Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Численные методы»

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

Дата:
Оценка:
Подпись:

Москва, 2020

Лабораторная работа №1

Вариант: 4

Задача:

1. Реализовать алгоритм LU - разложения матриц (с выбором главного элемента) в виде программы. Используя разработанное программное обеспечение, решить систему линейных алгебраических уравнений (СЛАУ). Для матрицы СЛАУ вычислить определитель и обратную матрицу.

2. Реализовать метод прогонки в виде программы, задавая в качестве входных данных ненулевые элементы матрицы системы и вектор правых частей. Используя разработанное программное обеспечение, решить СЛАУ с трехдиагональной матрицей.

3. Реализовать метод простых итераций и метод Зейделя в виде программ, задавая в качестве входных данных матрицу системы, вектор правых частей и точность вычислений. Используя разработанное программное обеспечение, решить СЛАУ. Проанализировать количество итераций, необходимое для достижения заданной точности.

4. Реализовать метод вращений в виде программы, задавая в качестве входных данных матрицу и точность вычислений. Используя разработанное программное обеспечение, найти собственные значения и собственные векторы симметрических матриц. Проанализировать зависимость погрешности вычислений от числа итераций.

5. Реализовать алгоритм QR – разложения матриц в виде программы. На его основе разработать программу, реализующую QR – алгоритм решения полной проблемы собственных значений произвольных матриц, задавая в качестве входных данных матрицу и точность вычислений. С использованием разработанного программного обеспечения найти собственные значения матрицы.

1 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 <vector>
2   #include <iostream>
3   #include "Matrix/matrix.h"
4
5   using namespace std;
6
7
8   void compute_solution(const Matrix& U, const Matrix& L, const vector<double>& b,
        vector<double>& x){
9       x.resize(U.get_m());
10      // Compute solution:
11      // vector z:
12
13      vector<double> z(L.get_m());
14      for(int i = 0; i < L.get_m(); ++i){
15          z[i] = b[i];
16          for(int j = 0; j < i; ++j){
17              z[i] -= (L[i][j] * z[j]);
18          }
19      }
20
21      // vector x:
22      for(int i = U.get_m() - 1; i >= 0; --i){
23          x[i] = z[i];
24          for(int j = i + 1; j < U.get_m(); ++j){
25              x[i] -= (x[j] * U[i][j]);
26          }
27          x[i] /= U[i][i];
28      }
29  }
30
31
32  // Compute LU matrix, SLAU solution for enterd matrix (if LU not exist - exception)
        and returns matrix determinant
33  double gauss_completely(const Matrix& matrix, Matrix& L, Matrix& U, Matrix& X, const
        vector<double>& b, vector<double>& x){
34      if(!matrix.is_quadratic()){
35          throw "LU not exist! Try simple gauss method or enter lines in another order!";
```

```
36      }
37      // init:
38      U = matrix;
39      L = Matrix(matrix.get_n(), matrix.get_m());
40      X = Matrix(matrix.get_n(), matrix.get_m());
41      double determinant = 1.0;
42
43      // First Gauss steps (compute L and U):
44      for(int j = 0; j < U.get_m(); ++j){
45          if(U[j][j] == 0.0){
46              throw "LU not exist! Try simple gauss method or enter lines in another
                    order!";
47          }
48          L[j][j] = 1.0;
49          for(int i = j + 1; i < U.get_n(); ++i){
50              double l_ij = U[i][j] / U[j][j];
51              L[i][j] = l_ij;
52              // line[i] - line[j]
53              for(int k = j; k < U.get_m(); ++k){
54                  U[i][k] -= U[j][k] * l_ij;
55              }
56          }
57      }
58
59      // Compute solution:
60      compute_solution(U, L, b, x);
61
62      // Compute back matrix like compute sol:
63      vector<double> b_1(b.size());
64      for(int i = 0; i < U.get_n(); ++i){
65          b_1[i] = 1.0;
66          compute_solution(U, L, b_1, X[i]);
67          b_1[i] = 0.0;
68      }
69      X.transpose();
70
71      // Compute determinant:
72      for(int i = 0; i < U.get_m(); ++i){
73          determinant *= U[i][i];
74      }
75
76      return determinant;
77  }
78
79  void ave(){
80      cout << "============================================================" << endl;
81      cout << "| LABORATORY WORK №1 |" << endl;
82      cout << "| NUMERICAL METHODS |" << endl;
83      cout << "| Task №1 |" << endl;
```

```
84      cout << "| Variant №4 |" << endl;
85      cout << "| |" << endl;
86      cout << "| Student: Bronnikov M.A. |" << endl;
87      cout << "| Date: 23.02.2020 |" << endl;
88      cout << "| |" << endl;
89      cout << "| |" << endl;
90      cout << "| Moscow, 2020 |" << endl;
91      cout << "================================================================" << endl;
92  }
93
94  void bye(){
95      cout << "================================================================" << endl;
96      cout << "| EXIT |" << endl;
97      cout << "================================================================" << endl;
98  }
99
100 int size_init(){
101     int size;
102     cin >> size;
103     return size;
104 }
105
106 void matrix_init(Matrix& A, int size){
107     A = Matrix(size, size);
108     for(int i = 0; i < size; ++i){
109         for(int j = 0; j < size; ++j){
110             cin >> A[i][j];
111         }
112     }
113 }
114
115
116 void vector_init(vector<double>& b, int size){
117     b.resize(size);
118     for(int i = 0; i < size; ++i){
119         cin >> b[i];
120     }
121 }
122
123 void print_vector_x(const vector<double>& x){
124     for(unsigned i = 0; i < x.size(); ++i){
125         cout << 'x' << i + 1 << " = " << x[i] << " ";
126     }
127     cout << endl;
128 }
129
130 void print_solution(const Matrix& U, const Matrix& L, const Matrix& B, const vector<
        double>& x, double determinant){
131     cout << "================================================================" << endl;
```

```
132     cout << "| ANSWER: |" << endl;
133     cout << "============================================================" << endl;
134     cout << "L matrix:" << endl;
135     cout << L << endl;
136     cout << "============================================================" << endl;
137     cout << "U matrix:" << endl;
138     cout << U << endl;
139     cout << "============================================================" << endl;
140     cout << "Back matrix:" << endl;
141     cout << B << endl;
142     cout << "============================================================" << endl;
143     cout << "x solution:" << endl;
144     print_vector_x(x);
145     cout << "============================================================" << endl;
146     cout << "Determinant: " << determinant << endl;
147     cout << "============================================================" << endl;
148     return;
149 }
150 
151 void print_statement(const Matrix& A, const vector<double>& b){
152     cout << "============================================================" << endl;
153     cout << "| EXERCSICE: |" << endl;
154     cout << "============================================================" << endl;
155     cout << "| We need to solve SLAU by LU separating matrix with Gauss |" << endl;
156     cout << "| method. Find determinant, back matrix, L and U matrix. |" << endl;
157     cout << "============================================================" << endl;
158     cout << "SLAU matrix:" << endl;
159     cout << A << endl;
160     cout << "Free vector:" << endl;
161     for(unsigned i = 0; i < b.size(); ++i){
162         cout.width(8);
163         cout << b[i] << endl;
164     }
165     cout << "============================================================" << endl;
166 }
167 
168 int main(){
169     ave();
170     Matrix L, U, B, A;
171     vector<double> b, x;
172     int size = size_init();
173     matrix_init(A, size);
174     vector_init(b, size);
175     print_statement(A, b);
176 
177     // Main function use:
178     double determinant = gauss_completely(A, L, U, B, b, x);
179 
180     print_solution(U, L, B, x, determinant);
```

```
181
182     bye();
183     return 0;
184 }
```

Задание №2:

```
1  #include <vector>
2  #include <iostream>
3  #include "Matrix/matrix.h"
4
5  using namespace std;
6
7  void ave(){
8      cout << "============================================================" << endl;
9      cout << "| LABORATORY WORK №1 |" << endl;
10     cout << "| NUMERICAL METHODS |" << endl;
11     cout << "| Task №2 |" << endl;
12     cout << "| Variant №4 |" << endl;
13     cout << "| |" << endl;
14     cout << "| Student: Bronnikov M.A. |" << endl;
15     cout << "| Date: 23.02.2020 |" << endl;
16     cout << "| |" << endl;
17     cout << "| |" << endl;
18     cout << "| Moscow, 2020 |" << endl;
19     cout << "============================================================" << endl;
20 }
21
22 void print_statement(const Matrix& A, const vector<double>& b){
23     cout << "============================================================" << endl;
24     cout << "| EXERCSICE: |" << endl;
25     cout << "============================================================" << endl;
26     cout << "| We need to solve SLAU by race method. |" << endl;
27     cout << "============================================================" << endl;
28     cout << "SLAU matrix:" << endl;
29     cout << A << endl;
30     cout << "Free vector:" << endl;
31     for(unsigned i = 0; i < b.size(); ++i){
32         cout.width(8);
33         cout << b[i] << endl;
34     }
35     cout << "============================================================" << endl;
36 }
37
38 void bye(){
39     cout << "============================================================" << endl;
40     cout << "| EXIT |" << endl;
41     cout << "============================================================" << endl;
42 }
43
```

```
44
45  int size_init(){
46      int size;
47      cin >> size;
48      return size;
49  }
50
51  void matrix_init(Matrix& A, int size){
52      A = Matrix(size, size);
53      for(int i = 0; i < size; ++i){
54          for(int j = 0; j < size; ++j){
55              cin >> A[i][j];
56          }
57      }
58  }
59
60  /*
61  void vector_init(vector<double>& b){
62      b.resize(5);
63      b[0] = -78.0;
64      b[1] = -73.0;
65      b[2] = -38.0;
66      b[3] = 77.0;
67      b[4] = 91.0;
68  }
69  */
70
71
72  void vector_init(vector<double>& b, int size){
73      b.resize(size);
74      for(int i = 0; i < size; ++i){
75          cin >> b[i];
76      }
77  }
78
79
80  void print_vector_x(const vector<double>& x){
81      for(unsigned i = 0; i < x.size(); ++i){
82          cout << 'x' << i + 1 << " = " << x[i] << " ";
83      }
84      cout << endl;
85  }
86
87  void matrix_to_vecs(const Matrix& matrix, vector<vector<double>>& vec){
88      if(!matrix.is_quadratic()){
89          throw "Wrong matrix";
90      }
91      vec.clear();
92      vec.assign(matrix.get_n(), vector<double>(3, 0.0));
```

```
93
94      for(int i = 0; i < matrix.get_n(); ++i){
95          vec[i][0] = i - 1 < 0 ? 0.0 : matrix[i][i - 1];
96          vec[i][1] = matrix[i][i];
97          vec[i][2] = i + 1 < matrix.get_m() ? matrix[i][i + 1] : 0.0;
98      }
99  }
100
101 void race_method(vector<vector<double>>& vec, const vector<double>& b, vector<double>&
        x){
102     x.assign(b.size(), 0.0);
103     vector<double> P(b.size()), Q(b.size());
104     P[0] = -vec[0][2] / vec[0][1];
105     Q[0] = b[0] / vec[0][1];
106     //cout << "P[" << 0 << "] = " << P[0] << " Q[" << 0 << "] = " << Q[0] << endl;
107     for(int i = 1; i < (int)x.size(); ++i){
108         double z = (vec[i][1] + vec[i][0] * P[i-1]);
109         P[i] = -vec[i][2];
110         P[i] /= z;
111         Q[i] = (b[i] - vec[i][0] * Q[i - 1]);
112         Q[i] /= z;
113         //cout << "P[" << i << "] = " << P[i] << " Q[" << i << "] = " << Q[i] << endl;
114     }
115     x.back() = Q.back();
116     for(int i = x.size() - 2; i >= 0; --i){
117         x[i] = P[i]*x[i+1] + Q[i];
118     }
119 }
120
121 void print_solution(const vector<double>& x){
122     cout << "============================================================" << endl;
123     cout << "| ANSWER: |" << endl;
124     cout << "============================================================" << endl;
125     cout << "x solution:" << endl;
126     print_vector_x(x);
127     cout << "============================================================" << endl;
128 }
129
130
131 int main(){
132     ave();
133
134     Matrix A;
135     vector<double> x, b;
136     vector<vector<double>> vec;
137     int size = size_init();
138     matrix_init(A, size);
139     vector_init(b, size);
140     print_statement(A, b);
```

```
141
142     if(A.is_three_diagonal()){
143         matrix_to_vecs(A, vec);
144     }else{
145         cout << "ERROR! MATRIX IS NOT THREE DIAGONAL! EXIT!" << endl;
146         bye();
147         return 0;
148     }
149
150     race_method(vec, b, x);
151
152     print_solution(x);
153     bye();
154
155     return 0;
156 }
```

Задание №3:

```
1  #include <vector>
2  #include <iostream>
3  #include "Matrix/matrix.h"
4  #include <cmath>
5
6  using namespace std;
7
8  void ave(){
9      cout << "============================================================" << endl;
10     cout << "| LABORATORY WORK №1 |" << endl;
11     cout << "| NUMERICAL METHODS |" << endl;
12     cout << "| Task №3 |" << endl;
13     cout << "| Variant №4 |" << endl;
14     cout << "| |" << endl;
15     cout << "| Student: Bronnikov M.A. |" << endl;
16     cout << "| Date: 23.02.2020 |" << endl;
17     cout << "| |" << endl;
18     cout << "| |" << endl;
19     cout << "| Moscow, 2020 |" << endl;
20     cout << "============================================================" << endl;
21 }
22
23 void print_statement(const Matrix& A, const vector<double>& b, double alfa){
24     cout << "============================================================" << endl;
25     cout << "| EXERCSICE: |" << endl;
26     cout << "============================================================" << endl;
27     cout << "| We need to solve SLAU by simple ittearation and Zeidel |" << endl;
28     cout << "| methods. Analyse count of iiteraions with accuracy. |" << endl;
29     cout << "============================================================" << endl;
30     cout << "SLAU matrix:" << endl;
31     cout << A << endl;
```

```
32      cout << "Free vector:" << endl;
33      for(unsigned i = 0; i < b.size(); ++i){
34          cout.width(8);
35          cout << b[i] << endl;
36      }
37      cout << endl;
38      cout << "Accuracy: " << alfa << endl;
39      cout << "============================================================" << endl;
40  }
41
42  void bye(){
43      cout << "============================================================" << endl;
44      cout << "| EXIT |" << endl;
45      cout << "============================================================" << endl;
46  }
47
48
49  int size_init(){
50      int size;
51      cin >> size;
52      return size;
53  }
54
55  void matrix_init(Matrix& A, int size){
56      A = Matrix(size, size);
57      for(int i = 0; i < size; ++i){
58          for(int j = 0; j < size; ++j){
59              cin >> A[i][j];
60          }
61      }
62  }
63
64  void vector_init(vector<double>& b, int size){
65      b.resize(size);
66      for(int i = 0; i < size; ++i){
67          cin >> b[i];
68      }
69  }
70
71
72
73  void print_vector_x(const vector<double>& x){
74      for(unsigned i = 0; i < x.size(); ++i){
75          cout << 'x' << i + 1 << " = " << x[i] << " ";
76      }
77      cout << endl;
78  }
79
80  void print_solution(const vector<double>& x, int itter){
```

```
81      cout << "============================================================" << endl;
82      cout << "| ANSWER: |" << endl;
83      cout << "============================================================" << endl;
84      cout << "x solution:" << endl;
85      print_vector_x(x);
86      cout << "============================================================" << endl;
87      cout << "Itterations: " << itter << endl;
88      cout << "============================================================" << endl;
89  }
90
91  vector<double> vector_minus(const vector<double>& a, const vector<double>& b){
92      vector<double> minus = a;
93      for(unsigned i = 0; i < minus.size(); ++i){
94          minus[i] -= b[i];
95      }
96      return minus;
97  }
98
99  vector<double> vector_plus(const vector<double>& a, const vector<double>& b){
100     vector<double> plus = a;
101     for(unsigned i = 0; i < plus.size(); ++i){
102         plus[i] += b[i];
103     }
104     return plus;
105 }
106
107 double norm_of_vector(const vector<double>& vec){
108     double norm = 0.0;
109     for(unsigned i = 0; i < vec.size(); ++i){
110         norm += vec[i] * vec[i];
111     }
112     return sqrt(norm);
113 }
114
115 int simple_itteration(const Matrix& A, const vector<double>& b, vector<double>& x,
        double alfa){
116     Matrix M = A;
117     x.resize(b.size());
118     vector<double> last(b.size(), 0.0), r = b;
119     double coeff = 0.0;
120
121     // commpute new matrix alfa and vector beta
122     if(!M.is_quadratic()){
123         throw "Wrong matrix! Try again!";
124     }
125     for(int i = 0; i < M.get_n(); ++i){
126         if(!A[i][i]){
127             throw "Wrong matrix! Try again!";
128         }
```

```
129         for(int j = 0; j < M.get_m(); ++j){
130             M[i][j] = i == j ? 0.0 : -A[i][j] / A[i][i];
131         }
132         r[i] /= A[i][i];
133     }
134 
135     x = r;
136     coeff = M.get_norm();
137     if(coeff < 1.0){
138         coeff /= 1 - coeff;
139     }else{
140         coeff = 1.0;
141     }
142 
143     /* MAKE ITTERATIONS HERE: */
144 
145     int itter = 0;
146 
147     for(itter = 0; coeff * norm_of_vector(vector_minus(x, last)) > alfa; ++itter){
148         x.swap(last);
149         x = vector_plus(r, M * last);
150     }
151 
152     return itter;
153 }
154 
155 int zeidels_method(const Matrix& A, const vector<double>& b, vector<double>& x, double
        alfa){
156     Matrix M = A;
157     x.resize(b.size());
158     vector<double> last(b.size(), 0.0), r = b;
159     double coeff = 0.0;
160 
161     // commpute new matrix alfa and vector beta
162     if(!M.is_quadratic()){
163         throw "Wrong matrix! Try again!";
164     }
165     for(int i = 0; i < M.get_n(); ++i){
166         if(!A[i][i]){
167             throw "Wrong matrix! Try again!";
168         }
169         for(int j = 0; j < M.get_m(); ++j){
170             M[i][j] = i == j ? 0.0 : -A[i][j] / A[i][i];
171         }
172         r[i] /= A[i][i];
173     }
174 
175     x = r;
176 
```

```
177     coeff = M.get_norm();
178     if(coeff < 1){
179         coeff = M.get_upper_norm() / (1 - coeff);
180     }else{
181         coeff = 1.0;
182     }
183 
184     // coeff /= 1 - M.get_norm();
185 
186     /* MAKE ITTERATIONS HERE: */
187 
188     int itter = 0;
189 
190     for(itter = 0; coeff * norm_of_vector(vector_minus(x, last)) > alfa; ++itter){
191         x.swap(last);
192         x = r;
193         for(int i = 0; i < M.get_n(); ++i){
194             for(int j = 0; j < i; ++j){
195                 x[i] += x[j] * M[i][j];
196             }
197             for(int j = i; j < M.get_m(); ++j){
198                 x[i] += last[j] * M[i][j];
199             }
200         }
201     }
202 
203     return itter;
204 }
205 
206 
207 int main(){
208     ave();
209 
210     Matrix A;
211     vector<double> x, b;
212     vector<vector<double>> vec;
213     double accuracy = 0.001;
214     int size = size_init();
215 
216     matrix_init(A, size);
217     vector_init(b, size);
218     cin >> accuracy;
219     print_statement(A, b, accuracy);
220 
221     //itterations:
222     cout << "Simple Itterations Method:" << endl;
223     int itter = simple_itteration(A, b, x, accuracy);
224     print_solution(x, itter);
225 
```

```
    //zeidel:
    cout << "Zeidels Method of Itterations:" << endl;
    itter = zeidels_method(A, b, x, accuracy);
    print_solution(x, itter);
    bye();

    return 0;
}
```

Задание №4:

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

using namespace std;

void ave(){
    cout << "============================================================" << endl;
    cout << "| LABORATORY WORK №1 |" << endl;
    cout << "| NUMERICAL METHODS |" << endl;
    cout << "| Task №4 |" << endl;
    cout << "| Variant №4 |" << endl;
    cout << "| |" << endl;
    cout << "| Student: Bronnikov M.A. |" << endl;
    cout << "| Date: 24.02.2020 |" << endl;
    cout << "| |" << endl;
    cout << "| |" << endl;
    cout << "| Moscow, 2020 |" << endl;
    cout << "============================================================" << endl;
}

void print_statement(const Matrix& A, double alfa){
    cout << "============================================================" << endl;
    cout << "| EXERCSICE: |" << endl;
    cout << "============================================================" << endl;
    cout << "| We need to find vectors and values of simmetric matrix |" << endl;
    cout << "| by rotate method. |" << endl;
    cout << "============================================================" << endl;
    cout << "Simmetric matrix:" << endl;
    cout << A << endl;
    cout << "Accuracy: " << alfa << endl;
    cout << "============================================================" << endl;
}

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

```
40 }
41
42
43 int size_init(){
44     int size;
45     cin >> size;
46     return size;
47 }
48
49 void matrix_init(Matrix& A, int size){
50     A = Matrix(size, size);
51     for(int i = 0; i < size; ++i){
52         for(int j = 0; j < size; ++j){
53             cin >> A[i][j];
54         }
55     }
56 }
57
58
59
60 void print_vector_x(const vector<double>& x){
61     for(unsigned i = 0; i < x.size(); ++i){
62         cout << 'l' << i + 1 << " = " << x[i] << " ";
63     }
64     cout << endl;
65 }
66
67 void print_solution(const vector<double>& x, const Matrix& U, int itter){
68     cout << "============================================================" << endl;
69     cout << "| ANSWER: |" << endl;
70     cout << "============================================================" << endl;
71     cout << "Values:" << endl;
72     print_vector_x(x);
73     cout << "============================================================" << endl;
74     cout << "Vectors:" << endl;
75     for(int j = 0; j < U.get_m(); ++j){
76         cout << "x" << j + 1 << ": " << endl;
77         for(int i = 0; i < U.get_n(); ++i){
78             cout.width(8);
79             cout << U[i][j] << endl;
80         }
81         cout << endl;
82     }
83     cout << "============================================================" << endl;
84     cout << "Itterations: " << itter << endl;
85     cout << "============================================================" << endl;
86 }
87
88 int rotate_method(const Matrix& A, Matrix& U, vector<double>& x, double alfa){
```

```
89      if(!A.is_simmetric()){
90          throw "Matrix not simmteric! Wrong!";
91      }
92
93      Matrix U_k(A.get_n(), A.get_m()), A_k = A;
94      int i_max = 0, j_max = 0, itter = 0;
95      double v_max = 0.0, fitta = 0.0, check = 0.0;
96
97      U = Matrix(A.get_n(), A.get_m());
98      U.make_ones();
99      x.resize(A.get_m());
100
101     do{
102         U_k.make_ones();
103
104         // search max elem in matrix:
105         v_max = 0.0;
106         i_max = j_max = 0;
107         for(int i = 0; i < A_k.get_n(); ++i){
108             for(int j = i + 1; j < A_k.get_m(); ++j){
109                 if(v_max < abs(A_k[i][j])){
110                     v_max = abs(A_k[i][j]);
111                     i_max = i;
112                     j_max = j;
113                 }
114             }
115         }
116
117         // create U:
118         fitta = A_k[i_max][i_max] == A_k[j_max][j_max] ?
119                 M_PI_4 :
120                 atan(2 * A_k[i_max][j_max] / (A_k[i_max][i_max] - A_k[j_max][j_max])) /
                        2;
121
122         U_k[i_max][j_max] = -sin(fitta);
123         U_k[i_max][i_max] = cos(fitta);
124         U_k[j_max][j_max] = cos(fitta);
125         U_k[j_max][i_max] = sin(fitta);
126
127         // multiply:
128         // U for vectors
129         U = U * U_k;
130
131         // A_k for values:
132         A_k = A_k * U_k;
133         U_k.transpose();
134         A_k = U_k * A_k;
135
136
```

```
        // commpute check:
        check = 0.0;
        for(int i = 0; i < A_k.get_n(); ++i){
            for(int j = i + 1; j < A_k.get_m(); ++j){
                check += A_k[i][j] * A_k[i][j];
            }
        }
        check = sqrt(check);
        ++itter;
    }while(check > alfa);

    for(int i = 0; i < A_k.get_n(); ++i){
        x[i] = A_k[i][i];
    }

    return itter;
}


int main(){
    ave();

    Matrix A, U;
    vector<double> x;
    double accuracy = 0.01;
    int size = size_init();
    matrix_init(A, size);
    cin >> accuracy;
    print_statement(A, accuracy);

    //rotate:
    cout << "Rotate Method:" << endl;
    int itter = rotate_method(A, U, x, accuracy);
    print_solution(x, U, itter);
    bye();

    return 0;
}
```

Задание №5:

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

using namespace std;

#define value_pair pair<pair<double, double>, pair<double, double>>
```

```
10
11  void ave(){
12      cout << "============================================================" << endl;
13      cout << "| LABORATORY WORK №1 |" << endl;
14      cout << "| NUMERICAL METHODS |" << endl;
15      cout << "| Task №5 |" << endl;
16      cout << "| Variant №4 |" << endl;
17      cout << "| |" << endl;
18      cout << "| Student: Bronnikov M.A. |" << endl;
19      cout << "| Date: 25.02.2020 |" << endl;
20      cout << "| |" << endl;
21      cout << "| |" << endl;
22      cout << "| Moscow, 2020 |" << endl;
23      cout << "============================================================" << endl;
24  }
25
26  void print_statement(const Matrix& A, double alfa){
27      cout << "============================================================" << endl;
28      cout << "| EXERCSICE: |" << endl;
29      cout << "============================================================" << endl;
30      cout << "| We need to find values of matrix using QR separation. |" << endl;
31      cout << "============================================================" << endl;
32      cout << "Matrix:" << endl;
33      cout << A << endl;
34      cout << "Accuracy: " << alfa << endl;
35      cout << "============================================================" << endl;
36  }
37
38  void bye(){
39      cout << "============================================================" << endl;
40      cout << "| EXIT |" << endl;
41      cout << "============================================================" << endl;
42  }
43
44  int size_init(){
45      int size;
46      cin >> size;
47      return size;
48  }
49
50  void matrix_init(Matrix& A, int size){
51      A = Matrix(size, size);
52      for(int i = 0; i < size; ++i){
53          for(int j = 0; j < size; ++j){
54              cin >> A[i][j];
55          }
56      }
57  }
58
```

```
59
60
61  void print_vector_x(const vector<pair<double, double>>& x){
62      for(unsigned i = 0; i < x.size(); ++i){
63          cout << 'l' << i + 1 << " = " << x[i].first;
64          if(x[i].second){
65              if(x[i].second > 0){
66                  cout << " + ";
67              }else{
68                  cout << " - ";
69              }
70              cout << abs(x[i].second) << "i";
71          }
72          cout << endl;
73      }
74      cout << endl;
75  }
76
77  void print_solution(const vector<pair<double, double>>& x, int itter){
78      cout << "============================================================" << endl;
79      cout << "| ANSWER: |" << endl;
80      cout << "============================================================" << endl;
81      cout << "Values:" << endl;
82      print_vector_x(x);
83      cout << "============================================================" << endl;
84      cout << "Itterations: " << itter << endl;
85      cout << "============================================================" << endl;
86  }
87
88  double mult_1xn_nx1_vecs(const vector<double>& left, const vector<double>& right){
89      double ans = 0.0;
90      if(left.size() != right.size()){
91          throw "Wrong sizes of vectors!";
92      }
93      for(unsigned i = 0; i < left.size(); ++i){
94          ans += right[i] * left[i];
95      }
96      return ans;
97  }
98
99  Matrix mult_nx1_1xn_vecs(const vector<double>& left, const vector<double>& right){
100     Matrix ans(left.size(), right.size());
101     for(int i = 0; i < ans.get_n(); ++i){
102         for(int j = 0; j < ans.get_m(); ++j){
103             ans[i][j] = left[i] * right[j];
104         }
105     }
106     return ans;
107 }
```

```
108 |
109 | double sign(double num){
110 |     if(!num){
111 |         return 0.0;
112 |     }
113 |     return num > 0 ? 1.0 : -1.0;
114 | }
115 |
116 | void solve_eq(double a, double b, double c, pair<pair<double, double>, pair<double,
        double>>& ans){
117 |     double D = b * b - 4.0 * a * c;
118 |     if(D >= 0.0){
119 |         ans.first.first = (-b + sqrt(D)) / (2 * a);
120 |         ans.first.second = 0.0;
121 |         ans.second.first = (-b - sqrt(D)) / (2 * a);
122 |         ans.second.second = 0.0;
123 |     }else{
124 |         ans.first.first = -b / (2 * a);
125 |         ans.first.second = sqrt(-D) / (2 * a);
126 |         ans.second.first = -b / (2 * a);
127 |         ans.second.second = -sqrt(-D) / (2 * a);;
128 |     }
129 | }
130 |
131 | double complex_check(const value_pair& last, const value_pair& cur){
132 |     pair<double, double> r1, r2;
133 |     // x[j]
134 |     r1.first = cur.first.first - last.first.first; // x
135 |     r1.second = cur.first.second - last.first.second; // y
136 |     // x[j+1]
137 |     r2.first = cur.second.first - last.second.first; // x
138 |     r2.second = cur.second.second - last.second.second; // y
139 |     return max(sqrt(r1.first*r1.first + r1.second*r1.second), sqrt(r2.first*r2.first +
            r2.second*r2.second));
140 | }
141 |
142 | // QR separate by Hausholder matrix:
143 | void QRseparate_method(const Matrix& A, Matrix& Q, Matrix& R){
144 |     Matrix E(A.get_n(), A.get_m());
145 |     E.make_ones();
146 |     Q = E;
147 |     R = A;
148 |
149 |     for(int j = 0; j < R.get_m() - 1; ++j){
150 |         vector<double> v(R.get_n(), 0.0);
151 |         double norm = 0.0;
152 |
153 |         // compute vector v:
154 |         // first non zero elem:
```

```
155         v[j] = R[j][j];
156         for(int i = j; i < R.get_n(); ++i){
157             norm += R[i][j] * R[i][j];
158         }
159         norm = sqrt(norm);
160         v[j] += sign(R[j][j]) * norm;
161 
162         // another elements:
163         for(int i = j + 1; i < R.get_n(); ++i){
164             v[i] = R[i][j];
165         }
166 
167         // compute Hausdorf matrix:
168         Matrix H = E - (2.0 / mult_1xn_nx1_vecs(v, v)) * mult_nx1_1xn_vecs(v, v);
169 
170         // update marix:
171         Q = Q * H;
172         R = H * R;
173     }
174 }
175 
176 int QRmethod_values(const Matrix& A, vector<pair<double, double>>& x, double alfa){
177     Matrix Q, R, A_k = A;
178     x.resize(A_k.get_m());
179     int itter = 0;
180     double check;
181     value_pair curr;
182     bool flag = true;
183 
184     for(itter = 0; flag; ++itter){
185         QRseparate_method(A_k, Q, R); // separate
186         A_k = R * Q; // update A_k
187 
188         //check:
189         flag = false;
190         for(int j = 0; j < A_k.get_m(); ++j){
191             check = 0.0;
192             for(int i = j + 1; i < A_k.get_n(); ++i){
193                 check += A_k[i][j] * A_k[i][j];
194             }
195             check = sqrt(check);
196 
197             if(check > alfa){
198                 solve_eq(1.0, -A_k[j][j]-A_k[j+1][j+1], A_k[j][j]*A_k[j+1][j+1]-A_k[j
                        +1][j]*A_k[j][j+1], curr);
199                 if(complex_check(curr, value_pair(x[j], x[j+1])) > alfa){
200                     flag = true;
201                 }
202                 x[j] = curr.first;
```

```
203                 x[j + 1] = curr.second;
204                 ++j;
205             }else{
206                 x[j].first = A_k[j][j];
207                 x[j].second = 0.0;
208             }
209         }
210         // end of check
211     }
212     // if flag == false => check is true and stop itteration
213     return itter;
214 }
215 
216 int main(){
217     ave();
218 
219     Matrix A;
220     vector<pair<double, double>> x;
221     double accuracy = 0.01;
222     int size = size_init();
223     matrix_init(A, size);
224     cin >> accuracy;
225     print_statement(A, accuracy);
226 
227     //QR:
228     cout << "QR Method:" << endl;
229     int itter = QRmethod_values(A, x, accuracy);
230     print_solution(x, itter);
231     bye();
232 
233     return 0;
234 }
```

2 Демонстрация работы

```
(base) max@max-Lenovo-B50-30:~/NumMethods/lab1$ make all
g++ -std=c++17 -Wall -pedantic -c task1.cpp
g++ -std=c++17 -Wall -pedantic -c Matrix/matrix.cpp -o Matrix/matrix.o
g++ task1.o Matrix/matrix.o -o task1
rm -rf task1.o
g++ -std=c++17 -Wall -pedantic -c task2.cpp
g++ task2.o Matrix/matrix.o -o task2
rm -rf task2.o
g++ -std=c++17 -Wall -pedantic -c task3.cpp
g++ task3.o Matrix/matrix.o -o task3
rm -rf task3.o
g++ -std=c++17 -Wall -pedantic -c task4.cpp
g++ task4.o Matrix/matrix.o -o task4
rm -rf task4.o
g++ -std=c++17 -Wall -pedantic -c task5.cpp
g++ task5.o Matrix/matrix.o -o task5
rm -rf task5.o
rm -rf Matrix/matrix.o
(base) max@max-Lenovo-B50-30:~/NumMethods/lab1$ ./task1 <data/m
m1.txt  m2.txt  m3.txt  m4.txt  m5.txt
(base) max@max-Lenovo-B50-30:~/NumMethods/lab1$ ./task1 <data/m1.txt
============================================================
|                     LABORATORY WORK №1                   |
|                     NUMERICAL METHODS                    |
|                          Task №1                         |
|                       Variant №4                         |
|                                                          |
|                                Student: Bronnikov M.A.   |
|                                   Date: 23.02.2020       |
|                                                          |
|                                                          |
|                       Moscow,2020                       |
============================================================
============================================================
|                       EXERCSICE:                         |
============================================================
| We need to solve SLAU by LU separating matrix with Gauss |
| method. Find determinant,back matrix,L and U matrix.     |
```

```
============================================================
SLAU matrix:
Matrix 4x4:
2        7       -8        6
4        4        0       -7
-1       -3        6        3
9       -7       -2       -8

Free vector:
-39
41
4
113
============================================================
============================================================
|                       ANSWER:                            |
============================================================
L matrix:
Matrix 4x4:
1        0        0        0
2        1        0        0
-0.5    -0.05        1        0
4.5     3.85 -9.85714        1

============================================================
U matrix:
Matrix 4x4:
2        7       -8        6
0      -10       16      -19
0        0      2.8     5.05
0        0        0  87.9286

============================================================
Back matrix:
Matrix 4x4:
0.102356 0.0718928 0.161251 0.0743298
0.039805 0.111292 0.034931 -0.0544273
-0.00365556 0.0867181 0.154955 -0.0205118
0.0812348 -0.0381803 0.112104 0.0113729

============================================================
```

```
x solution:
x1 = 8 x2 = -3 x3 = 2 x4 = -3
============================================================
Determinant: -4924
============================================================
============================================================
|                          EXIT                            |
============================================================
(base) max@max-Lenovo-B50-30:~/NumMethods/lab1$ ./task2 <data/m2.txt
============================================================
|                     LABORATORY WORK №1                   |
|                     NUMERICAL METHODS                    |
|                          Task №2                         |
|                       Variant №4                         |
|                                                          |
|                                Student: Bronnikov M.A.   |
|                                   Date: 23.02.2020       |
|                                                          |
|                                                          |
|                       Moscow,2020                       |
============================================================
============================================================
|                       EXERCSICE:                         |
============================================================
|  We need to solve SLAU by race method.                   |
============================================================
SLAU matrix:
Matrix 5x5:
-14       -6        0        0        0
-9       15       -1        0        0
0        1      -11        1        0
0        0       -7       12        3
0        0        0        6       -7

Free vector:
-78
-73
-38
77
91
============================================================
```

```
============================================================
|                       ANSWER:                            |
============================================================
x solution:
x1 = 5.99374 x2 = -0.985396 x3 = 4.27539 x4 = 10.0146 x5 = -4.41602
============================================================
============================================================
|                          EXIT                            |
============================================================
(base) max@max-Lenovo-B50-30:~/NumMethods/lab1$ ./task3 <data/m3.txt
============================================================
|                     LABORATORY WORK №1                   |
|                     NUMERICAL METHODS                    |
|                          Task №3                         |
|                       Variant №4                         |
|                                                          |
|                                Student: Bronnikov M.A.   |
|                                   Date: 23.02.2020       |
|                                                          |
|                                                          |
|                       Moscow,2020                       |
============================================================
============================================================
|                       EXERCSICE:                         |
============================================================
|  We need to solve SLAU by simple ittearation and Zeidel  |
|  methods. Analyse count of iiteraions with accuracy.     |
============================================================
SLAU matrix:
Matrix 4x4:
26       -9       -8        8
9      -21       -2        8
-3        2      -18        8
1       -6       -1       11

Free vector:
20
-164
140
-81
```

```
Accuracy: 0.001
============================================================
Simple Itterations Method:
============================================================
|                       ANSWER:                            |
============================================================
x solution:
x1 = 1.99999 x2 = 8.00001 x3 = -9 x4 = -4.00001
============================================================
Itterations: 40
============================================================
Zeidels Method of Itterations:
============================================================
|                       ANSWER:                            |
============================================================
x solution:
x1 = 1.99999 x2 = 7.99999 x3 = -9 x4 = -4
============================================================
Itterations: 11
============================================================
============================================================
|                         EXIT                             |
============================================================
(base) max@max-Lenovo-B50-30:~/NumMethods/lab1$ ./task4 <data/m4.txt
============================================================
|                     LABORATORY WORK №1                   |
|                     NUMERICAL METHODS                    |
|                          Task №4                         |
|                       Variant №4                         |
|                                                          |
|                                 Student: Bronnikov M.A.  |
|                                    Date: 24.02.2020      |
|                                                          |
|                                                          |
|                       Moscow,2020                       |
============================================================
============================================================
|                       EXERCSICE:                         |
============================================================
|  We need to find vectors and values of simmetric matrix  |
|  by rotate method.                                       |
```

```
============================================================
Simmetric matrix:
Matrix 3x3:
8        2       -1
2       -5       -8
-1       -8       -5

Accuracy: 0.01
============================================================
Rotate Method:
============================================================
|                        ANSWER:                           |
============================================================
Values:
l1 = 8.79893 l2 = -13.0241 l3 = 2.22517
============================================================
Vectors:
x1:
0.938312
0.265083
-0.222039

x2:
-0.0340734
0.709877
0.703501

x3:
0.344106
-0.652538
0.675118

============================================================
Itterations: 4
============================================================
============================================================
|                          EXIT                            |
============================================================
(base) max@max-Lenovo-B50-30:~/NumMethods/lab1$ ./task5 <data/m5.txt
============================================================
|                     LABORATORY WORK №1                   |
```

```
|                      NUMERICAL METHODS                       |
|                           Task №5                            |
|                         Variant №4                           |
|                                                              |
|                                  Student: Bronnikov M.A.     |
|                                     Date: 25.02.2020         |
|                                                              |
|                                                              |
|                         Moscow,2020                         |
============================================================
============================================================
|                         EXERCSICE:                           |
============================================================
|  We need to find values of matrix using QR separation.     |
============================================================
Matrix:
Matrix 3x3:
-4       -6       -3
-1        5       -5
6        2        5

Accuracy: 0.05
============================================================
QR Method:
============================================================
|                         ANSWER:                              |
============================================================
Values:
l1 = 7.10756
l2 = -0.553782 + 3.8273i
l3 = -0.553782 -3.8273i

============================================================
Itterations: 11
============================================================
============================================================
|                           EXIT                               |
============================================================
(base) max@max-Lenovo-B50-30:~/NumMethods/lab1$
```

3 Выводы

Выполнив первую лабораторную работу по курсу «Численные методы», я получил новые знания о методах решения систем линейных алгебраических уравнений.

Кроме этого я самостоятельно реализовал самые основные из них, а также узнал об основных проблемах этих методов, таких как наличие нулей на главной диагонали.

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