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

Varianta: 23 2

Codul $x3 + 7x - 2$: 3

Codul $lg(2x + 3) + 2x - 1$ 6

Concluzie 10

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=10^{-2}
3. Să se precizeze rădăcina obținută cu exactitatea ε=10^{-6} 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

$a)\ lg(2x+3)+2x-1$

$b)\ x^3+7x-2$

Ecuația x^3+7x-2 are două soluții pentru x, în intersecția valorilor [-0.9 , 0.9] , [-6.5 , 6.5].

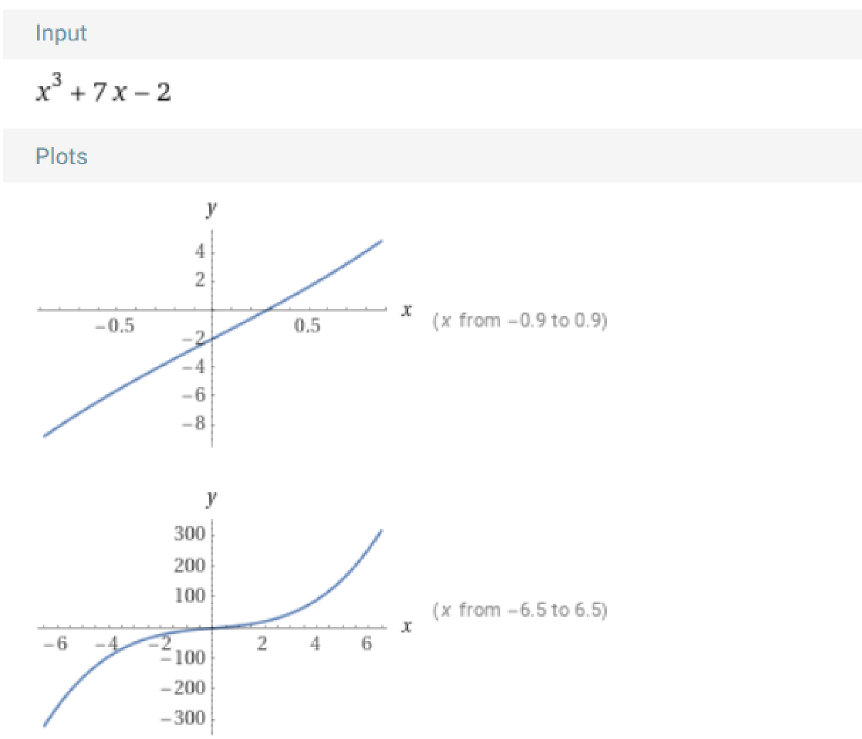

Figure 1 Graficul funcției x^3+7x-2 în Wolframalpha

Graficul funcției $lg(2x + 3) + 2x - 1$, se intersectează în punctul $lg(2x + 3)$și $- 2x + 1$.

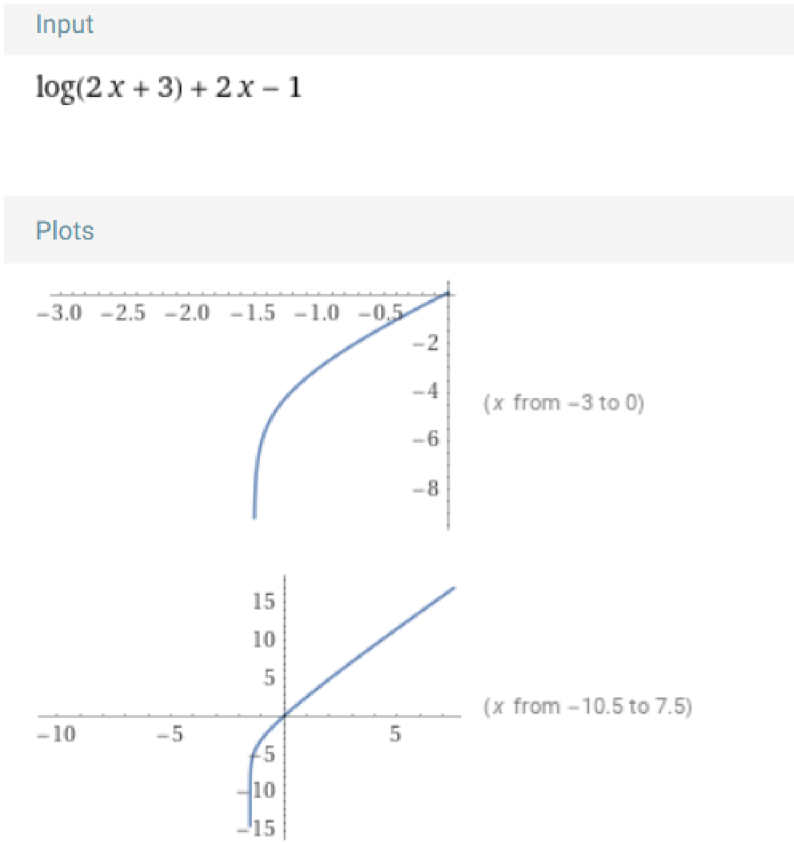

Figure 2: Graficul funcției lg(2x+3)+2x-1 în Wolframalpha

Codul $x^3 + 7x - 2$:

```
#include <stdio.h>
#include <math.h>

#define epsilon 0.0001

int count = 0;

// f(x) = x^3 + 7x - 2
double f(double x) {
    return pow(x, 3) + 7 * x - 2;
}
```

```
// g(x) = (x^3 - 2) / -7 (Modified as an iterative function approximation)
double fi(double x) {
    return (pow(x, 3) - 2) / -7;
}

// f'(x) = 3x^2 + 7
double fD(double x) {
    return 3 * pow(x, 2) + 7;
}

// f''(x) = 6x
double fDD(double x) {
    return 6 * x;
}

double injumatatire(double a, double b) {
    double c;
    if (f(a) * f(b) < 0) {
        while (fabs(b - a) > epsilon) {
            c = (a + b) / 2;
            count++;
            if (f(a) * f(c) < 0)
                b = c;
            else
                a = c;
        }
        return c;
    } else {
        return -1; // Return -1 if the method fails
    }
}

double aproximare(double a) {
    double x = a, y;
```

```
    do {
        y = fi(x);
        count++;
        if (fabs(y - x) <= epsilon)
            break;
        x = y;
    } while (1);
    return x;
}

double newton(double a) {
    double x = a, x1;
    do {
        x1 = x - f(x) / fD(x);
        count++;
        if (fabs(x1 - x) <= epsilon)
            break;
        x = x1;
    } while (1);
    return x1;
}

int main() {
    double a = -2, b = 2; // Adjusting the range to include the root
    printf("Metoda injumatatirii intervalului:\n");
    double rootInj = injumatatire(a, b);
    if (rootInj != -1) {
        printf("(%f, 0)\n", rootInj);
    } else {
        printf("Metoda injumatatirii nu a reusit.\n");
    }
    printf("Numar iteratii: %d\n", count);
    count = 0;

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

```
    printf("%f\n", aproximare(a));
    printf("Numar iteratii: %d\n", count);
    count = 0;

    printf("Metoda Newton:\n");
    printf("%f\n", newton(a));
    printf("Numar iteratii: %d\n", count);

    return 0;
}
```

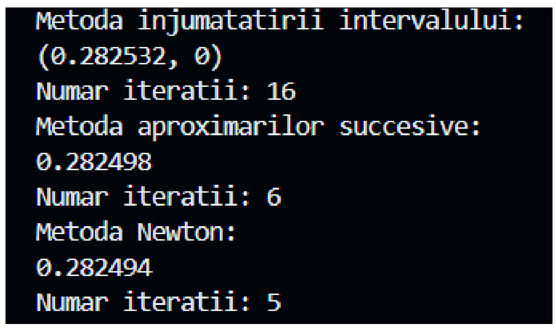

Figure 3: Rezultatul obținut a funcției x^3+7x-2 în program

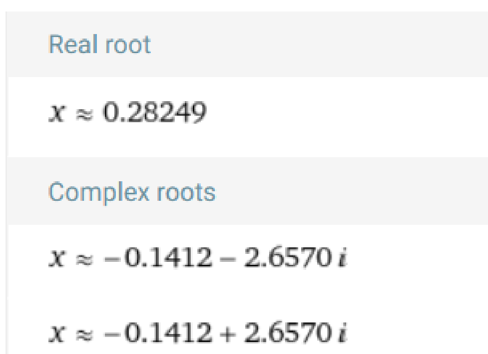

Figure 4: Rezultatul obținut a funcției x^3+7x-2 în Wolfram Alpha

Codul $lg(2x + 3) + 2x - 1$

```
#include <stdio.h>
#include <math.h>

#define epsilon 0.0001
```

```
int count = 0;

// f(x) = log(2x + 3) + 2x - 1
double f(double x) {
    return log10(2 * x + 3) + 2 * x - 1;
}

// g(x) = (log(2x + 3) - 1) / -2 (Modified as an iterative function approximation)
double fi(double x) {
    return (log10(2 * x + 3) - 1) / -2;
}

// f'(x) = 2 + 2 / (2x + 3)
double fD(double x) {
    return 2 + 2 / (2 * x + 3);
}

// f''(x) = -2 / (2x + 3)^2
double fDD(double x) {
    return -2 / pow(2 * x + 3, 2);
}

double injumatatire(double a, double b) {
    double c;
    if (f(a) * f(b) < 0) {
        while (fabs(b - a) > epsilon) {
            c = (a + b) / 2;
            count++;
            if (f(a) * f(c) < 0)
                b = c;
            else
                a = c;
        }
        return c;
```

```
    } else {
        return -1; // Return -1 if the method fails
    }
}

double aproximare(double a) {
    double x = a, y;
    do {
        y = fi(x);
        count++;
        if (fabs(y - x) <= epsilon)
            break;
        x = y;
    } while (1);
    return x;
}

double newton(double a) {
    double x = a, x1;
    do {
        x1 = x - f(x) / fD(x);
        count++;
        if (fabs(x1 - x) <= epsilon)
            break;
        x = x1;
    } while (1);
    return x1;
}

int main() {
    double a = -1, b = 1; // Adjusting the range to avoid invalid values for log
    printf("Metoda injumatatirii intervalului:\n");
    double rootInj = injumatatire(a, b);
    if (rootInj != -1) {
        printf("(%f, 0)\n", rootInj);
```

```
    } else {
        printf("Metoda injumatatirii nu a reusit.\n");
    }
    printf("Numar iteratii: %d\n", count);
    count = 0;

    printf("Metoda aproximarilor succesive:\n");
    printf("%f\n", aproximare(a));
    printf("Numar iteratii: %d\n", count);
    count = 0;

    printf("Metoda Newton:\n");
    printf("%f\n", newton(a));
    printf("Numar iteratii: %d\n", count);

    return 0;
}
```

```
Metoda injumatatirii intervalului:
(0.230408, 0)
Numar iteratii: 15
Metoda aproximarilor succesive:
0.230473
Numar iteratii: 6
Metoda Newton:
0.230407
Numar iteratii: 7
```

Figure 5: Rezultatul obținut a funcției lg(2x+3)+2x-1 în program

Numerical root

$x \approx 0.230410438973598\ldots$

Figure 6: Rezultatul obținut a funcției lg(2x+3)+2x-1 în Wolfram Alpha

Concluzie

În urma efectuării acestei lucrări de laborator s-a determinat soluția ecuației transcendente f(x) =0, prin precizezarea rădăcinei obţinute cu exactitatea ε utilizând cele 3=10-4, metode: înjumătățirii intervalului, aproximărilor succesive și Newton. 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.