Московский авиационный институт
(национальный исследовательский университет)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Лабораторные работы по курсу «Численные методы»

Студент: О. В. Гребнева
Преподаватель: Д. Е. Пивоваров
Группа: М8О-303Б-21
Дата:
Оценка:
Подпись:

Москва, 2024

1.1. Решение СЛАУ с помощью LU-разложения матриц

1 Постановка задачи

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: 5

$$\begin{cases} 3x_1 - 8x_2 + x_3 - 7x_4 = 96, \\ 6x_1 + 4x_2 + 8x_3 + 5x_4 = -13, \\ -x_1 + x_2 - 9x_3 - 3x_4 = -54, \\ -6x_1 + 6x_2 + 9x_3 - 4x_4 = 82. \end{cases}$$

2 Результаты работы

```
Решение системы:
x[0] = -3.00
x[1] = -6.00
x[2] = 8.00
x[3] = -7.00
Определитель матрицы: 7315.00
Обратная матрица:
0.06    0.13    0.10    -0.02
-0.02   0.09    0.13    0.05
0.02    0.01    -0.07   0.03
-0.09   -0.04   -0.11   -0.07
Матрица L:
1.00    0.00    0.00    0.00
2.00    1.00    0.00    0.00
-0.33   -0.08   1.00    0.00
-2.00   -0.50   -1.71   1.00
Матрица U:
3.00    -8.00   1.00    -7.00
0.00    20.00   6.00    19.00
0.00    0.00    -8.17   -3.75
0.00    0.00    0.00    -14.93
Матрица L*U:
3.00    -8.00   1.00    -7.00
6.00    4.00    8.00    5.00
-1.00   1.00    -9.00   -3.00
-6.00   6.00    9.00    -4.00
```

3 Исходный код

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

using namespace std;

vector<vector<double>> identity(int n) {
    vector<vector<double>> I(n, vector<double>(n, 0));
    for (int i = 0; i < n; ++i)
        I[i][i] = 1;
    return I;
}

void LU_decomposition(vector<vector<double>>& matrix, vector<int>& P, vector<vector<
    double>>& L, vector<vector<double>>& U) {
    int n = matrix.size();
    L = identity(n);
    U = matrix;

    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];
    }

}

vector<double> forward_substitution(const vector<vector<double>>& L, const vector<
    double>& b) {
    int n = L.size();
    vector<double> y(n, 0);
    for (int i = 0; i < n; ++i) {
        y[i] = b[i];
        for (int j = 0; j < i; ++j)
            y[i] -= L[i][j] * y[j];
    }
    return y;
}
```

```
46
47 vector<double> backward_substitution(const vector<vector<double>>& U, const vector<
       double>& y) {
48     int n = U.size();
49     vector<double> x(n, 0);
50     for (int i = n - 1; i >= 0; --i) {
51         x[i] = y[i];
52         for (int j = i + 1; j < n; ++j)
53             x[i] -= U[i][j] * x[j];
54         x[i] /= U[i][i];
55     }
56     return x;
57 }
58
59 double calculate_determinant(const vector<vector<double>>& U) {
60     int n = U.size();
61     double det = 1;
62     for (int i = 0; i < n; ++i)
63         det *= U[i][i];
64     return det;
65 }
66
67 vector<vector<double>> inverse_matrix(const vector<vector<double>>& L, const vector<
       vector<double>>& U) {
68     int n = U.size();
69     vector<vector<double>> inv_mat(n, vector<double>(n, 0));
70     vector<double> E(n, 0);
71     for (int i = 0; i < n; ++i) {
72         E[i] = 1;
73         vector<double> y = forward_substitution(L, E);
74         vector<double> x = backward_substitution(U, y);
75         for (int j = 0; j < n; ++j)
76             inv_mat[j][i] = x[j];
77         E[i] = 0;
78     }
79     return inv_mat;
80 }
81
82 vector<vector<double>> dot_product(const vector<vector<double>>& A, const vector<
       vector<double>>& B) {
83     int n = A.size();
84     int m = B[0].size();
85     int p = B.size();
86     vector<vector<double>> C(n, vector<double>(m, 0));
87
88     for (int i = 0; i < n; ++i)
89         for (int j = 0; j < m; ++j)
90             for (int k = 0; k < p; ++k)
91                 C[i][j] += A[i][k] * B[k][j];
```

```
92
93      return C;
94  }
95
96  int main() {
97      vector<vector<double>> matrix(4, vector <double>(4, 0));
98
99      vector<double> vec(4, 0);
100
101      ifstream fina("matrix.txt"), finb("column.txt");
102     for (int i = 0; i < matrix.size(); i++)
103     {
104         for (int j = 0; j < matrix.size(); j++)
105             fina >> matrix[i][j];
106     }
107     for (int i = 0; i < vec.size(); i++)
108     {
109         finb >> vec[i];
110     }
111
112     vector<int> P;
113     vector<vector<double>> L, U;
114
115     LU_decomposition(matrix, P, L, U);
116
117     vector<double> Pb = vec;
118     for (size_t i = 0; i < P.size(); ++i)
119         Pb[i] = vec[P[i]];
120
121     vector<double> y = forward_substitution(L, Pb);
122     vector<double> x = backward_substitution(U, y);
123
124     double det = calculate_determinant(U);
125
126     vector<vector<double>> inv_matrix = inverse_matrix(L, U);
127
128
129     ofstream fout("answer.txt");
130     fout.precision(2);
131     fout << fixed;
132
133
134
135
136     fout << " :" << endl;
137     for (size_t i = 0; i < x.size(); ++i)
138         fout << "x[" << i << "] = " << x[i] << endl;
139
140     fout << " : " << det << endl;
```

```
141     fout << " :" << endl;
142 
143     for (const auto& row : inv_matrix) {
144         for (const auto& elem : row)
145             fout << fixed << elem << "\t";
146         fout << endl;
147     }
148     fout << " L:" << endl;
149     for (const auto& row : L) {
150         for (const auto& elem : row)
151             fout << fixed << elem << "\t";
152         fout << endl;
153     }
154     fout << " U:" << endl;
155     for (const auto& row : U) {
156         for (const auto& elem : row)
157             fout << fixed << elem << "\t";
158         fout << endl;
159     }
160     fout << " L*U:" << endl;
161     for (const auto& row : dot_product(L, U)) {
162         for (const auto& elem : row)
163             fout << fixed << elem << "\t";
164         fout << endl;
165     }
166     return 0;
167 }
```

1.2. Решение СЛАУ методом прогонки

1 Постановка задачи

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: 5

$$\begin{cases} 8x_1 + 4x_2 = 48, \\ -5x_1 + 22x_2 + 8x_3 = 125, \\ -5x_2 - 11x_3 + x_4 = -43, \\ -9x_3 - 15x_4 + x_5 = 18, \\ x_4 + 7x_5 = -23. \end{cases}$$

2 Результаты работы

```
Решение системы:
x[0] = 3.00
x[1] = 6.00
x[2] = 1.00
x[3] = -2.00
x[4] = -3.00
```

3 Исходный код

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

using namespace std;

int main() {
    vector<vector<double>> matrix(5, vector <double>(3, 0));
    vector<double> vecd(5, 0);

    ifstream fina("matrix.txt"), finb("col.txt");
    for (int i = 0; i < matrix.size(); i++)
    {
        for (int j = 0; j < 3; j++)
        {
            if ((i == 0 && j == 0) || (i == 4 && j == 2))
                matrix[i][j] = 0;
            else
                fina >> matrix[i][j];
        }
    }
    for (int i = 0; i < vecd.size(); i++)
        finb >> vecd[i];

    vector<double> p(5, 0), q(5, 0), x(5, 0);
    p[0] = -matrix[0][2]/matrix[0][1];
    q[0] = vecd[0]/matrix[0][1];

    //
    for (int i = 1; i < matrix.size(); i++)
    {
        p[i] = -matrix[i][2]/(matrix[i][1] + matrix[i][0]*p[i-1]);
        q[i] = (vecd[i] - matrix[i][0]*q[i-1])/(matrix[i][1] + matrix[i][0]*p[i-1]);
    }
    x[4] = q[4];

    //
    for (int i = matrix.size() - 2; i > -1; i--)
    {
        x[i] = p[i]*x[i+1] + q[i];
    }

    ofstream fout("answer.txt");
    fout.precision(2);
    fout << fixed;

    fout << " :" << endl;
```

```
48      for (size_t i = 0; i < x.size(); ++i)
49          fout << "x[" << i << "] = " << x[i] << endl;
50
51      return 0;
52  }
```

1.3. Решение СЛАУ методом простых итераций и методом Зейделя

1 Постановка задачи

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: 5

$$\begin{cases} 20x_1 + 5x_2 + 7x_3 + x_4 = -117, \\ -x_1 + 13x_2 - 7x_4 = -1, \\ 4x_1 - 6x_2 + 17x_3 + 5x_4 = 49, \\ -9x_1 + 8x_2 + 4x_3 - 25x_4 = -21, \\ x_4 + 7x_5 = -23. \end{cases}$$

2 Результаты работы

```
Заданная точность: 0.20
Решение системы методом простых итераций:
x[0] = -7.98
x[1] = 1.97
x[2] = 3.99
x[3] = 4.97
Количество итераций: 8
Решение системы методом Зейделя:
x[0] = -7.99
x[1] = 1.99
x[2] = 4.00
x[3] = 4.99
Количество итераций: 7
Метод Зейделя сходится быстрее метода простых итераций
```

3 Исходный код

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

using namespace std;

double calculate_norma(const vector<vector<double>>& A) {
    double g = 0.0;

    for (int i = 0; i < A.size(); i++)
    {
        double h = 0;
        for (int j = 0; j < A.size(); j++)
        {
            if (A[i][j] > 0)
                h = h + A[i][j];
            else
                h = h - A[i][j];
        }
        if (h > g) g = h;
    }
    return g;
}

void jacobi_method(const vector<vector<double>> matrix, vector<vector<double>> &matA,
                                const vector<double> &vecd, vector<double> &vecb){
    for (int i = 0; i < matrix.size(); i++)
    {
        if (matrix[i][i] == 0)
            break;
        for (int j = 0; j < matrix.size(); j++)
        {
            if (i != j)
                matA[i][j] = -matrix[i][j]/matrix[i][i];
            else
                matA[i][j] = 0;
        }
        vecb[i] = vecd[i]/matrix[i][i];
    }
}

void simple_iteration(const vector<vector<double>> matA, const vector<double> &vecb,
    vector<double> &x, vector<double> &y,
                          double f, const double g, const double e, int &k, const int n
                              ){
    while (f > e)
    {
```

```
46          for (int i = 0; i < n; i++)
47              y[i] = x[i];
48          for (int i = 0; i < n; i++)
49          {
50              x[i] = vecb[i];
51              for (int j = 0; j < n; j++)
52                  x[i] = x[i] + matA[i][j]*y[j];
53          }
54          f = 0;
55          for (int i = 0; i < n; i++)
56          {
57              if ((x[i] - y[i]) > f) f = x[i] - y[i];
58              if ((y[i] - x[i]) > f) f = y[i] - x[i];
59          }
60          f = f*g/(1 - g);
61          k = k + 1;
62      }
63  }
64
65  void seidel_method(const vector<vector<double>> matA, const vector<double> &vecb,
        vector<double> &x, vector<double> &y,
66                             double f, const double g, const double e, int &k, const int n
                                   ){
67      while (f > e)
68      {
69          for (int i = 0; i < n; i++)
70              y[i] = x[i];
71          for (int i = 0; i < n; i++)
72          {
73              x[i] = vecb[i];
74              for (int j = 0; j < i-1; j++)
75                  x[i] = x[i] + matA[i][j]*x[j];
76              for (int j = i - 1; j < n; j++)
77                  x[i] = x[i] + matA[i][j]*y[j];
78          }
79          f = 0;
80          for (int i = 0; i < n; i++)
81          {
82              if ((x[i] - y[i]) > f) f = x[i] - y[i];
83              if ((y[i] - x[i]) > f) f = y[i] - x[i];
84          }
85          f = f*g/(1 - g);
86          k = k + 1;
87      }
88  }
89
90
91  int main() {
```

```
92      vector<vector<double>> matrix(4, vector <double>(4, 0)), matA(4, vector <double>(4,
            0));
93      vector<double> vecd(4, 0), vecb(4, 0), x(4, 0), y(4, 0);
94      double e;
95      int n = matrix.size();
96
97      ifstream fina("matrix.txt"), finb("col.txt");
98      for (int i = 0; i < n; i++)
99      {
100         for (int j = 0; j < n; j++)
101             fina >> matrix[i][j];
102     }
103     for (int i = 0; i < vecd.size(); i++)
104     {
105         finb >> vecd[i];
106     }
107     finb >> e;
108
109     ofstream fout("answer.txt");
110     fout.precision(2);
111     fout << fixed;
112
113     fout << " : " << e << endl;
114
115     jacobi_method(matrix, matA, vecd, vecb);
116
117     double g = calculate_norma(matA);
118
119     if (g >= 1)
120     {
121         fout << "   !" << endl;
122         return 0;
123     }
124
125     double f = 0;
126     for (int i = 0; i < n; i++)
127     {
128         x[i] = vecb[i];
129         if (x[i] > f) f = x[i];
130         if (-x[i] > f) f = -x[i];
131     }
132     f = f*g/(1 - g);
133     int k = 0, k0 = 0;
134
135     double f0 = f;
136     vector<double> x0 = x, y0 = y;
137
138     simple_iteration(matA, vecb, x, y, f, g, e, k, n);
139
```

```
140     fout << "    :" << endl;
141     for (size_t i = 0; i < x.size(); ++i)
142         fout << "x[" << i << "] = " << x[i] << endl;
143     fout << " : " << k << endl;
144 
145     f = f0;
146     y = y0;
147     x = x0;
148     k0 = k;
149     k = 0;
150 
151     seidel_method(matA, vecb, x, y, f, g, e, k, n);
152 
153     fout << "   :" << endl;
154     for (size_t i = 0; i < x.size(); ++i)
155         fout << "x[" << i << "] = " << x[i] << endl;
156     fout << " : " << k << endl;
157 
158     if (k0 > k)
159         fout << "      " << endl;
160     else if (k > k0)
161         fout << "      " << endl;
162     else
163         fout << "    " << endl;
164 
165     return 0;
166 }
```

1.4. Нахождение СЗ и СВ симметрических матриц методом вращений

1 Постановка задачи

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: 5

$$\begin{pmatrix} 0 & -7 & 7 \\ -7 & -9 & -5 \\ 7 & -5 & -1 \end{pmatrix}$$

2 Результаты работы

```
Собственные значения:
w[0] = 10.30
w[1] = -12.96
w[2] = -7.34
Собственные векторы:
0.69    0.41    -0.60
-0.41   0.90    0.16
0.60    0.13    0.79
Количество итераций: 4
```

3 Исходный код

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

#define M_PI 3.14159265358979323846

using namespace std;

void rotation_method(vector<vector<double>> matrix, vector<vector<double>> U, vector<vector<double>> &V,
                     vector<double> &w, const int n, const double e, int &k){

    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
        {
            if (i == j) V[i][j] = 1;
            else V[i][j] = 0;
        }

    double f = 0.0;

    for (int i = 0; i < n; i++)
        for (int j = i + 1; j < n; j++)
            f = f + matrix[i][j]*matrix[i][j];
    f = sqrt(f);

    double h = 0.0;
    vector<vector<double>> B(3, vector <double>(3, 0));

    while (f > e)
    {
        int l = 1, m = 2;
        double g = 0.0;

        for (int i = 0; i < n; i++)
        {
            for (int j = i + 1; j < n; j++)
            {
                if (matrix[i][j] > g){
                    g = matrix[i][j];
                    l = i;
                    m = j;
                }
                if (-matrix[i][j] > g){
                    g = -matrix[i][j];
                    l = i;
                    m = j;
```

```
47                  }
48              }
49          }
50          for (int i = 0; i < n; i++)
51          {
52              for (int j = 0; j < n; j++)
53              {
54                  if (i == j) U[i][j] = 1;
55                  else U[i][j] = 0;
56              }
57          }
58          if (matrix[l][l] == matrix[m][m]) h = M_PI/4;
59          else h = atan(2 * matrix[l][m]/(matrix[l][l] - matrix[m][m]))/2;
60
61          U[l][l] = cos(h);
62          U[l][m] = -sin(h);
63          U[m][l] = sin(h);
64          U[m][m] = cos(h);
65
66          for (int i = 0; i < n; i++)
67          {
68              for (int j = 0; j < n; j++)
69              {
70                  if ((i == l) || (i == m) || (j == l) || (j == m))
71                  {
72                      B[i][j] = 0;
73                      for (int p = 0; p < n; p++)
74                          B[i][j] = B[i][j] + U[p][i]*matrix[p][j];
75                  }
76              }
77          }
78          for (int i = 0; i < n; i++)
79          {
80              for (int j = 0; j < n; j++)
81              {
82                  if ((i == l) || (i == m) || (j == l) || (j == m))
83                  {
84                      matrix[i][j] = 0;
85                      for (int p = 0; p < n; p++)
86                          matrix[i][j] = matrix[i][j] + B[i][p]*U[p][j];
87                  }
88              }
89          }
90          for (int i = 0; i < n; i++)
91          {
92              for (int j = 0; j < n; j++)
93              {
94                  if ((i == l) || (i == m) || (j == l) || (j == m))
95                  {
```

```
                    B[i][j] = 0;
                    for (int p = 0; p < n; p++)
                        B[i][j] = B[i][j] + V[i][p]*U[p][j];
                }
            }
        }
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                if ((i == l) || (i == m) || (j == l) || (j == m))
                    V[i][j] = B[i][j];
            }
        }

        f = 0.0;
        for (int i = 0; i < n; i++)
            for (int j = i + 1; j < n; j++)
                f = f + matrix[i][j]*matrix[i][j];

        f = sqrt(f);
        k = k + 1;
    }

    for (int i = 0; i < n; i++)
    {
        w[i] = matrix[i][i];
    }
}

int main() {
    vector<vector<double>> matrix(3, vector <double>(3, 0)), V(3, vector <double>(3, 0)
        ), U(3, vector <double>(3, 0));
    double e;
    vector<double> w(3, 0);
    int n = matrix.size();

    ifstream fina("matrix.txt");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
            fina >> matrix[i][j];
    }
    fina >> e;

    int k = 0;

    rotation_method(matrix, U, V, w, n, e, k);

```

```
144     ofstream fout("answer.txt");
145     fout.precision(2);
146     fout << fixed;
147 
148     fout << " :" << endl;
149     for (size_t i = 0; i < w.size(); ++i)
150     {
151         fout << "w[" << i << "] = " << w[i] << endl;
152     }
153 
154     fout << " :" << endl;
155     for (const auto& row : V)
156     {
157         for (const auto& elem : row)
158             fout << fixed << elem << "\t";
159         fout << endl;
160     }
161 
162     fout << " : " << k << endl;
163 
164     return 0;
165 }
```

1.5. Нахождение СЗ и СВ матриц алгоритмом QR-разложения

1 Постановка задачи

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: 5

$$\begin{pmatrix} 5 & 8 & -2 \\ 7 & -2 & -4 \\ 5 & 8 & -1 \end{pmatrix}$$

2 Результаты работы

```
Матрица Q:
-0.50   0.52    -0.69
-0.70   -0.71   -0.02
-0.50   0.47    0.72
Матрица R:
-9.95   -6.63   4.32
0.00    9.38    1.33
-0.00   0.32    0.75
Собственные значения:
-5.27 4.68 2.59
```

3 Исходный код

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

using namespace std;

vector<vector<double>> transpose(const vector<vector<double>>& A) {
    int rows = A.size();
    int cols = A[0].size();
    vector<vector<double>> AT(cols, vector<double>(rows, 0.0));
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            AT[j][i] = A[i][j];
        }
    }
    return AT;
}

vector<vector<double>> multiply_m(const vector<vector<double>>& A, const vector<vector
    <double>>& B) {
    int rowsA = A.size();
    int colsA = A[0].size();
    int colsB = B[0].size();
    vector<vector<double>> C(rowsA, vector<double>(colsB, 0.0));
    for (int i = 0; i < rowsA; i++)
    {
        for (int j = 0; j < colsB; j++)
        {
            for (int k = 0; k < colsA; k++)
            {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    return C;
}

vector<vector<double>> multiply_s(const vector<vector<double>>& A, double scalar) {
    int rows = A.size();
    int cols = A[0].size();
    vector<vector<double>> C(rows, vector<double>(cols, 0.0));
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
```

```
47          {
48              C[i][j] = A[i][j] * scalar;
49          }
50      }
51      return C;
52  }
53
54  vector<vector<double>> division(const vector<vector<double>>& A, double B) {
55      int rows = A.size();
56      int cols = A[0].size();
57      vector<vector<double>> C(rows, vector<double>(cols, 0.0));
58      for (int i = 0; i < rows; i++)
59      {
60          for (int j = 0; j < cols; j++)
61          {
62              C[i][j] = A[i][j] / B;
63          }
64      }
65      return C;
66  }
67
68  vector<vector<double>> difference(const vector<vector<double>>& A, const vector<vector
        <double>>& B) {
69      int rows = A.size();
70      int cols = A[0].size();
71      vector<vector<double>> C(rows, vector<double>(cols, 0.0));
72      for (int i = 0; i < rows; i++)
73      {
74          for (int j = 0; j < cols; j++)
75          {
76              C[i][j] = A[i][j] - B[i][j];
77          }
78      }
79      return C;
80  }
81
82  double norm_vector(const vector<double>& b) {
83      double norm = 0.0;
84      for (int i = 0; i < b.size(); i++)
85      {
86          norm += b[i] * b[i];
87      }
88      return sqrt(norm);
89  }
90
91  vector<vector<double>> Householder(const vector<vector<double>>& H, int index) {
92      vector<vector<double>> H1 = H;
93      int n = H1[0].size();
94      vector<double> x1(n, 0.0);
```

```
95      vector<double> b(n, 0.0);
96      for (int k = 0; k < n; k++)
97      {
98          b[k] = H1[k][index];
99      }
100     double norm = norm_vector(b);
101     for (int i = 0; i < n; i++)
102     {
103         if (i == index) {
104             x1[i] = H1[i][index] + (H1[i][index] > 0 ? 1 : -1) * norm;
105         } else if (i < index) {
106             x1[i] = 0.0;
107         } else {
108             x1[i] = H1[i][index];
109         }
110     }
111     vector<vector<double>> v1(n, vector<double>(n, 0.0));
112     for (int i = 0; i < n; i++)
113     {
114         v1[i][0] = x1[i];
115     }
116     vector<vector<double>> v1_T = transpose(v1);
117     vector<vector<double>> E(n, vector<double>(n, 0.0));
118     for (int i = 0; i < n; i++)
119     {
120         E[i][i] = 1.0;
121     }
122     H1 = difference(E, multiply_s(division(multiply_m(v1, v1_T), multiply_m(v1_T, v1)
            [0][0]), 2.0));
123     return H1;
124 }
125
126 pair<vector<vector<double>>, vector<vector<double>>> QR(const vector<vector<double>>&
        A) {
127     vector<vector<double>> A0 = A;
128     vector<vector<vector<double>>> H;
129     int n = A.size();
130     for (int i = 0; i < n - 1; i++)
131     {
132         vector<vector<double>> H0 = Householder(A0, i);
133         A0 = multiply_m(H0, A0);
134         H.push_back(H0);
135     }
136     for (int i = 0; i < H.size() - 1; i++)
137     {
138         H[i + 1] = multiply_m(H[i], H[i + 1]);
139     }
140     return make_pair(A0, H.back());
141 }
```

```
142
143 vector<double> eigenvalues(const vector<vector<double>>& A, const double e) {
144     double m = 1.0;
145     vector<vector<double>> A0 = A;
146     while (m > e)
147     {
148         vector<vector<double>> Q0 = QR(A0).second;
149         vector<vector<double>> R0 = QR(A0).first;
150         vector<double> a;
151         A0 = multiply_m(R0, Q0);
152         int n = A0.size();
153         for (int i = 0; i < n; i++)
154         {
155             for (int j = 0; j < n; j++)
156             {
157                 if (i > j) a.push_back(A0[i][j]);
158             }
159         }
160         m = norm_vector(a);
161     }
162     vector<double> L;
163     int n = A0.size();
164     for (int i = 0; i < n; i++)
165     {
166         L.push_back(A0[i][i]);
167     }
168     return L;
169 }
170
171 int main() {
172     vector<vector<double>> A(3, vector <double>(3, 0));
173     double e;
174     int n = A.size();
175
176     ifstream fina("matrix.txt");
177     for (int i = 0; i < n; i++)
178     {
179         for (int j = 0; j < n; j++)
180             fina >> A[i][j];
181     }
182     fina >> e;
183
184     pair<vector<vector<double>>, vector<vector<double>>> QRresult = QR(A);
185     vector<vector<double>> Q = QRresult.second;
186     vector<vector<double>> R = QRresult.first;
187     vector<double> w = eigenvalues(A, e);
188
189     ofstream fout("answer.txt");
190     fout.precision(2);
```

```
191     fout << fixed;
192 
193     fout << " Q:" << endl;
194     for (const auto& row : Q)
195     {
196         for (const auto& elem : row)
197             fout << fixed << elem << "\t";
198         fout << endl;
199     }
200     fout << " R:" << endl;
201     for (const auto& row : R)
202     {
203         for (const auto& elem : row)
204             fout << fixed << elem << "\t";
205         fout << endl;
206     }
207     fout << " :" << endl;
208     for (double val : w) {
209         fout << val << " ";
210     }
211     fout << endl;
212 
213     return 0;
214 }
```