Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică și Microelectronică Departamentul Ingineria Software și Automatică

RAPORT

la lucrarea de laborator nr. 1 Metode si modele de calcul

**Tema:** REZOLVAREA NUMERICĂ A ECUAŢIILOR ALGEBRICE ŞI TRANSCENDENTE

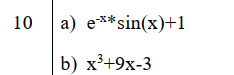
A efectuat: Popa Cătălin, TI-211

A verificat: Vladimir Patiuc

Chişinău 2022

1. ***Scopul lucrǎrii***
   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 ecuaţiei date cu ajutorul metodei înjumătăţirii intervalului cu o eroare mai mică decât ε*=10-2*.
   3. Să se precizeze rădăcina obţinută cu exactitatea ε*=10-6* utilizând

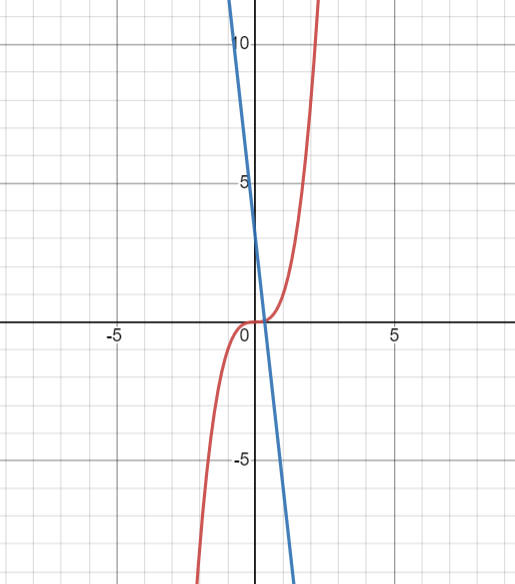
* metoda aproximațiilor succesive
* metoda tangentelor (Newton)
* metoda secantelor.
  1. Să se compare rezultatele luând în considerație numărul de iterații, evaluările pentru funcția şi derivată.



Pentru a separa rădăcinile primei ecuații, am folosit metoda grafică și metoda analitică.

**Metoda grafica**

**Graficul : Graficul pentru funcția f(x) = x^3 si g(x)= -9x+3**



Din grafic observam ca intersecțiile au loc în intervalele [0,1].

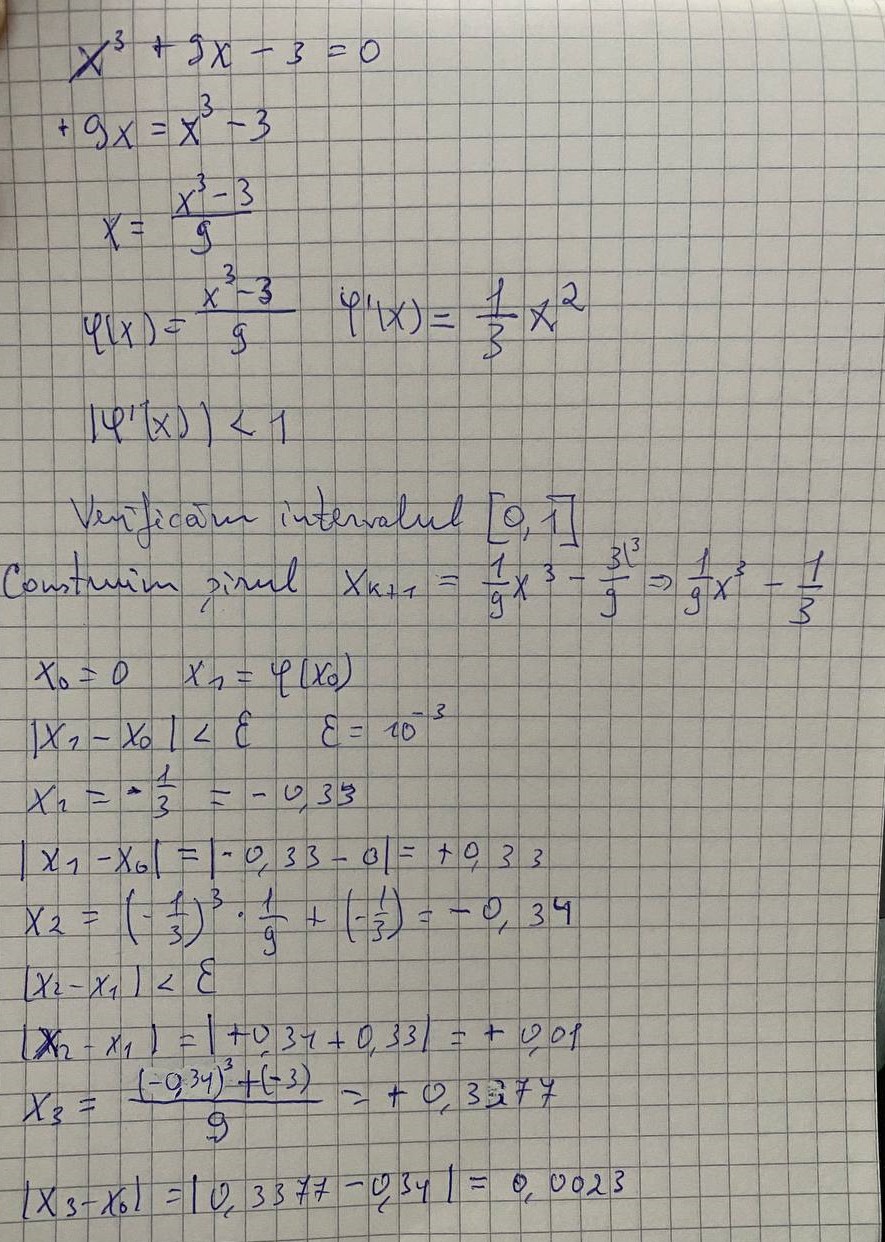
**Metoda analitică**

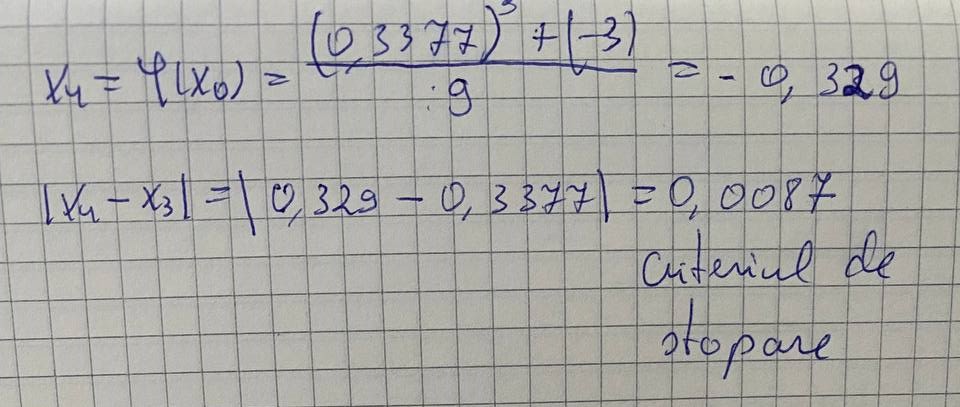
**Rândul de Rolle**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | -3 |  |  | 3 |
| f(x) | - | + | - | + |

**Metoda semi-interval**

**f(x)=x3 -9x-3**

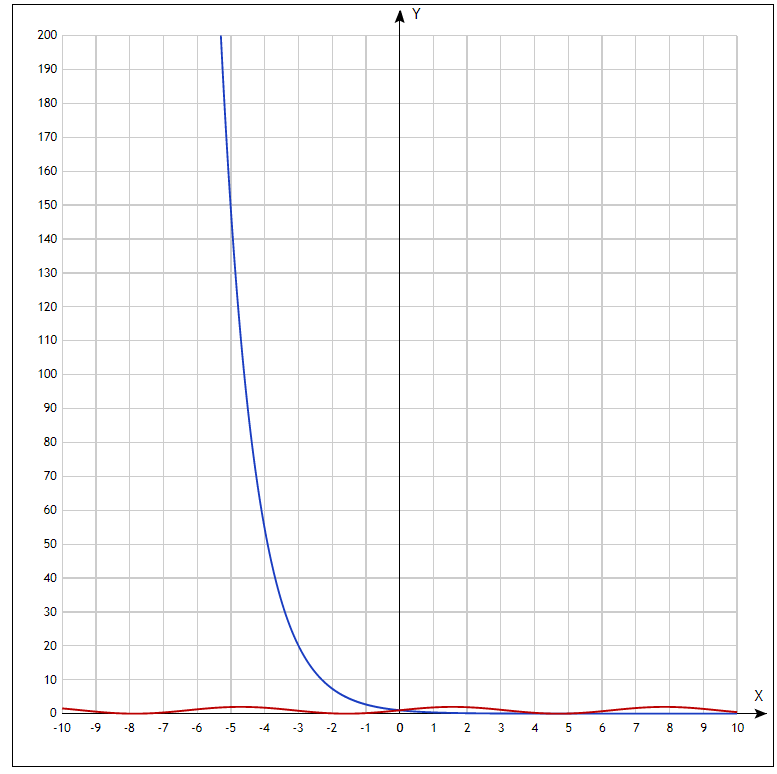
****

****

Pentru a doua ecuație, am folosit atât metoda grafică, cât și metoda analitică de separare a rădăcinilor:

**1**

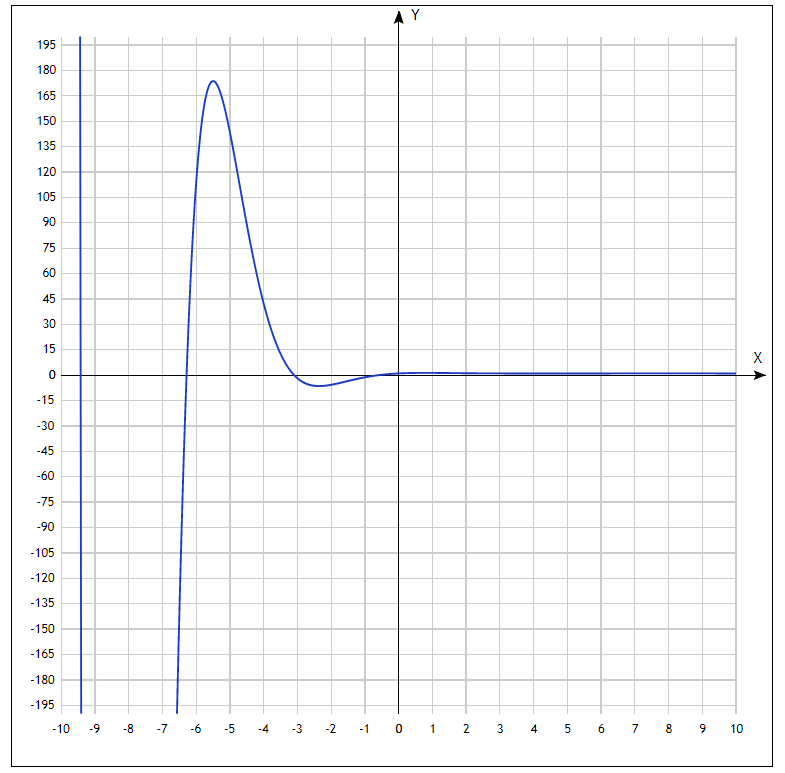
**a)** Graficul pentru **1**



**Graficul functiei e^-x si sin(x)+1**

Prin urmare, ecuatia data are o radacina reala ξ ∈(-1,1)

Metoda analitica



Graficul functiei e^-x \* sin(x)+1

Din grafic observam ca intersectia are loc in intervalul (-6,-5).

**Codul programului**

#include <iostream>

#include <conio.h>

#include <math.h>

using namespace std;

double (\*f)(double), (\*fn)(double), (\*fd)(double);

double f1(double x)

{

    return pow(x, 3) + 9 \* x - 3;

}

double f2(double x)

{

    return pow(2.71, -x) \* sin(x) + 1;

}

double fd1(double x)

{

    return 3 \* pow(x, 2) + 9;

}

double fd2(double x)

{

    return ((-sin(x)+cos(x))/pow(2.71, x));

}

double f3(double x)

{

    return (pow(x, 3) - 3) / 9;

}

double f4(double x)

{

    return ((-sin(x)+cos(x))/pow(2.71, x));

}

void aproximatie()

{

    cout <<"\nMetoda aproximatiei: \n";

    int k = 0;

    double x0, x1, eps = 0.000001;

    cout << " Introduceti valoare initiala x0" << endl;

    cout << " x0 = ";

    cin >> x0;

    while (1)

    {

        x1 = fn(x0);

        k++;

        if (abs(x1 - x0) < eps)

        {

            cout << " Radacina este: " << x0 << endl

                 << " Numarul de iteratii " << k << endl;

            break;

        }

        x0 = x1;

    }

    \_getch();

}

void injumatatire()

{

    cout <<"\nMetoda injumatatirii: \n";

    int k = 0;

    double a, b, c = 0, eps = 0.01;

    cout << " Introduceti intervalul " << endl;

    cout << " a = ";

    cin >> a;

    cout << " b = ";

    cin >> b;

    while ((b - a) > eps)

    {

        k++;

        c = (a + b) / 2;

        if (f(c) == 0)

            break;

        if (f(a) \* f(c) < 0)

            b = c;

        else

            a = c;

    }

    cout << " Radacina este: " << c << endl;

    cout << " Numarul de iteratii: " << k;

    \_getch();

}

void newton()

{

    cout <<"\nMetoda Newton: \n";

    int k = 0;

    double x0, x1, eps = 0.000001;

    cout << " Introduceti valoare initiala x0" << endl;

    cout << " x0 = ";

    cin >> x0;

    while (1)

    {

        x1 = x0 - f(x0) / fd(x0);

        k++;

        if (abs(x1 - x0) < eps)

        {

            cout << " Radacina este: " << x0 << endl

                 << " Numarul de iteratii " << k << endl;

            break;

        }

        x0 = x1;

    }

    \_getch();

}

void secante()

{

    cout <<"\nMetoda secantei: \n";

    double x2, x1, x3 = 0, y, eps = 0.000001;

    int n = 0;

    cout << " Introduceti intervalul " << endl;

    cout << " a = ";

    cin >> x1;

    cout << " b = ";

    cin >> x2;

    do

    {

        n++;

        y = x3;

        x3 = x2 - (f(x2) \* (x2 - x1) / (f(x2) - f(x1)));

        x1 = x2;

        x2 = x3;

    } while (fabs(y - x3) >= eps);

    cout << " Radacina este: " << x3 << endl;

    cout << " Numarul de iteratii : " << n << endl;

    \_getch();

}

void selectFunction()

{

    system("cls");

    cout << " 1. Functia f1(x) = x^3 + 9x - 3" << endl;

    cout << " 2. Functia f2(x) = (e^-x) \* sin(x) + 1" << endl;

    int optiune;

    do

    {

        optiune = \_getch();

    } while (optiune < '1' || optiune > '2');

    system("cls");

    switch (optiune)

    {

    case '1':

    {

        f = f1;

        fn = f3;

        fd = fd1;

        break;

    }

    case '2':

    {

        f = f2;

        fn = f4;

        fd = fd2;

        break;

    }

    }

}

int meniu()

{

    system("cls");

    if (f == f1)

        cout << " 1. Functia f1(x) = x^3 + 9x - 3" << endl;

    else

        cout << " 2. Functia f2(x) = (e^-x) \* sin(x) + 1" << endl;

    cout << " 1. Selectarea functiei" << endl;

    cout << " 2. Metoda aproximatiei succesive - exactitatea 10^(-6)" << endl;

    cout << " 3. Metoda injumatatirii - exactitatea 10^(-2)" << endl;

    cout << " 4. Metoda Newton – exactitatea 10^(-6)" << endl;

    cout << " 5. Metoda secantelor - exactitatea 10^(-6)" << endl;

    cout << " 6. Iesire" << endl;

    int optiune;

    do

    {

        optiune = \_getch();

    } while (optiune < '1' || optiune > '6');

    system("cls");

    return optiune - '0';

}

int main()

{

    int optiune;

    f = f1;

    fn = f3;

    fd = fd1;

    do

    {

        switch (optiune = meniu())

        {

        case 1:

        {

            selectFunction();

            break;

        }

        case 2:

        {

            aproximatie();

            break;

        }

        case 3:

        {

            injumatatire();

            break;

        }

        case 4:

        {

            newton();

            break;

        }

        case 5:

        {

            secante();

            break;

        }

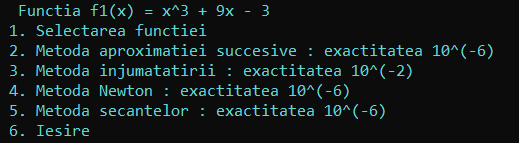
        }

    } while (optiune != 6);

}

**Afisare date**

1.Menu

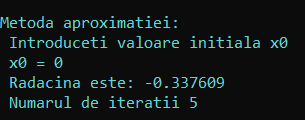


2. Functiile

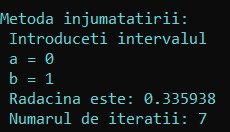


**Pentru functia x^3 + 9x - 3**

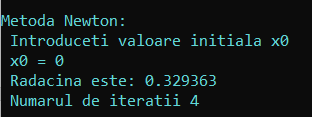
3.Metoda aproximatiei succesive



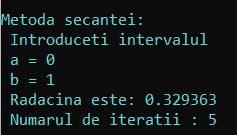
4.Metoda injumatarii



5.Metoda Newton

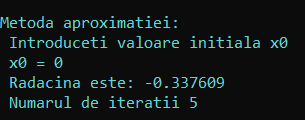


6.Metoda secantelor

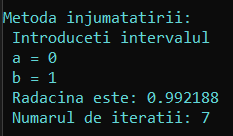


**Pentru functia (e^-x)\*sin(x) + 1**

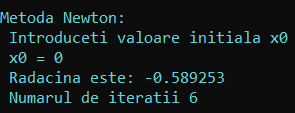
1.Metoda aproximatiei succesive



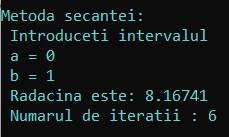
2.Metoda injumatarii



3.Metoda Newton



4.Metoda secantelor



**Tabelul rezultatelor**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metodă** | **Rădăcină** | | **Iterații** | |
| f1(x) | f2(x) | f1(x) | f2(x) |
| **Aproximare succesivă** | 0,337609 | -4,30278 | 5 | 9 |
| **A semiintervalului** | 0,335938 | 0,992188 | 7 | 7 |
| **De la Newton** | 0,329363 | -0.58925 | 4 | 6 |
| **Secantă** | 0,778902 | 8.16741 | 5 | 6 |

Făcând acest lucru de laborator, am învățat metodele de rezolvare a ecuațiilor algebrice și transcendentale, obținând aproximativ aceeași rădăcină. Metoda grafică mi s-a părut cea mai simplă deoarece nu necesită calcule și este ușor de determinat intervalul în care se găsește soluția, dar cea mai eficientă mi se pare metoda secantei, având în vedere că a făcut cele mai puține iterații. pana se gaseste. soluția.

Cunoștințele dobândite prin efectuarea acestei lucrări de laborator mă vor ajuta în studiul continuu și aprofundat al obiectului de studiu și îmi vor oferi posibilitatea de a dobândi o anumită experiență de lucru în rezoluția numerică a ecuațiilor algebrice și transcendente, dezvoltându-mi analitica gândire.