Ministerul Educaţiei, Tineretului şi Sportului al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

Demaptamentul Informatică și Ingineria Sistemelor

# RAPORT

Lucrare de laborator nr.1

la Metode Numerice

Tema: Rezolvarea numerica a ecuatiilor algebrice si trascendente

A efectuat: Cojocari Dragos st. gr. TI-214

A verificat: asistent univ.

V. Struna

Chişinău 2022

## Tema: Rezolvarea numerica a ecuatiilor algebrice si trascendente

**Scopul lucrarii:**

1. Sa se separe toate radacinile reale ale ecuatiei f(x) = 0 unde y = f(x) este o functie reala de variabila reala.
2. Sa se determine o radacina reala a ecuatiei date cu ajutorul metodei injumatatirii intervalului cu o erare mai mica decat
3. Sa se precizeze radacina obrinuta cu exactitatea utilizand:

* Metoda aproximatiilor successive
* Metoda tangentelor(Newton)

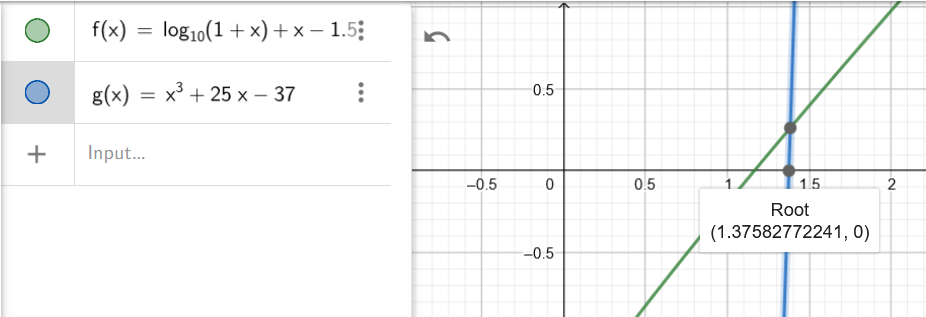
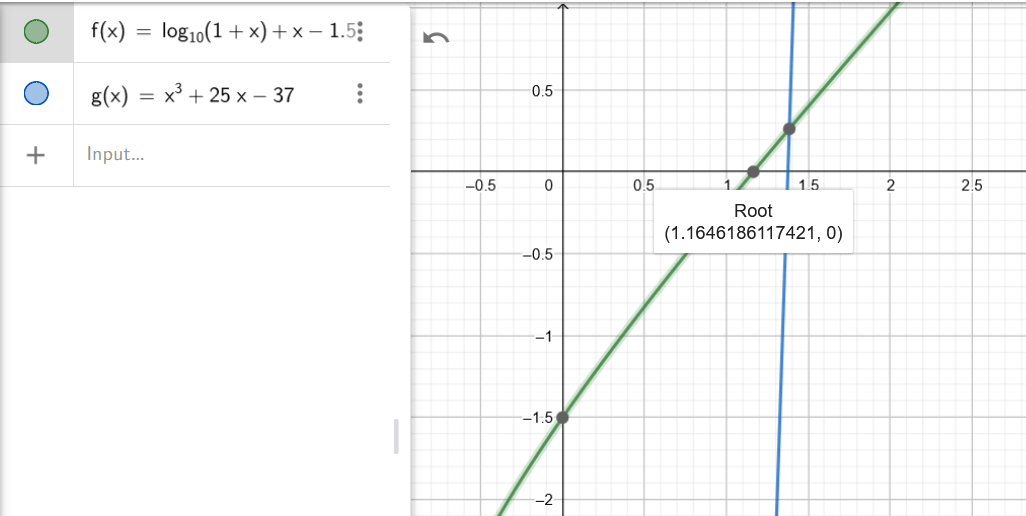
1. Sa se compare rezultatele luand in consideratie numarul de iteratii, evaluarile pentru functia si derivate.

**Probleme date spre rezolvare:**

**Var 7**

a)b)

**Metoda Grafica:**



**Metoda analitica:**

df =

x =

|  |  |  |  |
| --- | --- | --- | --- |
| x | 0 | 1.43 | 2.86 |
| y | -1.5 | 0.31 | 1.94 |

**f =**

df = => imaginar

**Codul sursa:**

import math

import pandas as pd

#log(1+x) + x - 1.5

def f1(x):

    return math.log10(1+x) + x - 1.5

# fi (pentru medota aproximarilor)

def f1fi(x):

    return 1.5 - math.log10(1+x)

#f1 derivat si dublu derivat pentru metoda newton

def f1d(x):

    return 1/(math.log10(10)\*(x+1)) + 1

def f1dd(x):

    return -1/(math.log(10)\*(x+1)\*\*2)

#x^3 + 25x - 37

def f2(x):

    return x\*\*3 + 25\*x - 37

# fi (pentru medota aproximarilor)

def f2fi(x):

    return (x\*\*3-37)/(-25)

#f2 derivat si dublu derivat pentru metoda newton

def f2d(x):

    return 3\*x\*\*2 + 25

def f2dd(x):

    return 6\*x

def afisare(x, i, err):

    print("x =", x, "Iteratii:", i,"Eroare:",err)

#metoda injumatatirii intervalelor (bisectoarelor)

def bisect(func,a,b):

    i = 0

    x = a

    y = func(x)

    if(func(a) \* func(b) < 0):

        while(abs(y) > 10\*\*(-6)):

            x = (a+b)/2

            y = func(x)

            if(func(a)\*y < 0):

                b = x

            else:

                a = x

            i += 1

    afisare(x,i,a-b)

    return x

#metoda aproximatiilor succesive

def aprox(func, a):

    i = 0

    while True:

        x = a

        a = func(x)

        i += 1

        if(abs(a - x) < 10\*\*(-6)):

            afisare(x,i,a-x)

            return x

#metoda newton

def newton(func, dFunc, ddFunc, a, b):

    i = 0

    if(func(a) \* ddFunc(a) > 0):

        x = x1 = a

    else:

        x = x1 = b

    while True:

        x = x1

        x1 = x - func(x) / dFunc(x)

        i += 1

        if(abs(x1-x) < 10\*\*(-6)):

            afisare(x,i,x1-x)

            return x1

#log(1+x) + x - 1.5

print("Pentru f = log(1+x) + x - 1.5")

print("Metoda injumatatirii:")

bisect(f1,0,2)

print("Metoda aproximatiilor:")

aprox(f1fi,0)

print("Metoda Newton:")

newton(f1,f1d,f1dd,0,1)

#x^3 + 25x - 37

print("\nPentru f = x^3 + 25x - 37")

print("Metoda injumatatirii:")

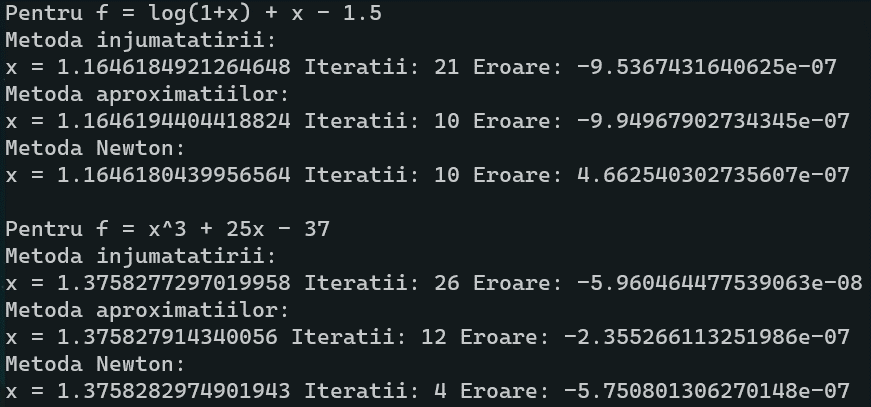
bisect(f2,-2,2)

print("Metoda aproximatiilor:")

aprox(f2fi,-2)

print("Metoda Newton:")

newton(f2,f2d,f2dd,-1,1)



**Concluzie:**

În cadrul acestei lucrări de laborator am separat rădăcinile reale ale funcției f. Am aflat radăcinile reale prin metoda înjumatatirii intervalului, metoda aproximatiilor succesive și metoda Newton.

Sunt de părerea ca metodele iterative sunt mai potrivite decât metodele directe pentru rezolvarea unor ecuații, rezolvarea cărora este imposibilă sau anevoioasă prin metode directe.

Am observat că numărul de iterații pentru fiecare metodă diferă. Metoda newton fiind cea mai eficientă din punct de vedere a numarului de opertii, deci foarte rapida și de încredere, de exemplu, fata de metoda aproximațiilor succesive, care des dă greș sau prin care este imposibil de a calcula radacinile, deoarece functia repede diverge.

De asemenea, ca alegerea valorii inițiale influențează asupra convergenței sau divergenței procesului iterativ. În cazul unui proces iterativ convergent, valoarea inițială afectează doar numărul de iterații necesar pentru atingerea unei erori impuse