

**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: Ю. В. Кон  
Преподаватель: Д. Е. Пивоваров  
Группа: М8О-303Б-21  
Дата:  
Оценка:  
Подпись:

**Москва, 2024**

## 2.1 Методы простой итерации и Ньютона

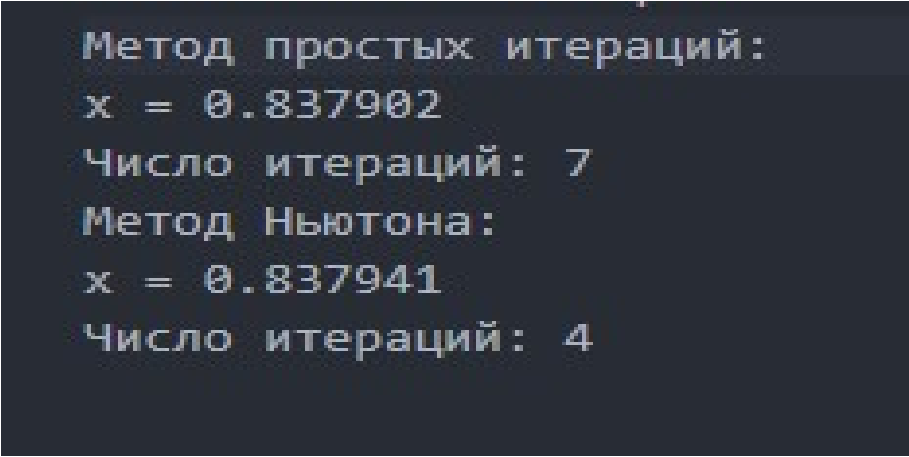
### 1 Постановка задачи

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: 12**

$$3^x - 5x^2 + 1 = 0$$

### 2 Результаты работы



```
Метод простых итераций:  
x = 0.837902  
Число итераций: 7  
Метод Ньютона:  
x = 0.837941  
Число итераций: 4
```

Рис. 1: Вывод программы в консоли

### 3 Исходный код

Файл с первым заданием второй лабораторной работы:

```
1  #include <iostream>
2  #include <cmath>
3  #include <fstream>
4
5  using namespace std;
6
7  double f(double x) {
8      return pow(3, x) - 5*x*x + 1;
9  }
10
11 double derivative(double x) {
12     return log(3)*pow(3, x)-10*x;
13 }
14
15 double phi(double x){
16     return sqrt((pow(3, x)+1)/5);
17 }
18
19 double simple_iterations(double x_0, double eps, int &iter){
20     double x_i, x;
21     x = x_0;
22     for (int i = 0; i < 10000; i++) {
23         x_i = phi(x);
24         if (fabs(x_i - x) < eps)
25             return x_i;
26         x = x_i;
27         iter++;
28     }
29     return x;
30 }
31
32 double newton(double x_0, double eps, int &iter){
33     double x;
34     x = x_0;
35     while (fabs(f(x)) > eps && iter < 1000) {
36         x = x - f(x) / derivative(x);
37         iter++;
38     }
39     return x;
40 }
41
42 int main(){
43     double x_0 = 0.5;
44     double eps = 0.0001;
45     int iterations = 0;
46     ofstream fout("output.txt");
```

```

47 | double answer = simple_iterations(x_0, eps, iterations);
48 | fout << " : " << endl;
49 | fout << "x = " << answer << endl;
50 | fout << " : " << iterations << endl;
51 | iterations = 0;
52 | answer = newton(x_0, eps, iterations);
53 | fout << " : " << endl;
54 | fout << "x = " << answer << endl;
55 | fout << " : " << iterations << endl;
56 | return 0;
57 | }

```

## 2.2 Методы простой итерации и Ньютона

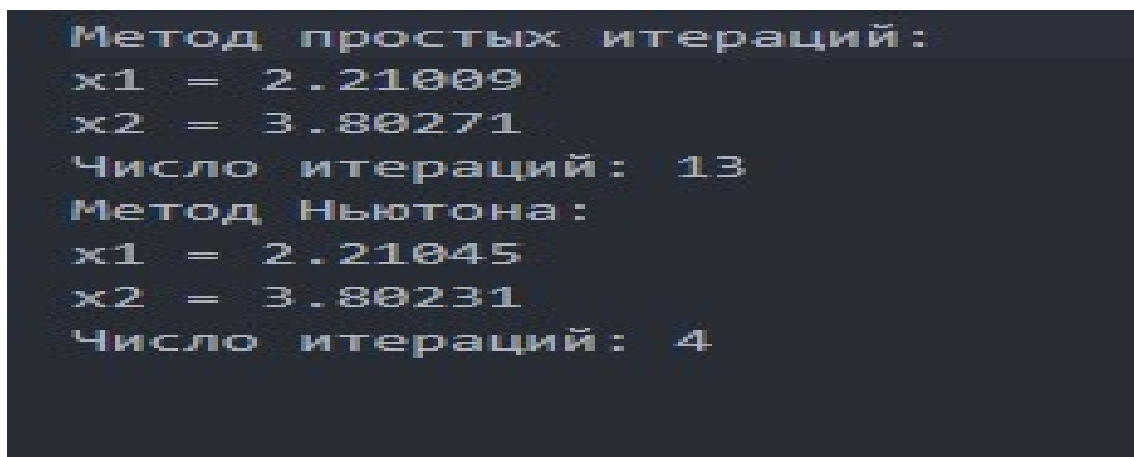
### 4 Постановка задачи

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: 12**

$$\begin{cases} x_1 - \cos(x_2) = 3 \\ x_2 - \sin(x_1) = 3 \end{cases}$$

### 5 Результаты работы



```
Метод простых итераций:  
x1 = 2.21009  
x2 = 3.80271  
Число итераций: 13  
Метод Ньютона:  
x1 = 2.21045  
x2 = 3.80231  
Число итераций: 4
```

Рис. 2: Вывод программы в консоли

## 6 Исходный код

Файл со вторым заданием второй лабораторной работы:

```
1  #include <iostream>
2  #include <cmath>
3  #include <fstream>
4
5  using namespace std;
6
7  double f1(double x1, double x2) {
8      return x1-cos(x2)-3;
9  }
10
11 double f2(double x1, double x2){
12     return x2-sin(x1)-3;
13 }
14
15 double df1dx1() {
16     return 1;
17 }
18
19 double df1dx2(double x2) {
20     return sin(x2);
21 }
22
23 double df2dx1(double x1) {
24     return -cos(x1);
25 }
26
27 double df2dx2() {
28     return 1;
29 }
30
31 double phi_1(double x2){
32     return cos(x2)+3;
33 }
34
35 double phi_2(double x1){
36     return sin(x1)+3;
37 }
38
39 pair<double, double> simple_iteration(double eps, double x1, double x2, int &iters) {
40     while (iters < 1000) {
41         double x1_new = phi_1(x2);
42         double x2_new = phi_2(x1);
43         if (fabs(x1_new - x1) < eps && fabs(x2_new - x2) < eps) {
44             return {x1_new, x2_new};
45         }
46         x1 = x1_new;
```

```

47     x2 = x2_new;
48     iters++;
49 }
50 return {x1, x2};
51 }
52
53 pair<double, double> newton(double eps, double x1, double x2, int &iters) {
54     while (iters < 1000) {
55         double det = df1dx1() * df2dx2() - df1dx2(x2) * df2dx1(x1);
56         double dx1 = (f1(x1, x2) * df2dx2() - f2(x1, x2) * df1dx2(x2)) / det;
57         double dx2 = (f2(x1, x2) * df1dx1() - f1(x1, x2) * df2dx1(x1)) / det;
58         x1 -= dx1;
59         x2 -= dx2;
60         if (fabs(dx1) < eps && fabs(dx2) < eps) {
61             return {x1, x2};
62         }
63         iters++;
64     }
65     return {x1, x2};
66 }
67
68 int main(){
69     double eps = 0.001;
70     double x1 = 0.5, x2 = 0.5;
71     int iterations = 0;
72     ofstream fout("output.txt");
73     auto solution_1 = simple_iteration(eps, x1, x2, iterations);
74     fout << " : " << endl;
75     fout << "x1 = " << solution_1.first << endl;
76     fout << "x2 = " << solution_1.second << endl;
77     fout << " : " << iterations << endl;
78     int iterations_2 = 0;
79     auto solution_2 = newton(eps, x1, x2, iterations_2);
80     fout << " : " << endl;
81     fout << "x1 = " << solution_2.first << endl;
82     fout << "x2 = " << solution_2.second << endl;
83     fout << " : " << iterations_2 << endl;
84     return 0;
85 }

```