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: Кучмистов Д.Р.
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: 14

$x^3 - 2x^2 - 10x + 15 = 0$

2 Результаты работы (Нашло два корня, оба из них верные)

```
Enter the value of epsilon:0.01
 Simple Iteration Method:
Root: 3.60572
Number of iterations: 13

Newton's Method:
Root: 1.38196
Number of iterations: 2
```

Рис. 1: Вывод программы

3 Исходный код

```cpp
#include <iostream>
#include <cmath>

using namespace std;

const double INITIAL_VALUE = 1.5;
const double EPSILON = 1e-6;

double function(double x) {
    return pow(x, 3) - 2 * pow(x, 2) - 10 * x + 15;
}

```

```
13 double derivative(double x) {
14     return 3 * pow(x, 2) - 4 * x - 10;
15 }
16
17 double iter_function(double x){
18     return pow(2 * pow(x, 2) + 10 * x - 15, 1.0 / 3.0);
19 }
20
21 double absolute(double a) {
22     return a > 0 ? a : -a;
23 }
24
25 double simple_iteration(double x0, double eps) {
26     double x = iter_function(x0);
27     double diff = abs(x - x0);
28     x0 = x;
29     int k = 0;
30     while (diff >= eps) {
31         x = iter_function(x0);
32         diff = abs(x - x0);
33         x0 = x;
34         k++;
35     }
36     cout << "Simple Iteration Method:" << endl;
37     cout << "Root: " << x << endl;
38     cout << "Number of iterations: " << k << endl;
39     return x;
40 }
41
42 double newton_method(double (*func)(double), double (*deriv)(double), double epsilon)
       {
43     double result, previous = INITIAL_VALUE;
44     int iterations = 0;
45     do {
46         result = previous - func(previous) / deriv(previous);
47         iterations++;
48         previous = result;
49     } while (absolute(func(result)) >= epsilon);
50     cout << endl;
51     cout << "Newton's Method:" << endl;
52     cout << "Root: " << result << endl;
53     cout << "Number of iterations: " << iterations << endl;
54     return result;
55 }
56
57
58 int main() {
59     double epsilon = EPSILON;
60
```

```
61      cout << "Enter the value of epsilon: ";
62      cin >> epsilon;
63
64      if (epsilon <= 0) {
65          cerr << "Error: Epsilon should be positive." << endl;
66          return 1;
67      }
68
69      simple_iteration(INITIAL_VALUE, epsilon);
70      newton_method(function, derivative, epsilon);
71
72      return 0;
73  }
```

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: 14

$$\begin{cases} \frac{x_1^2}{3} + \frac{x_2^2}{(\frac{3}{2})^2} - 1 = 0 \\ 3x_2 - exp(x_1) - x_1 = 0 \end{cases}$$

5 Результаты работы

```
Enter the value of epsilon:0.01

Newton's Method:
Root (x1): 1.11893
Root (x2): 1.39202
Number of iterations: 2

Simple Iteration Method:
Root (x1): 1.12646
Root (x2): 1.3921
Number of iterations: 50
```

Рис. 2: Вывод программы

6 Исходный код

```cpp
#include <iostream>
#include <cmath>
#include <utility>

using namespace std;

const int MAX_ITERATIONS = 1000;

double F1(double x1, double x2) {
    return pow(x1, 2) / pow(3, 2) + pow(x2, 2) / pow(3.0 / 2.0, 2) - 1;
}

double F2(double x1, double x2) {
    return 3 * x2 - exp(x1) - x1;
}

double G1(double x1, double x2) {
    return sqrt(1 - pow(x2, 2) / 4.5) * 3.0 / 2.0;
}

double G2(double x1, double x2) {
    return (exp(x1) + x1) / 3.0;
}

double dF1_dx1(double x1, double x2) {
    return 2 * x1 / pow(3, 2);
}

double dF1_dx2(double x1, double x2) {
    return 2 * x2 / pow(3.0 / 2.0, 2);
}

double dF2_dx1(double x1, double x2) {
    return -exp(x1) - 1;
}

double dF2_dx2(double x1, double x2) {
    return 3;
}

pair<double, double> simple_iteration(double accuracy, double x1, double x2, int &
    iters) {
    while (iters < 1000) {
        double new_x1 = G1(x1, x2);
        double new_x2 = G2(x1, x2);

        if (fabs(new_x1 - x1) < accuracy && fabs(new_x2 - x2) < accuracy) {
```

```
47              return {new_x1, new_x2};
48          }
49
50          x1 = new_x1;
51          x2 = new_x2;
52          iters++;
53      }
54
55      return {x1, x2};
56  }
57
58  pair<double, double> newton_method(double x1, double x2, double epsilon) {
59      double x1_new, x2_new;
60      int iterations = 0;
61
62      do {
63          double determinant = dF1_dx1(x1, x2) * dF2_dx2(x1, x2) - dF1_dx2(x1, x2) *
                dF2_dx1(x1, x2);
64          x1_new = x1 - (F1(x1, x2) * dF2_dx2(x1, x2) - F2(x1, x2) * dF1_dx2(x1, x2)) /
                determinant;
65          x2_new = x2 - (F2(x1, x2) * dF1_dx1(x1, x2) - F1(x1, x2) * dF2_dx1(x1, x2)) /
                determinant;
66
67          iterations++;
68          if (iterations > MAX_ITERATIONS) {
69              cout << "Newton's Method did not converge within max iterations." << endl;
70              break;
71          }
72
73          x1 = x1_new;
74          x2 = x2_new;
75
76      } while (abs(F1(x1, x2)) > epsilon || abs(F2(x1, x2)) > epsilon);
77
78      cout << endl;
79      cout << "Newton's Method:" << endl;
80      cout << "Root (x1): " << x1_new << endl;
81      cout << "Root (x2): " << x2_new << endl;
82      cout << "Number of iterations: " << iterations << endl;
83
84      return make_pair(x1_new, x2_new);
85  }
86
87  int main() {
88      double x1 = 1.5;
89      double x2 = 1.5;
90
91      double epsilon;
92      cout << "Enter the value of epsilon: ";
```

```
    cin >> epsilon;

    pair<double, double> result_newton = newton_method(x1, x2, epsilon);
    int iterations = 0;
    pair<double, double> result_simple = simple_iteration(epsilon, x1, x2, iterations);

    cout << endl;
    cout << "Simple Iteration Method:" << endl;
    cout << "Root (x1): " << result_simple.first << endl;
    cout << "Root (x2): " << result_simple.second << endl;
    cout << "Number of iterations: " << iterations << endl;

    return 0;
}
```