Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра Систем Штучного Інтелекту

Звіт

до лабораторної роботи № 5

з дисципліни

Чисельні методи

на тему:

“ Методи розв’язування нелінійних рівнянь з однією змінною ”

Варіант №24(Метод ітерацій)

Виконав: студент КН-217

Ратушняк Денис

Прийняла: доцент каф. СШІ

Мочурад Л. І.

Львів – 2022

Мета роботи: набути навиків розв’язування нелінійних операторних рівнянь, зокрема, нелінійних алгебраїчних і трансцендентних рівнянь.

Завдання.

Розв’язати нелінійне рівняння за методом ітерацій. Необхідно розробити основну програму, виконуючу опис даних, ввід початкових даних, відзив підпрограм розв’язання нелінійного рівняння, вивід початкових даних і отриманих результатів. Відокремити корені в рівнянні, що досліджується. Вибрати значення точності обчислень ε. На кожному відрізку уточнити корені, користуючись розробленою програмою.

24	$\ln x + (x+1)^3 = 0$	0.187

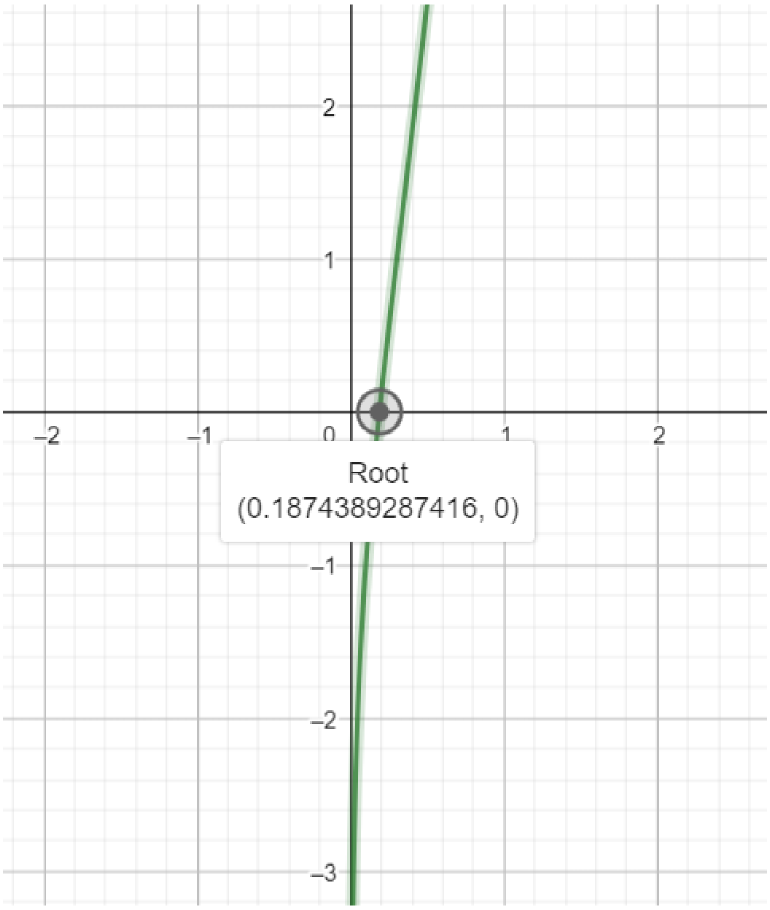
Програмна реалізація на мові програмування C++:

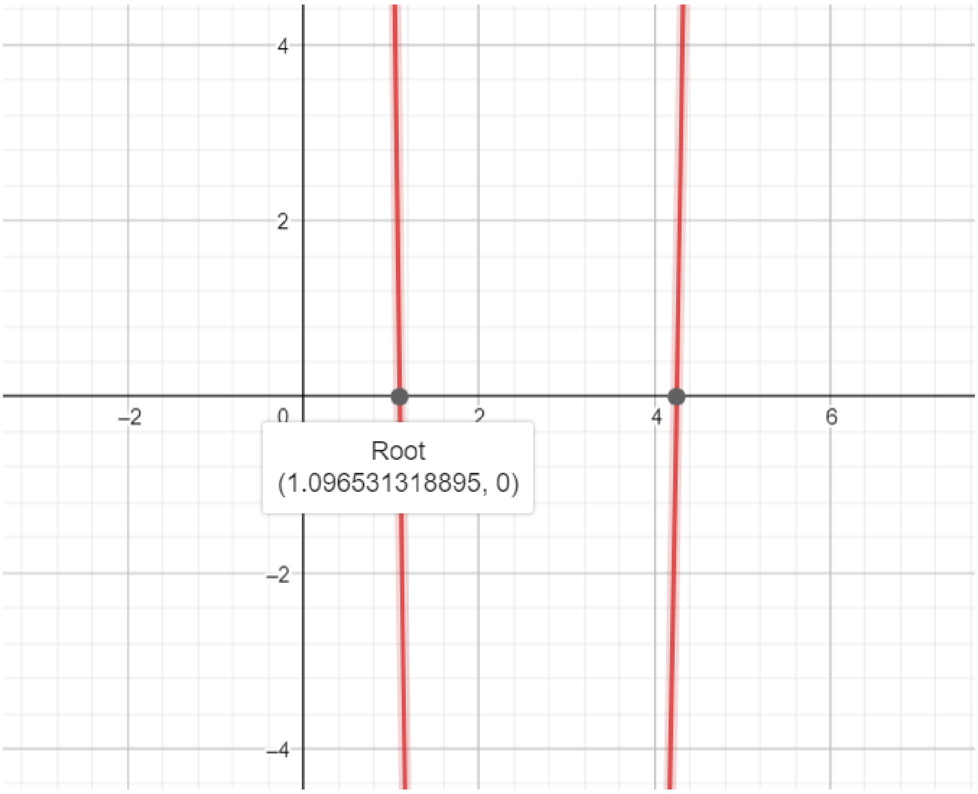
```
#include <bits/stdc++.h>
#include <windows.h>
using namespace std;

typedef long double ld;
typedef long long ll;
typedef vector< vector<ld > > matrix;
ld eps;

ld k1,k2,k3,k4,k5;
ll type;
ld f(ld x)
{
    if(type == 1) return log(x) + (x + 1) * (x + 1) * (x + 1);
    return k1 * log(x) + k2 * pow((k3 * x + k4), k5);
}

ld derivative(ld x)
{
    if(type == 1) return (1.0 / x) + 3 * (x + 1) * (x + 1);
    return k1 / x + k2 * k3 * k5 * pow((k3 * x + k4), k5 - 1);
}

ld F(ld x, ld m)
{
    return (x - m * f(x));
}

ll totalans = 0;


void find_ans(ld a, ld b){
    ld m = 1.0 / max(derivative(a), derivative(b));
    ld x0 = (a + b) / 2.0;
    vector<ld> x;
    x.emplace_back(x0);

    ld y;
    ll steps = 0;

    while(steps <= 10000000)
    {
```

```
            y = F(x.back(), m);
            steps ++;
            if(fabs(y - x.back()) < eps) break;
            x.emplace_back(y);
        }
        cout << fixed << setprecision(10) << "Found in range [" << a << ", " << b << " ] answer with " << steps << "
steps x = ";
        cout << fixed << setprecision(30) << x.back() << "\nf(x) = " << f(x.back()) << "\n";
        totalans++;
  }

  int main()
  {
        cout << "Input 1 for f(x) = ln(x) + (x + 1) ^ 3 and other number for f(x) = k1 * ln(x) + k2(k3 * x + k4) ^ k5\n";
        cin >> type;
        if(type == 1)
        {
            /// f(x) = ln(x) + (x + 1) ^ 3
            ld a,b,x0;
            cout << "Here you can input multiple test to solve equation ln(x) + (x + 1)^3 = 0\n";
            while(1)
            {
                cout << "Input range [a, b], then starting x0 (a <= x0 <= b) and eps\n";
                cin >> a >> b >> x0 >> eps;
                if(a > b)
                {
                    cout   << "a cannot be greater than b, try again\n";
                    continue;
                }

                if(a <= 0)
                {
                    cout   << "a cannot be less or equal than 0 due to ln(x), try again \n";
                    continue;
                }

                if(x0 < a || b < x0)
                {
                    cout   << "x0 must be in range [a, b] , try again\n";
                    continue;
                }
                if(f(a) * f(b) > 0)
                {
                    cout << "As function y = ln(x) + (x + 1)^3 is increasing and f(a) * f(b) > 0, there is no root in given
range [a, b] , try again\n";
                    continue;
                }

                ld m = 1.0 / max(derivative(a), derivative(b));

                vector<ld> x;
                x.emplace_back(x0);

                ld y;
                ll steps = 0;

                while(1)
                {
                    y = F(x.back(), m);
                    steps ++;
                    if(fabs(y - x.back()) < eps) break;
                    x.emplace_back(y);
```

```
        }
        cout << "Found answer with " << steps << " steps x = ";
        cout << fixed << setprecision(30) << x.back() << "\nf(x) = " << f(x.back()) << "\n";
    }


    return 0;
}

///f(x) = k1 * ln(x) + k2(k3 * x + k4) ^ k5

ld a,b,x0;
ll total_seg = 0;
cout << "Here you can input multiple test to solve equation k1 * ln(x) + k2(k3 * x + k4) ^ k5 = 0\n";
while(1)
{
    cout << "Input not zero k1, k2, k3, k4, k5\n";
    cin >> k1 >> k2 >> k3 >> k4 >> k5;
    totalans = 0;
    if(k1*k2*k3*k4*k5 == 0) {

        cout << "Not all k is zero, try again\n";
        continue;
    }

    cout << "\nInput range [a, b] where to find root(s), eps, and total segments for finding roots\n";
    cin >> a >> b >> eps >> total_seg;
    if(total_seg <= 0){
        cout   << "total segments must be greater than 0\n";
        continue;
    }
    if(a > b)
    {
        cout   << "a cannot be greater than b, try again\n";
        continue;
    }
    if(a <= 0)
    {
        cout   << "a cannot be less or equal than 0 due to ln(x), try again \n";
        continue;
    }

    ld len = (b - a) / total_seg;
    ld a1 = a;
    ld b1 = a1 + len;
    for(ll k = 0; k < total_seg; k++){
        if(f(a1) * f(b1) < 0) find_ans(a1, b1);
        a1 += len;
        b1 += len;
    }
    if(totalans == 0){
        cout << "No root found\n";
    }
}

/// 0.1 3 1 0.0001

/// -100 1 1 1 3
/// 0.1 5 0.00000000001 100

return 0;
}
```

Графіки протестованих функцій





Результати роботи програми

```
Input 1 for f(x) = ln(x) + (x + 1) ^ 3 and other number for f(x) = k1 * ln(x) + k2(k3 * x + k4) ^ k5
1
Here you can input multiple test to solve equation ln(x) + (x + 1)^3 = 0
Input range [a, b], then starting x0 (a <= x0 <= b) and eps
-3 3 0 0.0000001
a cannot be less or equal than 0 due to ln(x), try again
Input range [a, b], then starting x0 (a <= x0 <= b) and eps
1 3 0 0.0000001
x0 must be in range [a, b] , try again
Input range [a, b], then starting x0 (a <= x0 <= b) and eps
1 3 2 0.0000001
As function y = ln(x) + (x + 1)^3 is increasing and f(a) * f(b) > 0, there is no root in given range [a, b] , try again
Input range [a, b], then starting x0 (a <= x0 <= b) and eps
0.1 10 5 0.0000001
Found answer with 510 steps x = 0.18744262969235631840516070195
f(x) = 0.00003539994092461250761727509 2
Input range [a, b], then starting x0 (a <= x0 <= b) and eps
0.1 10 5 0.000000000000000001
Found answer with 1458 steps x = 0.1874389287305483183177423275 88
f(x) = 0.00000000000000000361256163872170
Input range [a, b], then starting x0 (a <= x0 <= b) and eps
0.1 1000 456 0.000000000000000001
Found answer with 9438911 steps x = 0.1874389287308613254249547649 95
f(x) = 0.00000000000029943069508250874 66
```

```
Input 1 for f(x) = ln(x) + (x + 1) ^ 3 and other number for f(x) = k1 * ln(x) + k2(k3 * x + k4) ^ k5
2
Here you can input multiple test to solve equation k1 * ln(x) + k2(k3 * x + k4) ^ k5 = 0
Input not zero k1, k2, k3, k4, k5
-100 1 1 1 3

Input range [a, b] where to find root(s), eps, and total segments for finding roots
0.1 1000 0.0000000000001 100000
Found in range [1.0899010000, 1.0999000000 ] answer with 6 steps x = 1.0965313188962666829037728710 72
f(x) = -0.0000000000000192327924419810614
Found in range [4.2495850000, 4.2595840000 ] answer with 6 steps x = 4.2498102140603754078504450397 70
f(x) = 0.0000000000001778827085630041438
Input not zero k1, k2, k3, k4, k5
-100 1 1 1 3

Input range [a, b] where to find root(s), eps, and total segments for finding roots
0.1 5 0.000001 100
Found in range [1.0800000000, 1.1290000000 ] answer with 4 steps x = 1.0965308548696881284006607315 54
f(x) = 0.0000361988789012940639922000 23
Found in range [4.2160000000, 4.2650000000 ] answer with 4 steps x = 4.2498102038320465486179178249 42
f(x) = -0.00000060501470634799492387401 2
```

Аналіз чисельних експериментів

Як і інші ітераційні методи метод ітерацій генерує проміжні результати і з кожним кроком їх уточнює відповідно до заданої точності та інших вхідних даних. Це за собою веде наявність проміжних результатів, що може бути корисним в багатьох випадках. Через можливість задання певних параметрів користувачем, можна фокусувати роботу програми на швидкодію, велику/малу точність, велику/малу кількість ітерацій.

Метод ітерацій збігається для довільного початкового значення x0∈[a, b]. Завдяки цьому він є самовиправляючим, тобто окрема помилка в обчисленнях не впливає на кінцевий результат, бо помилкове значення можна розглядати як нове початкове наближення x0. Ця властивість робить метод ітерацій **найнадійнішим методом** обчислень.

Метод ітерацій на малих вхідних даних працює швидко, точно, і виконує малу кількість ітерацій. Проте при збільшенні точності, проміжку, кількості відрізків у яких шукати корені, час його роботи збільшується, кінцева точність результатів значно зменшується.

Висновок

Я здобув на закріпив вміння та навички навиків розв’язування нелінійних рівнянь, зокрема, нелінійних алгебраїчних і трансцендентних рівнянь. Створив програму яка розв’язує рівняння ln(x) + (x+1)^3 == 0 на заданому проміжку з заданою точністю, а також яка розв’язує рівняння виду k1 * ln(x) + k2(k3 * x + k4) ^ k5 == 0, де k1*k2*k3*k4*k5 != 0.