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. С. Своеволин
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: 20

$$\begin{cases} 7x_1 + 8x_2 + 4x_3 - 6x_4 = -126 \\ -x_1 + 6x_2 - 2x_3 - 6x_4 = -42 \\ 2x_1 + 9x_2 + 6x_3 - 4x_4 = -115 \\ 5x_1 + 9x_2 + x_3 + x_4 = -67 \end{cases} \tag{1}$$

2 Результаты работы

```
LU разложение:
L =
1.00 0.00 0.00 0.00
-0.14 1.00 0.00 0.00
0.29 0.94 1.00 0.00
0.71 0.46 -0.19 1.00

U =
7.00 8.00 4.00 -6.00
0.00 7.14 -1.43 -6.86
0.00 0.00 6.20 4.16
0.00 0.00 0.00 9.25

LU=
7.00 8.00 4.00 -6.00
-1.00 6.00 -2.00 -6.00
2.00 9.00 6.00 -4.00
5.00 9.00 1.00 1.00

Решение системы:
-4.00
-5.00
-7.00
5.00

Обратная матрица:
0.15 -0.06 -0.12 0.03
-0.07 0.05 0.05 0.09
-0.01 -0.11 0.15 -0.07
-0.09 -0.07 0.02 0.11

Обратная на исходную:
1.00 0.00 -0.00 0.00
-0.00 1.00 0.00 0.00
0.00 0.00 1.00 0.00
-0.00 -0.00 0.00 1.00

Определитель: 2866.00
```

Рис. 1: Вывод в консоли

3 Исходный код

1.cpp

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

using namespace std;

struct matrix
{
    int rows = 0, cols = 0;
    vector <vector <double>> v;

    matrix() {}
    matrix(int _rows, int _cols)
    {
        rows = _rows;
        cols = _cols;
        v = vector <vector <double>>(rows, vector <double>(cols));
    }

    vector <double>& operator[](int row)
    {
        return v[row];
    }

    operator double()
    {
        return v[0][0];
    }
};

matrix operator*(matrix lhs, matrix rhs)
{
    if (lhs.cols != rhs.rows)
        return matrix(0, 0);
    matrix res(lhs.rows, rhs.cols);
    for (int i = 0; i < res.rows; i++)
    {
        for (int j = 0; j < res.cols; j++)
        {
            res[i][j] = 0;
            for (int k = 0; k < lhs.cols; k++)
                res[i][j] += lhs[i][k] * rhs[k][j];
        }
    }
    return res;
```

```
46 }
47
48 istream& operator>>(istream& stream, matrix& a)
49 {
50     for (int i = 0; i < a.rows; i++)
51     {
52         for (int j = 0; j < a.cols; j++)
53             stream >> a[i][j];
54     }
55     return stream;
56 }
57
58 ostream& operator<<(ostream& stream, matrix a)
59 {
60     for (int i = 0; i < a.rows; i++)
61     {
62         for (int j = 0; j < a.cols; j++)
63             stream << a[i][j] << ' ';
64         stream << '\n';
65     }
66     return stream;
67 }
68
69 vector <int> swp;
70
71 pair <matrix, matrix> lu_decomposition(matrix a)
72 {
73     int n = a.rows;
74     matrix l(n, n);
75     swp = vector <int>(0);
76     for (int k = 0; k < n; k++)
77     {
78         matrix prev = a;
79         int idx = k;
80         for (int i = k + 1; i < n; i++)
81         {
82             if (abs(prev[idx][k]) < abs(prev[i][k]))
83                 idx = i;
84         }
85         swap(prev[k], prev[idx]);
86         swap(a[k], a[idx]);
87         swap(l[k], l[idx]);
88         swp.push_back(idx);
89         for (int i = k + 1; i < n; i++)
90         {
91             double h = prev[i][k] / prev[k][k];
92             l[i][k] = h;
93             for (int j = k; j < n; j++)
94                 a[i][j] = prev[i][j] - h * prev[k][j];
```

```
95
96          }
97      }
98      for (int i = 0; i < n; i++)
99          l[i][i] = 1;
100     return { l, a };
101 }
102
103 matrix solve_triag(matrix a, matrix b, bool up)
104 {
105     int n = a.rows;
106     matrix res(n, 1);
107     int d = up ? -1 : 1;
108     int first = up ? n - 1 : 0;
109     for (int i = first; i < n && i >= 0; i += d)
110     {
111         res[i][0] = b[i][0];
112         for (int j = 0; j < n; j++)
113         {
114             if (i != j)
115                 res[i][0] -= a[i][j] * res[j][0];
116         }
117         res[i][0] = res[i][0] / a[i][i];
118     }
119     return res;
120 }
121
122 matrix solve_gauss(pair <matrix, matrix> lu, matrix b)
123 {
124     for (int i = 0; i < swp.size(); i++)
125         swap(b[i], b[swp[i]]);
126     matrix z = solve_triag(lu.first, b, false);
127     matrix x = solve_triag(lu.second, z, true);
128     return x;
129 }
130
131 matrix inverse(matrix a)
132 {
133     int n = a.rows;
134     matrix b(n, 1);
135     pair <matrix, matrix> lu = lu_decomposition(a);
136     matrix res(n, n);
137     for (int i = 0; i < n; i++)
138     {
139         b[max(i - 1, 0)][0] = 0;
140         b[i][0] = 1;
141         matrix col = solve_gauss(lu, b);
142         for (int j = 0; j < n; j++)
143             res[j][i] = col[j][0];
```

```
144     }
145     return res;
146 }
147 
148 double determinant(matrix a)
149 {
150     int n = a.rows;
151     pair <matrix, matrix> lu = lu_decomposition(a);
152     double det = 1;
153     for (int i = 0; i < n; i++)
154         det *= lu.second[i][i];
155     return det;
156 }
157 
158 int main()
159 {
160     ofstream fout("answer.txt");
161     fout.precision(2);
162     fout << fixed;
163     ifstream fina("matrix1.txt"), finb("column1.txt");
164     matrix a(4, 4), b(4, 1);
165     fina >> a;
166     finb >> b;
167     pair <matrix, matrix> p = lu_decomposition(a);
168     fout << "LU разложение:\nL =\n" << p.first << "\nU =\n" << p.second;
169     fout << "\nLU=\n" << p.first * p.second;
170     fout << "\Решениеn системы:\n" << solve_gauss(p, b);
171     fout << "\Обратнаяn матрица:\n" << inverse(a);
172     fout << "\Обратнаяn наисходную:\n" << a*inverse(a);
173     fout << "\Определительn: " << determinant(a) << "\n";
174 }
```

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: 20

$$\begin{cases} -6x_1 + 6x_2 = 30 \\ 2x_1 + 10x_2 - 7x_3 = -31 \\ -8x_2 + 18x_3 + 9x_4 = 108 \\ 6x_3 - 17x_4 - 6x_5 = -114 \\ 9x_4 + 14x_5 = 124 \end{cases} \tag{2}$$

5 Результаты работы

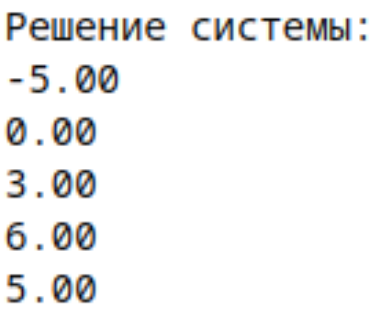

Рис. 2: Вывод в консоли

6 Исходный код

2.cpp

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

using namespace std;

struct matrix
{
    int rows = 0, cols = 0;
    vector <vector <double>> v;

    matrix() {}
    matrix(int _rows, int _cols)
    {
        rows = _rows;
        cols = _cols;
        v = vector <vector <double>>(rows, vector <double>(cols));
    }

    vector <double>& operator[](int row)
    {
        return v[row];
    }

    operator double()
    {
        return v[0][0];
    }
};

istream& operator>>(istream& stream, matrix& a)
{
    for (int i = 0; i < a.rows; i++)
    {
        for (int j = 0; j < a.cols; j++)
            stream >> a[i][j];
    }
    return stream;
}

ostream& operator<<(ostream& stream, matrix a)
{
    for (int i = 0; i < a.rows; i++)
    {
        for (int j = 0; j < a.cols; j++)
```

```
46              stream << a[i][j] << ' ';
47          stream << '\n';
48      }
49      return stream;
50  }
51
52
53  matrix solve_tridiagonal(matrix& a, matrix& b)
54  {
55      int n = a.rows;
56      vector <double> p(n), q(n);
57      p[0] = -a[0][1] / a[0][0];
58      q[0] = b[0][0] / a[0][0];
59      for (int i = 1; i < n; i++)
60      {
61          if (i != n - 1)
62              p[i] = -a[i][i + 1] / (a[i][i] + a[i][i - 1] * p[i - 1]);
63          else
64              p[i] = 0;
65              q[i] = (b[i][0] - a[i][i - 1] * q[i - 1]) / (a[i][i] + a[i][i - 1] * p[i -
                    1]);
66      }
67      matrix res(n, 1);
68      res[n - 1][0] = q[n - 1];
69      for (int i = n - 2; i >= 0; i--)
70          res[i][0] = p[i] * res[i + 1][0] + q[i];
71      return res;
72  }
73
74  int main()
75  {
76      ofstream fout("answer.txt");
77      fout.precision(2);
78      fout << fixed;
79      ifstream fina("matrix2.txt"), finb("column2.txt");
80      matrix a(5, 5), b(5, 1);
81      fina >> a;
82      finb >> b;
83      fout << "Решение системы:\n" << solve_tridiagonal(a, b);
84  }
```

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: 20

$$\begin{cases} 10x_1 - x_2 - 2x_3 + 5x_4 = -99 \\ 4x_1 + 28x_2 + 7x_3 + 9x_4 = 0 \\ 6x_1 + 5x_2 - 23x_3 + 4x_4 = 67 \\ x_1 + 4x_2 + 5x_3 - 15x_4 = 58 \end{cases} \tag{3}$$

8 Результаты работы

```
Решение системы методом итераций с e=0.01:
-8.00
4.00
-5.00
-5.00
Количество итераций: 12
Решение системы методом Зейделя с e=0.01:
-8.00
4.00
-5.00
-5.00
Количество итераций: 5
```

Рис. 3: Вывод в консоли

9 Исходный код

3.cpp

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

using namespace std;

struct matrix
{
    int rows = 0, cols = 0;
    vector <vector <double>> v;

    matrix() {}
    matrix(int _rows, int _cols)
    {
        rows = _rows;
        cols = _cols;
        v = vector <vector <double>>(rows, vector <double>(cols));
    }

    vector <double>& operator[](int row)
    {
        return v[row];
    }

    operator double()
    {
        return v[0][0];
    }
};

istream& operator>>(istream& stream, matrix& a)
{
    for (int i = 0; i < a.rows; i++)
    {
        for (int j = 0; j < a.cols; j++)
            stream >> a[i][j];
    }
    return stream;
}

ostream& operator<<(ostream& stream, matrix a)
{
    for (int i = 0; i < a.rows; i++)
    {
        for (int j = 0; j < a.cols; j++)
```

```
46              stream << a[i][j] << ' ';
47          stream << '\n';
48      }
49      return stream;
50  }
51
52  matrix operator*(matrix lhs, matrix rhs)
53  {
54      if (lhs.cols != rhs.rows)
55          return matrix(0, 0);
56      matrix res(lhs.rows, rhs.cols);
57      for (int i = 0; i < res.rows; i++)
58      {
59          for (int j = 0; j < res.cols; j++)
60          {
61              res[i][j] = 0;
62              for (int k = 0; k < lhs.cols; k++)
63                  res[i][j] += lhs[i][k] * rhs[k][j];
64          }
65      }
66      return res;
67  }
68
69  matrix operator*(double lhs, matrix rhs)
70  {
71      for (int i = 0; i < rhs.rows; i++)
72      {
73          for (int j = 0; j < rhs.cols; j++)
74              rhs[i][j] *= lhs;
75      }
76      return rhs;
77  }
78
79  matrix operator+(matrix lhs, matrix rhs)
80  {
81      if (lhs.rows != rhs.rows || rhs.cols != lhs.cols)
82          return matrix(0, 0);
83      matrix res(lhs.rows, lhs.cols);
84      for (int i = 0; i < rhs.rows; i++)
85      {
86          for (int j = 0; j < res.cols; j++)
87              res[i][j] = lhs[i][j] + rhs[i][j];
88      }
89      return res;
90  }
91
92  matrix operator-(matrix lhs, matrix rhs)
93  {
94      if (lhs.rows != rhs.rows || rhs.cols != lhs.cols)
```

```
95          return matrix(0, 0);
96      matrix res(lhs.rows, lhs.cols);
97      for (int i = 0; i < rhs.rows; i++)
98      {
99          for (int j = 0; j < res.cols; j++)
100             res[i][j] = lhs[i][j] - rhs[i][j];
101     }
102     return res;
103 }
104
105 double abs(matrix a)
106 {
107     double mx = 0;
108     for (int i = 0; i < a.rows; i++)
109     {
110         double s = 0;
111         for (int j = 0; j < a.cols; j++)
112             s += abs(a[i][j]);
113         mx = max(mx, s);
114     }
115     return mx;
116 }
117
118 matrix solve_iteration(matrix a, matrix b, double eps, int& iter)
119 {
120     int n = a.rows;
121     matrix alpha(n, n), beta(n, 1);
122     for (int i = 0; i < n; i++)
123     {
124         for (int j = 0; j < n; j++)
125             alpha[i][j] = -a[i][j] / a[i][i];
126         alpha[i][i] = 0;
127     }
128     for (int i = 0; i < n; i++)
129         beta[i][0] = b[i][0] / a[i][i];
130     matrix x = beta;
131     double m = abs(a);
132     double cur = m;
133     double epsk = 2 * eps;
134     iter = 0;
135     while (epsk > eps)
136     {
137         matrix prev = x;
138         x = beta + alpha * x;
139         if (m < 1)
140             epsk = cur / (1 - m) * abs(x - prev);
141         else
142             epsk = abs(x - prev);
143         cur = cur * m;
```

```
144         iter++;
145     }
146     return x;
147 }
148 
149 matrix solve_seidel(matrix a, matrix b, double eps, int& iter)
150 {
151     int n = a.rows;
152     matrix alpha(n, n), beta(n, 1);
153     for (int i = 0; i < n; i++)
154     {
155         for (int j = 0; j < n; j++)
156             alpha[i][j] = -a[i][j] / a[i][i];
157         alpha[i][i] = 0;
158     }
159     for (int i = 0; i < n; i++)
160         beta[i][0] = b[i][0] / a[i][i];
161     matrix x = beta;
162     double m = abs(a);
163     double cur = m;
164     double epsk = 2 * eps;
165     iter = 0;
166     while (epsk > eps)
167     {
168         matrix prev = x;
169         for (int i = 0; i < n; i++)
170         {
171             double cur = beta[i][0];
172             for (int j = 0; j < n; j++)
173                 cur += alpha[i][j] * x[j][0];
174             x[i][0] = cur;
175         }
176         if (m < 1)
177             epsk = cur / (1 - m) * abs(x - prev);
178         else
179             epsk = abs(x - prev);
180         cur = cur * m;
181         iter++;
182     }
183     return x;
184 }
185 
186 int main()
187 {
188     ofstream fout("answer.txt");
189     fout.precision(2);
190     fout << fixed;
191     ifstream fina("matrix3.txt"), finb("column3.txt");
192     matrix a(4, 4), b(4, 1);
```

```
193     fina >> a;
194     finb >> b;
195     int iter = 0;
196     fout << "Решение системыметодомитерацийсе=0.01:\n" << solve_iteration(a, b, 0.001,
            iter) << "Количество итераций: ";
197     fout << iter;
198     fout << "\Решениеn системыметодомЗейделясе=0.01:\n" << solve_seidel(a, b, 0.001,
            iter) << "Количество итераций: ";
199     fout << iter;
200 }
```

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: 20

$$\begin{pmatrix} -7 & -9 & 1 \\ -9 & 7 & 2 \\ 1 & 2 & 9 \end{pmatrix} \tag{4}$$

11 Результаты работы

```
Собственные значения с e=0.1:
-11.56
12.04
8.52

Собственные векторы с e=0.1:
0.89 -0.37 0.26
0.44 0.83 -0.35
-0.08 0.42 0.90
```

Рис. 4: Вывод в консоли

12 Исходный код

4.cpp

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

using namespace std;

using cmd = complex <double>;
const double pi = acos(-1);

struct matrix
{
    int rows = 0, cols = 0;
    vector <vector <double>> v;

    matrix() {}
    matrix(int _rows, int _cols)
    {
        rows = _rows;
        cols = _cols;
        v = vector <vector <double>>(rows, vector <double>(cols));
    }

    vector <double>& operator[](int row)
    {
        return v[row];
    }

    operator double()
    {
        return v[0][0];
    }
};

istream& operator>>(istream& stream, matrix& a)
{
    for (int i = 0; i < a.rows; i++)
    {
        for (int j = 0; j < a.cols; j++)
            stream >> a[i][j];
    }
    return stream;
}

ostream& operator<<(ostream& stream, matrix a)
```

```
{
    for (int i = 0; i < a.rows; i++)
    {
        for (int j = 0; j < a.cols; j++)
            stream << a[i][j] << ' ';
        stream << '\n';
    }
    return stream;
}

matrix transposition(matrix a)
{
    matrix res(a.cols, a.rows);
    for (int i = 0; i < a.rows; i++)
    {
        for (int j = 0; j < a.cols; j++)
            res[j][i] = a[i][j];
    }
    return res;
}

matrix operator*(matrix lhs, matrix rhs)
{
    if (lhs.cols != rhs.rows)
        return matrix(0, 0);
    matrix res(lhs.rows, rhs.cols);
    for (int i = 0; i < res.rows; i++)
    {
        for (int j = 0; j < res.cols; j++)
        {
            res[i][j] = 0;
            for (int k = 0; k < lhs.cols; k++)
                res[i][j] += lhs[i][k] * rhs[k][j];
        }
    }
    return res;
}

pair <matrix, matrix> method_jacobi(matrix a, double eps)
{
    int n = a.rows;
    double epsk = 2 * eps;
    matrix vec(n, n);
    for (int i = 0; i < n; i++)
        vec[i][i] = 1;
    while (epsk > eps)
    {
        int cur_i = 1, cur_j = 0;
        for (int i = 0; i < n; i++)
```

```
95          {
96              for (int j = 0; j < i; j++)
97              {
98                  if (abs(a[cur_i][cur_j]) < abs(a[i][j]))
99                  {
100                     cur_i = i;
101                     cur_j = j;
102                 }
103             }
104         }
105         matrix u(n, n);
106         double phi = pi / 4;
107         if (abs(a[cur_i][cur_i] - a[cur_j][cur_j]) > 1e-7)
108             phi = 0.5 * atan((2 * a[cur_i][cur_j]) / (a[cur_i][cur_i] - a[cur_j][cur_j
                ]));
109         for (int i = 0; i < n; i++)
110             u[i][i] = 1;
111         u[cur_i][cur_j] = -sin(phi);
112         u[cur_i][cur_i] = cos(phi);
113         u[cur_j][cur_i] = sin(phi);
114         u[cur_j][cur_j] = cos(phi);
115         vec = vec * u;
116         a = transposition(u) * a * u;
117         epsk = 0;
118         for (int i = 0; i < n; i++)
119         {
120             for (int j = 0; j < i; j++)
121                 epsk += a[i][j] * a[i][j];
122         }
123         epsk = sqrt(epsk);
124     }
125     matrix val(n, 1);
126     for (int i = 0; i < n; i++)
127         val[i][0] = a[i][i];
128     return { val, vec };
129 }
130
131 int main() {
132     ofstream fout("answer.txt");
133     fout.precision(2);
134     fout << fixed;
135     ifstream fin("matrix4.txt");
136     matrix a(3, 3);
137     fin >> a;
138     pair <matrix, matrix> p = method_jacobi(a, 0.1);
139     fout << "Собственные значениясе=0.1:\n" << p.first << "\Собственныеn векторысе
            =0.1:\n" << p.second;
140 }
```

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: 20

$$\begin{pmatrix} 6 & 5 & -6 \\ 4 & -6 & 9 \\ -6 & 6 & 1 \end{pmatrix} \tag{5}$$

14 Результаты работы

```
QR разложение:
Q =
-0.63960 0.24353 0.72911
-0.42640 -0.90159 -0.07291
0.63960 -0.35753 0.68050

R =
-9.38083 -4.47722 0.63960
0.00000 8.77237 -9.93304
0.00000 0.00000 -4.35035

Собственные значения QR методом с e=0.1:
(9.26799,0.00000) (-4.13399,-4.64087) (-4.13399,4.64087)
```

Рис. 5: Вывод в консоли

15 Исходный код

5.cpp

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

using namespace std;

using cmd = complex <double>;
const double pi = acos(-1);

struct matrix
{
    int rows = 0, cols = 0;
    vector <vector <double>> v;

    matrix() {}
    matrix(int _rows, int _cols)
    {
        rows = _rows;
        cols = _cols;
        v = vector <vector <double>>(rows, vector <double>(cols));
    }

    vector <double>& operator[](int row)
    {
        return v[row];
    }

    operator double()
    {
        return v[0][0];
    }
};

istream& operator>>(istream& stream, matrix& a)
{
    for (int i = 0; i < a.rows; i++)
    {
        for (int j = 0; j < a.cols; j++)
            stream >> a[i][j];
    }
    return stream;
}

ostream& operator<<(ostream& stream, matrix a)
```

```
46 {
47     for (int i = 0; i < a.rows; i++)
48     {
49         for (int j = 0; j < a.cols; j++)
50             stream << a[i][j] << ' ';
51         stream << '\n';
52     }
53     return stream;
54 }
55
56 matrix operator+(matrix lhs, matrix rhs)
57 {
58     if (lhs.rows != rhs.rows || rhs.cols != lhs.cols)
59         return matrix(0, 0);
60     matrix res(lhs.rows, lhs.cols);
61     for (int i = 0; i < rhs.rows; i++)
62     {
63         for (int j = 0; j < res.cols; j++)
64             res[i][j] = lhs[i][j] + rhs[i][j];
65     }
66     return res;
67 }
68
69 matrix operator-(matrix lhs, matrix rhs)
70 {
71     if (lhs.rows != rhs.rows || rhs.cols != lhs.cols)
72         return matrix(0, 0);
73     matrix res(lhs.rows, lhs.cols);
74     for (int i = 0; i < rhs.rows; i++)
75     {
76         for (int j = 0; j < res.cols; j++)
77             res[i][j] = lhs[i][j] - rhs[i][j];
78     }
79     return res;
80 }
81
82 matrix operator*(double lhs, matrix rhs)
83 {
84     for (int i = 0; i < rhs.rows; i++)
85     {
86         for (int j = 0; j < rhs.cols; j++)
87             rhs[i][j] *= lhs;
88     }
89     return rhs;
90 }
91
92 matrix operator*(matrix lhs, matrix rhs)
93 {
94     if (lhs.cols != rhs.rows)
```

```
95          return matrix(0, 0);
96      matrix res(lhs.rows, rhs.cols);
97      for (int i = 0; i < res.rows; i++)
98      {
99          for (int j = 0; j < res.cols; j++)
100         {
101             res[i][j] = 0;
102             for (int k = 0; k < lhs.cols; k++)
103                 res[i][j] += lhs[i][k] * rhs[k][j];
104         }
105     }
106     return res;
107 }
108
109 matrix transposition(matrix a)
110 {
111     matrix res(a.cols, a.rows);
112     for (int i = 0; i < a.rows; i++)
113     {
114         for (int j = 0; j < a.cols; j++)
115             res[j][i] = a[i][j];
116     }
117     return res;
118 }
119
120 vector <int> swp;
121
122 pair <matrix, matrix> lu_decomposition(matrix a)
123 {
124     int n = a.rows;
125     matrix l(n, n);
126     swp = vector <int>(0);
127     for (int k = 0; k < n; k++)
128     {
129         matrix prev = a;
130         int idx = k;
131         for (int i = k + 1; i < n; i++)
132         {
133             if (abs(prev[idx][k]) < abs(prev[i][k]))
134                 idx = i;
135         }
136         swap(prev[k], prev[idx]);
137         swap(a[k], a[idx]);
138         swap(l[k], l[idx]);
139         swp.push_back(idx);
140         for (int i = k + 1; i < n; i++)
141         {
142             double h = prev[i][k] / prev[k][k];
143             l[i][k] = h;
```

```
144             for (int j = k; j < n; j++)
145                 a[i][j] = prev[i][j] - h * prev[k][j];
146 
147         }
148     }
149     for (int i = 0; i < n; i++)
150         l[i][i] = 1;
151     return { l, a };
152 }
153 
154 matrix solve_triag(matrix a, matrix b, bool up)
155 {
156     int n = a.rows;
157     matrix res(n, 1);
158     int d = up ? -1 : 1;
159     int first = up ? n - 1 : 0;
160     for (int i = first; i < n && i >= 0; i += d)
161     {
162         res[i][0] = b[i][0];
163         for (int j = 0; j < n; j++)
164         {
165             if (i != j)
166                 res[i][0] -= a[i][j] * res[j][0];
167         }
168         res[i][0] = res[i][0] / a[i][i];
169     }
170     return res;
171 }
172 
173 matrix solve_gauss(pair <matrix, matrix> lu, matrix b)
174 {
175     for (int i = 0; i < swp.size(); i++)
176         swap(b[i], b[swp[i]]);
177     matrix z = solve_triag(lu.first, b, false);
178     matrix x = solve_triag(lu.second, z, true);
179     return x;
180 }
181 
182 matrix inverse(matrix a)
183 {
184     int n = a.rows;
185     matrix b(n, 1);
186     pair <matrix, matrix> lu = lu_decomposition(a);
187     matrix res(n, n);
188     for (int i = 0; i < n; i++)
189     {
190         b[max(i - 1, 0)][0] = 0;
191         b[i][0] = 1;
192         matrix col = solve_gauss(lu, b);
```

```
193         for (int j = 0; j < n; j++)
194             res[j][i] = col[j][0];
195     }
196     return res;
197 }
198 
199 double abs(matrix a)
200 {
201     double mx = 0;
202     for (int i = 0; i < a.rows; i++)
203     {
204         double s = 0;
205         for (int j = 0; j < a.cols; j++)
206             s += abs(a[i][j]);
207         mx = max(mx, s);
208     }
209     return mx;
210 }
211 
212 double sign(double x)
213 {
214     return x > 0 ? 1 : -1;
215 }
216 
217 pair <matrix, matrix> qr_decomposition(matrix a)
218 {
219     int n = a.rows;
220     matrix e(n, n);
221     for (int i = 0; i < n; i++)
222         e[i][i] = 1;
223     matrix q = e;
224     for (int i = 0; i < n - 1; i++)
225     {
226         matrix v(n, 1);
227         double s = 0;
228         for (int j = i; j < n; j++)
229             s += a[j][i] * a[j][i];
230         v[i][0] = a[i][i] + sign(a[i][i]) * sqrt(s);
231         for (int j = i + 1; j < n; j++)
232             v[j][0] = a[j][i];
233         matrix h = e - (2.0 / double(transposition(v) * v)) * (v * transposition(v));
234         q = q * h;
235         a = h * a;
236     }
237     return { q, a };
238 }
239 
240 vector <cmd> qr_eigenvalues(matrix a, double eps)
241 {
```

```
242     int n = a.rows;
243     vector <cmd> prev(n);
244     while (true)
245     {
246         pair <matrix, matrix> p = qr_decomposition(a);
247         a = p.second * p.first;
248         vector <cmd> cur;
249         for (int i = 0; i < n; i++)
250         {
251             if (i < n - 1 && abs(a[i + 1][i]) > 1e-7)
252             {
253                 double b = -(a[i][i] + a[i + 1][i + 1]);
254                 double c = a[i][i] * a[i + 1][i + 1] - a[i][i + 1] * a[i + 1][i];
255                 double d = b * b - 4 * c;
256                 cmd sgn = (d > 0) ? cmd(1, 0) : cmd(0, 1);
257                 d = sqrt(abs(d));
258                 cur.push_back(0.5 * (-b - sgn * d));
259                 cur.push_back( 0.5 * (-b + sgn * d));
260                 i++;
261             }
262             else
263                 cur.push_back(a[i][i]);
264         }
265         bool ok = true;
266         for (int i = 0; i < n; i++)
267             ok = ok && abs(cur[i] - prev[i]) < eps;
268         if (ok)
269             break;
270         prev = cur;
271     }
272     return prev;
273 }
274
275 int main()
276 {
277     ofstream fout("answer.txt");
278     fout.precision(5);
279     fout << fixed;
280     ifstream fin("matrix5.txt");
281     matrix a(3, 3);
282     fin >> a;
283     pair <matrix, matrix> p = qr_decomposition(a);
284     fout << "QR разложение:\nQ =\n" << p.first << "\nR =\n" << p.second;
285     cout << p.first * p.second << '\n' << transposition(p.first) << '\n' << inverse(p.
        first);
286     vector <cmd> v = qr_eigenvalues(a, 0.01);
287     fout << "\Собственныеn значенияQR методомсe=0.1:\n";
288     for (int i = 0; i < v.size(); i++)
289         fout << v[i] << ' ';
```

```
290 || }
```