

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

**Москва, 2024**

# 1 Методы решения задач линейной алгебры

## 1 Постановка задачи

1.1 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.

**Вариант: 19**

$$\begin{cases} -8x_1 + 5x_2 + 8x_3 - 6x_4 = -144 \\ 2x_1 + 7x_2 - 8x_3 - x_4 = 25 \\ -5x_1 - 4x_2 + x_3 - 6x_4 = -21 \\ 5x_1 - 9x_2 - 2x_3 + 8x_4 = 103 \end{cases} \quad (1)$$

## 2 Результаты работы

```
Original matrix
2 -8.00  5.00  8.00  -6.00
3 2.00  7.00 -8.00  -1.00
4 -5.00 -4.00  1.00  -6.00
5 5.00 -9.00 -2.00  8.00
6 U matrix
7 -8.00  5.00  8.00  -6.00
8 0.00  8.25 -6.00  -2.50
9 0.00  0.00 -9.18 -4.41
10 0.00  0.00  0.00  3.08
11 L matrix
12 1.00  0.00  0.00  0.00
13 -0.25  1.00  0.00  0.00
14 0.62 -0.86  1.00  0.00
15 -0.62 -0.71  0.14  1.00
16 L*U matrix
17 -8.00  5.00  8.00  -6.00
18 2.00  7.00 -8.00  -1.00
19 -5.00 -4.00  1.00  -6.00
20 5.00 -9.00 -2.00  8.00
21 Determinant:1867.00
22 Inversed matrix:
23 -0.39 -0.30 -0.10 -0.41
24 0.05  0.04 -0.08 -0.01
25 -0.09 -0.19 -0.09 -0.16
26 0.28  0.19 -0.04  0.32
27 Solution SLAU:
28 x[0] = 9.00
29 x[1] = -6.00
30 x[2] = -6.00
31 x[3] = -1.00
```

Рис. 1: Вывод в консоли

### 3 Исходный код

Lab1.1.cpp

```
1  #include <iostream>
2  #include <vector>
3  #include <fstream>
4
5  using namespace std;
6
7
8  void LU(vector <vector <double>> A, vector <vector <double>> &L,
9         vector <vector <double>> &U, int n) {
10     U = A;
11
12     for(int i = 0; i < n; i++)
13         for(int j = i; j < n; j++)
14             L[j][i]=U[j][i]/U[i][i];
15
16     for(int k = 1; k < n; k++)
17     {
18         for(int i = k-1; i < n; i++)
19             for(int j = i; j < n; j++)
20                 L[j][i]=U[j][i]/U[i][i];
21
22         for(int i = k; i < n; i++)
23             for(int j = k-1; j < n; j++)
24                 U[i][j]=U[i][j]-L[i][k-1]*U[k-1][j];
25     }
26 }
27
28
29 void proisv(vector <vector <double>> A, vector <vector <double>> B,
30            vector <vector <double>> &R, int n)
31 {
32     for(int i = 0; i < n; i++)
33         for(int j = 0; j < n; j++)
34             for(int k = 0; k < n; k++)
35                 R[i][j] += A[i][k] * B[k][j];
36 }
37
38 void show(vector <vector <double>> A, int n)
39 {
40     ofstream fout("answer.txt");
41     for(int i = 0; i < n; i++)
42     {
43         for(int j = 0; j < n; j++)
44         {
45             fout << A[i][j] << "\t";
```

```

46     }
47     fout << endl;
48 }
49 }
50
51 double determinant(vector <vector <double>> U, int n)
52 {
53     double det = 1;
54     for (int i=0; i<n; i++){
55         det *= U[i][i];
56     }
57     return det;
58 }
59
60
61 vector <double> solve_L_matrix(const vector <vector <double>> &L, const vector <double
    > &b, int n) {
62     vector <double> y(n, 0);
63     for (int i = 0; i < n; ++i) {
64         y[i] = b[i];
65         for (int j = 0; j < i; ++j)
66             y[i] -= L[i][j] * y[j];
67     }
68     return y;
69 }
70
71 vector <double> solve_U_matrix(const vector <vector <double>> &U, const vector <double
    > &y, int n){
72     vector<double> x(n, 0);
73     vector<double> E(n, 0);
74     for (int i = 0; i < n; ++i) {
75         E[i] = 1;
76         for (int i = n - 1; i >= 0; --i) {
77             x[i] = y[i];
78             for (int j = i + 1; j < n; ++j)
79                 x[i] -= U[i][j] * x[j];
80             x[i] /= U[i][i];
81         }
82         return x;
83     }
84 }
85
86 vector <double> solve_SLAU(vector <vector <double>> L, vector <vector <double>> U,
    vector <double> b, int n)
87 {
88     vector<double> y = solve_L_matrix(L, b, n);
89     vector<double> x = solve_U_matrix(U, y, n);
90     return x;
91 }

```

```

92
93 vector <vector <double>> inverse_matrix(vector <vector <double>> &A, vector <vector <
    double>> &L, vector <vector <double>> &U, int n){
94     vector<vector<double>> inverse_A(n, vector<double>(n, 0));
95     vector<double> E(n, 0);
96     for (int i = 0; i < n; ++i) {
97         E[i] = 1;
98         vector<double> y = solve_L_matrix(L, E, n);
99         vector<double> x = solve_U_matrix(U, y, n);
100         for (int j = 0; j < n; ++j)
101             inverse_A[j][i] = x[j];
102         E[i] = 0;
103     }
104     return inverse_A;
105 }
106
107
108 int main(){
109     const int n=4;
110     vector <vector <double>> A(n, vector <double>(n, 0));
111     vector <double> b(n,0);
112     ifstream fina("matrix.txt"), finb("column.txt");
113     for (int i = 0; i < A.size(); i++)
114     {
115         for (int j = 0; j < A.size(); j++)
116             fina >> A[i][j];
117     }
118     for (int i = 0; i < b.size(); i++)
119     {
120         finb >> b[i];
121     }
122
123     vector<vector<double>> L(n, vector<double>(0)), U(n, vector<double>(0)), RES(n,
        vector<double>(0));
124     for (int i = 0; i < n; i++){
125         for (int j=0; j<n; j++){
126             L[i].push_back(0);
127             U[i].push_back(0);
128             RES[i].push_back(0);
129         }
130     }
131
132     ofstream fout("answer.txt");
133
134     fout.precision(2);
135     fout << fixed;
136     LU(A,L,U,n);
137     fout << "Original matrix" << endl;
138     for(int i = 0; i < n; i++)

```

```

139 {
140     for(int j = 0; j < n; j++)
141     {
142         fout << A[i][j] << "\t";
143     }
144     fout << endl;
145 }
146 fout << "U matrix" << endl;
147 for(int i = 0; i < n; i++)
148 {
149     for(int j = 0; j < n; j++)
150     {
151         fout << U[i][j] << "\t";
152     }
153     fout << endl;
154 }
155 fout << "L matrix" << endl;
156 for(int i = 0; i < n; i++)
157 {
158     for(int j = 0; j < n; j++)
159     {
160         fout << L[i][j] << "\t";
161     }
162     fout << endl;
163 }
164 proisv(L,U,RES,n);
165 fout << "L*U matrix" << endl;
166 for(int i = 0; i < n; i++)
167 {
168     for(int j = 0; j < n; j++)
169     {
170         fout << RES[i][j] << "\t";
171     }
172     fout << endl;
173 }
174 fout << "Determinant:" << determinant(U, n) << endl;
175 fout << "Inversed matrix:" << endl;
176 vector <vector <double>> inversed = inverse_matrix(A, L, U, n);
177 for(int i = 0; i < n; i++)
178 {
179     for(int j = 0; j < n; j++)
180     {
181         fout << inversed[i][j] << "\t";
182     }
183     fout << endl;
184 }
185 fout << "Solution SLAU:" << endl;
186 vector <double> x = solve_SLAU(L,U,b,n);
187 for (size_t i = 0; i < x.size(); ++i)

```

```
188 |         fout << "x[" << i << "]" = " << x[i] << endl;  
189 |     return 0;  
190 | }
```



## 4 Постановка задачи

1.2. 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.

**Вариант:** 19

$$\begin{cases} 10x_1 - x_2 = 16 \\ -8x_1 + 16x_2 + x_3 = -110 \\ 6x_2 - 16x_3 + 6x_4 = 24 \\ -8x_3 + 16x_4 - 5x_5 = -3 \\ 5x_4 - 13x_5 = 87 \end{cases} \quad (2)$$

## 5 Результаты работы

```
1 Progon_method solution
2 x[0] = 1
3 x[1] = -6
4 x[2] = -6
5 x[3] = -6
6 x[4] = -9
7 |
```

Рис. 2: Вывод в консоли

## 6 Исходный код

Lab1.2.cpp

```
1  #include <iostream>
2  #include <vector>
3  #include <fstream>
4
5  using namespace std;
6
7  vector <double> progon_method(vector <vector <double>> A, vector <double> b, int n){
8      //
9      double y;
10     vector <double> alph(n,0), Bett(n,0), x(n,0);
11     y = A[0][0];
12     alph[0] = -A[0][1] / y;
13     Bett[0] = b[0] / y;
14     for (int i = 1; i < n; i++) {
15         y = A[i][i] + A[i][i - 1] * alph[i - 1];
16         alph[i] = -A[i][i + 1] / y;
17         Bett[i] = (b[i] - A[i][i - 1] * Bett[i - 1]) / y;
18     }
19
20     //
21     for (int i = n - 1; i >= 0; i--) {
22         x[i] = alph[i] * x[i + 1] + Bett[i];
23     }
24
25     return x;
26 }
27
28 int main(){
29     const int n=5;
30     vector <vector <double>> A(n, vector <double>(n, 0));
31     vector <double> b(n,0);
32     ifstream fina("matrix2.txt"), finb("column2.txt");
33     for (int i = 0; i < A.size(); i++)
34     {
35         for (int j = 0; j < A.size(); j++)
36             fina >> A[i][j];
37     }
38     for (int i = 0; i < b.size(); i++)
39     {
40         finb >> b[i];
41     }
42
43     ofstream fout("answer2.txt");
44     vector <double> x = progon_method(A, b, n);
45     for (size_t i = 0; i < x.size(); ++i)
```

```
46 |         fout << "x[" << i << "]" = " << x[i] << endl;  
47 |  
48 |     return 0;  
49 | }
```

## 7 Постановка задачи

1.3. 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.

**Вариант: 19**

$$\begin{cases} 15x_1 + 7x_3 + 5x_4 = 176 \\ -3x_1 - 14x_2 - 6x_3 + x_4 = -111 \\ -2x_1 + 9x_2 + 13x_3 + 2x_4 = 74 \\ 4x_1 - x_2 + 3x_3 + 9x_4 = 76 \end{cases} \quad (3)$$

## 8 Результаты работы

```
eps = 0.001  
  
Method Iteration  
Count of iterations:21  
x[0] = 9.00033  
x[1] = 5.00023  
x[2] = 3.00024  
x[3] = 4.00031  
  
Method Zeidel  
Count of iterations:9  
x[0] = 9.00013  
x[1] = 5.00007  
x[2] = 2.99998  
x[3] = 3.99996
```

Рис. 3: Вывод в консоли

## 9 Исходный код

Lab1.3.cpp

```
1  #include <iostream>
2  #include <vector>
3  #include <fstream>
4  #include <cmath>
5
6  using namespace std;
7
8  vector <double> method_iteration(vector <vector <double>> A, vector <double> b, int n,
   int &counter){
9
10     vector <double> x(n,0);
11     vector <double> xn(n,0);
12     double eps = 0.001;
13     for (int i = 0; i < n; i++) {
14         x[i] = b[i] / A[i][i];
15     }
16     do {
17         for (int i = 0; i < n; i++) {
18             xn[i] = b[i] / A[i][i];
19             for (int j = 0; j < n; j++) {
20                 if (i == j)
21                     continue;
22                 else {
23                     xn[i] -= A[i][j] / A[i][i] * x[j];
24                 }
25             }
26         }
27
28         bool flag = true;
29         for (int i = 0; i < n - 1; i++) {
30             if (fabs(xn[i] - x[i]) > eps) {
31                 flag = false;
32                 break;
33             }
34         }
35
36         for (int i = 0; i < n; i++) {
37             x[i] = xn[i];
38         }
39
40         if (flag)
41             break;
42         counter++;
43     } while (1);
44     return x;
```

```

45 }
46
47 vector<double> method_zeidel(vector<vector<double>> A, vector<double> b, int n,
48     int &counter){
49     vector<double> x(n, 0);
50     vector<double> x_prev(n, 0);
51     double eps = 0.001;
52     double diff;
53     int max_iter = 1000;
54
55     do {
56         x_prev = x;
57         for (int i = 0; i < n; i++) {
58             double sum1 = 0;
59             double sum2 = 0;
60             for (int j = 0; j < i; j++) {
61                 sum1 += A[i][j] * x[j];
62             }
63             for (int j = i + 1; j < n; j++) {
64                 sum2 += A[i][j] * x_prev[j];
65             }
66             x[i] = (b[i] - sum1 - sum2) / A[i][i];
67         }
68
69         diff = 0;
70         for (int i = 0; i < n; i++) {
71             diff += abs(x[i] - x_prev[i]);
72         }
73
74         counter++;
75     } while (diff > eps);
76
77     return x;
78 }
79
80 int main() {
81     const int n=4;
82     vector<vector<double>> A(n, vector<double>(n, 0));
83     vector<double> b(n,0);
84     ifstream fina("matrix3.txt"), finb("column3.txt");
85     for (int i = 0; i < A.size(); i++)
86     {
87         for (int j = 0; j < A.size(); j++)
88             fina >> A[i][j];
89     }
90     for (int i = 0; i < b.size(); i++)
91     {
92         finb >> b[i];

```



```

93     }
94
95     int counter = 0;
96     vector <double> x = method_iteration(A, b, n, counter);
97     ofstream fout("answer3.txt");
98     fout << "eps = 0.001\n" << endl;
99     fout << "Method Iteration" << endl;
100    fout << "Count of iterations:" << counter << endl;
101    for (size_t i = 0; i < x.size(); ++i)
102        fout << "x[" << i << "] = " << x[i] << endl;
103    counter = 0;
104    x = method_zeidel(A,b, n, counter);
105    fout << "\nMethod Zeidel" << endl;
106    fout << "Count of iterations:" << counter << endl;
107    for (size_t i = 0; i < x.size(); ++i)
108        fout << "x[" << i << "] = " << x[i] << endl;
109
110    return 0;
111 }

```

## 10 Постановка задачи

1.4. 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.

**Вариант: 19**

$$\begin{pmatrix} -8 & 9 & 6 \\ 9 & 9 & 1 \\ 6 & 1 & 8 \end{pmatrix} \quad (4)$$

## 11 Результаты работы

```
eps = 0.1  
  
Sobstn values:  
-13.1414  
14.7574  
7.38403  
  
Sobstn vectors:  
0.903166    0.429262    0.00495295  
-0.356296    0.755983    -0.549129  
-0.239465    0.49419    0.835723
```

Рис. 4: Вывод в консоли

## 12 Исходный код

Lab1.4.cpp

```
1  #include <iostream>
2  #include <cmath>
3  #include <vector>
4  #include <fstream>
5
6  using namespace std;
7
8  vector <vector <double>> proisv(vector <vector <double>> A, vector <vector <double>> B
    , int n){
9      vector <vector <double>> R(n, vector <double>(n,0));
10     for(int i = 0; i < n; i++)
11         for(int j = 0; j < n; j++)
12             for(int k = 0; k < n; k++)
13                 R[i][j] += A[i][k] * B[k][j];
14     return R;
15 }
16
17
18 vector<vector<double>> transpose(const vector<vector<double>>& matrix) {
19     int n = matrix.size();
20     vector<vector<double>> result(n, vector<double>(n));
21     for (int i = 0; i < n; ++i) {
22         for (int j = 0; j < n; ++j) {
23             result[i][j] = matrix[j][i];
24         }
25     }
26
27     return result;
28 }
29
30 void jacobiAlgorithm(vector<vector<double>>& A, vector<double>& sobs_values, vector<
    vector<double>>& sobs_vectors) {
31     int n = A.size();
32     sobs_vectors = vector<vector<double>>(n, vector<double>(n, 0));
33     for (int i = 0; i < n; ++i) sobs_vectors[i][i] = 1;
34
35     vector<vector<double>> B = A;
36     double eps = 0.1;
37     while (true) {
38         double maxVal = 0;
39         int p, q;
40         for (int i = 0; i < n; ++i) {
41             for (int j = i + 1; j < n; ++j) {
42                 if (fabs(B[i][j]) > maxVal) {
43                     maxVal = fabs(B[i][j]);
```

```

44         p = i;
45         q = j;
46     }
47 }
48 }
49 if (maxVal < eps) break;
50 double pi = acos(-1);
51 double theta = pi / 4;
52 if (abs(B[q][q] - B[p][p]) > 1e-7)
53     theta = 0.5 * atan((2 * B[q][p]) / (B[q][q] - B[p][p]));
54 double sinTheta = sin(theta);
55 double cosTheta = cos(theta);
56
57 vector<vector<double>> rotationMatrix(n, vector<double>(n, 0));
58 for (int i = 0; i < n; ++i) rotationMatrix[i][i] = 1;
59 rotationMatrix[p][p] = cosTheta;
60 rotationMatrix[p][q] = sinTheta;
61 rotationMatrix[q][p] = -sinTheta;
62 rotationMatrix[q][q] = cosTheta;
63
64 vector<vector<double>> res = proisv(transpose(rotationMatrix), B, n);
65 res = proisv(res, rotationMatrix, n);
66 B = res;
67 sobs_vectors = proisv(sobs_vectors, rotationMatrix, n);
68 }
69
70 sobs_values.resize(n);
71 for (int i = 0; i < n; ++i) {
72     sobs_values[i] = B[i][i];
73 }
74 }
75
76 int main() {
77     int n = 3;
78     ifstream fina("matrix4.txt");
79     ofstream fout("answer4.txt");
80     vector<vector<double>> A(n, vector<double>(n, 0));
81     for (int i = 0; i < A.size(); i++) {
82         for (int j = 0; j < A.size(); j++)
83             fina >> A[i][j];
84     }
85     vector<double> sobs_values;
86     vector<vector<double>> sobs_vectors;
87
88     jacobiAlgorithm(A, sobs_values, sobs_vectors);
89     fout << "eps = 0.1\n" << endl;
90     fout << "Sobstv values:" << endl;
91     for (size_t i = 0; i < sobs_values.size(); ++i)
92         fout << sobs_values[i] << endl;

```

```

93     fout << endl;
94     fout << "Sobstv vectors:" << endl;
95     for(int i = 0; i < n; i++) {
96         for(int j = 0; j < n; j++){
97             fout << sobs_vectors[i][j] << "\t";
98         }
99         fout << endl;
100     }
101
102     return 0;
103 }

```

## 13 Постановка задачи

1.5. 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.

**Вариант: 19**

$$\begin{pmatrix} 0 & -1 & 3 \\ -1 & 6 & -3 \\ -8 & 4 & 2 \end{pmatrix} \quad (5)$$

## 14 Результаты работы

```
Q matrix:
0    -0.180233    0.983624
-0.124035    0.976028    0.178841
-0.992278    -0.122004    -0.0223551

R matrix:
8.06226 -4.71332    -1.61245
3.33067e-16 5.54839 -3.71279
1.13798e-15 -9.71445e-16    2.36964

Sobsv_values:
(2.31769,5.04027)
(2.31769,-5.04027)
(3.36463,0)
```

Рис. 5: Вывод в консоли



## 15 Исходный код

Lab1.5.cpp

```
1  #include <iostream>
2  #include <vector>
3  #include <cmath>
4  #include <fstream>
5  #include <complex>
6
7  using namespace std;
8
9  vector<vector<double>> transpose(vector<vector<double>>matrix) {
10     int n = matrix.size();
11     vector<vector<double>> result(n, vector<double>(n));
12     for (int i = 0; i < n; ++i) {
13         for (int j = 0; j < n; ++j) {
14             result[i][j] = matrix[j][i];
15         }
16     }
17     return result;
18 }
19
20 double norma(vector <double> A){
21     int n = A.size();
22     double norm = 0;
23     for (int i=0; i<n; i++){
24         norm += A[i] * A[i];
25     }
26
27     return sqrt(norm);
28 }
29
30 double scalar_proisv(vector<double> first, vector<double> second){
31     double sum = 0;
32     int n = first.size();
33     for (int i=0; i<n; i++){
34         sum += first[i] * second[i];
35     }
36     return sum;
37 }
38
39
40 void normalize_matrix(vector<vector <double>>& A) {
41     int n = A.size();
42     for (int i = 0; i<n; i++){
43         double norm = norma(A[i]);
44         for (int j=0;j<n;j++){
45             A[i][j] /= norm;
```

```

46     }
47 }
48 }
49
50 vector<double> subtract_column(vector<double> first, vector<double> second){
51     int n = first.size();
52     vector<double> res(n,0);
53     for (int i = 0; i<n; i++){
54         res[i] = first[i] - second[i];
55     }
56     return res;
57 }
58
59 vector<vector<double>> gramSchmidt(vector<vector<double>> A) {
60     int n = A.size();
61     vector<vector<double>> transposed_matrix = transpose(A);
62     vector<vector<double>> B(n, vector<double>(n,0));
63     B[0] = transposed_matrix[0];
64     for (int i=1; i<n; i++){
65         vector<double> projection(n,0);
66         for (int count=0; count<i; count++){
67             double k = scalar_proisv(transposed_matrix[i], B[count]) / scalar_proisv(B[
68                 count], B[count]);
69             for (int j = 0; j<n; j++){
70                 projection[j] += k * B[count][j];
71             }
72             B[i] = subtract_column(transposed_matrix[i], projection);
73         }
74         return B;
75     }
76
77     vector<vector<double>> proisv(vector<vector<double>> A, vector<vector<double>> B
78         ){
79         int n = A.size();
80         vector<vector<double>> R(n, vector<double>(n,0));
81         for(int i = 0; i < n; i++)
82             for(int j = 0; j < n; j++)
83                 for(int k = 0; k < n; k++)
84                     R[i][j] += A[i][k] * B[k][j];
85         return R;
86     }
87
88     void QR_decomp(vector<vector<double>> &A, vector<vector<double>> &Q, vector<vector<
89         double>> &R){
90         Q = gramSchmidt(A);
91         normalize_matrix(Q);
92         Q = transpose(Q);

```

```

92     R = proisv(transpose(Q), A);
93 }
94
95 vector <complex<double>> sobs_values(vector<vector<double>> A, double eps)
96 {
97     int n = A.size();
98     vector <complex<double>> prev_sobs(n);
99     vector<vector<double>> Q, R;
100    vector <complex<double>> cur_sobs(n,0);
101    while (true) {
102        QR_decomp(A,Q,R);
103        A = proisv(R,Q);
104        for (int i = 0; i < n; i++) {
105            if (i < n - 1 && abs(A[i + 1][i]) > 1e-7) {
106                double b = -(A[i][i] + A[i + 1][i + 1]);
107                double c = A[i][i] * A[i + 1][i + 1] - A[i][i + 1] * A[i + 1][i];
108                double discriminant = b * b - 4 * c;
109
110                if (discriminant > 0) {
111                    cur_sobs[i] = 0.5 * (-b - sqrt(discriminant));
112                    cur_sobs[i + 1] = 0.5 * (-b + sqrt(discriminant));
113                    i++;
114                } else {
115                    cur_sobs[i] = complex<double>(-b / 2, sqrt(-discriminant) / 2);
116                    cur_sobs[i + 1] = complex<double>(-b / 2, -sqrt(-discriminant) / 2);
117                    i++;
118                }
119            } else {
120                cur_sobs[i] = A[i][i];
121            }
122        }
123        bool ok = true;
124        for (int i = 0; i < n; i++) {
125            ok = ok && abs(cur_sobs[i] - prev_sobs[i]) < eps;
126        }
127        if (ok)
128            break;
129        prev_sobs = cur_sobs;
130    }
131    return prev_sobs;
132 }
133
134 int main() {
135     int n = 3;
136     ifstream fina("matrix5.txt");
137     ofstream fout("answer5.txt");
138     vector <vector <double>> A(n, vector <double>(n, 0));
139     for (int i = 0; i < A.size(); i++){
140         for (int j = 0; j < A.size(); j++)

```

```

141         fina >> A[i][j];
142     }
143     vector<vector<double>> Q, R;
144     QR_decomp(A,Q,R);
145
146     fout << "Q matrix:" << endl;
147     for(int i = 0; i < n; i++){
148         for(int j = 0; j < n; j++){
149             fout << Q[i][j] << "\t";
150         }
151         fout << endl;
152     }
153
154     fout << "\nR matrix:" << endl;
155     for(int i = 0; i < n; i++){
156         for(int j = 0; j < n; j++){
157             fout << R[i][j] << "\t";
158         }
159         fout << endl;
160     }
161     double eps = 0.1;
162     vector<complex<double>> sobstv = sobs_values(A, eps);
163     fout << "\nSobsv_values:" << endl;
164     for(int i = 0; i < n; i++){
165         fout << sobstv[i] << "\t";
166     }
167     fout << endl;
168     return 0;
169 }

```