TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

………….🙦🕮🙤…………



BÁO CÁO MÔN

GIẢI TÍCH SỐ

**Đề tài:**

**Giải phương trình phi tuyến f(x) = 0 bằng phương pháp chia đôi**

Giảng viên: TS. Hà Thị Ngọc Yến

Nhóm thực hiện: 5

|  |  |
| --- | --- |
| Nguyễn Công Hiếu | 20195016 |
| Nguyễn Văn Triển | 20195934 |
| Mai Xuân Tuấn | 20195938 |
| Phạm Văn Thành | 20195918 |
| Trịnh Văn Hưng | 20195883 |
| Nguyễn Hoàng Lương | 20195899 |

MỤC LỤC:

1. Nghiệm và khoảng cách ly nghiệm:……………………………………………3
   1. Nghiệm của phương trình:………………………………………………….3
   2. Khoảng cách ly nghiệm (Hay khoảng phân ly nghiệm):…………………...3
2. Phương pháp chia đôi:…………………………………………………………5

2.1.Các bước thực hiện phương pháp chia đôi:………………………………..5

2.2.Xây dựng thuật toán:……………………………………………………....6

2.3.Đánh giá sai số:……………………………………………………………11

2.4.Số phép tính cần thực hiện:………………………………………………..12

III. Chương trình (Code) kiểm tra phương pháp chia đôi:………………………..12

3.1. Hậu nghiệm tối ưu:………………………………………………………..12

3.2.Tiên nghiệm:……………………………………………………………….15

3.3.Hậu nghiệm:……………………………………………………………….17

IV. Ưu, nhược điểm của phương pháp chia đôi:………………………………….20

**THÀNH VIÊN NHÓM BÁO CÁO:**

1. **Nguyễn Văn Triển**(MSSV 20195934):

* Nhóm trưởng;
* Thiết kế và chạy chương trình (code);
* Thiết kế và chạy slide;
* Đóng góp ý tưởng;
* Hỗ trợ thuyết trình;

1. **Phạm Văn Thành** (MSSV 20195918):

* Thuyết trình bài giảng;
* Thiết kế và chạy chương trình (code);
* Thiết kế và chạy slide;
* Đóng góp ý tưởng bài giảng;

1. **Nguyễn Công Hiếu** (MSSV 21095016):

* Thiết kế và chạy slide;
* Đóng góp ý tưởng bài giảng;
* Hỗ trợ thuyết trình;

1. **Mai Xuân Tuấn** (MSSV 20195988);

* Đóng góp ý tưởng;
* Tìm kiếm tư liệu bài giảng;

1. **Trịnh Văn Hưng** (MSSV 20195883):

* Đóng góp ý tưởng;
* Hỗ trợ thuyết trình;

1. **Nguyễn Hoàng Lương** (MSSV 20195899);

* Trình bày và hoàn thành báo cáo;

***Đặt vấn đề***: Xét bài toán sau: Giải phương trình phi tuyến f(x) = 0, với f(x) liên tục trên khoảng (a,b). Trong thực tế, việc tính đúng được nghiệm của phương trình là rất khó khăn. Thậm chí có thể không tính được nghiệm chính xác bằng các phép toán thông thường. Khi đó, ta chỉ có thể tính xấp xỉ được nghiệm với một độ chính xác cho trước. Có rất nhiều phương pháp cho phép ta làm điều này. Ở đây chúng ta sẽ tập trung vào phương pháp chia đôi (Bisection Method).

1. **Nghiệm và khoảng cách ly nghiệm:**
   1. ***Nghiệm của phương trình:***

Định nghĩa: Nghiệm α của 1 phương trình f(x) = 0 là giá trị thỏa mãn f(α) = 0.

VD: x=1, x=2 là nghiệm của phương trình x2 - 3x + 2 =0.

Áp dụng vào trong đồ thị hàm số, ta thấy rằng giá trị α cũng chính là giao điểm của trục hoành với đồ thị C của hàm số f(x).

* 1. ***Khoảng cách ly nghiệm (hay khoảng phân ly nghiệm):***

Khái niệm: Giả sử phương trình phi tuyến f(x) = 0 có nghiệm. Khoảng cách ly nghiệm của phương trình là khoảng (a;b) chứa đúng một nghiệm của phương trình.

VD: phương trình ln(x) = 1 có 1 nghiệm là x = *e* ≈ 2.7182. Như vậy khoảng cách ly nghiệm của phương trình là (2;3).

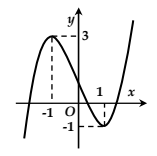
Ta có định lý sau: Nếu f(x) liên tục, đơn điệu trên (a;b) và f(a).f(b) < 0 thì (a;b) là khoảng cách ly nghiệm của phương trình f(x) = 0. Nói cách khác, nếu f(x) liên tục trên [a;b], f’(x) không đổi dấu trên (a;b) và f(a).f(b) < 0 thì (a,b) là khoảng cách lý nghiệm của phương trình.

Như vậy để nhận biết khoảng cách ly nghiệm của phương trình ta có 2 cách: khảo sát hàm số và vẽ đồ thị.

VD: Tìm một khoảng cách ly nghiệm của phương trình: x3 - 3x + 1 = 0

Giải: Xét f(x) = x3 – 3x + 1 => f’(x) = 3x2 – 3. Từ đó ta có bảng khảo sát hàm:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x |  | -1 |  | 1 |  |
| F(x) | + | 0 | - | 0 | + |



Xét khoảng (-2;-1) có f’(x) > 0 với mọi x (-2;-1) và f(-2).f(-1) < 0

Vậy (-2;-1) là khoảng cách ly nghiệm của phương trình trên.a

Sử dụng đồ thị ta cũng có được (-2;-1) là khoảng cách ly nghiệm của phương trình.

**Chú ý**: Chiều ngược lại không đúng. Phương trình x3 – x2  - x +1 = 0 có khoảng cách ly nghiệm là (0;2) (theo định nghĩa) nhưng f(0).f(2) > 0.

Dễ thấy các phương trình trên có thể giải ra nghiệm bằng các công thức toán học đã biết. Tuy nhiên trong thực tế, không phải phương trình nào cũng giải được như vậy. Ngoài ra, việc áp dụng vào trong đời sống, kỹ thuật,… đòi hỏi các số liệu chính xác, do đó những giá trị như , *e*, ,… cần phải quy ra số cụ thể (như 3.1412,…). Để có thể thuận tiện cho việc tính xấp xỉ được nghiệm của phương trình, chúng ta có thể áp dụng phương pháp chia đôi.

1. **Phương pháp chia đôi (Bisection method):**

- Nguyên tắc của phương pháp này là thu hẹp dần khoảng cách ly của phương trình đến một khoảng nhất định, từ đó áp dụng công thức sai số tính xấp xỉ ra giá trị nghiệm.

- Điều kiện để thực hiện phương pháp này bao gồm:

+ (a;b) là khoảng cách ly nghiệm.

+ f(x) liên tục.

+ f(x) trái dấu tại 2 đầu mút.

***2.1: Các bước thực hiện phương pháp chia đôi:***

Bài toán: Cho phương trình f(x) = 0 có khoảng cách ly nghiệm ban đầu là (a,b). Tìm giá trị xấp xỉ của nghiệm trong khoảng cách ly đó.

* Tính c =
* Đặt z = f(c)
* Kiểm tra điều kiện thuật toán:

+ Nếu z = 0 => x =c là một nghiệm của phương trình. Kết thúc thuật toán.

+ Nếu z.f(a) < 0 => x (a;c). Đặt b=c và quay lại bước đầu.

+ Nếu z.f(a) > 0 => x (c;b). Đặt a=c và quay lại bước đầu

Tiếp tục chia khoảng như trên cho đến khi ta có nghiệm xấp xỉ x0 thỏa mãn

|x - x0| < với là độ chính xác cho trước.

## *2.2: Xây dựng thuật toán:*

INPUT: Phương trình f(x), khoảng cách ly nghiệm (a,b) và độ chính xác .

OUTPUT: Nghiệm xấp xỉ x của phương trình.

THUẬT TOÁN:

B1: Đặt a0 = a, b0 = b

B2: Tính c = và đặt x0 = c.

/\* Kiểm tra điều kiện của z\*/

B3: Nếu z = 0 đến bước 7.

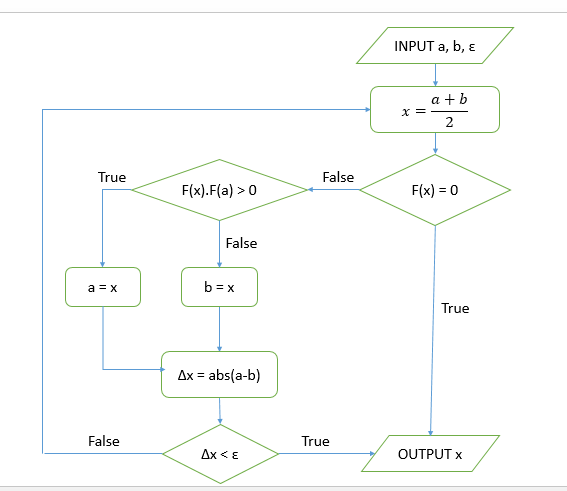
B4: Nếu z.f(a) < 0, gán b = c. Đến bước 6

B5: Nếu z.f(a) > 0, gán a = c. Đến bước 6

B6: Nếu | b-a | > thì quay về bước 1.

B7: Nghiệm xấp xỉ cần tìm là x = c.

Thuật toán có thể được biểu diễn đơn giản thông qua sơ đồ sau:



VD: Áp dụng phương pháp chia đôi tính nghiệm xấp xỉ của phương trình

x3+ 3x2 - 3 = 0 với độ chính xác = 10-3 , biết khoảng cách ly nghiệm là (-3,-2)

Giải:

Ta có: f(x) = x3 + 3x2 - 3. f’(x) = 3x2 + 6x

Vậy f’(x) = 0 ⬄ x = 0 hoặc x = 2

Bảng biến thiên:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x |  | -2 |  | 0 |  |
| f’(x) | + | 0 | - | 0 | + |

Ta có f’(x) > 0 với mọi x (-3,-2). Đồng thời f(-3).f(-2) = -3.1 = -3 <0

=> (-3,-2) là khoảng cách ly nghiệm của phương trình.

Áp dụng phương pháp chia đôi ta có:

+) c1 = = = -2,5

=> f(c1).f(a) = f(-2,5).f(-3) = 0,125.(-3) < 0

=> b = c1. Khoảng cách ly nghiệm là: (-3;-2,5)

Tính | (-2,5) – (-3) | < 10-3 => tiếp tục thuật toán.

+) c2 = = = -2,75

=> f(c2).f(a) = f(-2,5).f(-3) = 3,328 > 0

=> a = c2. Khoảng cách ly nghiệm là: (-2,75;-2,5)

Tính | (-2,5) – (-2,75) | < 10-3 => tiếp tục thuật toán.

+) c3 = = = -2,625

=> f(c3).f(a) = f(-2,625).f(-2,75) = 0,4615 >0

=> a = c3. Khoảng cách ly nghiệm là: (-2,625;-2,5)

Tính | (-2,5) – (-2,625) | < 10-3 => tiếp tục thuật toán.

+) c4 = = = -2.5625

=> f(c4).f(a) = f(-2.5625).f(-2,5) = 0,0529 > 0

=> a = c4. Khoảng cách ly nghiệm là: (-2,5625;-2,5)

Tính | (-2,5) – (-2,5625) | < 10-3 => tiếp tục thuật toán.

+) c5 = = = -2,53125

=> f(c5).f(a) = f(-2,53125).f(-2,5) = -0,0004309 < 0

=> b = c5. Khoảng cách ly nghiệm là: (-2,5625;-2,53125)

Tính | (-2,5625) – (-2,53125) | < 10-3 => tiếp tục thuật toán.

+) c6 = = = -2,546875

=> f(c6).f(a) = f(-2,546875).f(-2,5625) = 0,007730025 > 0

=> a = c6. Khoảng cách ly nghiệm là: (-2,546875;-2,53125)

Tính | (-2,546875) – (-2,53125) | < 10-3 => tiếp tục thuật toán.

+) c7 = = = -2,539063

=> f(c7).f(a) = f(-2,539063).f(-2,546875) = 0,001726557 >0

=> a = c7. Khoảng cách ly nghiệm là: (-2,539063;-2,53215)

Tính | (-2,539063) – (-2,53215) | < 10-3 => tiếp tục thuật toán.

+) c8 = = = -2,535156

=> f(c8).f(a) = f(-2,535156).f(-2,539063) =0,00035346 > 0

=> a = c8. Khoảng cách ly nghiệm là: (-2,535156;-2,53215)

Tính | (-2,535156) – (-2,53215) | < 10-3 => tiếp tục thuật toán.

+) c9 = = = -2,533203

=> f(c9).f(a) = f(-2,533203).f(-2,535156) = 0,000056102 > 0

=> a = c9. Khoảng cách ly nghiệm là: (-2,533203;-2,53215)

Tính | (-2,533203) – (-2,53215) | < 10-3 => tiếp tục thuật toán.

+) c10 = = = -2,532227

=> f(c10).f(a) = f(-2,532227).f(-2,53215) = 0,0000025097 > 0

=> a = c10. Khoảng cách ly nghiệm là: (-2,532227;-2,53215)

Tính | (-2,532227) – (-2,53215) | > 10-3 => kết thúc thuật toán.

Vậy nghiệm xấp xỉ của phương trình: x3 + 3x2 – 3 = 0 là

x = -2,532227

\* Việc tính toán giá trị của thuật toán có thể tóm gọn trong bảng sau:



## *2.3: Đánh giá sai số:*

Giả sử ta đặt x\* là nghiệm. Ta có:

Bước 0: = |x\* - x0| , x0 = ;

Bước 1: = |x\* - x1| , x1 = ;

……

Bước n: = |x\* - xn| , xn = 1 ;

Như vậy, ta có sai số tuyệt đối: err = xn =

Mặt khác ta kiểm tra sự hội tụ về nghiệm:

|x\* - xn| ≤ (b-a)/2n+1  => ≤ = 0

* = 0

Vậy dãy xn hội tụ về nghiệm x\* của phương trình n ra vô cùng.

## *2.4: Số phép tính cần thực hiện:*

Với phương pháp chia đôi, ta còn có thể tính được số lần lặp lại của thuật toán:

err = xn = => = => n =

1. **Chương trình (Code) kiểm tra phương pháp chia đôi:**

**( Sử dụng ngôn ngữ lập trình Python)**

*3.1:Đối với công thức tiên nghiệm:*

# -\*- coding: utf-8 -\*-

"""bisection\_1.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/16htYqqs\_SRLxHOFAgoHblO3B11rdC9IX

"""

# -\*- coding: utf-8 -\*-

"""bisection\_1.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/16htYqqs\_SRLxHOFAgoHblO3B11rdC9IX

"""

#Tiên nghiệm

import numpy as np

import matplotlib.pyplot as plt

import math

from numpy.lib import scimath

import timeit

def f(x):

return np.log(3 \* x) + pow(2, x) - 2 \* x -2

# Vẽ đồ thị hàm số

plt.xlabel("Giá trị x biến thiên")

plt.ylabel("Giá trị y biến thiên")

plt.title("Đồ thị hàm số của phương trình f(x) ")

x = np.linspace(0,3.5 , 1000)

plt.plot(x, f(x))

plt.plot(0, 0, '+')

plt.plot(0, )

plt.grid()

plt.show()

# Bisection method

def bisection():

def try\_():

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

print("Bạn có muốn sử dụng lại chương trình bisection method không? Yes/No? Y/N?")

request = str(input())

a = request.upper()

if (a == 'YES' or a == 'Y'):

bisection()

elif (a == 'NO' or a == 'N'):

print("Cảm ơn. Hẹn gặp lại ♥ ")

try:

print("Khoảng cách ly nghiệm là khoảng sao trong khoảng a, b có duy nhất 1 nghiệm của phương trình")

print("Xác định cận dưới a của khoảng cách ly nghiệm. ")

a = float(input("a = "))

print("Xác định cận trên b của khoảng cách ly nghiệm. ")

b = float(input("b = "))

print("Độ chính xác epsilon.")

eps = float(input("epsilon = "))

except:

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

print("Yêu cầu xác định lại khoảng cách ly nghiệm hoặc epsilon (số thực).")

print("Vui lòng xác định lại.")

bisection()

else:

if (a >= b or eps >= 1 or eps <= 0):

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

print("Yêu xác định lại a < b và 0 < epsilon < 1.")

bisection()

elif (f(a) \* f(b) >= 0):

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

print("Khoảng cách ly nghiệm không hợp lệ yêu cầu xác định lại.")

try\_()

#Vì đã kiểm tra điều kiện nghiêm ngặt của a, b ở trên nên chắc chắn rằng a thỏa mãn khoảng cách ly

elif ( f(a)\*f((a+b)/2) == 0):

print("Nghiệm gần đúng của phưởng trình là: ",((a+b)/2))

print("Số lần lặp là: 1 lần")

try\_()

#Bisection-Method

else:

#Số lần có thể phải lặp qua

n = math.ceil(scimath.log2((b - a) / (2\*eps)))

print("Số lần lặp ước chừng khoảng: ", n, " lần")

# làm tròn eps

e = eps

demss = 0

while e < 1:

demss += 1

e \*= 10

count = 0

print("{0:^15}|{1:^15}|{2:^15}|{3:^15}|{4:^15}|{5:^15}|{6:^15}".format("Số lần lặp", "a", "b", "c", "f(a)",

"f(b)", "f(c)"))

for i in range (0,n):

c = (a + b) / 2.0

mid = f(a) \* f(c)

print("{0:^15}|{1:<15}|{2:<15}|{3:<15}|{4:<15}|{5:<15}|{6:<15}".format(count, round(a, demss),

round(b, demss),

round(((a + b) / 2), demss),

round(f(a), demss),

round(f(b), demss),

round(f((a + b) / 2), demss)))

if (mid > 0):

a = c

elif (mid < 0):

b = c

else:

print("- Số lần lặp: ", count)

print("=> Nghiệm gần đúng của phương trình là: x = ", round(c, demss))

try\_()

count += 1

print("=> Nghiệm gần đúng của phương trình là: x = ", round(c, demss))

start = timeit.default\_timer()

stop = timeit.default\_timer()

print('- Time: ', (stop - start) \* 1000, "ms")

try\_()

bisection()

*3.2: Đối với công thức hậu nghiệm:*

# -\*- coding: utf-8 -\*-

"""Bisection\_2.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1RCtc0h2GPC0eBgeVX15Q8NJi0gA4fCdy

"""

# Hậu nghiệm

import numpy as np

import matplotlib.pyplot as plt

import math

from numpy.lib import scimath

import timeit

# Nhập vào hàm số

def f(x):

return np.log(3 \* x) + pow(2, x) - 2 \* x -2

# Vẽ đồ thị hàm số

plt.xlabel("Giá trị x biến thiên")

plt.ylabel("Giá trị y biến thiên")

plt.title("Đồ thị hàm số của phương trình f(x) ")

x = np.linspace(0, 3.5, 1000)

plt.plot(x, f(x))

plt.plot(0, 0, '+')

plt.grid()

plt.show()

# Bisection method

def bisection():

def try\_():

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

print("Bạn có muốn sử dụng lại chương trình bisection method không? Yes/No? Y/N?")

request = str(input())

a = request.upper()

if (a == 'YES' or a == 'Y'):

bisection()

elif (a == 'NO' or a == 'N'):

print("Cảm ơn. Hẹn gặp lại ♥ ")

try:

print("Khoảng cách ly nghiệm là khoảng sao trong khoảng a, b có duy nhất 1 nghiệm của phương trình")

print("Xác định cận dưới a của khoảng cách ly nghiệm. ")

a = float(input("a = "))

print("Xác định cận trên b của khoảng cách ly nghiệm. ")

b = float(input("b = "))

print("Độ chính xác epsilon.")

eps = float(input("epsilon = "))

except:

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

print("Yêu cầu xác định lại khoảng cách ly nghiệm hoặc epsilon (số thực).")

print("Vui lòng xác định lại.")

bisection()

else:

if (a >= b or eps >= 1 or eps <= 0):

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

print("Yêu xác định lại a < b và epsilon < 1 và khác epsilon >.")

bisection()

elif (f(a) \* f(b) >= 0):

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

print("Khoảng cách ly nghiệm không hợp lệ yêu cầu xác định lại.")

bisection()

# Vì đã kiểm tra điều kiện nghiêm ngặt của a, b ở trên nên chắc chắn rằng a thỏa mãn khoảng cách ly

elif (f(a) \* f((a + b) / 2) == 0):

print("Nghiệm gần đúng của phưởng trình là: ", ((a + b) / 2))

print("Số lần lặp là: 1 lần")

try\_()

# Bisection-Method

else:

# làm tròn eps

e = eps

demss = 0

while e < 1:

demss += 1

e \*= 10

# Lặp đến khi sai số tuyệt đối < sai số cần tìm thì dừng.

count = 0

print("{0:^15}|{1:^15}|{2:^15}|{3:^15}|{4:^15}|{5:^15}|{6:^15}".format("Số lần lặp", "a", "b", "c", "f(a)",

"f(b)", "f(c)"))

while ((math.fabs(b - a)/2.0)> eps):

c = (a + b) / 2.0

mid = f(a) \* f(c)

print("{0:^15}|{1:<15}|{2:<15}|{3:<15}|{4:<15}|{5:<15}|{6:<15}".format(count, round(a, demss),

round(b, demss),

round(((a + b) / 2), demss),

round(f(a), demss),

round(f(b), demss),

round(f((a + b) / 2), demss)))

if (mid > 0):

a = c

elif (mid < 0):

b = c

else:

print("- Số lần lặp: ", count)

print("=> Nghiệm gần đúng của phương trình là: x = ", round(c, demss))

try\_()

count += 1

print("- Số lần lặp: ", count)

print("=> Nghiệm gần đúng của phương trình là: x = ", round(c, demss))

start = timeit.default\_timer()

stop = timeit.default\_timer()

print('- Time: ', (stop - start) \* 1000, "ms")

try\_()

bisection()

*3.3:Đối với công thức hậu nghiệm tối ưu:*

# -\*- coding: utf-8 -\*-

"""bisection\_3.pynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1\_-6R9xDsQdSDfVDWN26h2awVqp89DR6r

"""

# Hậu nghiệm tối ưu

'''\*\*\*Với phương pháp tối ưu trên ta đã giảm được việc tính toán f(a) và f(c).f(a) nhiều lần

do dựa vào tính chất cùng hoặc khác phía của toán học

'''

import numpy as np

import matplotlib.pyplot as plt

import math

from numpy.lib import scimath

import timeit

# Nhập vào hàm số

def f(x):

return np.log(3 \* x) + pow(2, x) - 2 \* x -2

# Vẽ đồ thị hàm số

plt.xlabel("Giá trị x biến thiên")

plt.ylabel("Giá trị y biến thiên")

plt.title("Đồ thị hàm số của phương trình f(x) ")

x = np.linspace(0, 3.5, 1000)

plt.plot(x, f(x))

plt.plot(0, 0, '+')

plt.grid()

plt.show()

# Bisection method

def bisection():

def try\_():

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

print("Bạn có muốn sử dụng lại chương trình bisection method không? Yes/No? Y/N?")

request = str(input())

a = request.upper()

if (a == 'YES' or a == 'Y'):

bisection()

elif (a == 'NO' or a == 'N'):

print("Cảm ơn. Hẹn gặp lại ♥ ")

try:

print("Xác định cận dưới a của khoảng cách ly nghiệm. ")

a = float(input("a = "))

print("Xác định cận trên b của khoảng cách ly nghiệm. ")

b = float(input("b = "))

print("Độ chính xác epsilon.")

eps = float(input("epsilon = "))

except:

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

print("Yêu cầu xác định lại khoảng cách ly nghiệm hoặc epsilon (số thực).")

print("Vui lòng xác định lại.")

bisection()

else:

if (a > b or abs(a - b) == 0 or eps >= 1 or eps <= 0):

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

print("Yêu xác định lại a < b và epsilon < 1 và khác epsilon >.")

bisection()

elif (f(a) \* f(b) >= 0):

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

print("Khoảng cách ly nghiệm không hợp lệ yêu cầu xác định lại.")

bisection()

elif ( f(a)\*f((a+b)/2) == 0):

print("Nghiệm gần đúng của phưởng trình là: ",((a+b)/2))

print("Số lần lặp là: 1 lần")

try\_()

#Bisection-Method

else:

# Tính F(a) một lần duy nhất

f\_a = f(a)

# Gán F(a) cho mệnh đề logic

if (f\_a > 0):

f\_a = True

else:

f\_a = False

# làm tròn eps

e = eps

demss = 0

while e < 1:

demss += 1

e \*= 10

# Lặp đến khi sai số tuyệt đối < sai số cần tìm thì dừng.

print("{0:^15}|{1:^15}|{2:^15}|{3:^15}|{4:^15}|{5:^15}|{6:^15}".format("Số lần lặp","a","b","c","f(a)","f(b)","f(c)"))

count = 0

while (math.fabs(b-a)/2 >= eps):

c = (a + b) / 2.0

f\_c = f(c)

if (count >=0):

print("{0:^15}|{1:<15}|{2:<15}|{3:<15}|{4:<15}|{5:<15}|{6:<15}".format(count,round(a,demss),round(b,demss),round(((a+b)/2),demss),round(f(a),demss),round(f(b),demss),round(f((a+b)/2),demss)))

# Gán F(c) cho mệnh đề logic

if (f\_c > 0):

f\_c = True

elif (f\_c<0):

f\_c = False

else:

print("- Số lần lặp: ", count)

print("=> Nghiệm gần đúng của phương trình là: x = ", round(c, demss))

try\_()

# Kiểm tra tính cùng phía của đồ thị

if (f\_a != f\_c): # =>> f(a) trái dấu với f(c)

b = c

# f(a) cùng dấu với f(c)

else:

a = c

count += 1

print("- Số lần lặp: ", count)

print("=> Nghiệm gần đúng của phương trình là: x = ", round(c, demss))

print("\*\*\*Với phương pháp tối ưu trên ta đã giảm được việc tính toán f(a) và f(c).f(a) nhiều lần \n do dựa vào tính chất cùng hoặc khác phía của toán học ")

start = timeit.default\_timer()

stop = timeit.default\_timer()

print('- Time: ', (stop - start) \* 1000, "ms")

try\_()

bisection()

1. **Ưu, nhược điểm của phương pháp chia đôi:**

* *Về ưu điểm:*
* Dãy giá trị xấp xỉ xn, như đã chứng minh ở trên, luôn luôn hội tụ về giá trị nghiệm đúng của phương trình.
* Dễ hiểu và dễ cài đặt chương trình dịch.
* Độ tin cậy cao.
* Không có nhiều điều kiện rang buộc nên có thể giải quyết hầu hết các phương trình.
* *Về nhược điểm:*
* Tốc độ hội tụ chậm.
* Phụ thuộc vào sự trái dấu của 2 đầu mút của khoảng cách ly nghiệm.
* Không thể xác định nhiều nghiệm
* Với độ chính xác cao thì cần sử dụng nhiều vòng lặp, rất mất thời gian cũng như tài nguyên máy tính.
* Sự hội tụ sẽ chậm hơn nếu giá trị của nghiệm nằm gần 2 đầu mút của khoảng cách ly nghiệm.