

**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: Ю. В. Кон
Преподаватель: Д. Е. Пивоваров
Группа: М8О-303Б-21
Дата:
Оценка:
Подпись:

Москва, 2024

1.1 LU - разложение 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: 12

$$\begin{cases} -x_1 - 8x_2 + 5x_4 = -60 \\ 6x_1 - 6x_2 + 2x_3 + 4x_4 = -10 \\ 9x_1 - 5x_2 - 6x_3 + 4x_4 = 65 \\ -5x_1 - 9x_3 + x_4 = 18 \end{cases}$$

2 Результаты работы

```
Решение системы:
7
6
-6
-1

Определитель матрицы: -356

Обратная матрица:
-0.0280899  -0.0421348  0.0955056  -0.0730337
-0.904494   1.64326  -0.724719   0.848315
-0.123596   0.314607   -0.179775   0.0786517
-1.25281    2.62079  -1.14045    1.3427
```

Рис. 1: Вывод программы в консоли

3 Исходный код

Файл с первым заданием лабораторной работы:

```
1 #include <iostream>
2 #include <vector>
3 #include <fstream>
4
5 using namespace std;
6
7 void LU(vector<vector<double>>& matrix, vector<vector<double>>& L, vector<vector<
    double>>& U, int n) {
8     U = matrix;
9
10    for(int i = 0; i < n; i++)
11        for(int j = i; j < n; j++)
12            L[j][i]=U[j][i]/U[i][i];
13
14    for(int k = 1; k < n; k++)
15    {
16        for(int i = k-1; i < n; i++)
17            for(int j = i; j < n; j++)
18                L[j][i]=U[j][i]/U[i][i];
19
20        for(int i = k; i < n; i++)
21            for(int j = k-1; j < n; j++)
22                U[i][j]=U[i][j]-L[i][k-1]*U[k-1][j];
23    }
24 }
25
26 vector<vector<double>> result(vector<vector<double>>& matrix_1, vector<vector<double
    >>& matrix_2, vector<vector<double>>& L, vector<vector<double>>& U, int n) {
27
28     LU(matrix_1, L, U, n);
29     vector<vector<double>> res = matrix_2;
30     int m = res[0].size();
31     for (int k=0; k<m; k++)
32         for (int i=0; i<n; i++)
33             for (int j=0; j<i; j++)
34                 res[i][k] -= res[j][k]*L[i][j];
35     for (int k=0; k<m; k++) {
36         for (int i=n-1; i>-1; i--) {
37             for (int j=i+1; j<n; j++) {
38                 res[i][k] -= res[j][k]*U[i][j];
39             }
40             res[i][k] /= U[i][i];
41         }
42     }
43     return res;
44 }
```

```

45
46 double determinant(const vector<vector<double>>& U) {
47     int n = U.size();
48     double det = 1;
49     for (int i = 0; i < n; ++i)
50         det *= U[i][i];
51     return det;
52 }
53
54 int main(){
55     int n = 4;
56     vector<vector<double>> matrix(n, vector<double>(n, 0)), b(n, vector<double>(n, 0)
57         );
58     ifstream in1("matrix.txt"), in2("b.txt");
59     for (int i = 0; i < n; i++)
60     {
61         for (int j = 0; j < n; j++){
62             in1 >> matrix[i][j];
63         }
64     }
65     for (int i = 0; i < n; i++)
66     {
67         in2 >> b[i][0];
68     }
69     vector<vector<double>> L(n, vector<double>(0)), U(n, vector<double>(0)), E(n,
70         vector<double>(0));
71     for (int i = 0; i < n; i++){
72         for (int j=0; j<n; j++){
73             L[i].push_back(0);
74             U[i].push_back(0);
75             if (i == j){
76                 E[i].push_back(1);
77             }else E[i].push_back(0);
78         }
79     }
80     LU(matrix, L, U, n);
81     double det = 1;
82     det = determinant(U);
83     vector<vector<double>> res = result(matrix, b, L, U, n);
84     vector<vector<double>> invert = result(matrix, E, L, U, n);
85     ofstream out("output.txt");
86     out << " : " << endl;
87     for (int i = 0; i < n; ++i)
88     {
89         out << res[i][0] << endl;
90         out << endl << " : " << det << endl;
91         out << endl << " : " << endl;
92     }
93     for(int i = 0; i < n; i++)
94     {
95         for(int j = 0; j < n; j++)

```

```
92 | {  
93 |     out << invert[i][j] << "\t";  
94 | }  
95 | out << endl;  
96 | }  
97 | return 0;  
98 | }
```

1.2 Метод прогонки

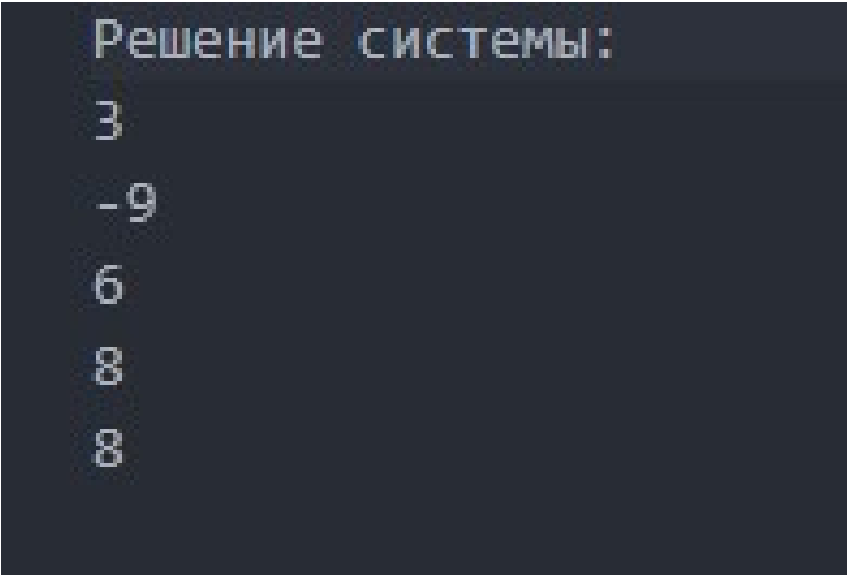
4 Постановка задачи

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: 12

$$\begin{cases} -11x_1 + 9x_2 = -114 \\ x_1 - 8x_2 + x_3 = 81 \\ -2x_2 - 11x_3 + 5x_4 = -8 \\ 3x_3 - 14x_4 + 7x_5 = -38 \\ 8x_4 + 10x_5 = 144 \end{cases}$$

5 Результаты работы



```
Решение системы:
3
-9
6
8
8
```

Рис. 2: Вывод программы в консоли

6 Исходный код

Файл со вторым заданием лабораторной работы:

```
1  #include <iostream>
2  #include <vector>
3  #include <fstream>
4
5  using namespace std;
6
7  vector<double> the_run_through_method(vector<vector<double>> matrix, vector<double> b,
      int n){
8      vector<double> P(n, 0.0), Q(n, 0.0), x(n, 0.0);
9      //a=matrix[i][0], b=matrix[i][1], c=matrix[i][2], d=b[i]
10     P[0] = -matrix[0][2]/matrix[0][1];
11     Q[0] = b[0]/matrix[0][1];
12     for (int i = 1; i < n; i++)
13     {
14         P[i] = -matrix[i][2]/(matrix[i][1] + matrix[i][0]*P[i-1]);
15         Q[i] = (b[i] - matrix[i][0]*Q[i-1])/(matrix[i][1] + matrix[i][0]*P[i-1]);
16     }
17     P[n-1] = 0;
18     x[n-1] = Q[n-1];
19     for (int i = n - 2; i >= 0; i--)
20     {
21         x[i] = P[i]*x[i+1] + Q[i];
22     }
23     return x;
24 }
25
26 int main(){
27     int n = 5;
28     vector<vector<double>> matrix(n, vector<double>(n, 0.0));
29     vector<double> b(n, 0);
30     ifstream in1("matrix.txt"), in2("b.txt");
31     for (int i = 0; i < n; i++)
32     {
33         for (int j = 0; j < n-2; j++){
34             in1 >> matrix[i][j];
35         }
36     }
37     for (int i = 0; i < n; i++)
38     {
39         in2 >> b[i];
40     }
41     vector<double> res = the_run_through_method(matrix, b, n);
42
43     ofstream out("output.txt");
44
45     out << " : " << endl;
```

```
46 |     for (int i = 0; i < n; ++i)
47 |         out << res[i] << endl;
48 |
49 |     return 0;
50 | }
```


1.3 Метод простых итераций. Метод Зейделя

7 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: 12

$$\begin{cases} 14x_1 - 4x_2 - 2x_3 + 3x_4 = 38 \\ -3x_1 + 23x_2 - 6x_3 - 9x_4 = -195 \\ -7x_1 - 8x_2 + 21x_3 - 5x_4 = -27 \\ -2x_1 - 2x_2 + 8x_3 + 18x_4 = 142 \end{cases}$$

8 Результаты работы

```
Решение системы методом простых итераций:
-0.999999
-6
-1.99992
8.00003
Количество итераций: 10
Решением системы методом Зейделя:
-1.00012
-6.00007
-2.00006
8.00001
Количество итераций: 9
```

Рис. 3: Вывод программы в консоли

9 Исходный код

Файл с третьим заданием лабораторной работы:

```
1  #include <iostream>
2  #include <vector>
3  #include <cmath>
4  #include <fstream>
5
6  using namespace std;
7
8  vector<double> simple_iteration(vector<vector<double>>& matrix, vector<double>& b,
   double eps, int &iter, int n) {
9
10     vector<double> x(n, 0.0);
11     while (true) {
12         iter++;
13         vector<double> x_new(n, 0.0);
14         for (int i = 0; i < n; i++) {
15             double sum = 0.0;
16             for (int j = 0; j < n; j++) {
17                 if (j != i) {
18                     sum += matrix[i][j] * x[j];
19                 }
20             }
21             x_new[i] = (b[i] - sum) / matrix[i][i];
22         }
23         double diff = 0.0;
24         for (int i = 0; i < n; i++) {
25             diff = max(diff, fabs(x[i] - x_new[i]));
26         }
27         if (diff < eps) {
28             break;
29         }
30         x = x_new;
31     }
32     return x;
33 }
34
35 vector<double> Seidel(vector<vector<double>>& matrix, vector<double>& b, double eps,
   int& iter, int n) {
36
37     vector<double> x(n, 0.0);
38     vector<double> x_new(n, 0.0);
39     while (true) {
40         iter++;
41         for (int i = 0; i < n; i++) {
42             double sum1 = 0.0;
43             double sum2 = 0.0;
44             for (int j = 0; j < i; j++) {
```

```

45         sum1 += matrix[i][j] * x_new[j];
46     }
47     for (int j = i + 1; j < n; j++) {
48         sum2 += matrix[i][j] * x[j];
49     }
50     x_new[i] = (b[i] - sum1 - sum2) / matrix[i][i];
51 }
52 double diff = 0.0;
53 for (int i = 0; i < n; i++) {
54     diff = max(diff, fabs(x[i] - x_new[i]));
55 }
56 if (diff < eps) {
57     break;
58 }
59 x = x_new;
60 }
61 return x;
62 }
63
64 int main() {
65     int n = 4;
66     vector<vector<double>> matrix(n, vector<double>(n, 0));
67     vector<double> b(n, 0);
68     ifstream in1("matrix.txt"), in2("b.txt");
69     for (int i = 0; i < n; i++)
70     {
71         for (int j = 0; j < n; j++){
72             in1 >> matrix[i][j];
73         }
74     }
75
76     for (int i = 0; i < n; i++)
77     {
78         in2 >> b[i];
79     }
80     double eps = 0.0001;
81     int iteration_1 = 0;
82     int iteration_2 = 0;
83
84     vector<double> x_1 = simple_iteration(matrix, b, eps, iteration_1, n);
85     ofstream out("output.txt");
86     out << "      : " << endl;
87     for (int i = 0; i < n; ++i)
88         out << x_1[i] << endl;
89     out << " : " << iteration_1 << endl;
90
91     vector<double> x_2 = Seidel(matrix, b, eps, iteration_2, n);
92     out << "      : " << endl;
93     for (int i = 0; i < n; ++i)

```

```
94 |         out << x_2[i] << endl;
95 |     out << " : " << iteration_2 << endl;
96 |
97 |     return 0;
98 | }
```

1.4 Метод вращений

10 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: 12

$$\begin{pmatrix} 7 & 3 & -1 \\ 3 & -7 & -8 \\ -1 & -8 & -2 \end{pmatrix}$$

11 Результаты работы

```
Собственные значения:  
8.57605  
-13.0509  
2.47486  
Собственные векторы:  
0.873738    -0.092421    0.477536  
0.344594    0.810511    -0.473632  
-0.343274    0.578386    0.740022  
Количество итераций: 6
```

Рис. 4: Вывод программы в консоли

12 Исходный код

Файл с четвертым заданием лабораторной работы:

```
1  #include <iostream>
2  #include <vector>
3  #include <fstream>
4  #include <cmath>
5
6  const double eps = 1e-4;
7
8  using namespace std;
9
10 void rotation_method(vector<vector<double>> matrix, vector<vector<double>> U, vector<
    vector<double>> &V, vector<double> &w, int n, int &k){
11     for (int i = 0; i < n; i++)
12         for (int j = 0; j < n; j++)
13             {
14                 if (i == j) V[i][j] = 1;
15                 else V[i][j] = 0;
16             }
17
18     double f = 0.0;
19     vector<vector<double>> B(n, vector<double>(n, 0));
20     for (int i = 0; i < n; i++)
21         for (int j = i + 1; j < n; j++)
22             f = f + matrix[i][j]*matrix[i][j];
23     f = sqrt(f);
24
25     double phi = 0.0;
26
27     while (f > eps)
28     {
29         int p = 1, q = 2;
30         double g = 0.0;
31
32         for (int i = 0; i < n; i++)
33             {
34                 for (int j = i + 1; j < n; j++)
35                     {
36                         if (matrix[i][j] > g){
37                             g = matrix[i][j];
38                             p = i;
39                             q = j;
40                         }
41                         if (-matrix[i][j] > g){
42                             g = -matrix[i][j];
43                             p = i;
44                             q = j;
45                         }
46                     }
47             }
```

```

46     }
47 }
48 for (int i = 0; i < n; i++)
49 {
50     for (int j = 0; j < n; j++)
51     {
52         if (i == j) U[i][j] = 1;
53         else U[i][j] = 0;
54     }
55 }
56 if (matrix[p][p] == matrix[q][q]) phi = M_PI/4;
57 else phi = atan(2 * matrix[p][q]/(matrix[p][p] - matrix[q][q]))/2;
58
59 U[p][p] = cos(phi);
60 U[p][q] = -sin(phi);
61 U[q][p] = sin(phi);
62 U[q][q] = cos(phi);
63
64 for (int i = 0; i < n; i++)
65 {
66     for (int j = 0; j < n; j++)
67     {
68         if ((i == p) || (i == q) || (j == p) || (j == q))
69         {
70             B[i][j] = 0;
71             for (int p = 0; p < n; p++)
72                 B[i][j] = B[i][j] + U[p][i]*matrix[p][j];
73         }
74     }
75 }
76 for (int i = 0; i < n; i++)
77 {
78     for (int j = 0; j < n; j++)
79     {
80         if ((i == p) || (i == q) || (j == p) || (j == q))
81         {
82             matrix[i][j] = 0;
83             for (int p = 0; p < n; p++)
84                 matrix[i][j] = matrix[i][j] + B[i][p]*U[p][j];
85         }
86     }
87 }
88 for (int i = 0; i < n; i++)
89 {
90     for (int j = 0; j < n; j++)
91     {
92         if ((i == p) || (i == q) || (j == p) || (j == q))
93         {
94             B[i][j] = 0;

```

```

95         for (int p = 0; p < n; p++)
96             B[i][j] = B[i][j] + V[i][p]*U[p][j];
97     }
98 }
99 }
100 for (int i = 0; i < n; i++)
101 {
102     for (int j = 0; j < n; j++)
103     {
104         if ((i == p) || (i == q) || (j == p) || (j == q))
105             V[i][j] = B[i][j];
106     }
107 }
108
109 f = 0.0;
110 for (int i = 0; i < n; i++)
111     for (int j = i + 1; j < n; j++)
112         f = f + matrix[i][j]*matrix[i][j];
113
114 f = sqrt(f);
115 ++k;
116 }
117
118 for (int i = 0; i < n; i++)
119 {
120     w[i] = matrix[i][i];
121 }
122 }
123
124 int main() {
125     int n = 3;
126     vector<vector<double>> matrix(n, vector <double>(n, 0)), Vectors(n, vector <double>
127         >(n, 0)), U(n, vector <double>(n, 0));
128     vector<double> w(n, 0);
129     int k = 0;
130     ifstream in("matrix.txt");
131     for (int i = 0; i < n; i++)
132     {
133         for (int j = 0; j < n; j++)
134             in >> matrix[i][j];
135     }
136
137     rotation_method(matrix, U, Vectors, w, n, k);
138
139     ofstream out("output.txt");
140
141     out << " :" << endl;
142     for (size_t i = 0; i < w.size(); ++i)
143     {

```



```

143     out << w[i] << endl;
144 }
145
146 out << " : " << endl;
147 for (const auto& row : Vectors)
148 {
149     for (const auto& elem : row)
150         out << elem << "\t";
151     out << endl;
152 }
153
154 out << " : " << k << endl;
155
156 return 0;
157 }

```

1.5 QR – разложение матриц

13 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: 12

$$\begin{pmatrix} 5 & -1 & -2 \\ -4 & 3 & -3 \\ -2 & -1 & 1 \end{pmatrix}$$

14 Результаты работы

```
Матрица Q:
0.745356      0.272166      -0.608581
-0.596285     0.680414      -0.426006
-0.298142     -0.680414     -0.669439

Матрица R:
6.7082  -2.23607    5.55112e-017
0      2.44949  -3.26599
0      0      1.82574

Собственные значения:
6.38443  3.83943  -1.22386
```

Рис. 5: Вывод программы в консоли

15 Исходный код

Файл с пятым заданием лабораторной работы:

```
1  #include <iostream>
2  #include <vector>
3  #include <fstream>
4  #include <cmath>
5
6  using namespace std;
7
8  void qr_decomposition(vector<vector<double>>& A, vector<vector<double>>& Q, vector<
    vector<double>>& R, double eps, int n) {
9      int m = A[0].size();
10     R = vector<vector<double>>(m, vector<double>(m, 0.0));
11     Q = A;
12
13     for (int j = 0; j < m; ++j) {
14         for (int k = 0; k < j; ++k) {
15             double dot_prod = 0.0;
16             for (int i = 0; i < n; ++i) {
17                 dot_prod += Q[i][j] * Q[i][k];
18             }
19             for (int i = 0; i < n; ++i) {
20                 Q[i][j] -= dot_prod * Q[i][k];
21             }
22         }
23
24         double norma = 0.0;
25         for (int i = 0; i < n; ++i) {
26             norma += Q[i][j] * Q[i][j];
27         }
28         norma = sqrt(norma);
29
30         for (int i = 0; i < n; ++i) {
31             Q[i][j] /= norma;
32             R[j][j] = norma;
33         }
34
35         for (int k = j + 1; k < m; ++k) {
36             double dot_prod = 0.0;
37             for (int i = 0; i < n; ++i) {
38                 dot_prod += Q[i][j] * A[i][k];
39             }
40             R[j][k] = dot_prod;
41         }
42     }
43 }
44
```

```

45 vector<vector<double>> multiply(vector<vector<double>>& A, vector<vector<double>>& B,
46     int n1, int n2) {
47     int m = B[0].size();
48     vector<vector<double>> res(n1, vector<double>(m, 0.0));
49
50     for (int i = 0; i < n1; ++i) {
51         for (int j = 0; j < m; ++j) {
52             for (int k = 0; k < n2; ++k) {
53                 res[i][j] += A[i][k] * B[k][j];
54             }
55         }
56     }
57
58     return res;
59 }
60
61 vector<double> comp_eigenvalues(vector<vector<double>>& A, int iter, double eps, int n
62     ) {
63     vector<vector<double>> Ak = A;
64
65     for (int i = 0; i < iter; ++i) {
66         vector<vector<double>> Q, R;
67         qr_decomposition(Ak, Q, R, eps, n);
68         int n_1 = Q.size();
69         int n_2 = R.size();
70         Ak = multiply(R, Q, n_1, n_2);
71     }
72
73     vector<double> eigenvalues(n);
74
75     for (int i = 0; i < n; ++i) {
76         eigenvalues[i] = Ak[i][i];
77     }
78
79     return eigenvalues;
80 }
81
82 int main(){
83     int n = 3;
84     vector<vector<double>> A(n, vector<double>(n, 0));
85     double eps = 1e-4;
86
87     ifstream in("matrix.txt");
88     for (int i = 0; i < n; i++)
89     {
90         for (int j = 0; j < n; j++)
91             in >> A[i][j];

```

```

92     }
93
94     vector<vector<double>> Q;
95     vector<vector<double>> R;
96     int iter = 45;
97     qr_decomposition(A, Q, R, eps, n);
98     vector<double> eigenvalues = comp_eigenvalues(A, iter, eps, n);
99
100     ofstream out("output.txt");
101
102     out << " Q:" << endl;
103     for (const auto& row : Q)
104     {
105         for (const auto& elem : row)
106             out << elem << "\t";
107         out << endl;
108     }
109     out << " R:" << endl;
110     for (const auto& row : R)
111     {
112         for (const auto& elem : row)
113             out << elem << "\t";
114         out << endl;
115     }
116     out << " :" << endl;
117     for (double val : eigenvalues) {
118         out << val << " ";
119     }
120     out << endl;
121
122     return 0;
123 }

```