Như vậy, ta có điểm $(x_1, f(x_1))$, lặp lại quá trình trên đối với đoạn có hai đầu mút là x_1 và d.

$$\boxed{x_n = x_{n-1} - \frac{d - x_{n-1}}{f(d) - f(x_{n-1})}.f(x_{n-1})} \quad (1)$$

Công thức (1) trên được gọi là *Công thức lặp theo phương pháp Dây cung*

Chú ý:
Việc xác định ***khoảng phân ly nghiệm (a,b) và điểm Fourier*** là vô cùng quan trọng.

Mặc dù khi thay tùy ý $a,\ b$ làm $x_0,\ d$ ta vẫn có được công thức lặp trên nhưng khi tính toán, kết quả lặp có thể sẽ không hội tụ về x^* (Hình minh họa). Hoặc nếu có hội tụ thì quá trình tính toán bị kéo dài và phải thực hiện nhiều phép tính hơn.

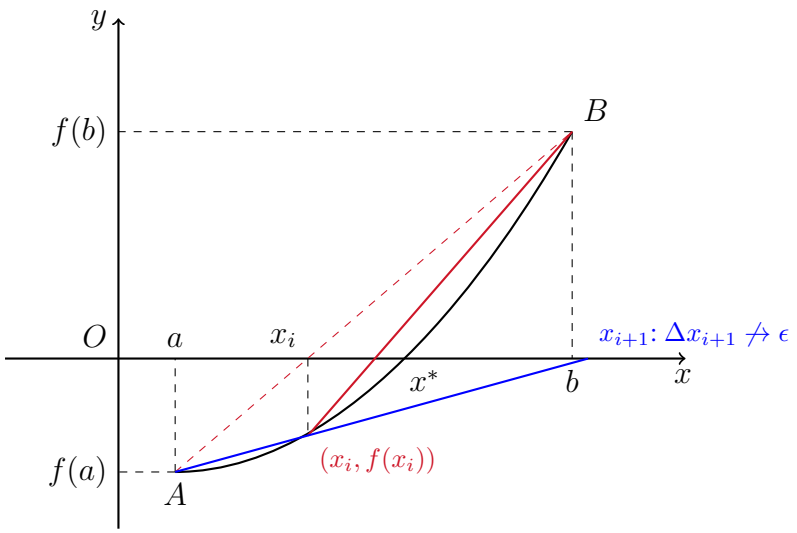
Và cũng có rất nhiều trường hợp, mặc dù đã lựa chọn được khoảng (a, b) và là khoảng phân ly nghiệm và đâu là điểm Fourier nhưng ta không thể thu được dãy các nghiệm xấp xỉ có xu hướng hội tụ về một giá trị nào đó. Ví dụ như khi thực hiện tính toán nghiệm của $f(x) = x^3 = 0$ trong khoảng từ $(-0.5, 1)$ thì ta cũng không thu được kết quả (bảng 1). Nhưng khi thực hiện tiến hành ở khoảng $(-0.5, 0.5)$ thì ta lại "vô tình" thu được kết quả (bảng 2).

i	x_i	d	$f(x_i)$	$f(d)$	δ
1	-0.5000000000	1.0000000000	-0.1250000000	1.0000000000	
2	-0.3333333333	1.0000000000	-0.0370370370	1.0000000000	0.1666666667
3	-0.2857142857	1.0000000000	-0.0233236152	1.0000000000	0.0476190476
4	-0.2564102564	1.0000000000	-0.0168580050	1.0000000000	0.0293040293
5	-0.2355808286	1.0000000000	-0.0130743418	1.0000000000	0.0208294278
6	-0.2196349050	1.0000000000	-0.0105950761	1.0000000000	0.0159459236
7	-0.2068482558	1.0000000000	-0.0088502510	1.0000000000	0.0127866491
8	-0.1962610453	1.0000000000	-0.0075596610	1.0000000000	0.0105872106
9	-0.1872855688	1.0000000000	-0.0065692069	1.0000000000	0.0089754764
10	-0.1795369465	1.0000000000	-0.0057871069	1.0000000000	0.0077486223
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...
78	-0.0737133446	1.0000000000	-0.0004005330	1.0000000000	0.0004377653
79	-0.0732834591	1.0000000000	-0.0003935663	1.0000000000	0.0004298855
80	-0.0728612171	1.0000000000	-0.0003868025	1.0000000000	0.0004222420

Bảng 1: Áp dụng phương pháp Dây cung cho $f(x) = x^3$ và trong khoảng $(-0.5, 1)$

i	x_i	x_{last}	$f(x_i)$	$f(x_{last})$	δ
1	-0.5000000000	0.5000000000	-0.1250000000	0.1250000000	
2	0.00000000000	0.5000000000	0.00000000000	0.1250000000	0.50000000000
3	0.00000000000	0.5000000000	0.00000000000	0.1250000000	0.50000000000

Bảng 2: Áp dụng phương pháp Dây cung cho $f(x) = x^3$ và trong khoảng $(-0.5, 0.5)$



Minh họa: Các giao điểm thu được có xu hướng không hội tụ đến x^*

3 Sự hội tụ của phương pháp dây cung

3.1 Tính lồi lõm của đồ thị hàm

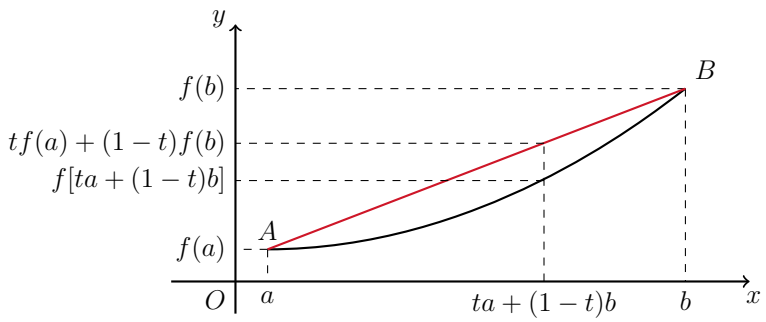
Đầu tiên, chúng ta sẽ nhắc lại kiến thức về hàm lồi, hàm lõm mà ta đã được học trong Giải tích I

- **Hàm lồi:**

  Đồ thị của hàm $y = f(x)$ được gọi là *lồi - Convex Graph* trong khoảng $I \subset \mathbb{R}$ nếu: $a, b \in I, a < b$, cung AB của nó ứng với khoảng $(a, b) \supset I$ luôn nằm dưới dây cung AB.

- **Hàm lõm:**

  Đồ thị của hàm $y = f(x)$ được gọi là *lõm - Convex Graph* trong khoảng $I \subset \mathbb{R}$ nếu: $a, b \in I, a < b$, cung AB của nó ứng với khoảng $(a, b) \supset I$ luôn nằm trên dây cung AB. Hay có thể nói $-f(x)$ là lồi trong I.



Bất đẳng thức hàm lồi: Hàm $f(x)$ là *lồi* trong I *nếu và chỉ nếu* $a, b \forall I, a < b$ thỏa mãn

$$f[ta + (1-t)b] < tf(a) + (1-t)f(b) \ \forall t \in [0, 1]$$

Định lý về tính lồi lõm của hàm: Giả sử hàm f khả vi hai lần trong khoảng I. Khi đó nếu

- $f"(x) \geq 0$ thì $f(x)$ lồi trong I
- $f"(x) \leq 0$ thì $f(x)$ lõm trong I

Như vậy, nếu $f"(x) > 0 \ \forall x \in (a, b)$ thì $f(x)$ là lồi trong khoảng (a, b) hay mọi dây cung nối hai điểm $A(a_i, f(a_i)), B(b_i, f(b_i))$ với $a_i, b_i \in (a, b)$ đều nằm phía trên đồ thị hàm $f(x)$. Vì vậy, ta cần chọn (a, b) sao cho $f"(x)$ mang dấu không đổi để ta có thể áp dụng phương pháp Dây cung với mọi bài toán và thu được kết quả đúng đắn.

Để có thể đáp ứng yêu cầu này, ta áp dụng ***Phương pháp Gradient - Descent*** vào thuật toán. Cụ thể, với $[a, b]$ cho trước, Phương pháp này sẽ cho ta biết được trong khoảng (a, b) thì dấu của $f"(x)$ có thay đổi hay không.

3.2 Điểm Fourier

Xét phương trình $f(x) = 0$ và khoảng cách ly nghiệm là (a, b).
Với 2 điểm $A(a, f(a))$ và $B(b, f(b))$, ta cần xác định một trong hai điểm trên là điểm Fourier.
Trước hết ta cần nêu lại định nghĩa về điểm Fourier:

$$M(d, f(d)) \textbf{ được gọi là điểm Fourier nếu } f(d).f"(d) > 0.$$

Từ đó, ta sẽ chọn điểm x_0 sao cho $f(x_0).f(d) < 0$ và đặt $M_0(x_0, f(x_0))$

3.3 Định lý về sự hội tụ

Giả sử (a, b) là khoảng phân ly nghiệm x^* của phương trình $f(x) = 0$. Nếu:
Hàm số $f(x)$ liên tục và có đạo hàm liên tục đến cấp 2 trên $|a, b|$. $f'(x)$, $f"(x)$ không đổi dấu trên cả đoạn đó. Đồng thời xác định được điểm $M(d, f(d))$ - Fourier và $M_0(x_0, f(x_0))$ thì. Dãy x_n xác định theo công thức lặp sẽ hội tụ đơn điệu tới x^*.

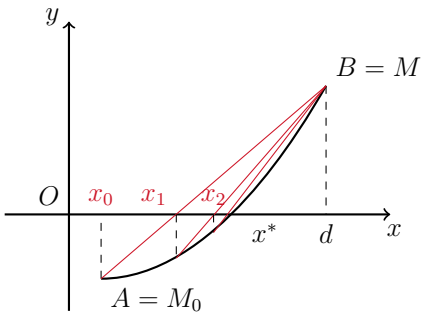
$$\lim_{n\to\infty} x_n = \lim_{n\to\infty} \left[x_{n-1} - \frac{d - x_{n-1}}{f(d) - f(x_{n-1})}.f(x_{n-1}) \right] = x^*$$

Cụ thể: khi $n \to \infty$, nếu:

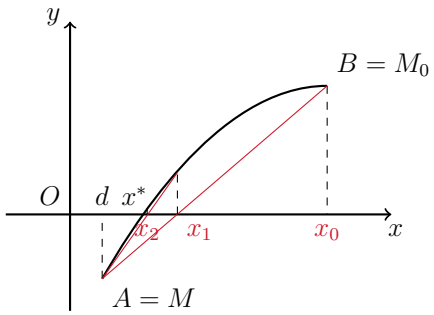
a) $f'(x).f"(x) > 0$ thì dãy x_n đơn điệu tăng đến x^*

b) $f'(x).f"(x) < 0$ thì dãy x_n đơn điệu giảm đến x^*

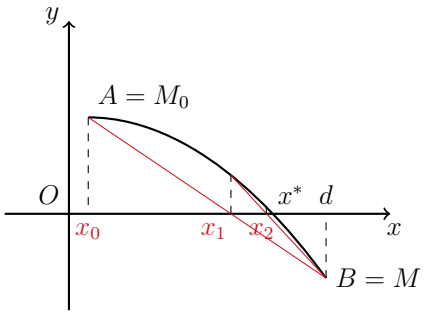
∗ **Minh họa định lý**



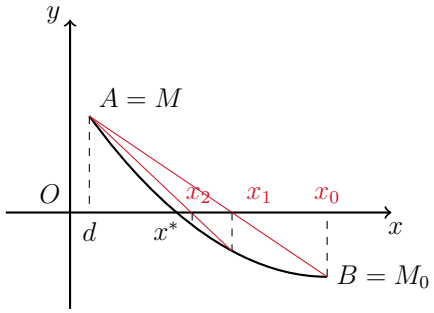
$f' > 0,\ f" > 0$



$f' > 0,\ f" < 0$



$f' < 0,\ f" < 0$



$f' < 0,\ f" > 0$

Để làm rõ định lý, ta sẽ chứng minh $\lim\limits_{n\to\infty} x_n = x^*$

∗ **Chứng minh** (với $f'(x).f"(x) < 0$)

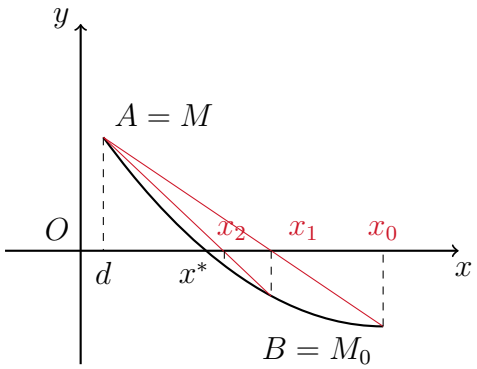
a) $\{x_n\}$ **là một dãy đơn điệu**

Thật vậy, giả sử trên khoảng ly nghiệm (a, b), hàm f có tính chất:

$$f'(x) < 0,\ f"(x) > 0$$

Ta có $A(a, f(a))$ là điểm Fourier $M(d, f(d))$ *(vì* $f(a) > f(x^*) = 0,\ f"(a) > 0$*)* và $B(b, f(b))$ là điểm $M_0(x_0, f(x_0))$

Gọi x_i là nghiệm xấp xỉ thứ i của x^*, như vậy, x_i là hoành độ giao điểm của MM_{i-1} với Ox, trong đó $M(a, f(a))$ và $M_{i-1}(x_{i-1}, f(x_{i-1}))$



Theo công thức (1) ta có:

$$x_i = x_{i-1} - \frac{a - x_{i-1}}{f(a) - f(x_{i-1})}.f(x_{i-1})$$

hay

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f(x_{i-1}) - f(a)}.(x_{i-1} - a)$$

$$\Rightarrow x_i - x_{i-1} = \frac{-f(x_{i-1})}{f(x_{i-1}) - f(a)}.(x_{i-1} - a) \quad (2)$$

Theo **Định lý Lagrange** cho hàm $f(x)$ liên tục trên đoạn $[a, x_{i-1}]$ và khả vi trong khoảng (a, x_{i-1}) sẽ $\exists\, c \in (a, x_{i-1})$ để:

$$f(x_{i-1}) - f(a) = (x_{i-1} - a).f'(c)$$

hay

$$\frac{f(x_{i-1}) - f(a)}{x_{i-1} - a} = f'(c)$$

Áp dụng vào (2) ta thu được

$$x_i - x_{i-1} = \frac{-f(x_{i-1})}{f(x_{i-1}) - f(a)}.(x_{i-1} - a) = -\frac{f(x_{i-1})}{f'(c)}$$

- Có x_i là hoành độ giao điểm của dây cung MM_{i-1} với M nằm phía trên trục hoành nên M_{i-1} phải nằm ở phía dưới trục Ox hay $f(x_{i-1}) < f(x^*) = 0$
- Hơn nữa, c cũng thuộc (a, b) và $f'(x) < 0\, \forall x \in (a, b) \Rightarrow f'(c) < 0$ hay $f'(c)$ cùng dấu với $f(x_{i-1})$

$$\Rightarrow x_i - x_{i-1} = -\frac{f(x_{i-1})}{f'(c)} < 0$$

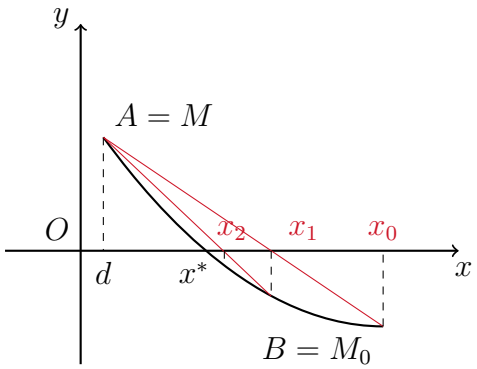
hay $x_i < x_{i-1}$ nên $\{x_n\}$ đơn điệu giảm ■

b) **Chứng minh dãy $\{x_n\}$ bị chặn**

Ta có $M(d, f(d))$ và $M_{i-1}(x_{i-1}, f(x_{i-1}))$ là hai đầu mút của các dây cung thu được. Và $\{x_n\}$ là dãy đơn điệu giảm.

Nhận xét:

- x_1 là giao điểm của dây cung $MM_0 \Rightarrow d < x^* < x_1 < x_0$
- x_2 là giao điểm của dây cung $MM_1 \Rightarrow d < x^* < x_2 < x_1 < x_0$
- ...
- x_n là giao điểm của dây cung MM_{n-1} $\Rightarrow d < x^* < x_n < \ldots < x_1 < x_0$



Ta có thể chứng minh quy nạp để thu được $x_k > x^* \, (\forall x_k \in \{x_n\})$ hay $\{x_n\}$ bị chặn dưới bởi x^* ■

Như vậy ta đã chứng minh được **a)** và **b)** nên ta có thể kết luận: $\lim\limits_{n\to\infty} x_n = x^*$

Hay định lý đã được chứng minh.

Tương tự: với $f'(x).f"(x) > 0$ ***thì*** B ***là điểm Fourier và định lý cũng được chứng minh tương tự và thành công.***

Ví dụ Test thử: *Tìm nghiệm của phương trình* $f(x) = x^3 + 2x^2 - 3x - 1$ *trong khoảng* $(1, 2)$ *với 10 lần lặp.*

Nhận xét:

- $f'(x) = 3x^2 + 4x - 3 > 0 \; \forall x \in (1, 2)$
- $f"(x) = 6x + 4 > 0 \; \forall x \in (1, 2)$ $\Rightarrow f'(x).f"(x) > 0$
- $f(2).f"(2) = 9 \times 16 > 0$ - xác định điểm **Fourier**
  $f(2).f(1) < 0$ - xác định điểm M_0 nên $x_0 = 1, \; d = 2$

Áp dụng **Công thức lặp** 1 với $d = 2$

$$x_n = x_{n-1} - \frac{2 - x_{n-1}}{9 - f(x_{n-1})}.f(x_{n-1})$$

- Bảng kết quả với 10 chữ số sau dấu phẩy

n	$(x_{n-1,\,d})$	x_n
1	(1.0000000000, 2.0000000000)	1.1000000000
2	(1.1000000000, 2.0000000000)	1.1517436381
3	(1.1517436381, 2.0000000000)	1.1768409100
4	(1.1768409100, 2.0000000000)	1.1886276733
5	(1.1886276733, 2.0000000000)	1.1940789113
6	(1.1940789113, 2.0000000000)	1.1965820882
7	(1.1965820882, 2.0000000000)	1.1977277544
8	(1.1977277544, 2.0000000000)	1.1982513178
9	(1.1982513178, 2.0000000000)	1.1984904185
10	(1.1984904185, 2.0000000000)	1.1985995764

4 Đánh giá sai số

Ý nghĩa: Sai số tốt sẽ cho bài toán ra kết quả đúng đắn mà vẫn dảm bảo cho chương trình thực thi hiệu quả.

4.1 Công thức sai số 1 - *Công thức mục tiêu*

Giả sử (a, b) là khoảng phân ly nghiệm x^* của phương trình $f(x), x_i \in (a, b)$ là nghiệm gần đúng tìm được. Nếu $|f'(x)| \geq m > 0 \ \ \forall x \in [a, b]$, $m = \text{const}$ (hay $m = \min\limits_{x\in[a,b]} |f'(x)|$) thì

$$\boxed{|x_i - x^*| \leq \frac{|f(x_i)|}{m}} \tag{3}$$

Hơn nữa, công thức 3 có thể áp dụng cho cả phương pháp *Newton* và phương pháp *Dây cung*.

∗**Chứng minh:**

- Xét $f(x_i) = f(x_i) - f(x^*)$
  Áp dụng **Định lý Lagrange** ta có
  $$f(x_i) = f(x_i) - f(x^*) = f'(c)(x_i - x^*) \text{ với } c \text{ nằm giữa } x^* \text{ và } x_i$$
- Như vậy, $c \in (a, b) \Rightarrow |f'(c)| \geq m > 0$
  Và
  $$|f(x_i)| = |f'(c)||x_i - x^*| \geq m|x_i - x^*|$$
  Hay
  $$|x_i - x^*| \leq \frac{|f(x_i)|}{m}$$

■

4.2 Công thức sai số 2 - *Đánh giá sai số theo hai bước lặp liên tiếp*

Với (a, b) là khoảng phân ly nghiệm x^* của $f(x)$
Đặt $M = \max\limits_{x\in[a,b]} |f'(x)|$, $m = \min\limits_{x\in[a,b]} |f'(x)|$, ta có công thức

$$\boxed{|x_{i+1} - x^*| \leq \frac{M - m}{m}|x_{i+1} - x_i|} \tag{4}$$

∗**Chứng minh:**

- Đặt $M = \max\limits_{x\in[a,b]} |f'(x)|$, $m = \min\limits_{x\in[a,b]} |f'(x)|$
- Từ **Công thức lặp** (1)
  $$x_i = x_{i-1} - \frac{d - x_{i-1}}{f(d) - f(x_{i-1})}.f(x_{i-1})$$
  và $f(x^*) = 0$ nên
  $$[f(d) - f(x_{i-1})](x_i - x_{i-1}) = [f(x^*) - f(x_{i-1})](d - x_{i-1})$$

- Áp dụng **Định lý Lagrange** vào hai ngoặc vuông:

$$f'(c_1)(d - x_{i-1})(x_i - x_{i-1}) = f'(c_2)(x^* - x_{i-1})(d - x_{i-1})$$

với $c_1,\ c_2$ là các trị trung gian đều thuộc (a, b) Từ đó suy ra:

$$\begin{aligned} & f'(c_1)(x_i - x_{i-1}) = f'(c_2)(x^* - x_{i-1}) \\ \Rightarrow \quad & f'(c_1)(x_i - x_{i-1}) = f'(c_2)[(x^* - x_i) + (x_i - x_{i-1})] \\ \Rightarrow \quad & x^* - x_i = \frac{f'(c_1) - f'(c_2)}{f'(c_1)}(x_i - x_{i-1}) \end{aligned}$$

- Do $f'(x)$ không đổi dấu nên $f'(c_1) - f'(c_2) \leq M - m$

$$\Rightarrow \quad |x^* - x_i| \leq \frac{M - m}{m}|x_i - x_{i-1}| \qquad \blacksquare$$

Nhận xét:

- Áp dụng với phương pháp Dây cung
- Cho thấy cấp độ hội tụ phương pháp là bậc 1 (tuyến tính)
  $\Rightarrow$ Tốc độ hội tụ nghiệm còn chậm
  VD: Phương pháp tiếp tuyến có cấp độ hội tụ bậc 2, . . .
- **Đặc biệt khi** $M \leq 2m$ **thì** $|x^* - x_i| \leq |x_i - x_{i-1}|$

II Cài đặt thuật toán

1 Hoàn thiện thuật toán

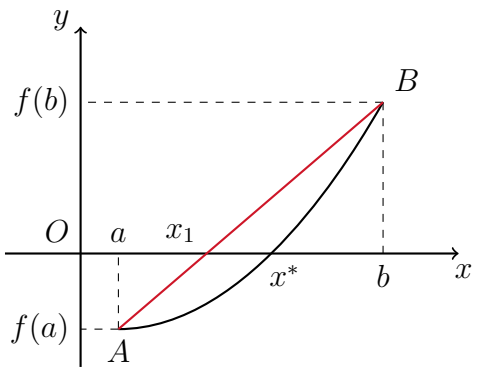
Sau khi đã phân tích, ta có được một thuật toán hoàn chỉnh như sau:

Bước 1: **Tìm khoảng phân ly nghiệm** (a, b) **thỏa mãn** $f(a).f(b) < 0$

Đồng thời dùng **Gradient - Descent** để kiểm tra xem trên (a, b) : $f'(x)$ và $f"(x)$ có giữ nguyên dấu không.

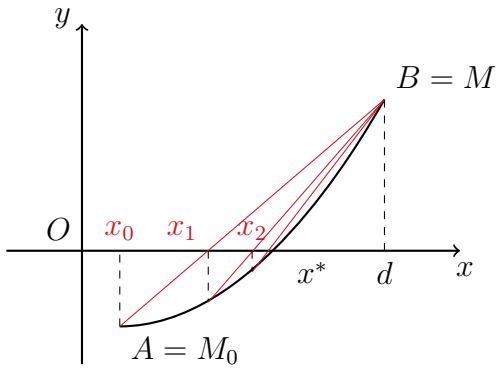
- Nếu không thì phải khảo sát lại để tìm (a, b). Thuật toán kết thúc
- Nếu có thì tìm $m = \min\limits_{x\in[a,b]} |f'(x)|$
  để áp dụng công thức sai số

Ta có thể xác định nhanh chóng được
$m = \min(|f'(a)|, |f'(b)|)$
(Do $f"(x)$ là giữ nguyên dấu trên (a, b), $f'(x) \neq 0$ trên (a, b))



Bước 2: **Xác định hoành độ giao điểm của dây cung AB với trục hoành là x_1.**

$$x_1 = \frac{af(b) - bf(a)}{f(b) - f(a)}$$



Bước 3: **Tính $f(x_1)$ và so sánh $f(x_1).f(a) > 0$**

- Nếu Đúng thì $x_0 = a,\ d = b.$
  (x_1, b) là khoảng phân ly nghiệm mới
- Nếu Sai thì $x_0 = b,\ d = a.$
  (a, x_1) là khoảng phân ly nghiệm mới

Bước 4: **Tính toán các nghiệm xấp xỉ**

- Áp dụng **Công thức lặp** (1)

$$x_n = x_{n-1} - \frac{d - x_{n-1}}{f(d) - f(x_{n-1})}.f(x_{n-1})$$

- Và **Công thức sai số 1** (3)

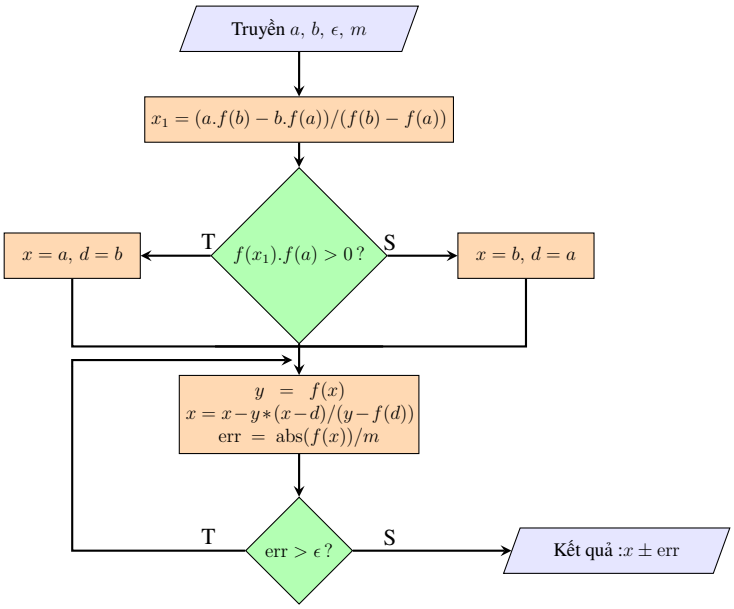
$$\text{err} = \frac{|f(x_n)|}{m}$$

Bước 5: **So sánh** $\text{err} > \epsilon$?

- Nếu Sai thì x_n là nghiệm cần tìm với sai số $\text{err} \leq \epsilon$
- Nếu Đúng thì quay về **Bước 4**

2 Sơ đồ khối

Áp dụng khi đã tìm được khoảng ly nghiệm (a, b) - ***Gradient Descent***



3 Trình bày code (ngôn ngữ C++)

Để quá trình thực hiện chương trình thực hiện dễ dàng và thuận tiện trong quá trình nghiên cứu, việc nhập **INPUT** và **OUTPUT** sẽ được thực hiện thông qua file text (.txt).

3.1 Bước khai báo đầu chương trình

Khi chạy chương trình, ta cần những thông tin sau:

```
#include <iostream>
#include <math.h>
#include <iomanip>
#define LD long double
using namespace std;
const LD h = 1e-7;
LD eta = 0.001;
const int MAX_LOOP = 1000000;
LD a, b, x, x_old, fa, fb, fx, eps, m1, M1;
```

Trong đó :

- `#include<iostream>` - thư viện giúp kiểm soát luồng vào - ra dữ liệu.
- `#include<math.h>` - giúp ta khai thác hàm toán cần thiết trong quá trình thực hiện.
- `#include<iomanip>` - thư viện con của `std` giúp ta định dạng được output (về nhu cầu xác định độ chính xác của kết quả).

Về phần các biến toàn cục cần thiết: Trong chương trình, ta sẽ sử dụng nhiều giá trị mang kiểu `long double` nên ta sẽ tạo một Macro: `#define LD long double` cho thuận tiện việc sử dụng.

- `LD a,b, eps`: để lưu giá trị của hai đầu khoảng phân ly nghiệm, ϵ đầu vào.
- `LD fa, fb, fx`: để lưu, tính giá trị của $f(a),\ f(b),\ f(x_n)$
- `LD m1, M1`: lưu giá trị min max cảu $f'(x)$ trong (a, b).
- `LD x, x_old`: dùng để lưu hai giá trị x_n thu được trong hai lần liên tiếp (phục vụ cho quá trình tính toán và áp dụng trong công thức sai số)
- Một số biến cấp thêm như `const LD h = 1e-7` được phục vụ cho quá trình tính đạo hàm cấp 1 và cấp 2, `const int MAX_LOOP = 1000000` được sử dụng để tránh quá trình lặp bị kéo dài vô hạn (khi ta cho input với số epsilon quá bé với khoảng 10^{-20} hay những điều kiện khác nữa dễ gây lặp vô hạn)

3.2 Các hàm thực hiện trong chương trình

Dựa theo thuật toán ở phần **Hoàn thiện thuật toán**, tổng quát lại ta cần các hàm :

- Hàm *Gradient - Descent*: thực hiện chức năng đánh giá trên khoảng phân ly nghiệm đã cho xem liệu đạo hàm cấp 1 và cấp 2 trên khoảng đó có đổi dấu hay không. Hàm đóng vai trò trong việc xác định tính hợp lệ của đầu vào.

- Các hàm tính giá trị hàm $f(x)$, $f'(x)$, $f"(x)$ để thực hiện tính toán và khảo sát trong Gradient - Descent.
- Hàm thực hiện *Phương pháp Dây cung* như trong sơ đồ khối

3.3 Code

Trong phần code dưới đây, chương trình đang tính nghiệm hàm $f(x) = \sin(x) + \cos(x) + 1$.

```
#include <iostream>
#include <math.h>
#include <iomanip>
#define LD long double
using namespace std;
const LD h = 1e-7;
LD eta = 0.001;
const int MAX_LOOP = 1000000;
LD a, b, x, x_old, fa, fb, fx, eps, m1, M1;
LD f(LD x)                          //f(x)=0
{
  return sin(x)+cos(x)+1;
}
LD f1(LD x)                         //f`(x)
{
  return (f(x + h) - f(x - h)) / (2 * h);
}
LD f2(LD x)                         //f``(x)
{
  return (f1(x + h) - f1(x - h)) / (2 * h);
}
LD random_num(LD a, LD b)
{
  return a + (LD)rand() / RAND_MAX * (b - a);
}
int re_sign(LD (*g)(LD x))              //Gradient-Decent
{
  int i;
  for (i = 1; i < 100; i++)           //Kiem tra ham dang xet co la ham hang
   khong
    if (g(random_num(a, b)))
      break;
  if (i == 100)
    return 0;
  LD gx, sign;
  x = a;
  gx = g(x);
  if (gx == 0)
  {
    x += eps;
    gx = g(x);
  }
  sign = (gx > 0) ? 1 : -1;
  int cnt = 0;
  while (x < b && ++cnt <= MAX_LOOP)        //Gradient-Decent rat hiem khi co
   truong hop g(x)=0
```

```
45    {                          //Tuy nhien de de phong lap vo han toi van gioi han so lan lap
46      x += eta * abs(gx);
47      if (x > b)
48        x = b;
49      gx = g(x);
50      if (gx==0) {x+=eps;continue;}       //Van nhat g(x) = 0 that thi toi van muon ham cho KQ dung
51      if (sign * gx < 0)
52        return 1;
53    }
54    return 0;
55  }
56  bool stop1()
57  {
58    return abs(fx) / m1 < eps;
59  }
60  bool stop2()
61  {
62    return (M1-m1) / m1 * abs(x-x_old) < eps;
63  }
64  void The_Secant_Method()              //PP_day_cung
65  {
66    x = a - fa * (a - b) / (fa - fb);
67    fx = f(x);
68    if (f(x) * fa < 0)                  //Xac dinh diem Fourier
69    {                         //Neu a la diem Fourier, gan b = a
70      b = a;                      //Nhu vay diem b luon la diem Fourier
71      fb = fa;
72    }
73    cout << "f(" << x << ") = " << fx << '\n';
74    int cnt = 0;
75    while (!stop1() && ++cnt <= MAX_LOOP)       //stop1() la dieu kien dung f(x)/m1 < eps
76    {                         //stop2() la dieu kien dung (M1-m1)/m1 * abs(x[n]-x[n-1]) < eps
77      x_old = x;
78      x = x - fx * (x - b) / (fx - fb);
79      fx = f(x);
80      cout << "f(" << x << ") = " << fx << '\n';
81    }
82  }
83  void Input()
84  {
85
86    freopen("INP.txt", "r", stdin);freopen("OUT.txt", "w", stdout);ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0); //dat dong nay vao comment neu muon nhap tu console
87
88    cin >> a >> b >> eps;
89    if (a > b)
90      swap(a, b);
91    cout << fixed << setprecision(-log10(eps) + 1);
92  }
93  int Input_Invalid() //kiem tra du lieu dau vao
94  {
```

```
95    fa = f(a);
96    fb = f(b);
97    if (fa * fb > 0)
98    {
99      cout << "khoang li nghiem ko hop le";
100     return 1;
101   }
102   if (re_sign(f1))
103   {
104     cout << "ham khong don dieu";
105     return 1;
106   }
107   if (re_sign(f2))
108   {
109     cout << "dao ham khong don dieu";
110     return 1;
111   }
112
113   m1 = abs(f1(a));
114   M1 = abs(f1(b));
115   if (m1 > M1)
116     swap(m1, M1);
117   return 0;
118 }
119 int main()
120 {
121   Input();
122
123   srand(time(NULL));              //cau lenh nay chi dung cho ham random khong
       lien quan thuat toan chinh
124
125   if (Input_Invalid())
126     return 0;
127
128   The_Secant_Method();
129
130   return 0;
131 }
```

4 Chương trình chạy thử

4.1 Tìm nghiệm của phương trình $\sin(x)$ để xác định số π

Khi giải phương trình trên trên giấy, giải bình thường ta dễ dàng thu được kết quả là $x = k\pi$ với $k \in \mathbb{Z}$. Với $k = 1$, ta thu được giá trị $x = \pi \approx 3.14159\ldots$, vì vậy ta tạm thử khoảng $(3, 4)$ là khoảng phân ly nghiệm.

Thực hiện áp dụng chương trình giải $f(x) = \sin(x) = 0$

- Thay hàm $f(x)$ trong hàm `LD f(LD x)` để `return sin(x);`
- Mở INP.txt, nhập giá trị lần lượt là `3 4 1e-10` cách nhau bởi dấu cách " ".
- Thực hiện chạy code, xem ở file OUTPUT.txt

Input đầu vào file INP.txt:
```
3 4 1e-10
```

Kết quả trong file OUTPUT.txt:
```
dao ham khong don dieu
```

Từ kết quả trên, ta thấy được khoảng $(3, 4)$ không phải là khoảng phân ly nghiệm lý tưởng, ta cần phải chỉnh sửa lại khoảng này và nhìn nhận lại đặc điểm của hàm.
Thử lại lần nữa với các khoảng nhỏ hơn $(3, 3.2)\ldots$

Input đầu vào file INP.txt:
```
3 3.2 1e-10
```

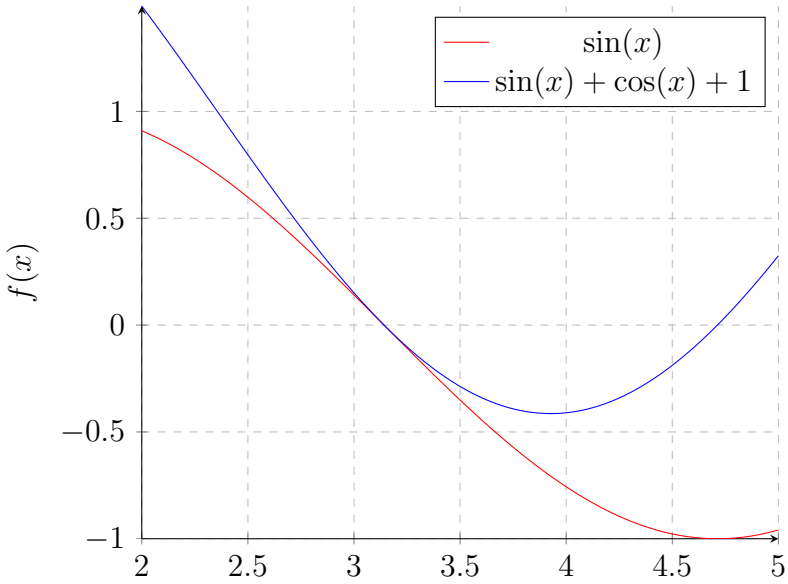
Kết quả trong file OUTPUT.txt:
```
dao ham khong don dieu
```

Input đầu vào file INP.txt:
```
3.1 3.16 1e-10
```

Kết quả trong file OUTPUT.txt:
```
dao ham khong don dieu
```

Như vậy, dù cho thu nhỏ khoảng phân ly nghiệm, ta cũng không thể thu được nghiệm $\pi = 3.14159\ldots$ như ta mong muốn. Kết quả trả về luôn là "***Đạo hàm không đơn điệu***" nên ta cần nhìn nhận lại chính hàm $f(x) = \sin(x)$ và nhận thấy có điều bất thường. Đó chính là tại $x = \pi$ ta có được $f"(\pi) = 0$ hay $x = \pi$ là hoành độ của điểm uốn đồ thị hàm số. Chính vì vậy nên bài toán trên tưởng chừng đơn giản nhưng lại không thể áp dụng phương pháp Dây cung thực hiện được vì đã vi phạm điều kiện về đầu vào của bài toán.

Để xác định được số π thông qua giải nghiệm phương trình $f(x) = 0$, ta cần đưa ra một phương trình hàm khác. Một hàm tương tự với $f(\pi) = 0$ nhưng có tránh được lỗi vi phạm như hàm $f(x) = \sin(x)$.



Đồ thị 1: $\sin(x)$ *và* $\sin(x) + \cos(x) + 1$ *trên khoảng đoạn* $[2, 5]$

4.2 Xác định số π với hàm $f(x) = \sin(x) + \cos(x) + 1$

Ta lựa chọn ví dụ hàm $f(x) = \sin(x) + \cos(x) + 1$ vì nó thỏa mãn những điều kiện sau:

- $f(\pi) = \sin(\pi) + \cos(\pi) + 1 = 0 + (-1) + 1 = 0$
- $f"(\pi) = -\sin(\pi) - \cos(\pi) = 1 \neq 0$

Trước hết ta cũng làm tương tự như trước nhưng là với hàm $f(x) = \sin(x) + \cos(x) + 1$

- Thay hàm $f(x)$ trong hàm `LD f(LD x)` để `return sin(x)+cos(x)+1;`
- Mở INP.txt, nhập giá trị lần lượt là `3 4 1e-10` cách nhau bởi dấu cách " ".
- Thực hiện chạy code, xem ở file OUTPUT.txt

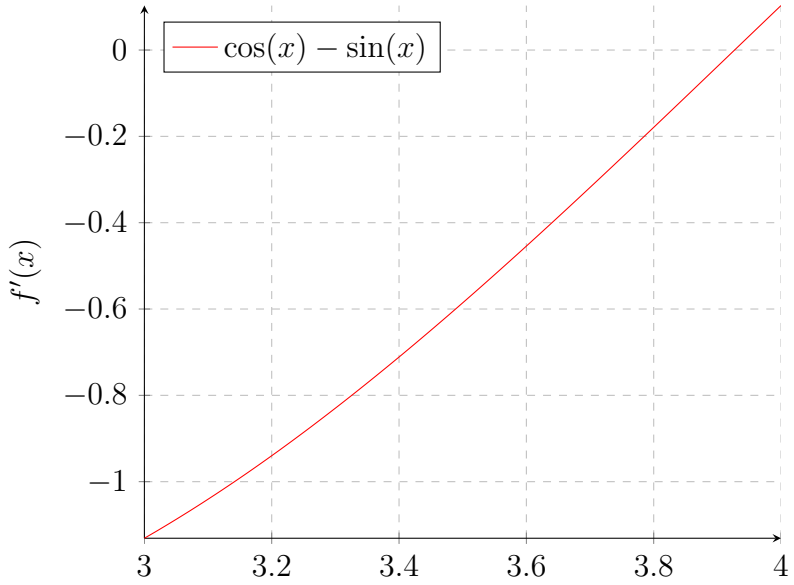
Input đầu vào file INP.txt:

```
3 4 1e-10
```

Kết quả trong file OUTPUT.txt:

```
ham khong don dieu
```

Như vậy, trên khoảng $(3, 4)$ thì ta thu được kết quả trên. Lý do nằm ở $f'(x) = \cos(x) - \sin(x)$ có dấu thay đổi trên khoảng này. Thế nên, ta cần phải thu nhỏ tiếp khoảng ly nghiệm



Đồ thị 2: *Đạo hàm của $f(x)$ trên $(3, 4)$*

Dựa vào **đồ thị 1**, ta thử lựa chọn khoảng phân ly nghiệm là $(3, 3.5)$

Input đầu vào file INP.txt:

```
3 3.5 1e-10
```

Kết quả trong file OUTPUT.txt:

```
f(3.17237538916) = -0.03030412331
f(3.14358391046) = -0.00198927300
f(3.14171848729) = -0.00012582579
f(3.14160059343) = -0.00000793981
f(3.14159315453) = -0.00000050094
f(3.14159268519) = -0.00000003161
f(3.14159265558) = -0.00000000199
f(3.14159265372) = -0.00000000013
f(3.14159265360) = -0.00000000001
```

Như vậy, sau 9 lần lặp, ta đã xác định nghiệm của $f(x)$ và xác định được số $\pi \approx 3.14159265360$ với độ chính xác 11 số sau dấu phẩy. Hơn nữa, ta hiện đang sử dụng công thức sai số thứ nhất - *Công thức mục tiêu* để xác định điều kiện dừng bài toán. Thử với Công thức sai số thứ 2 - *Công thức đánh giá sai số theo hai lần lặp liên tiếp*

Ta thực hiện các bước tương tự như vừa rồi, chỉ khác ở chỗ dòng code thứ 75, ta không lựa chọn trong vòng `while` là `!stop1()` nữa mà sử dụng `!stop2()`

Input đầu vào file INP.txt:

```
3 3.5 1e-10
```

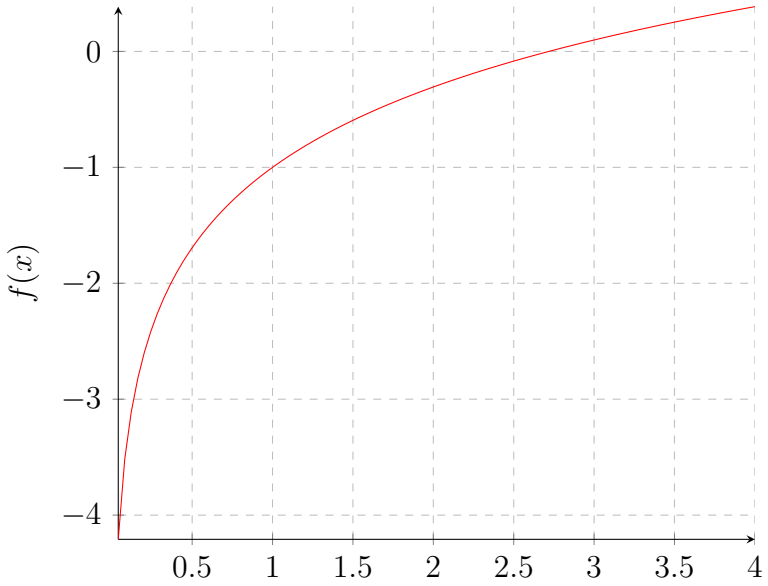
Kết quả trong file OUTPUT.txt:

```
f(3.17237538916) = -0.03030412331
f(3.14358391046) = -0.00198927300
f(3.14171848729) = -0.00012582579
f(3.14160059343) = -0.00000793981
f(3.14159315453) = -0.00000050094
f(3.14159268519) = -0.00000003161
f(3.14159265558) = -0.00000000199
f(3.14159265372) = -0.00000000013
f(3.14159265360) = -0.00000000001
f(3.14159265359) = -0.00000000000
```

Mặc dù chưa có sự khác biệt quá rõ ràng giữa việc sử dụng 2 công thức khi mà thực hiện lần này, ta mất nhiều hơn 1 lần lặp và chính xác hơn kết quả trước 1×10^{-11} cùng với đó là kết quả xấp xỉ $f(x) \approx 0$.

4.3 Xác định số e thông qua giải nghiệm hàm

Để xác định hàm số e, ta lựa chọn hàm $f(x) = \ln(x) - 1$.



Đồ thị 3:*Đồ thị hàm* $\ln(x) - 1$ *trong* $(0, 4)$

Lần này, ta thay giá trị hàm trong hàm `LD f(LD x)` là `return log(x)-1;` và thử tìm nghiệm trên khoảng $(2, 4)$, sai số ϵ vẫn là 10^{-11} và công thức sai số 1.

Input đầu vào file INP.txt:

```
2 4 1e-10
```

Kết quả trong file OUTPUT.txt:

```
f(2.88539008178) = 0.05966010114
f(2.74126839094) = 0.00842072969
f(2.72146964549) = 0.00117204523
f(2.71872442860) = 0.00016281024
f(2.71834328939) = 0.00002260996
f(2.71829036332) = 0.00000313979
f(2.71828301367) = 0.00000043601
f(2.71828199305) = 0.00000006055
f(2.71828185131) = 0.00000000841
f(2.71828183163) = 0.00000000117
f(2.71828182890) = 0.00000000016
f(2.71828182852) = 0.00000000002
```

Và đây là khi áp dụng Công thức sai số 2

Input đầu vào file INP.txt:

```
2 4 1e-10
```

Kết quả trong file OUTPUT.txt:

```
f(2.88539008178) = 0.05966010114
f(2.74126839094) = 0.00842072969
f(2.72146964549) = 0.00117204523
f(2.71872442860) = 0.00016281024
f(2.71834328939) = 0.00002260996
f(2.71829036332) = 0.00000313979
f(2.71828301367) = 0.00000043601
f(2.71828199305) = 0.00000006055
f(2.71828185131) = 0.00000000841
f(2.71828183163) = 0.00000000117
f(2.71828182890) = 0.00000000016
f(2.71828182852) = 0.00000000002
f(2.71828182847) = 0.00000000000
```

Có thể thấy, với những ví dụ đã thực hiện, việc tìm nghiệm đã thành công với hầu hết các hàm, ngoại trừ các trường hợp đặc biệt như $f(x) = \sin(x)$ ta không thể xác định được nghiệm $x = \pi$ do không thỏa mãn điều kiện đầu vào của phương pháp. Hơn nữa, khi áp dụng công thức số 2 - *Công thức đánh giá sai số giữa hai lần lặp* - mặc dù phải lặp nhiều hơn nhưng cho kết quả có độ chính xác cao hơn so với công thức 1 - *Công thức mục tiêu*.

Tài liệu

[1] Lê Trọng Vinh. *Giáo trình Giải tích số* (Dùng cho sinh viên ngành Toán tin ứng dụng và các ngành kỹ thuật). NXB Khoa học và Kỹ thuật. 2000.

[2] Gradient - Descent, Wikipedia: https://en.wikipedia.org/wiki/Gradient_descent