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

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

2 Результаты работы

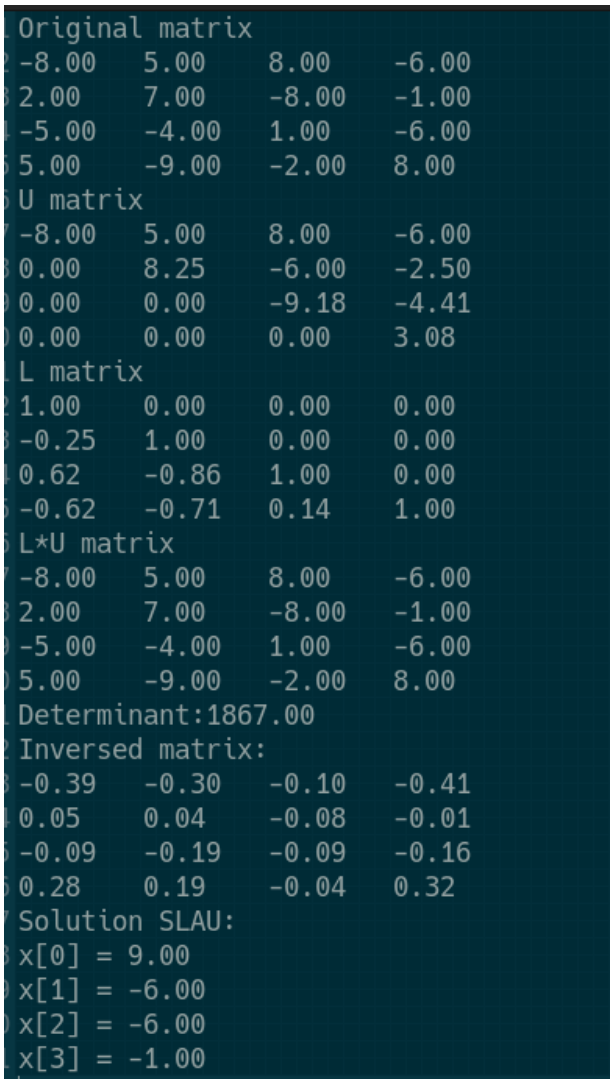

Рис. 1: Вывод в консоли

3 Исходный код

Lab1.1.cpp

```cpp
#include <iostream>
#include <vector>
#include <fstream>

using namespace std;


void LU(vector <vector <double>> A, vector <vector <double>> &L,
       vector <vector <double>> &U, int n) {
    U = A;

  for(int i = 0; i < n; i++)
    for(int j = i; j < n; j++)
      L[j][i]=U[j][i]/U[i][i];

  for(int k = 1; k < n; k++)
  {
    for(int i = k-1; i < n; i++)
      for(int j = i; j < n; j++)
        L[j][i]=U[j][i]/U[i][i];

    for(int i = k; i < n; i++)
      for(int j = k-1; j < n; j++)
        U[i][j]=U[i][j]-L[i][k-1]*U[k-1][j];
  }
}


void proisv(vector <vector <double>> A, vector <vector <double>> B,
     vector <vector <double>> &R, int n)
{
  for(int i = 0; i < n; i++)
    for(int j = 0; j < n; j++)
      for(int k = 0; k < n; k++)
        R[i][j] += A[i][k] * B[k][j];
}

void show(vector <vector <double>> A, int n)
{
  ofstream fout("answer.txt");
  for(int i = 0; i < n; i++)
  {
    for(int j = 0; j < n; j++)
    {
      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++)
```

```
{
  for(int j = 0; j < n; j++)
  {
    fout << A[i][j] << "\t";
  }
  fout << endl;
}
fout << "U matrix" << endl;
for(int i = 0; i < n; i++)
{
  for(int j = 0; j < n; j++)
  {
    fout << U[i][j] << "\t";
  }
  fout << endl;
}
fout << "L matrix" << endl;
for(int i = 0; i < n; i++)
{
  for(int j = 0; j < n; j++)
  {
    fout << L[i][j] << "\t";
  }
  fout << endl;
}
proisv(L,U,RES,n);
fout << "L*U matrix" << endl;
for(int i = 0; i < n; i++)
{
  for(int j = 0; j < n; j++)
  {
    fout << RES[i][j] << "\t";
  }
  fout << endl;
}
fout << "Determinant:" << determinant(U, n) << endl;
fout << "Inversed matrix:" << endl;
vector <vector <double>> inversed = inverse_matrix(A, L, U, n);
for(int i = 0; i < n; i++)
{
  for(int j = 0; j < n; j++)
  {
    fout << inversed[i][j] << "\t";
  }
  fout << endl;
}
fout << "Solution SLAU:" << endl;
vector <double> x = solve_SLAU(L,U,b,n);
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

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

5 Результаты работы

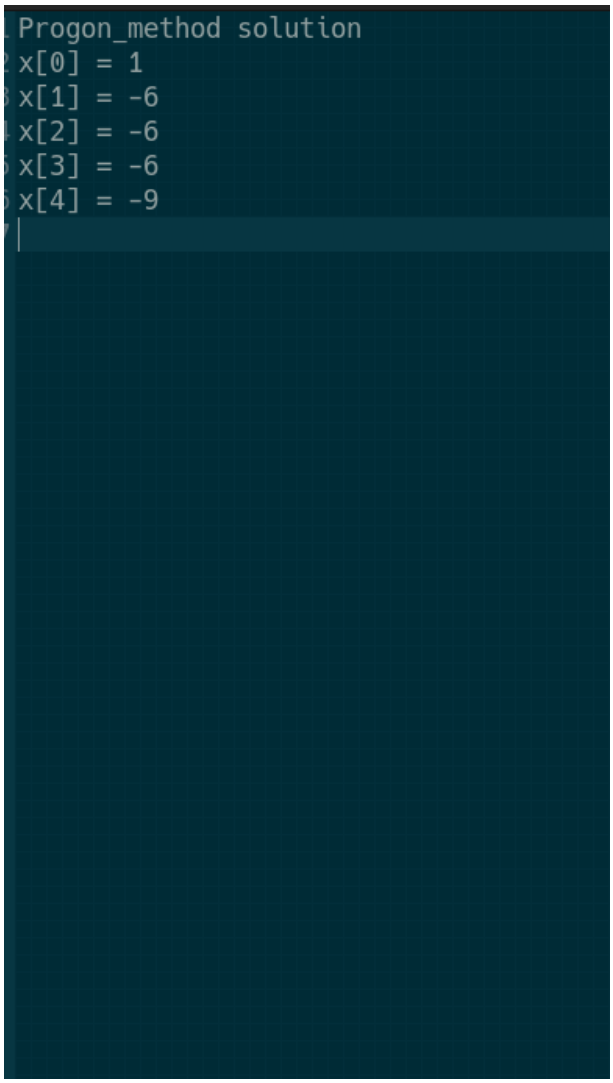

Рис. 2: Вывод в консоли

6 Исходный код

Lab1.2.cpp

```cpp
#include <iostream>
#include <vector>
#include <fstream>

using namespace std;

vector <double> progon_method(vector <vector <double>> A, vector <double> b, int n){
    //
    double y;
    vector <double> alph(n,0), Bett(n,0), x(n,0);
    y = A[0][0];
    alph[0] = -A[0][1] / y;
    Bett[0] = b[0] / y;
    for (int i = 1; i < n; i++) {
        y = A[i][i] + A[i][i - 1] * alph[i - 1];
        alph[i] = -A[i][i + 1] / y;
        Bett[i] = (b[i] - A[i][i - 1] * Bett[i - 1]) / y;
    }

    //
    for (int i = n - 1; i >= 0; i--) {
        x[i] = alph[i] * x[i + 1] + Bett[i];
    }

    return x;
}

int main(){
  const int n=5;
  vector <vector <double>> A(n, vector <double>(n, 0));
  vector <double> b(n,0);
  ifstream fina("matrix2.txt"), finb("column2.txt");
    for (int i = 0; i < A.size(); i++)
    {
        for (int j = 0; j < A.size(); j++)
            fina >> A[i][j];
    }
    for (int i = 0; i < b.size(); i++)
    {
        finb >> b[i];
    }

    ofstream fout("answer2.txt");
    vector <double> x = progon_method(A, b, n);
    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

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

8 Результаты работы

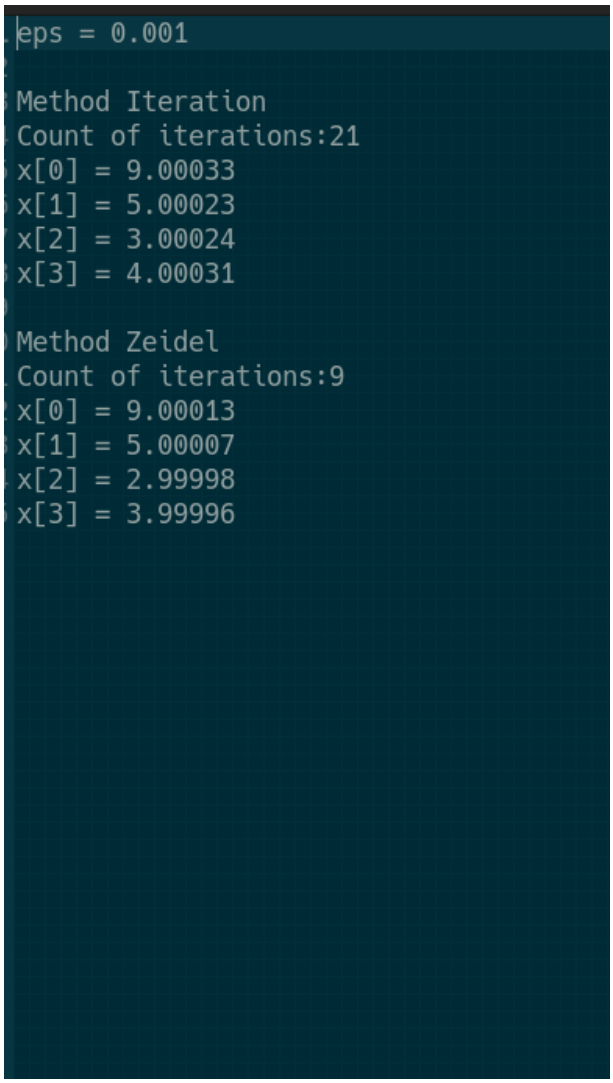

Рис. 3: Вывод в консоли

9 Исходный код

Lab1.3.cpp

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

using namespace std;

vector <double> method_iteration(vector <vector <double>> A, vector <double> b, int n,
     int &counter){

    vector <double> x(n,0);
    vector <double> xn(n,0);
    double eps = 0.001;
    for (int i = 0; i < n; i++) {
    x[i] = b[i] / A[i][i];
  }
    do {
    for (int i = 0; i < n; i++) {
      xn[i] = b[i] / A[i][i];
      for (int j = 0; j < n; j++) {
        if (i == j)
          continue;
        else {
          xn[i] -= A[i][j] / A[i][i] * x[j];
        }
      }
    }

    bool flag = true;
    for (int i = 0; i < n - 1; i++) {
      if (fabs(xn[i] - x[i]) > eps) {
        flag = false;
        break;
      }
    }

    for (int i = 0; i < n; i++) {
      x[i] = xn[i];
    }

    if (flag)
      break;
        counter++;
  } while (1);
    return x;
```

```
}

vector <double> method_zeidel(vector <vector <double>> A, vector <double> b, int n,
    int &counter){
    vector<double> x(n, 0);
    vector<double> x_prev(n, 0);
    double eps = 0.001;
    double diff;
    int max_iter = 1000;


    do {
        x_prev = x;
        for (int i = 0; i < n; i++) {
            double sum1 = 0;
            double sum2 = 0;
            for (int j = 0; j < i; j++) {
                sum1 += A[i][j] * x[j];
            }
            for (int j = i + 1; j < n; j++) {
                sum2 += A[i][j] * x_prev[j];
            }
            x[i] = (b[i] - sum1 - sum2) / A[i][i];
        }

        diff = 0;
        for (int i = 0; i < n; i++) {
            diff += abs(x[i] - x_prev[i]);
        }

        counter++;
    } while (diff > eps);

    return x;
}

int main() {
    const int n=4;
  vector <vector <double>> A(n, vector <double>(n, 0));
  vector <double> b(n,0);
  ifstream fina("matrix3.txt"), finb("column3.txt");
    for (int i = 0; i < A.size(); i++)
    {
        for (int j = 0; j < A.size(); j++)
            fina >> A[i][j];
    }
    for (int i = 0; i < b.size(); i++)
    {
        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

$$\begin{pmatrix} -8 & 9 & 6 \\ 9 & 9 & 1 \\ 6 & 1 & 8 \end{pmatrix} \tag{4}$$

11 Результаты работы

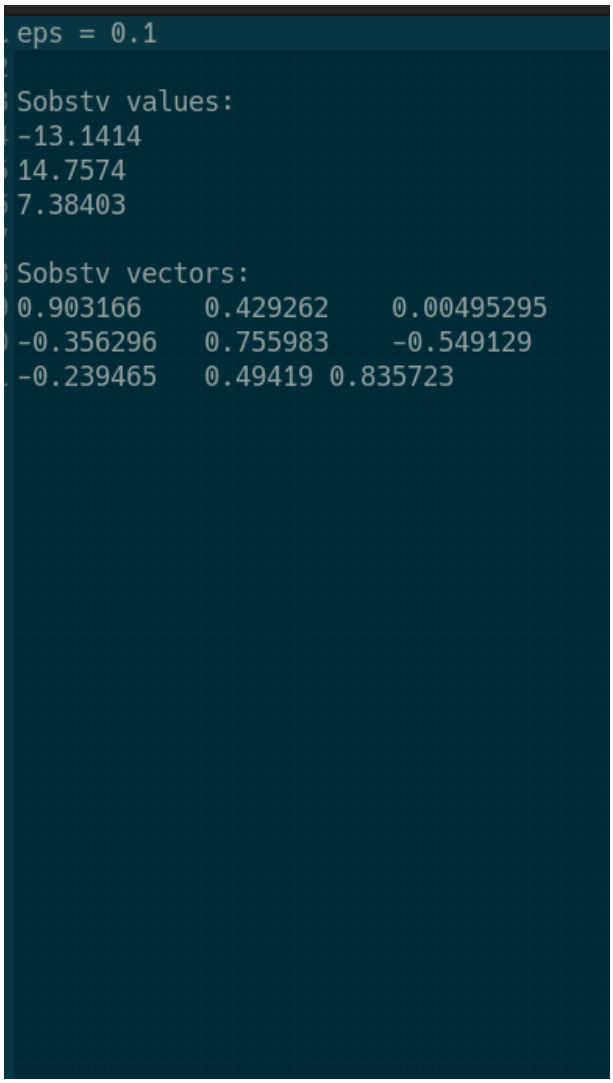

Рис. 4: Вывод в консоли

12 Исходный код

Lab1.4.cpp

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

using namespace std;

vector <vector <double>> proisv(vector <vector <double>> A, vector <vector <double>> B
    , int n){
    vector <vector <double>> R(n, vector <double>(n,0));
  for(int i = 0; i < n; i++)
    for(int j = 0; j < n; j++)
      for(int k = 0; k < n; k++)
        R[i][j] += A[i][k] * B[k][j];
    return R;
}


vector<vector<double>> transpose(const vector<vector<double>>& matrix) {
    int n = matrix.size();
    vector<vector<double>> result(n, vector<double>(n));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            result[i][j] = matrix[j][i];
        }
    }

    return result;
}

void jacobiAlgorithm(vector<vector<double>>& A, vector<double>& sobs_values, vector<
    vector<double>>& sobs_vectors) {
    int n = A.size();
    sobs_vectors = vector<vector<double>>(n, vector<double>(n, 0));
    for (int i = 0; i < n; ++i) sobs_vectors[i][i] = 1;

    vector<vector<double>> B = A;
    double eps = 0.1;
    while (true) {
        double maxVal = 0;
        int p, q;
        for (int i = 0; i < n; ++i) {
            for (int j = i + 1; j < n; ++j) {
                if (fabs(B[i][j]) > maxVal) {
                    maxVal = fabs(B[i][j]);
```

```
                    p = i;
                    q = j;
                }
            }
        }
        if (maxVal < eps) break;
        double pi = acos(-1);
        double theta = pi / 4;
        if (abs(B[q][q] - B[p][p]) > 1e-7)
            theta = 0.5 * atan((2 * B[q][p]) / (B[q][q] - B[p][p]));
        double sinTheta = sin(theta);
        double cosTheta = cos(theta);

        vector<vector<double>> rotationMatrix(n, vector<double>(n, 0));
        for (int i = 0; i < n; ++i) rotationMatrix[i][i] = 1;
        rotationMatrix[p][p] = cosTheta;
        rotationMatrix[p][q] = sinTheta;
        rotationMatrix[q][p] = -sinTheta;
        rotationMatrix[q][q] = cosTheta;

        vector <vector <double>> res = proisv(transpose(rotationMatrix), B, n);
        res = proisv(res, rotationMatrix, n);
        B = res;
        sobs_vectors = proisv(sobs_vectors, rotationMatrix, n);
    }

    sobs_values.resize(n);
    for (int i = 0; i < n; ++i) {
        sobs_values[i] = B[i][i];
    }
}

int main() {
    int n = 3;
    ifstream fina("matrix4.txt");
    ofstream fout("answer4.txt");
    vector <vector <double>> A(n, vector <double>(n, 0));
    for (int i = 0; i < A.size(); i++) {
        for (int j = 0; j < A.size(); j++)
            fina >> A[i][j];
    }
    vector<double> sobs_values;
    vector<vector<double>> sobs_vectors;

    jacobiAlgorithm(A, sobs_values, sobs_vectors);
    fout << "eps = 0.1\n" << endl;
    fout << "Sobstv values:" << endl;
    for (size_t i = 0; i < sobs_values.size(); ++i)
        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

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

14 Результаты работы

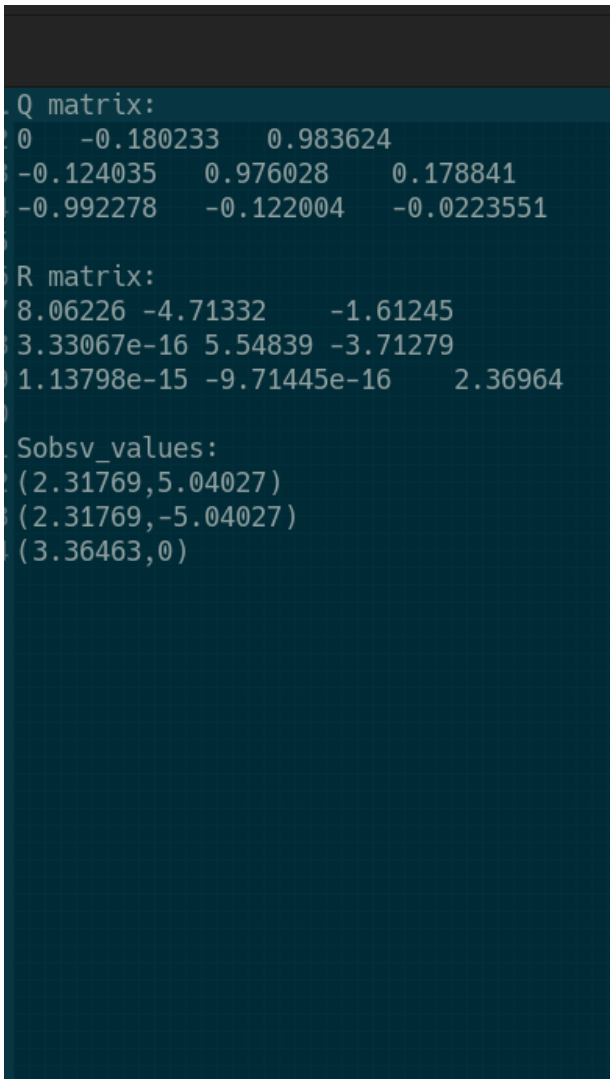

Рис. 5: Вывод в консоли

15 Исходный код

Lab1.5.cpp

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <fstream>
#include <ccomplex>

using namespace std;

vector<vector<double>> transpose(vector<vector<double>>matrix) {
    int n = matrix.size();
    vector<vector<double>> result(n, vector<double>(n));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            result[i][j] = matrix[j][i];
        }
    }
    return result;
}

double norma(vector <double> A){
    int n = A.size();
    double norm = 0;
    for (int i=0; i<n; i++){
        norm += A[i] * A[i];
    }

    return sqrt(norm);
}

double scalar_proisv(vector<double> first, vector<double> second){
    double sum = 0;
    int n = first.size();
    for (int i=0; i<n; i++){
        sum += first[i] * second[i];
    }
    return sum;
}


void normalize_matrix(vector<vector <double>>& A) {
    int n = A.size();
    for (int i = 0; i<n; i++){
        double norm = norma(A[i]);
        for (int j=0;j<n;j++){
            A[i][j] /= norm;
```

```
46          }
47      }
48  }
49
50  vector <double> substract_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[
                    count], B[count]);
68              for (int j = 0; j<n; j++){
69                  projection[j] += k * B[count][j];
70              }
71          }
72          B[i] = substract_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      int n = A.size();
79      vector <vector <double>> R(n, vector<double>(n,0));
80    for(int i = 0; i < n; i++)
81      for(int j = 0; j < n; j++)
82        for(int k = 0; k < n; k++)
83          R[i][j] += A[i][k] * B[k][j];
84      return R;
85  }
86
87
88  void QR_decomp(vector<vector <double>> &A, vector<vector <double>> &Q, vector<vector <
        double>> &R){
89      Q = gramSchmidt(A);
90      normalize_matrix(Q);
91      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++)
```

```
            fina >> A[i][j];
    }
    vector<vector<double>> Q, R;
    QR_decomp(A,Q,R);

    fout << "Q matrix:" << endl;
    for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
      fout << Q[i][j] << "\t";
    }
    fout << endl;
  }

    fout << "\nR matrix:" << endl;
    for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
      fout << R[i][j] << "\t";
    }
    fout << endl;
  }
    double eps = 0.1;
    vector<complex<double>> sobstv = sobs_values(A, eps);
    fout << "\nSobsv_values:" << endl;
    for(int i = 0; i < n; i++){
        fout << sobstv[i] << "\t";
    fout << endl;
  }
    return 0;
}
```