**MINISTERUL EDUCAŢIEI**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare Informatică și Microelectronică**

**Tehnologia Informației**

**Raport**

**Disciplina:**Metode numerice

**Lucrarea de laborator nr. 1**

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

Varianta: 23

|  |  |  |
| --- | --- | --- |
| **Student:** | **\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Raevschi Grigore TI-231** |
| **Coordonator:** | **\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Conf. univ. Pațiuc Vladimir** |
|  |  |  |

Chișinău 2024

Cuprins

[Scopul lucrării 2](#_Toc178019661)

[Varianta: 23 2](#_Toc178019662)

[Funcția a 2](#_Toc178019663)

[Funcția b 3](#_Toc178019664)

[Codul 5](#_Toc178019665)

[Codul 10](#_Toc178019666)

[Concluzie 13](#_Toc178019667)

# 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 ε=
3. Să se precizeze rădăcina obținută cu exactitatea ε= utilizând
   * + metoda aproximațiilor succesive
     + metoda tangentelor (Newton)
     + metoda secantelor.
4. Să se compare rezultatele luând în considerație numărul de iterații, evaluările pentru funcția și derivată.

# Varianta: 23

## Funcția a

*Din faptul că avem o expresie >0 rezultă că pe acest interval există soluții.*



*;*

*;*

A graph with a line drawn on it

Description automatically generated

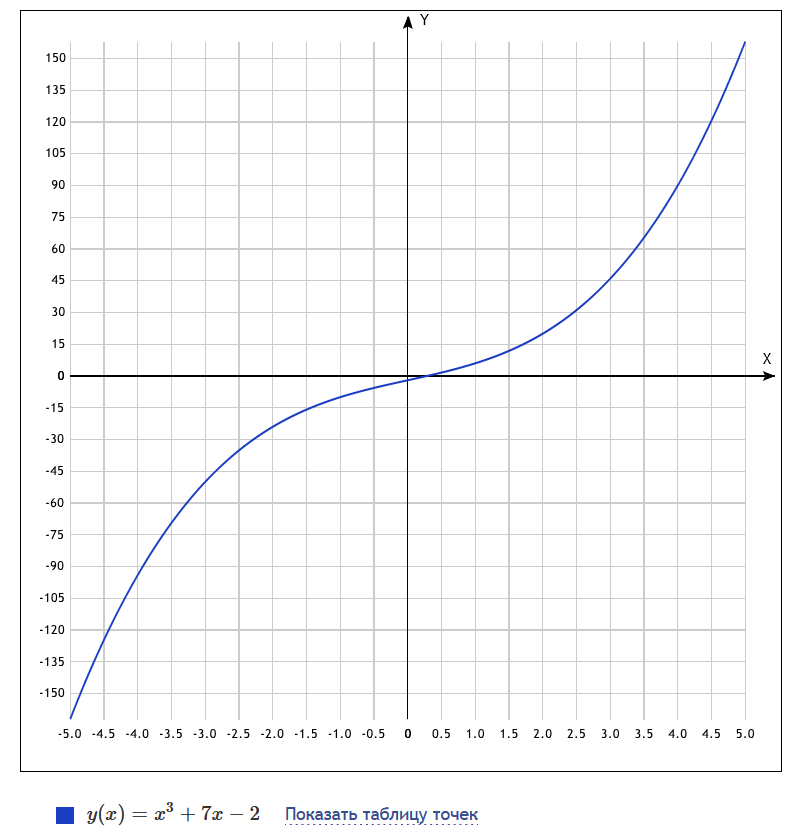
**Figura 1**:Graficul funcției lg(2x+3)+2x-1

## Funcția b

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Observăm că semnele diferă ( + și - )

Aproximare succesivă



**Figura 1**: Graficul funcției

# Codul

#include <stdio.h>

#include <math.h>

#define epsilon 0.0001

int numar\_iteratii = 0;

double functie(double x)

{

    return log10(2 \* x + 3) + 2 \* x - 1;

}

double functie\_iterata(double x)

{

    return (log10(2 \* x + 3) - 1) / -2;

}

double derivata\_functie(double x)

{

    return 2 + 2 / (2 \* x + 3);

}

double derivata\_doua\_functie(double x)

{

    return -2 / pow(2 \* x + 3, 2);

}

double metoda\_injumatatirii(double a, double b)

{

    double c;

    if (functie(a) \* functie(b) < 0)

    {

        while (fabs(b - a) > epsilon)

        {

            c = (a + b) / 2;

            numar\_iteratii++;

            printf("Pasul %d: a = %f, b = %f, c = %f\n", numar\_iteratii, a, b, c);

            if (functie(a) \* functie(c) < 0)

                b = c;

            else

                a = c;

        }

        return c;

    }

    else

    {

        return -1;

    }

}

double metoda\_aproximarilor(double a)

{

    double x = a, y;

    do

    {

        y = functie\_iterata(x);

        numar\_iteratii++;

        printf("Pasul %d: x = %f, y = %f\n", numar\_iteratii, x, y);

        if (fabs(y - x) <= epsilon)

            break;

        x = y;

    } while (1);

    return x;

}

double metoda\_newton(double a)

{

    double x = a, x1;

    do

    {

        x1 = x - functie(x) / derivata\_functie(x);

        numar\_iteratii++;

        printf("Pasul %d: x = %f, x1 = %f\n", numar\_iteratii, x, x1);

        if (fabs(x1 - x) <= epsilon)

            break;

        x = x1;

    } while (1);

    return x1;

}

double metoda\_secantelor(double x0, double x1)

{

    double x2;

    do

    {

        x2 = x1 - (functie(x1) \* (x1 - x0)) / (functie(x1) - functie(x0));

        numar\_iteratii++;

        printf("Pasul %d: x0 = %f, x1 = %f, x2 = %f\n", numar\_iteratii, x0, x1, x2);

        if (fabs(x2 - x1) <= epsilon)

            break;

        x0 = x1;

        x1 = x2;

    } while (1);

    return x2;

}

int main()

{

    double a = -1, b = 1;

    printf("Metoda injumatatirii intervalului:\n");

    double radacina\_injumatatire = metoda\_injumatatirii(a, b);

    if (radacina\_injumatatire != -1)

    {

        printf("Rădăcina: (%f, 0)\n", radacina\_injumatatire);

    }

    else

    {

        printf("Metoda injumatatirii nu a reusit.\n");

    }

    printf("Numar iteratii: %d\n", numar\_iteratii);

    numar\_iteratii = 0;

    printf("Metoda aproximarilor succesive:\n");

    printf("Rădăcina: %f\n", metoda\_aproximarilor(a));

    printf("Numar iteratii: %d\n", numar\_iteratii);

    numar\_iteratii = 0;

    printf("Metoda Newton:\n");

    printf("Rădăcina: %f\n", metoda\_newton(a));

    printf("Numar iteratii: %d\n", numar\_iteratii);

    numar\_iteratii = 0;

    printf("Metoda secantelor:\n");

    printf("Rădăcina: %f\n", metoda\_secantelor(a, b));

    printf("Numar iteratii: %d\n", numar\_iteratii);

    return 0;

}

A computer screen shot of a number

Description automatically generated

**Figura 2**: Rezultatul obținut a funcției lg(2x+3)+2x-1în program

# Codul

#include <stdio.h>

#include <math.h>

#define epsilon 0.0001

int numar\_iteratii = 0;

double functie(double x)

{

    return pow(x, 3) + 7 \* x - 2;

}

double derivata\_functie(double x)

{

    return 3 \* pow(x, 3) + 7;

}

double metoda\_injumatatirii(double a, double b)

{

    double c;

    if (functie(a) \* functie(b) < 0)

    {

        while (fabs(b - a) > epsilon)

        {

            c = (a + b) / 2;

            numar\_iteratii++;

            printf("Pasul %d: a = %f, b = %f, c = %f\n", numar\_iteratii, a, b, c);

            if (functie(a) \* functie(c) < 0)

                b = c;

            else

                a = c;

        }

        return c;

    }

    else

    {

        return -1;

    }

}

double metoda\_newton(double a)

{

    double x = a, x1;

    do

    {

        x1 = x - functie(x) / derivata\_functie(x);

        numar\_iteratii++;

        printf("Pasul %d: x = %f, x1 = %f\n", numar\_iteratii, x, x1);

        if (fabs(x1 - x) <= epsilon)

            break;

        x = x1;

    } while (1);

    return x1;

}

double metoda\_secantelor(double x0, double x1)

{

    double x2;

    do

    {

        x2 = x1 - (functie(x1) \* (x1 - x0)) / (functie(x1) - functie(x0));

        numar\_iteratii++;

        printf("Pasul %d: x0 = %f, x1 = %f, x2 = %f\n", numar\_iteratii, x0, x1, x2);

        if (fabs(x2 - x1) <= epsilon)

            break;

        x0 = x1;

        x1 = x2;

    } while (1);

    return x2;

}

int main()

{

    double a = -1, b = 1;

    printf("Metoda injumatatirii intervalului:\n");

    double radacina\_injumatatire = metoda\_injumatatirii(a, b);

    if (radacina\_injumatatire != -1)

    {

        printf("Rădăcina: (%f, 0)\n", radacina\_injumatatire);

    }

    else

    {

        printf("Metoda injumatatirii nu a reusit.\n");

    }

    printf("Numar iteratii: %d\n", numar\_iteratii);

    numar\_iteratii = 0;

    printf("Metoda Newton:\n");

    printf("Rădăcina: %f\n", metoda\_newton(a));

    printf("Numar iteratii: %d\n", numar\_iteratii);

    numar\_iteratii = 0;

    printf("Metoda secantelor:\n");

    printf("Rădăcina: %f\n", metoda\_secantelor(a, b));

    printf("Numar iteratii: %d\n", numar\_iteratii);

    return 0;

}

A screenshot of a computer screen

Description automatically generated

**Figura 3**: Rezultatul obținut a funcției 2 în program

# Concluzie

În urma efectuării acestei lucrări de laborator s-a determinat soluția ecuației transcendente f(x) =0, prin precizarea rădăcinii obținute cu exactitatea ε utilizând cele , metode: înjumătățirii intervalului, aproximărilor succesive, Newton și secantelor. În cazul ambelor ecuații metoda Newton s-a dovedit a fi mai eficientă, obținându-se o soluție exactă și cu o complexitatea în timp mai mică.