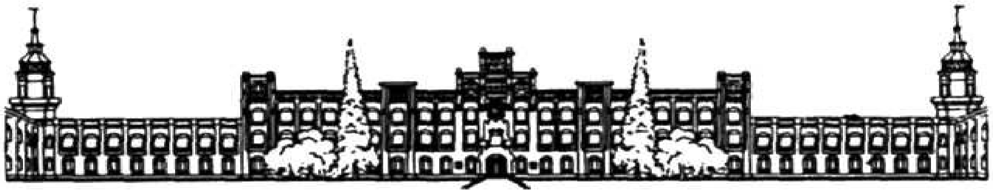

Міністерство освіти, науки, молоді та спорту України
Національний технічний університет України
«Київський політехнічний інститут»
Фізико-технічний інститут

Лабораторна робота №2

Методи обчислень
«РОЗВ'ЯЗАННЯ СИСТЕМ ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ (СЛАР) ПРЯМИМИ МЕТОДАМ»

Варіант 4

Підготував:
студент 3 курсу
групи ФІ-84
Коломієць Андрій Юрійович

Викладач:
Стьопочкіна Ірина Валеріївна

Київ – 2021

Вхідна система

Матриця A

2,12	0,42	1,34	0,88
0,42	3,95	1,87	0,43
1,34	1,87	2,98	0,46
0,88	0,43	0,46	4,44

Вектор B

11,172
0,115
0,009
9,349

Результати роботи програми

```
_________________________________________________
Work of program which solve matrix
_________________________________________________
Matrix left side
        2.12    0.42    1.34    0.88
        0.42    3.95    1.87    0.43
        1.34    1.87    2.98    0.46
        0.88    0.43    0.46    4.44
Matrix right side
                11.172
                0.115
                0.009
                9.349
_________________________________________________
Process of finding solving matrix...
Direct course factorization:
         -step with index i=0 j=0:
        1.45602         0               0               0
                0               0               0               0
                0               0               0               0
                0               0               0               0
         -step with index i=1 j=0:
        1.45602         0               0               0
        0.288457                0               0               0
                0               0               0               0
                0               0               0               0
         -step with index i=2 j=0:
        1.45602         0               0               0
        0.288457                0               0               0
        0.920316                0               0               0
                0               0               0               0
         -step with index i=3 j=0:
        1.45602         0               0               0
```

```
0.288457                0               0               0
0.920316                0               0               0
0.604386                0               0               0
 -step with index i=1 j=1:
1.45602         0               0               0
0.288457        1.96642         0               0
0.920316                0               0               0
0.604386                0               0               0
 -step with index i=2 j=1:
1.45602         0               0               0
0.288457        1.96642         0               0
0.920316        0.815966                0               0
0.604386                0               0               0
 -step with index i=2 j=2:
1.45602         0               0               0
0.288457        1.96642         0               0
0.920316        0.815966        1.21129         0
0.604386                0               0               0
 -step with index i=3 j=1:
1.45602         0               0               0
0.288457        1.96642         0               0
0.920316        0.815966        1.21129         0
0.604386        0.130013                0               0
 -step with index i=3 j=2:
1.45602         0               0               0
0.288457        1.96642         0               0
0.920316        0.815966        1.21129         0
0.604386        0.130013        -0.167023               0
 -step with index i=3 j=3:
1.45602         0               0               0
0.288457        1.96642         0               0
0.920316        0.815966        1.21129         0
0.604386        0.130013        -0.167023       2.00747
 -step with index i=0 j=1:
1.45602         0               0               0
0.288457        1.96642         0               0
0.920316        0.815966        1.21129         0
0.604386        0.130013        -0.167023       2.00747
 -step with index i=0 j=2:
1.45602         0               0               0
0.288457        1.96642         0               0
0.920316        0.815966        1.21129         0
0.604386        0.130013        -0.167023       2.00747
 -step with index i=0 j=3:
1.45602         0               0               0
0.288457        1.96642         0               0
0.920316        0.815966        1.21129         0
0.604386        0.130013        -0.167023       2.00747
```

```
         -step with index i=1 j=2:
        1.45602         0               0               0
        0.288457        1.96642         0               0
        0.920316        0.815966        1.21129         0
        0.604386        0.130013        -0.167023       2.00747
         -step with index i=1 j=3:
        1.45602         0               0               0
        0.288457        1.96642         0               0
        0.920316        0.815966        1.21129         0
        0.604386        0.130013        -0.167023       2.00747
         -step with index i=2 j=3:
        1.45602         0               0               0
        0.288457        1.96642         0               0
        0.920316        0.815966        1.21129         0
        0.604386        0.130013        -0.167023       2.00747
Temp solving step:
         -step with index i=0:
                7.67296
                0
                0
                0
         -step with index i=1:
                7.67296
                -1.06708
                0
                0

         -step with index i=2:
                7.67296
                -1.06708
                -5.10353
                0
         -step with index i=3:
                7.67296
                -1.06708
                -5.10353
                1.99151
Matrix of factorization:
        1.45602         0               0               0
        0.288457        1.96642         0               0
        0.920316        0.815966        1.21129         0
        0.604386        0.130013        -0.167023       2.00747
Transponate matrix of factorization:
        1.45602 0.288457        0.920316        0.604386
                0       1.96642 0.815966        0.130013
                0               0       1.21129 -0.167023
                0               0               0       2.00747
```

```
Solving step:
         -step with index i=4:
                0
                0
                0
                0.992054

         -step with index i=2:
                0
                0
                -4.07652
                0.992054

         -step with index i=1:
                0
                1.08331
                -4.07652
                0.992054
         -step with index i=0:
                7.22006
                1.08331
                -4.07652
                0.992054
Inconspicuous vector:
                1.77636e-15
                2.63678e-16
                1.75207e-16
                0
_________________________________________________
```

Код

```
#include <iostream>
#include <string>
#include <cmath>
#include <stdio.h>
#include <windows.h>
#include <conio.h>

using namespace std;

#define _type_ long double
#define line
cout<<endl<<"_________________________________________________"<<endl;


_type_** find_lower_matrix(_type_** matrix_left, _type_** matrix_lower, int
size_matrix);
_type_* temp_solving(_type_** matrix_lower, _type_* temp_matrix, _type_*
matrix_right, int size_matrix);
_type_* finding_solving(_type_** matrix_lower, _type_* temp_matrix, _type_*
solving, int size_matrix);
_type_* inconspicuous(_type_** matrix_left, _type_* matrix_right, _type_*
solving, _type_* temp_matrix, int size_matrix);

void output_vector(_type_* matrix_right, int size_matrix);
void output_double_matrix(_type_** matrix_left, int size_matrix);

int main()
{

                  int size_matrix=4;


      _type_** matrix_left = new _type_*[size_matrix];

      for (int i = 0; i < size_matrix; i++)
      {
            matrix_left[i] = new _type_[size_matrix];
      }

      for (int i = 0; i < size_matrix; i++)
      {
            for (int j = 0; j < size_matrix; j++)
            {
                  matrix_left[i][j] = 0;
            }
      }


      _type_* matrix_right = new _type_[size_matrix];

      for (int i = 0; i < size_matrix; i++)
      {
            matrix_right[i] = 0;
      }


      _type_** matrix_lower = new _type_*[size_matrix];

      for (int i = 0; i < size_matrix; i++)
      {
            matrix_lower[i] = new _type_[size_matrix];
      }

      for (int i = 0; i < size_matrix; i++)
      {
          for (int j = 0; j < size_matrix; j++)
          {
              matrix_lower[i][j] = 0;
          }
      }

      _type_* temp_matrix = new _type_[size_matrix];

      for (int i = 0; i < size_matrix; i++)
      {
              temp_matrix[i] = 0;
      }

      _type_* solving = new _type_[size_matrix];

      for (int i = 0; i < size_matrix; i++)
      {
          solving[i] = 0;
      }


      /////////////////////////////Progam start/////////////////////////////

      line

      cout << endl << "Work of program which solve matrix" << endl;

      line

      cout << endl << "Matrix left side" << endl << endl;

      matrix_left[0][0] = 2.12;     matrix_left[0][1] = 0.42;    matrix_left[0][2] = 1.34;    matrix_left[0][3] = 0.88;        matrix_right[0] =11.172;

      matrix_left[1][0] = 0.42;     matrix_left[1][1] = 3.95;    matrix_left[1][2] = 1.87;    matrix_left[1][3] = 0.43;        matrix_right[1] =0.115;

      matrix_left[2][0] = 1.34;     matrix_left[2][1] = 1.87;    matrix_left[2][2] = 2.98;    matrix_left[2][3] = 0.46;        matrix_right[2] =0.009;

      matrix_left[3][0] = 0.88;     matrix_left[3][1] = 0.43;    matrix_left[3][2] = 0.46;    matrix_left[3][3] = 4.44;        matrix_right[3] =9.349;

      output_double_matrix(matrix_left,size_matrix);


      cout << endl << "Matrix right side" << endl << endl;

      output_vector(matrix_right, size_matrix);

      line

      cout << endl << "Process of finding solving matrix..." << endl << endl;

      find_lower_matrix(matrix_left, matrix_lower, size_matrix);

      temp_solving(matrix_lower, temp_matrix, matrix_right, size_matrix);

      finding_solving(matrix_lower,temp_matrix,solving,size_matrix);

      inconspicuous(matrix_left, matrix_right, solving,temp_matrix, size_matrix);


      line
```

```
      //////////////////////////////Progam end//////////////////////////////

      delete[] solving;

      delete[] temp_matrix;


      for (int i = 0; i < size_matrix; i++)
      {
            delete matrix_lower[i];
      }

      delete[] matrix_lower;

      delete[] matrix_right;


      for (int i = 0; i < size_matrix; i++)
      {
            delete matrix_left[i];
      }

      delete[] matrix_left;


      return 0;

}


_type_** find_lower_matrix(_type_** matrix_left, _type_** matrix_lower, int
size_matrix)
{

      cout << endl << "Direct course factorization:" << endl;

      matrix_lower[0][0] = sqrt(matrix_left[0][0]);

      cout << endl << "\t -step with index i=" << 0 << " j=" << 0 <<":"<< endl;

      output_double_matrix( matrix_lower,size_matrix);

      for (int i = 1; i < size_matrix; i++)
      {
            matrix_lower[i][0] = (matrix_left[i][0]) / matrix_lower[0][0];

            cout << endl << "\t -step with index i=" << i << " j=" << 0 << ":" <<
endl;

            output_double_matrix(matrix_lower, size_matrix);
      }

      for (int i = 1; i < size_matrix; i++)
      {
            for (int j = 1; j < size_matrix; j++)
            {
                  if (i > j)
                  {
                        _type_ temp = 0;

                        for (int k = 0; k <= j - 1; k++)
                        {
                              temp = temp + matrix_lower[i][k] * matrix_lower[j]
[k];
```

```
                        }

                        temp = matrix_left[i][j] - temp;

                        matrix_lower[i][j] = temp / matrix_lower[j][j];

                        cout << endl << "\t -step with index i=" << i << " j=" <<
j << ":" << endl;

                        output_double_matrix(matrix_lower, size_matrix);
                  }
            }


            _type_ temp = 0;

            for (int j = 0; j <=i-1; j++)
            {
                  temp = temp + pow(matrix_lower[i][j], 2);
            }

            temp = matrix_left[i][i] - temp;

            matrix_lower[i][i] = sqrt(temp);

            cout << endl << "\t -step with index i=" << i << " j=" << i << ":" <<
endl;

            output_double_matrix(matrix_lower, size_matrix);
      }





      for (int i = 0; i < size_matrix; i++)
      {
            for (int j = 0; j < size_matrix; j++)
            {
                  if (j > i)
                  {
                        matrix_left[i][j] = 0;

                        cout << endl << "\t -step with index i=" << i << " j=" <<
j << ":" << endl;

                        output_double_matrix(matrix_lower, size_matrix);
                  }
            }
      }


      return matrix_lower;
}

_type_* temp_solving(_type_** matrix_lower, _type_* temp_matrix, _type_*
matrix_right, int size_matrix)
{
      cout << endl << "Temp solving step:" << endl;

      temp_matrix[0] = matrix_right[0] / matrix_lower[0][0];

      cout << endl << "\t -step with index i=" << 0 << ":" << endl;
```

```
    output_vector(temp_matrix, size_matrix);

    for (int i = 1; i < size_matrix; i++)
    {
        _type_ temp = 0;

        for (int j = 0; j <i; j++)
        {
            temp = temp + matrix_lower[i][j] * temp_matrix[j];
        }


        temp =  - temp+ matrix_right[i];

        temp_matrix[i] = temp / matrix_lower[i][i];

        cout << endl << "\t -step with index i=" << i << ":" << endl;

        output_vector(temp_matrix, size_matrix);
    }

    return  temp_matrix;
}

_type_* finding_solving(_type_** matrix_lower, _type_* temp_matrix, _type_*
solving, int size_matrix)
{

    cout << endl << "Matrix of factorization:" << endl;

    output_double_matrix(matrix_lower, size_matrix);

    for (int i = 0; i < size_matrix; i++)
    {
        for (int j = 0; j < i; j++)
        {
            _type_ temp;
            temp = matrix_lower[i][j];
            matrix_lower[i][j] = matrix_lower[j][i];
            matrix_lower[j][i] = temp;
        }
    }

    cout << endl << "Transponate matrix of factorization:" << endl;

    output_double_matrix(matrix_lower, size_matrix);

    cout << endl << "Solving step:" << endl;

    solving[size_matrix - 1] = temp_matrix[size_matrix - 1] /
matrix_lower[size_matrix - 1][size_matrix - 1];

    cout << endl << "\t -step with index i=" << size_matrix << ":" << endl;

    output_vector(solving, size_matrix);

    for (int i = size_matrix - 2; i >=0; i--)
    {
        _type_ temp = 0;

        for (int j = i; j <size_matrix; j++)
        {
            temp = temp + matrix_lower[i][j+1] * solving[j+1];
        }
```

```
            temp = -temp + temp_matrix[i];

            solving[i] = temp / matrix_lower[i][i];

            cout << endl << "\t -step with index i=" << i << ":" << endl;

            output_vector(solving, size_matrix);
      }
      return solving;
}

_type_* inconspicuous(_type_** matrix_left, _type_* matrix_right, _type_*
solving, _type_* temp_matrix, int size_matrix)
{
      matrix_left[0][0] = 2.12;     matrix_left[0][1] = 0.42;    matrix_left[0]
[2] = 1.34;    matrix_left[0][3] = 0.88;
      matrix_left[1][0] = 0.42;     matrix_left[1][1] = 3.95;    matrix_left[1]
[2] = 1.87;    matrix_left[1][3] = 0.43;
      matrix_left[2][0] = 1.34;     matrix_left[2][1] = 1.87;    matrix_left[2]
[2] = 2.98;    matrix_left[2][3] = 0.46;
      matrix_left[3][0] = 0.88;     matrix_left[3][1] = 0.43;    matrix_left[3]
[2] = 0.46;    matrix_left[3][3] = 4.44;

      for (int i = 0; i < size_matrix; i++)
      {
            temp_matrix[i] = 0;
      }

      for (int i = 0; i < size_matrix; i++)
      {
            _type_ temp = 0;

            for (int j = 0; j < size_matrix; j++)
            {
                  temp = temp + matrix_left[i][j] * solving[j];
            }

            temp_matrix[i] = abs(temp-matrix_right[i]);
      }

      cout << endl << "Inconspicuous vector:" << endl;

      output_vector(temp_matrix, size_matrix);

      return temp_matrix;
}

void output_double_matrix(_type_** matrix_left, int size_matrix)
{
      cout << endl;

      for (int i = 0; i < size_matrix; i++)
      {
            for (int j = 0; j < size_matrix; j++)
            {
                  if (matrix_left[i][j]==0)
                  {
                        cout << "\t\t" << matrix_left[i][j];
                  }
                  else
                  {
                        cout << "\t" << matrix_left[i][j];
                  }

            }
```

```
		cout << endl;
	}

	cout << endl;
}

void output_vector(_type_* matrix_right, int size_matrix)
{
	cout << endl;

	for (int i = 0; i < size_matrix; i++)
	{
		cout << "\t\t" << matrix_right[i];
		cout << endl << endl;
	}

	cout << endl;

}
```