**Лабораторная работа**

**«Методы простой итерации и Ньютона**

**решения нелинейных уравнений»**

**Арсений Анищенко**

**Группа 2**

1. **Постановка задачи.**

Вычислить вещественные корни собственного многочлена четвертой степени *P*(λ), полученного из канонической формы Фробениуса лабораторной работы «Метод Данилевского».

1. Произвести отделение корней многочлена *P*(λ).

1.1. Для определения промежутков монотонности функции *P*(λ) решить уравнение *P'*(λ)=0:

*а)* методом простой итерации,

*б)* методом Ньютона.

Предварительно произвести отделение корней многочлена *P'*(λ).

Если требуется, при отделении корней многочлена *P'*(λ) найти промежутки монотонности функции *P'*(λ): решить квадратное уравнение *P''*(λ)=0.

1.2. (Задание дополнительное, для повышения оценки текущей успеваемости.) Если отделения корней не произошло, то вещественных корней многочлен *P*(λ) не имеет. Тогда два (наибольших по модулю) корня образуют комплексно-сопряженную пару и имеет место Случай 4 файла «Степенной метод». Получить собственные значения и соответствующие им собственные векторы согласно теории Случая 4. Проверить, что полученные собственные значения являются корнями собственного многочлена *P*(λ).

2. Приближенно вычислить вещественные корни собственного многочлена:

*а)* применить метод простой итерации,

*б)* применить метод Ньютона.

3. (Задание дополнительное, для повышения оценки текущей успеваемости.) Хотя бы для одного найденного собственного значения (корня собственного многочлена): с помощью матриц *M*3, *M*2, *M*1, полученных в лабораторной работе «Метод Данилевского», найти соответствующий собственный вектор *u*. Проверить выполнение (приближенное) равенства *Au*–λ*u*=0.

1. **Входные данные.**

**x^4 +124.806x^3 +156.095x^2 -178174x^1 +2.34203e+006x^0**

1. **Листинг программы.**

#include<iostream>

#include<math.h>

#include<algorithm>

using namespace std;

void func(float\*a,float\*b,float\* c,float\* res11,float\* res12,float\* res21,float\* res22,float\* xy) {

for (int i = 0; i < 4; ++i)

b[i] = a[i] \* (4 - i);

for (int i = 0; i < 3; ++i)

c[i] = b[i] \* (3 - i);

float x1, x2;

int A = c[0];

int B = c[1];

int C = c[2];

float D = B\*B - 4 \* A\*C;

if (D > 0) {

x1 = (-B + sqrt(D)) / (2 \* A);

x2 = (-B - sqrt(D)) / (2 \* A);

}

else

return;

xy[0] = x1;

xy[1] = x2;

//Нахождение корней P'(λ)

//Метод простой итерации

float x = min(x1, x2) - 50;

for (int i = 0; i < 1000; ++i) {

x = pow(fabs((b[1] \* x \* x + b[2] \* x + b[3]) / b[0]), 1.0 / 3);

if (-(b[1] \* x\*x + b[2] \* x + b[3]) / b[0] < 0)

x = -x;

}

res11[0] = x;

x = min(x1,x2)+(max(x1, x2) - min(x1, x2)) / 2;

for (int i = 0; i < 10000; i++) {

x = pow(fabs((b[0] \*x\* x\*x + b[2] \* x + b[3]) / b[1]), 1.0 / 2);

if (-(b[0] \* x\* x\*x + b[2] \* x + b[3]) / b[1]<0)

x = -x;

}

res11[1] = x;

x = max(x1, x2) + 50;

for (int i = 0; i < 10000; ++i) {

x = pow(fabs((b[0] \* x\* x\*x + b[2] \* x + b[3]) / b[1]), 1.0 / 2);

if (-(b[0] \* x\* x\*x + b[2] \* x + b[3]) / b[1]<0)

x = -x;

}

res11[2] = x;

//Метод Ньютона

x = min(x1, x2) - 100;

for (int i = 0; i < 1000; ++i) {

x = x - (b[0] \* x\*x\*x + b[1] \* x\*x + b[2] \* x + b[3]) / (b[0] \* 3 \* x\*x + b[1] \* 2 \* x + b[2]);

}

res21[0] = x;

x = min(x1, x2) + (max(x1, x2) - min(x1, x2)) / 2;

for (int i = 0; i < 10000; ++i) {

x = x - (b[0] \* x\*x\*x + b[1] \* x\*x + b[2] \* x + b[3]) / (b[0] \* 3 \* x\*x + b[1] \* 2 \* x + b[2]);

}

res21[1] = x;

x = max(x1, x2) + 100;

for (int i = 0; i < 10000; ++i) {

x = x - (b[0] \* x\*x\*x + b[1] \* x\*x + b[2] \* x + b[3]) / (b[0] \* 3 \* x\*x + b[1] \* 2 \* x + b[2]);

}

res21[2] = x;

//Нахождение вещественных корней собственного многочлена четвертой степени P(λ)

//Метод простой итерации

float temp = res11[0];

x = res11[0]-100;

for (int i = 0; i < 100; ++i) {

x = pow(fabs((a[1] \*x\*x\*x + a[2]\*x\*x + a[3]\*x+a[4]) / a[0]), 1.0 / 4);

if (-(a[1] \* x\*x\*x + a[2] \* x\*x + a[3] \* x + a[4]) / a[0]< 0)

x = -x;

}

res12[0] = x;

x = (res11[1] -temp) / 2 + temp;

for (int i = 0; i < 100; ++i) {

x = pow(fabs((a[0] \* x\* x\* x\*x + a[2] \* x \* x + a[3] \* x + a[4]) / a[1]), 1.0 / 3);

if (-(a[0] \* x \* x \* x\*x + a[2] \* x \* x + a[3] \* x + a[4]) / a[1]<0)

x = -x;

}

temp = res11[1];

res12[1] = x;

x = (res11[2] - temp) / 2 + temp;

for (int i = 0; i < 100; ++i) {

x = pow(fabs((a[0] \* x\* x\* x\*x + a[2] \* x\* x + a[3] \* x + a[4]) / a[1]), 1.0 / 3);

if (-(a[0] \* x\* x\* x\*x + a[2] \* x\* x + a[3] \* x + a[4]) / a[1]<0)

x = -x;

}

temp = res11[2];

res12[2] = x;

x = temp + 100;

for (int i = 0; i < 100; ++i) {

x = pow(fabs((a[0] \* x\* x\* x\*x + a[2] \* x\* x + a[3] \* x + a[4]) / a[1]), 1.0 / 3);

if (-(a[0] \* x\* x\* x\*x + a[2] \* x\* x + a[3] \* x + a[4]) / a[1]<0)

x = -x;

}

res12[3] = x;

//Метод Ньютона

temp = res21[0];

x = res21[0] - 100;

for (int i = 0; i < 100; ++i) {

x = x - (a[0]\*x \* x\*x\*x + a[1]\*x \* x\*x + a[2]\*x \* x + a[3]\*x+a[4]) / (b[0] \* x\*x\*x + b[1] \* x\*x + b[2] \* x + b[3]);

}

res22[0] = x;

x = (res21[1] - temp) / 2 + temp;

for (int i = 0; i < 100; ++i) {

x = x - (a[0] \* x \* x\*x\*x + a[1] \* x \* x\*x + a[2] \* x \* x + a[3] \* x + a[4]) / (b[0] \* x\*x\*x + b[1] \* x\*x + b[2] \* x + b[3]);

}

temp = res21[1];

res22[1] = x;

x = (res21[2] - temp) / 2 + temp;

for (int i = 0; i < 100; ++i) {

x = x - (a[0] \* x \* x\*x\*x + a[1] \* x \* x\*x + a[2] \* x \* x + a[3] \* x + a[4]) / (b[0] \* x\*x\*x + b[1] \* x\*x + b[2] \* x + b[3]);

}

temp = res21[2];

res22[2] = x;

x = temp + 100;

for (int i = 0; i < 100; ++i) {

x = x - (a[0] \* x \* x\*x\*x + a[1] \* x \* x\*x + a[2] \* x \* x + a[3] \* x + a[4]) / (b[0] \* x\*x\*x + b[1] \* x\*x + b[2] \* x + b[3]);

}

res22[3] = x;

}

int main() {

setlocale(LC\_ALL, ".1251");

float \*a = new float[5];

a[0] = 1;

a[1] = 124.8058;

a[2] = 156.095;

a[3] = -178174.0625;

a[4] = 2342025.5;

float \*b = new float[4];

float \*c = new float[4];

float \*res11 = new float[4];

float \*res12 = new float[4];

float \*res21 = new float[4];

float \*res22 = new float[4];

float \*xy = new float[2];

func(a, b, c, res11, res12, res21, res22, xy);

//Входные данные

cout << "x^4";

for (int i = 0; i < 4; ++i){

cout << ' ';

if (a[i+1] >= 0)

cout << "+";

cout << a[i+1] << "x^" << 3-i;

if (i == 3)

cout << endl;

}

cout << endl;

//Выходные данные

cout << "The equation P''(x): ";

for (int i = 0; i < 3; i++)

{

cout << c[i] << "x^" << 2 - i;

if (i!= 2&&c[i+1] >= 0)

cout << "+";

}

cout << endl;

cout << "The roots of equation P''(x): ";

cout << xy[0] << " " << xy[1] << endl << endl;

cout << "The equation P'(x): ";

for (int i = 0; i < 4; ++i) {

cout << b[i] << "x^" << 3 - i;

if (i != 3 && b[i+1] >= 0)

cout << '+';

}

cout << endl;

cout << "The roots of equation P'(x):" << endl;

cout << "Simple iteration method: ";

for (int i = 0; i < 3; ++i)

cout << res11[i] << " ";

cout << endl;

cout << "Newton's method: ";

for (int i = 0; i < 3; ++i)

cout << res21[i] << " ";

cout << endl << endl;

cout << "The real roots of equation P(x):" << endl;

cout << "Simple iteration method: ";

for (int i = 0; i < 4; ++i)

cout << res12[i] << " ";

cout << endl;

cout << "Newton's method: ";

for (int i = 0; i < 4; ++i)

cout << res22[i] << " ";

return 0;

}

1. **Выходные данные**

The equation P''(x): 12x^2+748.835x^1+312.19x^0

The roots of equation P''(x): -0.419941 -61.9134

The equation P'(x): 4x^3+374.417x^2+312.19x^1-178174x^0

The roots of equation P'(x):

Simple iteration method: -86.7919 19.5028 19.5028

Newton's method: -86.7918 -26.3153 19.5028

The real roots of equation P(x):

Simple iteration method: -105.217 -58.5844 20.0555 20.0544

Newton's method: -105.217 -58.5875 18.9442 20.055

**5. Выводы**

В ходе лабораторной работы были вычислены вещественные корни собственного многочлена четвертой степени P(λ), полученного из канонической формы Фробениуса лабораторной работы «Метод Данилевского». В результате многочлен имеет всего 4 вещественных корня.