# Bahria University,

## Karachi Campus



LAB EXPERIMENT NO.

**\_03\_**

LIST OF TASKS

|  |  |
| --- | --- |
| **TASK NO** | **OBJECTIVE** |
| 01 | Write a python program for approximating the roots of following functions using the bisection method:  a. \(x^3 - 9x + 1\) starting with the interval \([2, 4]\)  b. \(3x = \sqrt{1 + \sin{x}}\) starting with the interval \([0, 1]\) |
| 02 | Write a python program for approximating the roots of following functions using the false-position method:  a. \(f(x) = (x - 4)^2(x + 2)\) starting with the interval \([-2.5, -1.0]\)  b. \(f(x) = e^x(3.2 \sin{x} - 0.5 \cos{x})\) starting with the interval \([3, 4]\) |

Submitted On:

Date: 8/10/2024

**Task No 01:** Write a python program for approximating the roots of following functions using the bisection method:

a. \(x^3 - 9x + 1\) starting with the interval \([2, 4]\)

b. \(3x = \sqrt{1 + \sin{x}}\) starting with the interval \([0, 1]\)

**Solution:**

def func(x):

return x\*\*3-(9\*x)+1  # Change the equation here

def bisection(a, b, num\_iterations):

if (func(a) \* func(b) >= 0):

print("The chosen interval does not bracket a root.")

return

for \_ in range(num\_iterations):

root = (a + b) / 2

if func(root) == 0:

print("Found the exact root:", root)

return

if func(root) \* func(a) < 0:

b = root

else:

a = root

print("The approximate root after", num\_iterations, "iterations is:", root)

a = 2

b = 4

num\_iterations = 10

bisection(a, b, num\_iterations)

import math

def func(x):

try:

return (9 \* (x\*\*2) - 1) / math.sin(x)  # Handle possible division by zero

except ZeroDivisionError:

return float('inf')

def bisection(a, b, num\_iterations):

if (func(a) \* func(b) >= 0):

print("The chosen interval does not bracket a root.")

return

for \_ in range(num\_iterations):

root = (a + b) / 2

if func(root) == 0:

print("Found the exact root:", root)

return

if func(root) \* func(a) < 0:

b = root

else:

a = root

print("The approximate root after", num\_iterations, "iterations is:", root)

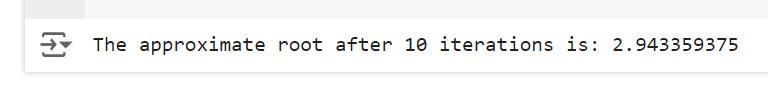
a = 0.1  # Avoid sin(0) = 0 to prevent division by zero

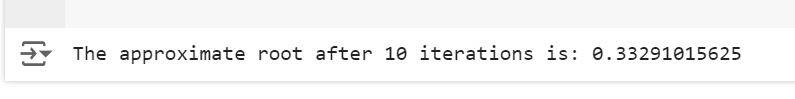
b = 1

num\_iterations = 10

bisection(a, b, num\_iterations)

**Output:**

****

****

**Task No 02:** Write a python program for approximating the roots of following functions using the false-position method:

a. \(f(x) = (x - 4)^2(x + 2)\) starting with the interval \([-2.5, -1.0]\)

b. \(f(x) = e^x(3.2 \sin{x} - 0.5 \cos{x})\) starting with the interval \([3, 4]\)

**Solution:**

MAX\_ITER = 10

def function(x):

return (((x-4)\*\*2) \* (x+2))

def regulaFalsi(a, b):

if function(a) \* function(b) >= 0:

print("The chosen values of 'a' and 'b' are not suitable.")

return -1

c = a

for i in range(MAX\_ITER):

c = a - ((function(a) \* (b - a)) / (function(b) - function(a)))

if function(c) == 0:

break

elif function(c) \* function(a) < 0:

b = c

else:

a = c

print("The estimated root is: ", '%.4f' % c)

a = -2.5

b = -1.0

regulaFalsi(a, b)

import math

MAX\_ITER = 10

def function(x):

return (math.exp(x) \*(3.2\*math.sin(x)-0.5\*math.cos(x)))

def regulaFalsi(a, b):

if function(a) \* function(b) >= 0:

print("The chosen values of 'a' and 'b' are not suitable.")

return -1

c = a

for i in range(MAX\_ITER):

c = a - ((function(a) \* (b - a)) / (function(b) - function(a)))

if function(c) == 0:

break

elif function(c) \* function(a) < 0:

b = c

else:

a = c

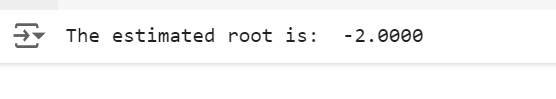
print("The estimated root is: ", '%.4f' % c)

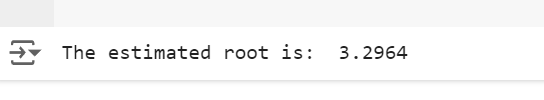
a =3

b = 4

regulaFalsi(a, b)

**Output:**

****

****