Московский авиационный институт
(национальный исследовательский университет)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Лабораторные работы по курсу «Численные методы»

Студент:	С. П. Сабурова
Преподаватель:	Д. Е. Пивоваров
Группа:	М8О-303Б-21
Дата:	
Оценка:	
Подпись:	

Москва, 2024

2.1 Методы простой итерации и Ньютона

1 Постановка задачи

Реализовать методы простой итерации и Ньютона решения нелинейных уравнений в виде программ, задавая в качестве входных данных точность вычислений. С использованием разработанного программного обеспечения найти положительный корень нелинейного уравнения (начальное приближение определить графически). Проанализировать зависимость погрешности вычислений от количества итераций.

Вариант: 18
$ln(x+1) - x^3 + 1 = 0$

2 Результаты работы

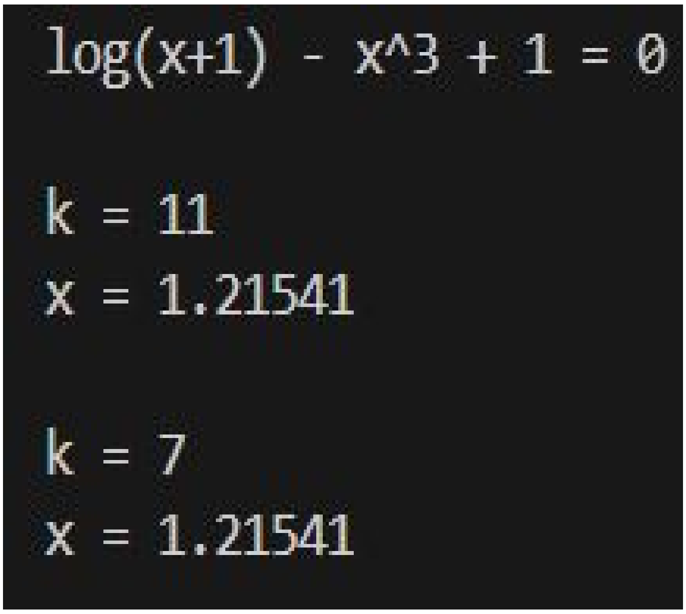

Рис. 1: Вывод программы в консоли

2.2 Методы простой итерации и Ньютона

3 Постановка задачи

Реализовать методы простой итерации и Ньютона решения систем нелинейных уравнений в виде программного кода, задавая в качестве входных данных точность вычислений. С использованием разработанного программного обеспечения решить систему нелинейных уравнений (при наличии нескольких решений найти то из них, в котором значения неизвестных являются положительными); начальное приближение определить графически. Проанализировать зависимость погрешности вычислений от количества итераций.

Вариант: 18

$$\begin{cases} 4x_1 - \cos x_2 = 0 \\ 4x_2 - e^{x_1} = 0 \end{cases}$$

4 Результаты работы

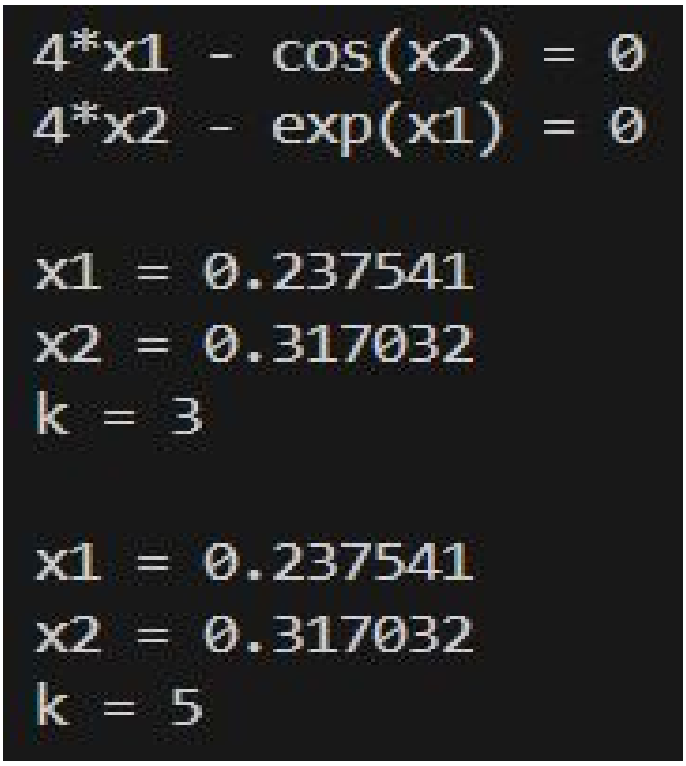

Рис. 2: Вывод программы в консоли

5 Исходный код

```
#include <bits/stdc++.h>

using namespace std;
using func_matrix = vector<vector<function<double(double, double)>>>;
using matrix = vector<double>;

pair<int, double> iteration_solution(double x_start, double q, double e){
    double r = x_start, l = x_start*5;
    int n=0;

    while (q*abs(r - l)/(1-q) >= e){
        n++;
        l = r;
        r = pow(log(r+1) + 1, 1.0/3);
    }
    return {n, r};
}

pair<int, double> newton_solution(double x_start, double e){
    double r = x_start - (log(x_start+1) - pow(x_start, 3) + 1)/(1/(x_start+1) - 3*pow(
        x_start, 2)), l = x_start;
    int n=0;

    while (abs(r - l) >= e){
        n++;
        l = r;
        r = r - (log(r+1) - pow(r, 3) + 1)/(1/(r+1) - 3*pow(r, 2));
    }
    return {n, r};
}

double eps(const matrix& z1, const matrix& z2, double q) {
    double d = 0.0;
    for(int i = 0; i < z1.size(); i++)
        d = max(d, abs(z1[i] - z2[i]));
    if (q == -1)
        return d;
    return q*d/(1-q);
}

double det(double x1, double x2, func_matrix& m) {
    return m[0][0](x1, x2) * m[1][1](x1, x2) - m[0][1](x1, x2) * m[1][0](x1, x2);
}

tuple<double, double, int> newton_system_solution(double x_start_1, double x_start_2,
    double e) {
```

```
    func_matrix jacobian = {{[](double x1, double x2) {return -exp(x1);}, [](double x1,
         double x2) {return 4;}}, {[](double x1, double x2) {return 4;}, [](double x1,
        double x2) {return sin(x2);}}};
    func_matrix first_A = {{[](double x1, double x2) {return 4*x2 - exp(x1);}, [](
        double x1, double x2) {return 4;}}, {[](double x1, double x2) {return 4*x1 -
        cos(x2);}, [](double x1, double x2) {return sin(x2);}}};
    func_matrix second_A = {{[](double x1, double x2) {return -exp(x1);}, [](double x1,
         double x2) {return 4*x2 - exp(x1);}}, {[](double x1, double x2) {return 4;},
        [](double x1, double x2) {return 4*x1 - cos(x2);}}};

    int n = 0;
    double x_r_1, x_r_2, x_l_1 = x_start_1, x_l_2 = x_start_2;
    x_r_1 = x_start_1 - det(x_start_1, x_start_2, first_A)/det(x_start_1, x_start_2,
        jacobian);
    x_r_2 = x_start_2 - det(x_start_1, x_start_2, second_A)/det(x_start_1, x_start_2,
        jacobian);

    while (eps({x_l_1, x_l_2}, {x_r_1, x_r_2}, -1) >= e){
        n += 1;
        x_l_1 = x_r_1;
        x_l_2 = x_r_2;
        x_r_1 = x_r_1 - det(x_r_1, x_r_2, first_A)/det(x_r_1, x_r_2, jacobian);
        x_r_2 = x_r_2 - det(x_r_1, x_r_2, second_A)/det(x_r_1, x_r_2, jacobian);
    }

    return {x_r_1, x_r_2, n};
}

tuple<double, double, int> iteration_system_solution(double x_start_1, double
    x_start_2, double q, double e) {
    int n = 0;
    double x_r_1 = x_start_1, x_r_2 = x_start_2, x_l_1 = x_start_1*5 + 1, x_l_2 =
        x_start_2*5 + 1;

    while (eps({x_l_1, x_l_2}, {x_r_1, x_r_2}, q) >= e){
        n += 1;
        x_l_1 = x_r_1;
        x_l_2 = x_r_2;
        x_r_1 = cos(x_r_2)/4;
        x_r_2 = exp(x_r_1)/4;
    }

    return {x_r_1, x_r_2, n};
}

int main(){
    pair<int, double> newton_algo = newton_solution(0.5, 0.000001), iteration_algo =
        iteration_solution(0.5, 0.5, 0.000001);
```

```
    cout << "-------------------------------" << endl << "log(x+1) - x^3 + 1 = 0" <<
        endl << endl << "k = " << newton_algo.first << endl << "x = " << newton_algo.
        second << endl << endl << "k = " << iteration_algo.first << endl << "x = " <<
        iteration_algo.second << endl << endl;

    double x_1, x_2;
    int n;
    cout << "-------------------------------" << endl << "4*x1 - cos(x2) = 0" << endl
        << "4*x2 - exp(x1) = 0" << endl;
    tie(x_1, x_2, n) = newton_system_solution(-2.0, 0.0, 0.000001);
    cout << endl << "x1 = " << x_1 << endl << "x2 = " << x_2 << endl << "k = " << n <<
        endl;
    tie(x_1, x_2, n) = iteration_system_solution(0.2, 0.2, 0.5, 0.000001);
    cout << endl << "x1 = " << x_1 << endl << "x2 = " << x_2 << endl << "k = " << n <<
        endl << "-------------------------------" << endl;

    return 1;
}
```