**Tiên nghiệm:**

(source: )

#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}"**.f ormat(**"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}"**.f ormat(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()

**Hậu nghiệm:**

(source: )

# 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.

#Tạo bảng xét các lần lặp

count = 0

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

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

#Phương pháp

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}"**.f ormat(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()

**Hậu nghiệm tối ưu:**

(source: )

# 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}"**.f ormat(**"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}"**.f ormat(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()