

**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: Наумов Г.К.  
Преподаватель: Пивоваров Д.Е.  
Группа: М8О-303Б-21  
Дата:  
Оценка:  
Подпись:

**Москва, 2024**

## 2.1 Методы простой итерации и Ньютона

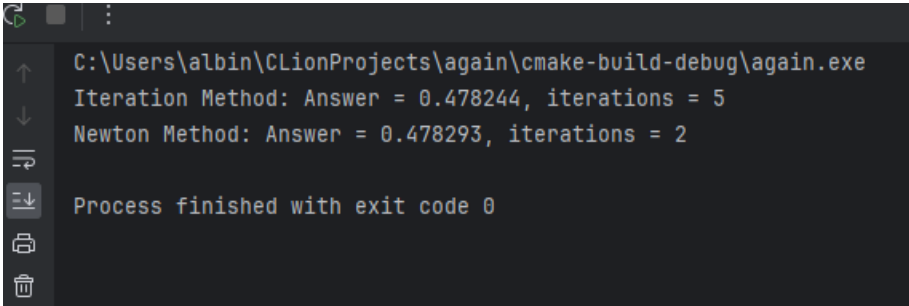
### 1 Постановка задачи

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: 16**

$$xe^x + x^2 - 1 = 0$$

### 2 Результаты работы



```
C:\Users\albin\CLionProjects\again\cmake-build-debug\again.exe
Iteration Method: Answer = 0.478244, iterations = 5
Newton Method: Answer = 0.478293, iterations = 2
Process finished with exit code 0
```

Рис. 1: Вывод программы

### 3 Исходный код

```
1 | #include <iostream>
2 | #include <cmath>
3 |
4 | double f(double x) {
5 |     return x * exp(x) + x * x - 1;
6 | }
7 |
8 | double f_derivative(double x) {
9 |     return exp(x) + x * exp(x) + 2 * x;
10 | }
11 |
12 |
13 | double g(double x) {
14 |     return x - (x * std::exp(x) + x * x - 1) / (std::exp(x) + 2 * x);
```

```

15 }
16
17 double Iteration(double initialGuess, double accuracy, int &iterations) {
18     double x = initialGuess;
19     while (iterations < 1000) {
20         double x1 = g(x);
21         if (std::fabs(x1 - x) < accuracy) {
22             return x1;
23         }
24         x = x1;
25         iterations++;
26     }
27     return x;
28 }
29
30 double Newton(double initialGuess, double accuracy, int &iterations) {
31     double x = initialGuess;
32     while (fabs(f(x)) > accuracy && iterations < 1000) {
33         x = x - f(x) / f_derivative(x);
34         iterations++;
35     }
36     return x;
37 }
38
39 int main() {
40     double initialGuess = 0.6;
41     double accuracy = 0.001;
42     int iterations = 0;
43
44     double answer = Iteration(initialGuess, accuracy, iterations);
45     std::cout << "Iteration Method: Answer = " << answer << ", iterations = " <<
        iterations << std::endl;
46
47     iterations = 0;
48     answer = Newton(initialGuess, accuracy, iterations);
49     std::cout << "Newton Method: Answer = " << answer << ", iterations = " <<
        iterations << std::endl;
50
51     return 0;
52 }

```

## 2.2 Методы простой итерации и Ньютона

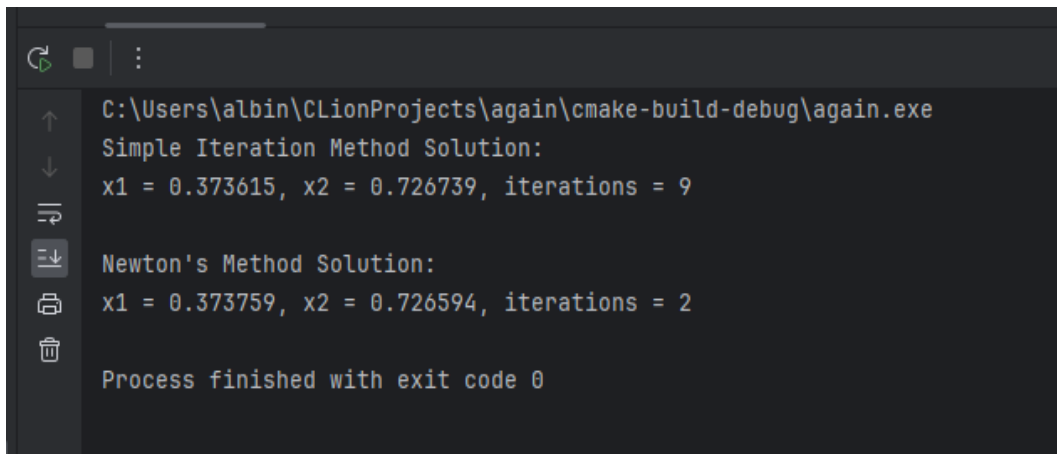
### 4 Постановка задачи

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: 16**

$$\begin{cases} 2x_1 - \cos(x_2) = 0 \\ 2x_2 - \exp(x_1) = 0 \end{cases}$$

### 5 Результаты работы



```
C:\Users\albin\CLionProjects\again\cmake-build-debug\again.exe
Simple Iteration Method Solution:
x1 = 0.373615, x2 = 0.726739, iterations = 9

Newton's Method Solution:
x1 = 0.373759, x2 = 0.726594, iterations = 2

Process finished with exit code 0
```

Рис. 2: Вывод программы

## 6 Исходный код

```
1 | #include <iostream>
2 | #include <cmath>
3 |
4 | double f1(double x1, double x2) {
5 |     return 2 * x1 - std::cos(x2);
6 | }
7 |
8 | double f2(double x1, double x2) {
9 |     return 2 * x2 - std::exp(x1);
10 | }
11 |
12 | double df1_dx1(double x2) {
13 |     return 2;
14 | }
15 |
16 | double df1_dx2(double x1) {
17 |     return std::sin(x1);
18 | }
19 |
20 | double df2_dx1(double x1) {
21 |     return -std::exp(x1);
22 | }
23 |
24 | double df2_dx2() {
25 |     return 2;
26 | }
27 |
28 | std::pair<double, double> simple_iteration(double accuracy, double x1, double x2, int
    &iters) {
29 |     while (iters < 1000) {
30 |         double new_x1 = std::cos(x2) / 2.0;
31 |         double new_x2 = std::exp(x1) / 2.0;
32 |
33 |         if (std::fabs(new_x1 - x1) < accuracy && std::fabs(new_x2 - x2) < accuracy) {
34 |             return {new_x1, new_x2};
35 |         }
36 |
37 |         x1 = new_x1;
38 |         x2 = new_x2;
39 |         iters++;
40 |     }
41 |
42 |     return {x1, x2};
43 | }
44 |
45 | //
46 | std::pair<double, double> newton_method(double accuracy, double x1, double x2, int &
```

```

47     iters) {
48     while (iters < 1000) {
49         double det = df1_dx1(x2) * df2_dx2() - df1_dx2(x1) * df2_dx1(x1);
50         double dx1 = (f1(x1, x2) * df2_dx2() - f2(x1, x2) * df1_dx2(x1)) / det;
51         double dx2 = (f2(x1, x2) * df1_dx1(x2) - f1(x1, x2) * df2_dx1(x1)) / det;
52
53         x1 -= dx1;
54         x2 -= dx2;
55
56         if (std::fabs(dx1) < accuracy && std::fabs(dx2) < accuracy) {
57             return {x1, x2};
58         }
59         iters++;
60     }
61     return {x1, x2};
62 }
63
64 int main() {
65     double accuracy = 0.001;
66     double x1 = 0.5, x2 = 0.5;
67     int iters = 0;
68     auto simple_solution = simple_iteration(accuracy, x1, x2, iters);
69     std::cout << "Simple Iteration Method Solution:\n";
70     std::cout << "x1 = " << simple_solution.first << ", x2 = " << simple_solution.
71         second << ", iterations = " << iters << std::endl;
72     std::cout << std::endl;
73
74     iters = 0;
75     auto newton_solution = newton_method(accuracy, x1, x2, iters);
76     std::cout << "Newton's Method Solution:\n";
77     std::cout << "x1 = " << newton_solution.first << ", x2 = " << newton_solution.
78         second << ", iterations = " << iters << std::endl;
79     return 0;
80 }

```