Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу «Численные методы»

Студент: М. А. Бронников
Преподаватель: Д. Л. Ревизников
Группа: М8О-307Б

Дата:
Оценка:
Подпись:

Москва, 2020

Лабораторная работа №2

Вариант: 4

Задача:

1. 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.
   Уравнение:

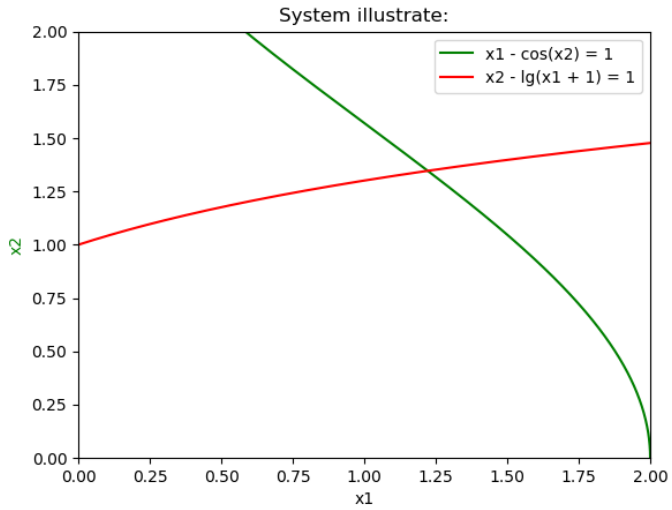
   $$x^3 + x^2 - x - 0.5 = 0$$

2. 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.
   Система:

   $$\begin{cases} x_1 - \cos x_2 = 1 \\ x_2 - \operatorname{tg}(x_1 + 1) = a \end{cases}$$

1 Графичесоке отображение

Перед тем, как решить задачу из задания №2, я построил график сиситемы для более четкого понимания того, в какой окрестности мне стоит задавать начальную точку.



Код для отрисоки графика:

```
import numpy as np
import matplotlib.pyplot as plt
import numpy as np
import math

t = np.arange(0.0, 2.2, 0.001)
y1 = list(t)
x1 = list(map(lambda y: math.cos(y) + 1.0, y1))
x2 = list(t)
y2 = list(map(lambda x: math.log10(x + 1.0) + 1.0, x2))

fig = plt.figure()
ax1 = fig.add_subplot(111)
line1, = ax1.plot(x1, y1, 'g', label="x1 - cos(x2) = 1")
ax1.set_xlabel('x1')
ax1.set_ylabel('x2', color='g')
line2, = ax1.plot(x2, y2, 'r', label="x2 - lg(x1 + 1) = 1")
plt.title('System illustrate:')
plt.xlim(0.0, 2.0)
plt.ylim(0.0, 2.0)
plt.legend((line1, line2), ("x1 - cos(x2) = 1", "x2 - lg(x1 + 1) = 1"))

plt.show()
```

2 Исходный код

Исходный код я писал на языке *C++*.

Реализация матрицы:

```
#ifndef MATRIX_H
#define MATRIX_H

#include <vector>
#include <iostream>
#include <cmath>


using namespace std;

// class with matrix functions:
class Matrix{
public:
    Matrix();
    Matrix(int n, int m);

    void make_ones();
    void transpose();

    vector<double>& operator[](const int index);
    const vector<double>& operator[](const int index) const;

    friend const Matrix operator+(const Matrix& left, const Matrix& right);
    friend const Matrix operator-(const Matrix& left, const Matrix& right);

    friend const Matrix operator*(const Matrix& left, const Matrix& right);
    //friend const Matrix operator*(const vector<double>& left, const vector<double>&
        right);
    friend const vector<double> operator*(const Matrix& left, const vector<double>&
        right);
    friend const Matrix operator*(const Matrix& left, double right);
    friend const Matrix operator*(double left, const Matrix& right);

    const Matrix& operator=(const Matrix& right);

    friend std::ostream& operator<<(std::ostream& os, const Matrix& matrix);

    double get_norm() const;
    double get_upper_norm() const;

    int get_n() const;
    int get_m() const;

    bool is_three_diagonal() const;
```

```
43      bool is_simmetric() const;
44
45      bool is_quadratic() const;
46
47  private:
48      vector<vector<double>> _matrix;
49      int n_size;
50      int m_size;
51  };
```

Задание №1

```
1   #include <iostream>
2   #include <vector>
3   #include <algorithm>
4   #include <cmath>
5
6   using namespace std;
7
8   const double epsilon = 0.001;
9   const double delta = 5.0;
10  const double min_d = 0.05;
11
12  double dfitta_dx(double x);
13
14  void ave(){
15      cout << "============================================================" << endl;
16      cout << "| LABORATORY WORK №2 |" << endl;
17      cout << "| NUMERICAL METHODS |" << endl;
18      cout << "| Task №1 |" << endl;
19      cout << "| Variant №4 |" << endl;
20      cout << "| |" << endl;
21      cout << "| Student: Bronnikov M.A. |" << endl;
22      cout << "| Date: 28.02.2020 |" << endl;
23      cout << "| |" << endl;
24      cout << "| |" << endl;
25      cout << "| Moscow, 2020 |" << endl;
26      cout << "============================================================" << endl;
27  }
28
29  void bye(){
30      cout << "============================================================" << endl;
31      cout << "| EXIT |" << endl;
32      cout << "============================================================" << endl;
33  }
34
35  void print_statement(double alfa, double x0){
36      cout << "============================================================" << endl;
37      cout << "| EXERCSICE: |" << endl;
38      cout << "============================================================" << endl;
```

```
39      cout << "| We need to solve equation by simple itteraion and Newton |" << endl;
40      cout << "| method. |" << endl;
41      cout << "============================================================" << endl;
42      cout << "Equation:" << endl;
43      cout << "x^3 + x^2 - x + 0.5 = 0" << endl;
44      cout << "Accuracy: " << alfa << endl;
45      cout << "x0 = " << x0 << endl;
46      cout << "============================================================" << endl;
47  }
48
49
50  void print_solution(double x_n, int itter_n, double x_i, int itter_i, double a, double
        b){
51      cout << "============================================================" << endl;
52      cout << "| ANSWER: |" << endl;
53      cout << "============================================================" << endl;
54      cout << "| NEWTON METHOD: |" << endl;
55      cout << "============================================================" << endl;
56      cout << "x = " << x_n << endl;
57      cout << "Itterations: " << itter_n << endl;
58      cout << "============================================================" << endl;
59      cout << "| SIMPLE ITTERATIONS METHOD: |" << endl;
60      cout << "============================================================" << endl;
61      cout << "x = " << x_i << endl;
62      cout << "Itterations: " << itter_i << endl;
63      cout << "x0 in [" << a << ", " << b << "]" << endl;
64      cout << "============================================================" << endl;
65  }
66
67
68  // dihotomy methood for search supr:
69  double get_sup(double a, double b){
70      double F1, F2;
71      while(abs(b - a) >= epsilon){
72          double x = (a + b) / 2.0;
73          F2 = dfitta_dx(x + epsilon);
74          F1 = dfitta_dx(x - epsilon);
75          if(F1 < F2){
76              a = x;
77          }else{
78              b = x;
79          }
80      }
81      return dfitta_dx((b + a) / 2.0);
82  }
83
84
85
86
```

```
87  double f(double x){
88      return x*x*x + x*x - x - 0.5;
89  }
90
91  double df_dx(double x){
92      return 3.0*x*x + 2.0*x - 1.0;
93  }
94
95  double d2f_dx2(double x){
96      return 6.0*x + 2.0;
97  }
98
99  bool newton_condition(double x0){
100     return f(x0) * d2f_dx2(x0) > 0;
101 }
102
103 double fitta(double x){
104     return pow(0.5 + x - x*x, 1.0 / 3.0);
105 }
106
107 double dfitta_dx(double x){
108     return (1.0 - 2.0*x) / (3.0 * pow(0.5 + x - x*x, 2.0 / 3.0));
109 }
110
111
112 // Method searc [a, b] where true condition
113 bool itteration_condition(double x0, double& a, double& b){
114     double d = delta;
115     a = x0 - d;
116     b = x0 + d;
117     while(d > min_d){
118         if(get_sup(a, b) < 1){
119             return true;
120         }
121         d /= 2.0;
122         a = x0 - d;
123         b = x0 + d;
124     }
125     return false;
126 }
127
128 double newton_method(double x0, double alfa, int& itter){
129     double x_j, x_k = x0;
130     itter = 0;
131     do{
132         x_j = x_k;
133         x_k -= f(x_j) / df_dx(x_j);
134         ++itter;
135     }while(abs(x_k - x_j) >= alfa);
```

```
136     return x_k;
137 }
138 
139 double itteration_method(double x0, double alfa, int& itter, double a, double b){
140     double x_j, x_k = x0;
141     // search q here:
142     double q = get_sup(a, b);
143     q /= (1 - q);
144 
145     itter = 0;
146     do{
147         x_j = x_k;
148         x_k = fitta(x_j);
149         ++itter;
150     }while(q * abs(x_k - x_j) >= alfa);
151     return x_k;
152 }
153 
154 int main(){
155     double x0, alfa, x_n, x_i;
156     double a, b;
157     int itter_n, itter_i;
158 
159     ave();
160     cin >> alfa;
161     cin >> x0;
162     print_statement(alfa, x0);
163 
164     // Check conditions for x0:
165     if(!newton_condition(x0)){
166         cout << "Wrong x0 for Newton method. Please try to choice another x0." << endl;
167         bye();
168         return 0;
169     }
170 
171     // [a, b] - set of points where good solutions
172     if(!itteration_condition(x0, a, b)){
173         cout << "Wrong x0 for simmple itteration method. Please try to choice another
            x0." << endl;
174         bye();
175         return 0;
176     }
177 
178     x_n = newton_method(x0, alfa, itter_n);
179     x_i = itteration_method(x0, alfa, itter_i, a, b);
180 
181     print_solution(x_n, itter_n, x_i, itter_i, a, b);
182     bye();
183     return 0;
```

```
}
```

Задание №2:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include "Matrix/matrix.h"

using namespace std;

// params:
const double a = 1.0; // 4 variant
const int n = 2; // 2 variables in equation
const double start_delta = 0.2;
const double min_delta = 0.01;
const double search_step = 0.01;

void ave(){
    cout << "============================================================" << endl;
    cout << "| LABORATORY WORK №2 |" << endl;
    cout << "| NUMERICAL METHODS |" << endl;
    cout << "| Task №2 |" << endl;
    cout << "| Variant №4 |" << endl;
    cout << "| |" << endl;
    cout << "| Student: Bronnikov M.A. |" << endl;
    cout << "| Date: 28.02.2020 |" << endl;
    cout << "| |" << endl;
    cout << "| |" << endl;
    cout << "| Moscow, 2020 |" << endl;
    cout << "============================================================" << endl;
}

void bye(){
    cout << "============================================================" << endl;
    cout << "| EXIT |" << endl;
    cout << "============================================================" << endl;
}

void print_statement(double alfa, const vector<double>& x0){
    cout << "============================================================" << endl;
    cout << "| EXERCSICE: |" << endl;
    cout << "============================================================" << endl;
    cout << "| We need to solve equation system by simple itteraion and |" << endl;
    cout << "| Newton methods. |" << endl;
    cout << "============================================================" << endl;
    cout << "Equation system:" << endl;
    cout << endl;
    cout << "x1 - cos(x2) = 1" << endl;
```

```
47      cout << "x2 - lg(x1 + 1) = 1" << endl;
48      cout << endl;
49      cout << "Accuracy: " << alfa << endl;
50      cout << "x0 = (" << x0[0];
51      for(int i = 1; i < n; ++i){
52          cout << ", " << x0[i];
53      }
54      cout << ")" << endl;
55      cout << "============================================================" << endl;
56  }
57
58
59  void print_solution(const vector<double>& x_n, int itter_n, const vector<double>& x_i,
         int itter_i, const double q){
60      cout << "============================================================" << endl;
61      cout << "| ANSWER: |" << endl;
62      cout << "============================================================" << endl;
63      cout << "| NEWTON METHOD: |" << endl;
64      cout << "============================================================" << endl;
65      cout << "x = (" << x_n[0];
66      for(int i = 1; i < n; ++i){
67          cout << ", " << x_n[i];
68      }
69      cout << ")" << endl;
70      cout << "Itterations: " << itter_n << endl;
71      cout << "============================================================" << endl;
72      cout << "| SIMPLE ITTERATIONS METHOD: |" << endl;
73      cout << "============================================================" << endl;
74      if(q >= 1.0){
75          cout << "Sufficient condition not done!" << endl;
76      }else{
77          cout << "Sufficient condition done with q: " << q << endl;
78      }
79      cout << "x = (" << x_i[0];
80      for(int i = 1; i < n; ++i){
81          cout << ", " << x_i[i];
82      }
83      cout << ")" << endl;
84      cout << "Itterations: " << itter_i << endl;
85      cout << "============================================================" << endl;
86  }
87
88  // plus and minus:
89  vector<double> vector_minus(const vector<double>& a, const vector<double>& b){
90      vector<double> minus = a;
91      for(unsigned i = 0; i < minus.size(); ++i){
92          minus[i] -= b[i];
93      }
94      return minus;
```

```
95  }
96
97  vector<double> vector_plus(const vector<double>& a, const vector<double>& b){
98      vector<double> plus = a;
99      for(unsigned i = 0; i < plus.size(); ++i){
100         plus[i] += b[i];
101     }
102     return plus;
103 }
104
105 // norm for methods stop
106 double norm_of_vector(const vector<double>& vec){
107     double norm = 0.0;
108     for(unsigned i = 0; i < vec.size(); ++i){
109         norm += vec[i] * vec[i];
110     }
111     return sqrt(norm);
112 }
113
114 // norm for zeidel stop
115 double norm_of_vectors(const vector<double>& x1, const vector<double>& x2){
116     double norm = 0.0;
117     if(x1.size() != x2.size()){
118         throw "Wrong sizes of vectors";
119     }
120     for(unsigned i = 0; i < x1.size(); ++i){
121         norm = max(abs(x1[i] - x2[i]), norm);
122     }
123     return norm;
124 }
125
126 // For SLAU solve:
127 void zeidels_method(const Matrix& A, const vector<double>& b, vector<double>& x,
        double alfa){
128     Matrix M = A;
129     x.resize(b.size());
130     vector<double> last(b.size(), 0.0), r = b;
131     if(!M.is_quadratic()){
132         throw "Wrong matrix! Try again!";
133     }
134     for(int i = 0; i < M.get_n(); ++i){
135         if(!A[i][i]){
136             throw "Wrong matrix! Try again!";
137         }
138         for(int j = 0; j < M.get_m(); ++j){
139             M[i][j] = i == j ? 0.0 : -A[i][j] / A[i][i];
140         }
141         r[i] /= A[i][i];
142     }
```

```
    x = r;
    for(int itter = 0; norm_of_vector(vector_minus(x, last)) > alfa; ++itter){
        x.swap(last);
        x = r;
        for(int i = 0; i < M.get_n(); ++i){
            for(int j = 0; j < i; ++j){
                x[i] += x[j] * M[i][j];
            }
            for(int j = i; j < M.get_m(); ++j){
                x[i] += last[j] * M[i][j];
            }
        }
    }
}

/* This methods set my equation. redefine for your task */

double f1(const vector<double>& x){
    return x[0] - cos(x[1]) - 1.0;
}

double df1_dx1(const vector<double>& x){
    return 1.0;
}

double df1_dx2(const vector<double>& x){
    return sin(x[1]);
}


double f2(const vector<double>& x){
    return x[1] - log10(x[0] + 1.0) - a;
}


double df2_dx2(const vector<double>& x){
    return 1.0;
}

double df2_dx1(const vector<double>& x){
    return -1.0 / ((x[0] + 1) * log(10));
}

double fitta1(const vector<double>& x){
    return cos(x[1]) + 1.0;
}

double dfitta1_dx1(const vector<double>& x){
    return 0.0;
```

```
}

double dfitta1_dx2(const vector<double>& x){
    return -sin(x[1]);
}

double fitta2(const vector<double>& x){
    return log10(x[0] + 1.0) + a;
}

double dfitta2_dx1(const vector<double>& x){
    return 1.0 / ((x[0] + 1) * log(10));
}

double dfitta2_dx2(const vector<double>& x){
    return 0.0;
}


void set_matrix(Matrix& A, const vector<double>& x){
    A = Matrix(n, n);
    A[0][0] = df1_dx1(x);
    A[0][1] = df1_dx2(x);
    A[1][0] = df2_dx1(x);
    A[1][1] = df2_dx2(x);
}

void set_dfitta_matrix(Matrix& A, const vector<double>& x){
    A = Matrix(n, n);
    A[0][0] = dfitta1_dx1(x);
    A[0][1] = dfitta1_dx2(x);
    A[1][0] = dfitta2_dx1(x);
    A[1][1] = dfitta2_dx2(x);
}

void set_vector(vector<double>& fx, const vector<double>& x){
    fx.resize(n);
    fx[0] = -f1(x);
    fx[1] = -f2(x);
}

/* End of methods for redefine */


// Main methods:
vector<double> nex_step(const vector<double>& x){
    vector<double> ans(n);
    ans[0] = fitta1(x);
    ans[1] = fitta2(x);
```

```
241     return ans;
242 }
243 
244 int newton_method(const vector<double>& x0, vector<double>& x, double alfa){
245     int itter = 0;
246     vector<double> x_i;
247     vector<double> fx, dx;
248     Matrix J;
249     x = x0;
250     do{
251         x_i.swap(x);
252         set_matrix(J, x_i);
253         set_vector(fx, x_i);
254         zeidels_method(J, fx, dx, alfa);
255         x = vector_plus(x_i, dx);
256         ++itter;
257     }while(norm_of_vectors(x, x_i) > alfa);
258     return itter;
259 }
260 
261 double find_q(const vector<double>& x0){
262     double delta = start_delta * 2.0;
263     vector<double> ans_x = x0;
264     Matrix Dfitta;
265     double ans;
266     do{
267         delta /= 2.0;
268         // search x with max norm
269         for(unsigned i = 0; i < x0.size(); ++i){
270             double maximum = 0.0;
271             vector<double> x = x0;
272             for(double v = x0[i] - delta; v <= x0[i] + delta; v += search_step){
273                 x[i] = v;
274                 set_dfitta_matrix(Dfitta, x);
275                 double norm = Dfitta.get_norm();
276                 if(norm > maximum){
277                     ans_x[i] = v;
278                     maximum = norm;
279                 }
280             }
281         }
282         set_dfitta_matrix(Dfitta, ans_x);
283         ans = Dfitta.get_norm();
284     }while(ans >= 1.0 && delta >= min_delta);
285     return ans;
286 }
287 
288 int itteration_method(const vector<double>& x0, vector<double>& x, double alfa, double
    & q){
```

```
289     int itter = 0;
290     vector<double> x_i;
291     vector<double> fx, dx;
292     q = find_q(x0);
293     if(q < 1.0){
294         alfa *= (1.0 - q);
295         alfa /= q;
296     }
297     x = x0;
298     do{
299         x_i.swap(x);
300         x = nex_step(x_i);
301         ++itter;
302     }while(norm_of_vectors(x, x_i) > alfa);
303     return itter;
304 }
305 
306 int main(){
307     ave();
308     vector<double> x0(n), x_n(n), x_i(n, 0);
309     double alfa, q;
310     int itter_n, itter_i = 0;
311     // read accuracy and x0:
312     cin >> alfa;
313     for(int i = 0; i < n; ++i){
314         cin >> x0[i];
315     }
316     print_statement(alfa, x0);
317 
318     itter_n = newton_method(x0, x_n, alfa);
319     itter_i = itteration_method(x0, x_i, alfa, q);
320 
321     print_solution(x_n, itter_n, x_i, itter_i, q);
322     bye();
323     return 0;
324 }
```

3 Демонстрация работы

```
(base) max@max-Lenovo-B50-30:~/NumMethods/lab2$ ls
data      Matrix       task1.cpp  task2_graphic.ipynb
Makefile  methods.pdf  task2.cpp  tasks
(base) max@max-Lenovo-B50-30:~/NumMethods/lab2$ make all
g++ -std=c++17 -Wall -pedantic task1.cpp -o task1
g++ -std=c++17 -Wall -pedantic -c task2.cpp
g++ -std=c++17 -Wall -pedantic -c Matrix/matrix.cpp -o Matrix/matrix.o
g++ task2.o Matrix/matrix.o -o task2
rm -rf task2.o
rm -rf Matrix/matrix.o
(base) max@max-Lenovo-B50-30:~/NumMethods/lab2$ ./task1 <data/d1.txt
============================================================
|                     LABORATORY WORK №2                   |
|                     NUMERICAL METHODS                    |
|                          Task №1                         |
|                        Variant №4                        |
|                                                          |
|                                 Student: Bronnikov M.A.  |
|                                    Date: 28.02.2020      |
|                                                          |
|                                                          |
|                       Moscow,2020                       |
============================================================
============================================================
|                       EXERCSICE:                         |
============================================================
| We need to solve equation by simple itteraion and Newton |
| method.                                                  |
============================================================
Equation:
x^3 + x^2 -x + 0.5 = 0
Accuracy: 0.005
x0 = 0.97
============================================================
============================================================
|                       ANSWER:                            |
============================================================
|                    NEWTON METHOD:                        |
```

```
============================================================
x = 0.854638
Itterations: 3
============================================================
|               SIMPLE ITTERATIONS METHOD:                 |
============================================================
x = 0.850108
Itterations: 3
x0 in [0.345,1.595]
============================================================
============================================================
|                          EXIT                            |
============================================================
(base) max@max-Lenovo-B50-30:~/NumMethods/lab2$ ./task2 <data/d2.txt
============================================================
|                     LABORATORY WORK №2                   |
|                     NUMERICAL METHODS                    |
|                          Task №2                         |
|                       Variant №4                         |
|                                                          |
|                                Student: Bronnikov M.A.   |
|                                   Date: 28.02.2020       |
|                                                          |
|                                                          |
|                       Moscow,2020                       |
============================================================
============================================================
|                       EXERCSICE:                         |
============================================================
| We need to solve equation system by simple itteraion and |
| Newton methods.                                          |
============================================================
Equation system:

x1 -cos(x2) = 1
x2 -lg(x1 + 1) = 1

Accuracy: 0.0005
x0 = (1,1.1)
============================================================
============================================================
```

```
|                       ANSWER:                                |
================================================================
|                    NEWTON METHOD:                            |
================================================================
x = (1.22216,1.34676)
Itterations: 3
================================================================
|                SIMPLE ITTERATIONS METHOD:                    |
================================================================
Sufficient condition done with q: 0.960835
x = (1.22216,1.34678)
Itterations: 14
================================================================
================================================================
|                          EXIT                                |
================================================================
(base) max@max-Lenovo-B50-30:~/NumMethods/lab2$
```

4 Выводы

Выполнив вторую лабораторную работу по курсу «Численные методы», я ближе познакомился с методами Ньютона и простых итераций решения нелинейных уравнений, а также систем с ними. Я самостоятельно реализовал эти методы, что позволило узнать все особенности и подводные камни этих методов.

При решении задания №2 мне пригодились знания и алгоритмы из первой лабораторной, что дало мне очередной пример того, как и где применяются изученные мною методы.

Я рад, что я познакомился с этой темой и уверен что полученные знания и опыт я еще не раз применю на практике в будущем.