Ministerul Educației, Culturii și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

**Raport la**

**Lucrarea de laborator Nr. 1**

*Disciplina: Metode și modele de calcul 1*

**Tema: ”Rezolvarea numerică a ecuațiilor algebrice și transcendente”**

**Varianta 21**

Au efectuat: Diana Rîbac, SI-201

A verificat: Conf. univ., dr. Leonid Dohotaru

Chișinău – 2021

**Scopul lucrării:**

1) Să se separe toate rădăcinile reale ale ecuației f(x)=0 unde y=f(x) este o funcție reală de variabilă reală.

2) Să se determine o rădăcină reală a ecuației, date cu ajutorul metodei înjumătățirii intervalului cu o eroare mai mică decât .

3) Să se precizeze rădăcina obținută cu exactitatea utilizând:

*- metoda înjumătățirii intervalului;*

*- metoda aproximațiilor succesive;*

*- metoda tangentelor (Newton).*

**Varianta 21**

a) lg(3x)+x-2;

b) +2;

**Codul sursa:**

#include <iostream>

#include <math.h>

#include <iomanip>

#include <windows.h>

using namespace std;

void setcolor(unsigned short color)

{

HANDLE hcon = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hcon, color);

}

double fx(double x)

{

return log10(3 \* x) + x - 2; //v21a

//return x \* x \* x - 25 \* x + 2; //v21b

}

void injumatatireaIntervalului()

{

int a = -100, b;

double epsilon = pow(10, -4);

int k = 0;

double c, d, x;

int radacini = 0;

cout << setw(25);

cout << endl << "\t\t "; setcolor(2); cout << " Metoda injumatatirii intervalului "; setcolor(15); cout << " \t " << endl;

cout << "\t\t\t";

while (radacini != 1)

{

//if (a == 0) { a++; continue; }

b = a + 1;

double fx1 = fx(a);

if (!fx1) {

cout << "\t\t\t\t Pe intervalul [" << a << ", " << a + 1 << "] am gasit\t ";

cout << "\t\t\t Radacina " << ++radacini << " : " << fixed << "cand x = " << a << " f(x) = " << fx1 << "\t ";

a++;

break;

}

double fx2 = fx(b);

if ((fx1 > 0 && fx2 < 0) || (fx1 < 0 && fx2 > 0))

{

radacini++;

cout << "\n\t\t\t\t Pe intervalul [" << a << ", " << (a + 1) << "] am gasit: \t ";

c = a;

if (fx1 < 0) c = b;

d = a + b - c;

k = 0;

while (abs(c - d) > epsilon)

{

x = c + (d - c) / 2;

k++;

fx1 = fx(x);

if (fx1 == 0) break;

if (fx1 > 0) c = x;

else d = x;

}

cout << "\n\t\t\t Radacina " << radacini << " : " << fixed << "cand x = " << d << " f(x) = " << fx1 << endl;

}

a++;

}

cout << "\t\t\t";

cout << endl;

}

void metodaAproximatiilorSuccesive() {

double eps = pow(10, -4);

double a = 1;

double b = 2;

double e, x, xnou, xvechi;

x = a - fx(a) / (fx(b) - fx(a)) \* (b - a);

if (fx(x) \* fx(a) > 0)

{

e = b;

xnou = a;

}

else

{

e = a;

xnou = b;

}

cout << "\n\n\t\t\t";

cout << endl << "\t\t\t\t"; setcolor(2); cout << "Metoda aproximatiilor succesive "; setcolor(15); cout << " \t " << endl;

cout << "\t\t\t";

if (fx(0) == 0)

{

cout << "\n\t\t\t\t x = " << fixed << 0 << setw(12) << " f(x) = " << fx(0) << " \t\t \n";

}

do

{

xvechi = xnou;

xnou = xvechi - fx(xvechi) / (fx(e) - fx(xvechi)) \* (e - xvechi);

if (xnou < 0)

cout << "\n\t\t\t\t x = " << fixed << xnou << " f(x) = " << fx(xnou) << " \t\t ";

else

cout << "\n\t\t\t\t x = " << fixed << xnou << " f(x) = " << fx(xnou) << " \t\t ";

} while (abs(xnou - xvechi) > eps);

cout << "\n\t\t\t";

cout << endl << endl;

}

double derivata(double x)

{

return 3 / 3 \* x \* log(10); //v21a

//return 2 \* x - 25; //v21b

}

void metodaNewton()

{

double a, b, x, c;

int i, n;

// la v21a : 1 2

// la v21b : -5.5 -1

a = 1;

b = 2;

n = 10;

i = 0;

c = a - fx(a) / (fx(b) - fx(a)) \* (b - a);

if (fx(c) \* fx(a) > 0) x = a;

else x = b;

cout << "\n\t\t\t";

cout << endl << "\t\t\t"; setcolor(2); cout << " \t\t\ Metoda Newton "; setcolor(15); cout << " \t\t ";

cout << "\n\t\t\t";

cout << endl;

if (fx(0) == 0)

{

cout << left << "\t\t\t i = " << setw(2) << i << " x = " << setw(12) << 0 << " f(x) = " << setw(12) << fx(0) << " \t \n" << endl;

}

while (i < n)

{

i = i + 1;

x = x - fx(x) / derivata(x);

cout << left << "\t\t\t i = " << setw(2) << i << " x = " << setw(12) << x << " f(x) = " << setw(12) << fx(x) << " \t " << endl;

}

cout << "\t\t\t";

cout << endl << endl << endl << endl;

}

int main()

{

injumatatireaIntervalului();

metodaAproximatiilorSuccesive();

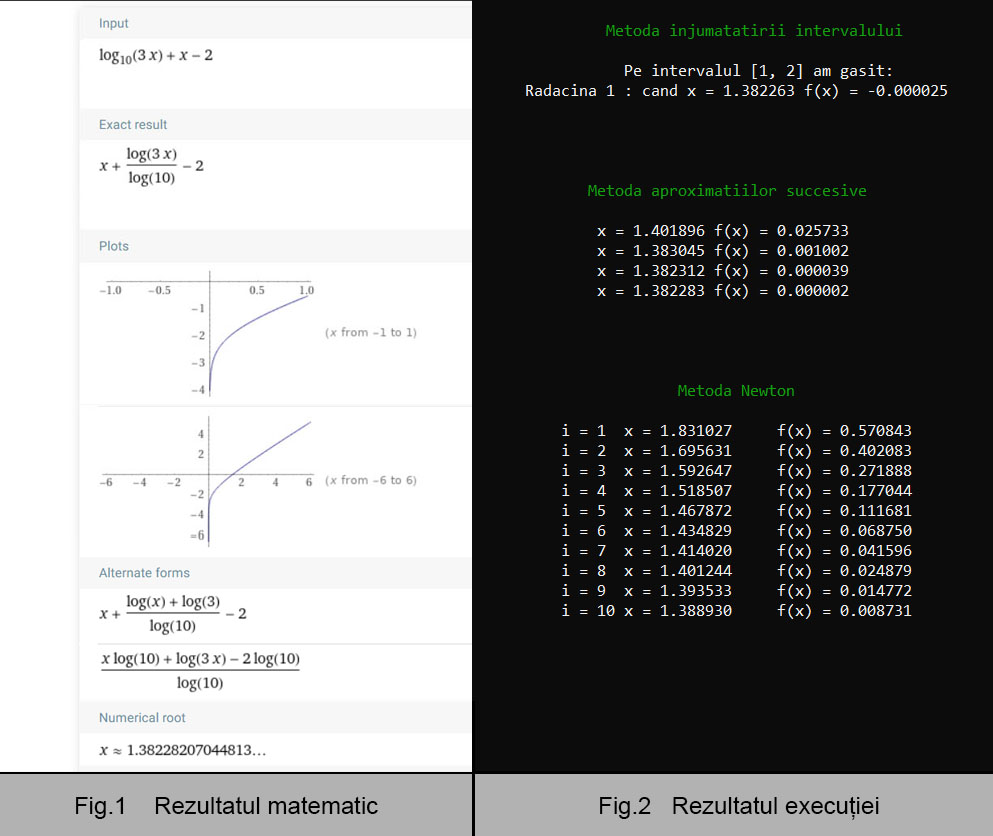
metodaNewton();

return 0;

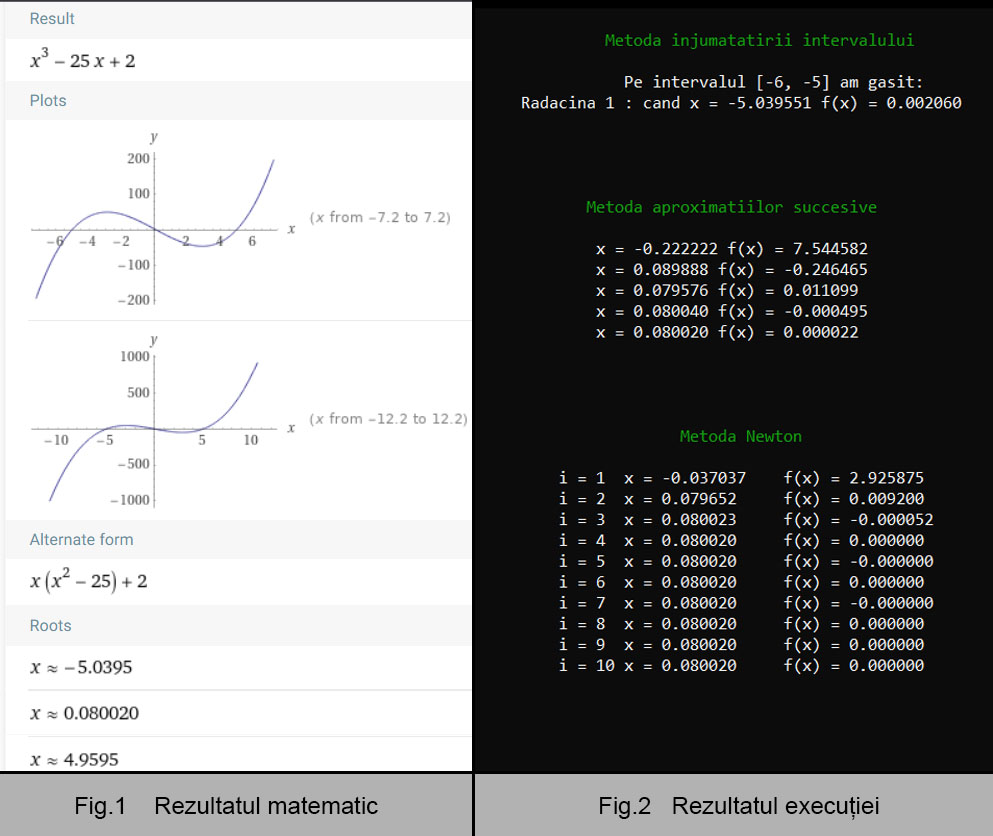
}

**Rezultatele:**

1. Pentru funcția lg(3x)+x-2;



1. Pentru funcția +2;



**Concluzie:**

În urma efectuării acestei lucrări de laborator am învățat cum să determin rădăcinile reale ale ecuațiilor prin diferite metode. Am avut să separ toate rădăcinile reale ale ecuației f(x), după care am determinat rădăcinile cu ajutorul metodei înjumătățirii intervalului cu eroarea dată și la fel am precizat rădăcina obținută cu exactitate prin metoda aproximării succesive și Newton. Am comparat ulterior rezultatele obținute în mod matematic, cu ajutorul programului de calcul Wolfram, cu ce am obținut in consola programului și astfel am ajuns la un rezultat comun.

Rezultatele obținute matematic:

[https://www.wolframalpha.com/input/?i=log10%283+\*+x%29+%2B+x+-+2](https://www.wolframalpha.com/input/?i=log10%283+*+x%29+%2B+x+-+2)

<https://www.wolframalpha.com/input/?i=x%5E3-25%2B2>