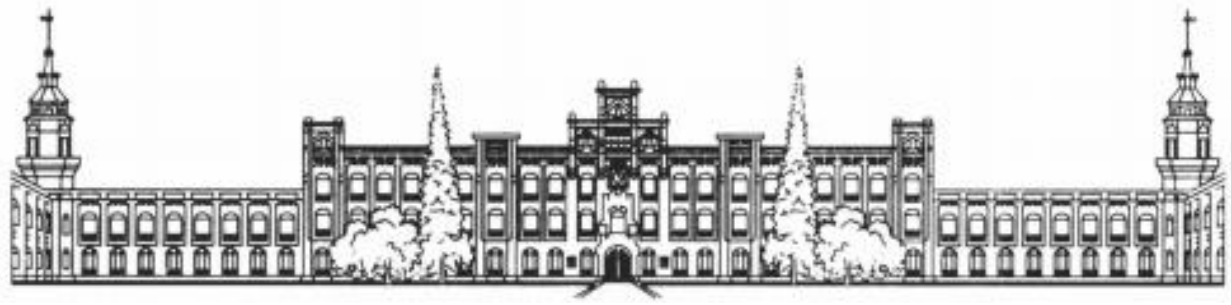

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря
Сікорського» Інститут прикладного системного аналізу

Лабораторна робота № 1

з курсу «Чисельні методи 1»

з теми «Методи розв'язування нелінійних алгебраїчних рівнянь»

Варіант № 16

Виконав студент 2 курсу групи КА-91
Панченко Єгор Станіславович
перевірила старший викладач
Хоменко Ольга Володимирівна

Київ-2021

Завдання: знайти корені рівняння $31 \cdot x^7 - 210 \cdot x^6 - 449 \cdot x^5 + 850 \cdot x^4 + 916 \cdot x^3 - 809 \cdot x^2 - 139 \cdot x + 25 = 0$, використовуючи чисельні методи.

1. **Теоретичні відомості**

   У даній роботі використовуються теореми з лекційного матеріалу, а саме Теореми 3 та 9 з Лекції 1. Також використовується критерій існування кратних коренів.

2. **Опис процесу відокремлення коренів**

   Для початку програма знаходить у якому проміжку [L, R] знаходяться модулі коренів (Лекція 1, Теорема 3). Для даного рівняння $L \approx 0.0265674814$ і $R \approx 30.5483870968$. Оскільки L мале, то для спрощення алгоритму задача розв'язувалася не на множині $[-R, -L] \cup [L, R]$, а на $[-R, R]$. Далі, за теоремою Штурма (Лекція 1, Теорема 9) будувалася послідовність $\{f_i\}_{i=0}^{n}$. Варто зазначити, що даний за умовою многочлен не має кратних коренів (у ході обчислення послідовності $\{f_i\}_{i=0}^{n}$ було встановлено, що похідна многочлена та сам многочлен не мають спільного многочлена дільника, ступеня більшого за нуль, тобто кратних коренів бути не може), тому метод Штурма можна застосовувати для пошуку усіх коренів. Програма йшла з кроком $step = 0.1$, і знаходила проміжки, де послідовність змінює знак різну кількість разів. Результатом даного кроку є:

   - Перший корінь належить інтервалу [-2.0483870968, -1.9483870968];
   - Другий корінь належить інтервалу [-1.5483870968, -1.4483870968];
   - Третій корінь належить інтервалу [-0.2483870968, -0.1483870968];
   - Четвертий корінь належить інтервалу [0.0516129032, 0.1516129032];
   - П'ятий корінь належить інтервалу [0.7516129032, 0.8516129032];
   - Шостий корінь належить інтервалу [1.4516129032, 1.5516129032];

- Сьомий корінь належить інтервалу [8.0516129032, 8.1516129032];

3. Текст програм

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <algorithm>
#include <deque>
#include <queue>
#include <map>
#include <set>
#include <cmath>
#include <ctime>
#include <random>

using namespace std;

const double eps = 1e-5;
vector < double > a = {25, -139, -809, 916, 850, -449, -210, 31};
vector < vector < double > > f;
double step = 0.1;
double L, R;
vector < vector < int > > Table;

void FindLR() {
    double A, B;
    A = fabs(a[0]);
    B = fabs(a.back());
    for (int i = 1; i + 1 < a.size(); i++) {
        A = max(A, fabs(a[i]));
        B = max(B, fabs(a[i]));
    }
    L = 1.0 / (1 + B / fabs(a[0]));
    R = 1 + A / fabs(a.back());
}

bool equal(double a, double b) {
    return fabs(a - b) <= eps;
}

int sgn(double x) {
    return (equal(x, 0) ? 0 : (x > 0 ? 1 : -1));
}

double calculate(vector < double > P, double x) {
    double res = 0;
    for (int i = 0; i < P.size(); i++)
        res += P[i] * pow(x, i);
    return res;
}

vector < double > Mod(vector < double > P, vector < double > Q) {
    while (Q.size() <= P.size()) {
```

```
        int deg = (int)P.size() - Q.size();
        double coefficient = P.back() / Q.back();
        for (int i = 0; i < Q.size(); i++) {
            P[i + deg] -= coefficient * Q[i];
        }
        while (!P.empty() && equal(P.back(), 0))
            P.pop_back();
    }
    if (P.empty()) {
        return {0};
    }
    return P;
}

vector < double > Derivative(vector < double > P) {
    for (int i = 0; i + 1 < (int)P.size(); i++) {
        P[i] = (i + 1) * P[i + 1];
    }
    P.pop_back();
    return P;
}

void Print(vector < double > P) {
    printf("Pol: ");
    for (int i = (int)P.size() - 1; i >= 0; i--) {
        if (P[i] >= 0 && i + 1 != P.size()) {
            printf("+");
        }
        printf("%0.2lf", P[i]);
        if (i != 0) {
            printf("*x^%d ",i);
        }
    }
    printf("\n");
}

void MultyBy(vector < double > &P, double c) {
    for (auto &i : P) {
        i *= c;
    }
}

void FindRootByBinaryDividing(vector < double > P, double l, double r) {
    bool isinc = calculate(P, r) > calculate(P, l);
    while (r - l > eps) {
        printf("Interval is [%0.10lf, %0.10lf] and values are [%0.10lf, %0.10lf]\n",
                l, r, calculate(P, l), calculate(P, r));
        double m = (r + l) / 2;
        double x = calculate(P, m);
        if (equal(x, 0)) {
            l = m;
            break;
        }
        (((x > 0 && isinc) || (x < 0 && !isinc)) ? r : l) = m;
    }
    printf("The root found by bin. method is %0.10lf\n", l);
}
```

```
void FindRootByChord(vector < double > P, double l, double r) {
    int sign = sgn(calculate(Derivative(Derivative(P)), l));
    bool moveleft = ((sign < 0 && calculate(P, l) > 0) ||
            (sign > 0 && calculate(P, l) < 0));
    vector < double > x;
    for (/*r - l > eps*/ /*int k = 30; k; k--*/; ;) {
        printf("Interval is [%0.10lf, %0.10lf] and values are [%0.10lf, %0.10lf]\n",
               l, r, calculate(P, l), calculate(P, r));
        double c = l - calculate(P, l) * (r - l) / (calculate(P, r) - calculate(P,
l));
        if (!x.empty() && (fabs(x.back() - c) < eps || fabs(calculate(P, x.back()))
< eps)) {
            break;
        }
        x.push_back(c);
        (moveleft ? l : r) = c;
    }
    printf("The root found by crd. method is %0.10lf\n",
            /*l - calculate(P, l) * (r - l) / (calculate(P, r) - calculate(P, l))*/
x.back());

}

void FindRootByNewton(vector < double > P, double l, double r) {
    vector < double > x(1, (calculate(P, l) > 0 ? l : r));
    for (; ;) {
        printf("Point is [%0.10lf] and value is [%0.10lf]\n",
               x.back(), calculate(P, x.back()));
        double c = x.back() - calculate(P, x.back()) / calculate(Derivative(P),
x.back());
        if (fabs(x.back() - c) < eps || fabs(calculate(P, x.back())) < eps) {
            break;
        }
        x.push_back(c);
    }
    printf("The root found by ntw. method is %0.10lf\n",
            /*l - calculate(P, l) * (r - l) / (calculate(P, r) - calculate(P, l))*/
x.back());
}

void MakeTable(double l, double r) {
    Table.resize(f.size());
    vector < int > changes;
    for (double val = l; val <= r; val += step) {
        for (int i = 0; i < Table.size(); i++) {
            Table[i].push_back(sgn(calculate(f[i], val)));
        }
        changes.push_back(0);
        for (int i = 0; i + 1 < Table.size(); i++) {
            changes.back() += Table[i].back() * Table[i + 1].back() == -1;
        }
    }

    for (double val = l, i = 0; (int)i + 1 < changes.size(); val += step, i += 1) {
        if (changes[i] - changes[i + 1] == 1) {
            printf("There is only one root in [%0.10lf, %0.10lf]\n", val, val +
```

```
step);
            FindRootByBinaryDividing(a, val, val + step);
            FindRootByChord(a, val, val + step);
            FindRootByNewton(a, val, val + step);
        }
    }
}

int main() {
    FindLR();
    printf("L and R are [%0.10lf] and [%0.10lf]\n", L, R);
    f.push_back(a);
    f.push_back(Derivative(a));
    while (true) {
        auto x = Mod(f[f.size() - 2], f.back());
        MultyBy(x, -1);
        f.push_back(x);
        if (x.size() == 1) {
            break;
        }
    }
    /*for (auto i : f) {
        Print(i);
    }*/
    MakeTable(-R, R);
}
```

4. Результат роботи програм

```
C:\Users\panen\CLionProjects\NumericalMethods\cmake-build-debug\NumericalMethods.exe
L and R are [0.0265674814] and [30.5483870968]
There is only one root in [-2.0483870968, -1.9483870968]
Interval is [-2.0483870968, -1.9483870968] and values are [-4.3361768020, 513.2970039855]
Interval is [-2.0483870968, -1.9983870968] and values are [-4.3361768020, 307.2949185597]
Interval is [-2.0483870968, -2.0233870968] and values are [-4.3361768020, 165.9685665608]
Interval is [-2.0483870968, -2.0358870968] and values are [-4.3361768020, 84.6074944393]
Interval is [-2.0483870968, -2.0421370968] and values are [-4.3361768020, 41.1051420490]
Interval is [-2.0483870968, -2.0452620968] and values are [-4.3361768020, 18.6295945030]
Interval is [-2.0483870968, -2.0468245968] and values are [-4.3361768020, 7.2083315798]
Interval is [-2.0483870968, -2.0476058468] and values are [-4.3361768020, 1.4515262994]
Interval is [-2.0479964718, -2.0476058468] and values are [-1.4384576118, 1.4515262994]
Interval is [-2.0479964718, -2.0478011593] and values are [-1.4384576118, 0.0075005768]
Interval is [-2.0478988155, -2.0478011593] and values are [-0.7152368747, 0.0075005768]
Interval is [-2.0478499874, -2.0478011593] and values are [-0.3538077488, 0.0075005768]
Interval is [-2.0478255733, -2.0478011593] and values are [-0.1731384873, 0.0075005768]
Interval is [-2.0478133663, -2.0478011593] and values are [-0.0828151807, 0.0075005768]
The root found by bin. method is -2.0478072628
Interval is [-2.0483870968, -1.9483870968] and values are [-4.3361768020, 513.2970039855]
Interval is [-2.0483870968, -2.0475494038] and values are [-4.3361768020, 1.8684727343]
Interval is [-2.0483870968, -2.0478016673] and values are [-4.3361768020, 0.0037421788]
The root found by crd. method is -2.0478016673
Point is [-1.9483870968] and value is [513.2970039855]
Point is [-2.1090455299] and value is [-555.3055403652]
Point is [-2.0579010299] and value is [-77.3300882699]
Point is [-2.0481363110] and value is [-2.4749183067]
Point is [-2.0478025548] and value is [-0.0028237697]
The root found by ntw. method is -2.0478025548
```

5. Висновок

У ході виконання лабораторної роботи було програмно реалізовано методи виділення коренів (знаходження інтервалу, якому належать всі корені; знаходження усіх інтервалів, де знаходиться рівно один корінь) та методи знаходження кореня на проміжку (методи половинного ділення, хорд та дотичних).