МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет прикладной математики и информатики
Кафедра информационных систем управления

Отчет
По методам вычислений

Выполнил студент группы № 13
Семенович Дмитрий Анатольевич
Вариант 12

Минск 2021

Оглавление

12.1.

Написать программу, которая обращает симметричную матрицу методом квадратного корня. Кроме матрицы A-1 программа должна выводить матрицу R и главную диагональ матрицы D. Применить программу к следующим ниже входным данным и вывести результат.

1-я матрица

```
Matrix A =
9  9  -12  12  15

9  18  -27  0  30

-12  -27  25  -8  -57

12  0  -8  19  -9

15  30  -57  -9  66



Matrix D =
9  0  0  0  0

0  18  0  0  0

0  0  25  0  0

0  0  0  19  0

0  0  0  0  66


Matrix U =
3  3  -4  4  5
0  3  -5  -4  5
0  0  4  -3  -3
0  0  0  4,69041575982343  -3,83761289440099
0  0  0  0  2,77979724571285
```

```
Matrix A^1 =
2,95400891840439 | -2,26077542981159 | -0,990487836615418 | -0,800609031491385 | -0,341176470588235 |
-----------------------------------------------------------------------------------------------------
-2,23488044951908 | 1,90981520065539 | 0,82411055037937 | 0,593399683105566 | 0,219607843137255 |
-----------------------------------------------------------------------------------------------------
-1,86473515231905 | 1,60776561245522 | 0,814960892736635 | 0,51666678424788 | 0,301629675616949 |
-----------------------------------------------------------------------------------------------------
0,206310889462354 | -0,204064077213144 | -0,0618763846118593 | -0,0393549906170191 | 0,0148300481142325 |
-----------------------------------------------------------------------------------------------------
-3,99825206724492 | 3,43704975297473 | 1,72014719062411 | 1,07866649578197 | 0,617631538103368 |
-----------------------------------------------------------------------------------------------------
Conditionality number = 1920,75922691705
```

2-я матрица

```
Matrix A =
25  0  -5  5  25  -10  -10  -20  5

0  9  -6  3  -6  9  3  9  6

-5  -6  21  -15  19  -20  -8  18  -9

5  3  -15  12  -8  12  1  -20  7

25  -6  19  -8  34  -58  -38  -11  -12

-10  9  -20  12  -58  -15  43  10  -5

-10  3  -8  1  -38  43  -7  13  -10

-20  9  18  -20  -11  10  13  100  8

5  6  -9  7  -12  -5  -10  8  -3


Matrix D =
25  0  0  0  0  0  0  0  0

0  9  0  0  0  0  0  0  0

0  0  21  0  0  0  0  0  0

0  0  0  12  0  0  0  0  0

0  0  0  0  34  0  0  0  0

0  0  0  0  0  -15  0  0  0

0  0  0  0  0  0  -7  0  0

0  0  0  0  0  0  0  100  0

0  0  0  0  0  0  0  0  -3
```

```
Matrix U =
5  0  -1  1  5  -2  -2  -4  1
0  3  -2  1  -2  3  1  3  2
0  0  4  -3  5  -4  -2  5  -1
0  0  0  1  4  -1  -4  -4  1
0  0  0  0  6  -3  0  1  -2
0  0  0  0  0  7,34846922834953  3,2659863237109  1,63299316185545  -2,44948974278318
0  0  0  0  0  0  6,53197264742181  -1,42886901662352  0
0  0  0  0  0  0  0  5,31899113241098  3,94811712921228
0  0  0  0  0  0  0  0  5,96553676260396
```

```
Matrix A^1 =
-1,61070393766492 | 5,23759250909266 | 6,50210534915302 | 5,67110824852878 | -0,226620124263504 | -0,0852985934411912 | 0,0682363466270861 | 0,123842346485727 | -0,07540797605253 |
-0,914801009010425 | 2,98028766731807 | 3,24369425229175 | 2,55800009067899 | -0,0996747956515042 | -0,0225727400247475 | 0,0191635259006184 | -0,0116016593749508 | -0,0174938843826445 |
-3,24931513187541 | 9,96572624338513 | 12,3074800471953 | 10,704439622488 | -0,408968483404639 | -0,155404350341396 | 0,132430605744765 | 0,212539911976067 | -0,127112302047122 |
-4,16981097724318 | 12,6039193453767 | 15,9076589252619 | 14,2196576859704 | -0,551940794794183 | -0,243474194000409 | 0,181500937750116 | 0,353528096381485 | -0,197180378524527 |
0,932185259437031 | -3,09309368480687 | -4,04631985574956 | -3,84277980762612 | 0,174952002445552 | 0,097505612500655 | -0,0370410051645941 | -0,145520785514335 | 0,118337195828505 |
2,46203188328559 | -7,65136145486586 | -9,92297837345362 | -9,15163687691193 | 0,371251440851016 | 0,203560226956196 | -0,148051530463937 | -0,29585461545419 | 0,160486674391657 |
-0,495019435248836 | 1,48914682423082 | 1,93742810279312 | 1,77584188457909 | -0,0660097189537018 | -0,0357798457391104 | 0,0355984801877889 | 0,0555930522870352 | -0,0228128621089224 |
-0,071126774109701 | 0,25475399405489 | 0,480976919439287 | 0,576007818281832 | -0,0270184097100512 | -0,0231891922948057 | 0,011118610457407 | 0,050827933519575 | -0,0208574739281576 |
0,123568279670278 | -0,376334139697847 | -0,488756603768404 | -0,496910003862495 | 0,0208574739281576 | 0,0160293549633063 | -0,00456257242178447 | -0,0208574739281576 | 0,0280996523754345
Conditionality number = 10218,4496517489
```

3-я матрица

```
Matrix A =
81  -3240  41580  -249480  810810  -1513512  1621620  -926640  218790
-3240  172800  -2494800  15966720  -54054000  103783680  -113513400  65894400  -15752880
41580  -2494800  38419920  -256132800  891891000  -1748106360  1942340400  -1141620480  275675400
-249480  15966720  -256132800  1756339200  -6243237000  12430978560  -13984850880  8302694400  -2021619600
810810  -54054000  891891000  -6243237000  22545022500  -45450765360  51648597000  -30918888000  7581073500
-1513512  103783680  -1748106360  12430978560  -45450765360  92554285824  -106051785840  63930746880  -15768632880
1621620  -113513400  1942340400  -13984850880  51648597000  -106051785840  122367445200  -74205331200  18396738360
-926640  65894400  -1141620480  8302694400  -30918888000  63930746880  -74205331200  45229916160  -11263309200
218790  -15752880  275675400  -2021619600  7581073500  -15768632880  18396738360  -11263309200  2815827300

Matrix D =
81  0  0  0  0  0  0  0  0
0  172800  0  0  0  0  0  0  0
0  0  38419920  0  0  0  0  0  0
0  0  0  1756339200  0  0  0  0  0
0  0  0  0  22545022500  0  0  0  0
0  0  0  0  0  92554285824  0  0  0
0  0  0  0  0  0  122367445200  0  0
0  0  0  0  0  0  0  45229916160  0
0  0  0  0  0  0  0  0  2815827300

Matrix U =
9  -360  4620  -27720  90090  -168168  180180  -102960  24310
0  207,846096908265  -4001,03736548411  28807,4690314856  -104026,971502587  208053,943005174  -234060,68588082  138702,628670116  -33684,9241055995
0  0  1033,0634056049  -12396,7608672588  57556,3897408446  -134298,242728637  167872,803410797  -107438,59418291  27673,5772895374
0  0  0  2095,43503836316  -17025,4096867006  52967,9412475131  -79451,9118712697  57783,2086336507  -16371,9091128675
0  0  0  0  2145  -12012  24570  -21840  7140,00000000089
0  0  0  0  0  1207,25142368937  -4527,19283883512  5571,92964779707  -2255,30485743851
0  0  0  0  0  0  378,582883923719  -865,332306111357  490,354973473179
0  0  0  0  0  0  0  61,9677335393187  -65,8407168547464
0  0  0  0  0  0  0  0  4,12310585691765

Matrix A^-1 =
-6645443767520,74  3757892936709,83  4683501380376,2  3503494438138,61  2245596551641,85  1307326949952,63  675798419593,949  265430553443,502  -29717382,0573478
-71743,9877200999  40574,7351438235  50566,9524394584  37826,5469427554  24245,4738419592  14115,3716141066  7297,03922024642  2866,48512188718  0,434461758909914
-849,145765417179  480,179353140666  598,452221395194  447,672326902055  286,939680039108  167,048754214293  86,3528250167258  33,9165310925331  -0,00365061450494018
353984,3185465  -200185,422892602  -249488,367912952  -186629,637899156  -119622,521077401  -69641,8818759548  -36001,0605240233  -14141,2351453845  -0,477608048797449
1368898091678,55  -774090136021,35  -964756655291,734  -721686469471,884  -462571491358,383  -269296894178,592  -139208034758,219  -54676164725,4402  6121494,4597714
-47135,1904141233  26649,6195998639  33215,40171992  24846,8603595981  15925,6226781565  9271,19020368607  4792,22398098038  1881,78248086141  -0,936841751051892
-1085342,78646369  613744,113035098  764915,72513674  572195,409164123  366753,839734686  213514,39034138  110372,304000466  43350,4738734543  -4,85349087036092
544601,859211668  -307952,614147143  -383808,757464811  -287108,418824988  -184024,259804658  -107133,354162793  -55379,7481854956  -21750,1914746357  4,22541976045711
0  0,252270300065645  0,18311854112014  0,129248867174775  0,0978608056311777  0,0796178099285015  0,0688504582213492  0,0624999929584717  0,0588235228119432
```

12.2

Число обусловленности матрицы из второго задания в матричной максимум-норме равно:

```
Conditionality number = 10218,4496517489
```

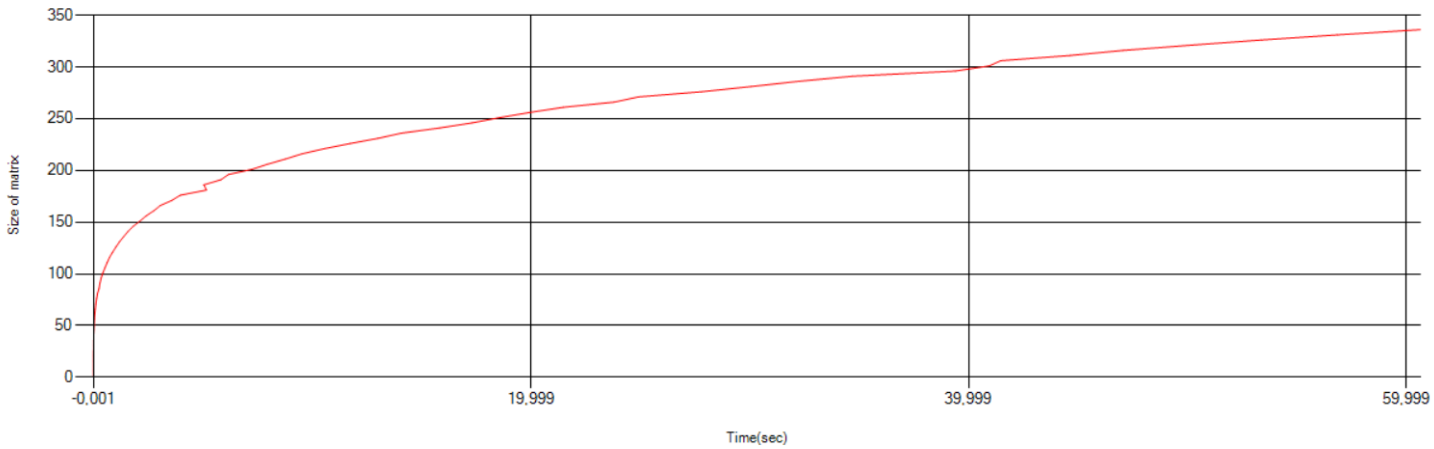
- Данную точность удалось достичь за счет использвания типа Double. **Double** — 64-битный тип, вмещающий 15 или 16 цифр.

12.3

В результате тестов мне удалось достигнуть обращения матрицы 276X276. Могу предположить, что можно было бы достигнуть лучшего результата реализуя данную задачу на более быстром языке программирования и используя более оптимальный алгоритм решения СЛАУ.

```
**********
Time = 60,646
size = 341
**********
```

График размера матрицы от времени:



Реализация:

```
using System;
using System.Diagnostics;
using System.Windows.Forms;

namespace lab1
{
    class Program
    {
        public static double[,] fillMtxU(double[,] mtx, int size)
        {
            double[,] U = new double[size, size];

            U[0, 0] = Math.Sqrt(mtx[0, 0]);

            for (int i = 1; i < size; ++i)
                U[0, i] = mtx[0, i] / U[0, 0];


            for (int i = 1; i < size; ++i)
            {
                for (int j = i; j < size; ++j)
                {
                    if (i == j)
```

```
            {
                double sum = 0.0;

                for (int k = 0; k <= i - 1; ++k)
                    sum += Math.Pow(U[k, i], 2);

                U[i, j] = Math.Sqrt(Math.Abs(mtx[i, j] - sum));

            }
            else
            {
                double sum = 0.0;

                for (int k = 0; k <= i - 1; ++k)
                    sum += (U[k, i] * U[k, j]);

                U[i, j] = (mtx[i, j] - sum) / U[i, i];
            }
        }
    }
    /*
    Console.Write("\nMatrix U = \n");
    for (int i = 0; i < size; ++i)
    {
        for (int j = 0; j < size; ++j)
        {
            Console.Write(U[i, j] + "  ");
        }
        Console.Write("\n");
    }
    */
    return U;
}

static void swap<T>(ref T lhs, ref T rhs)
{
    T temp;
    temp = lhs;
    lhs = rhs;
    rhs = temp;
}
static T[,] Copy<T>(T[,] array)
{
    T[,] newArray = new T[array.GetLength(0), array.GetLength(1)];
    for (int i = 0; i < array.GetLength(0); i++)
        for (int j = 0; j < array.GetLength(1); j++)
            newArray[i, j] = array[i, j];
    return newArray;
}
public static double[] solveEquation(double[,] mtx, double[] b, int size)
{
    double[,] a = Copy(mtx);
    double[] x = new double[size];
    b.CopyTo(x, 0);

    int[] js = new int[size];

    bool ok = true;
    int n1 = size - 1;
```

```
double d = 0.0;

for (int k = 0; k <= size - 2; ++k)
{
    int _i = 0;
    d = 0.0;
    for (int i = k; i <= size - 1; ++i)
    {
        for (int j = k; j <= size - 1; ++j)
        {
            double t = Math.Abs(a[i, j]);
            if (t > d) {
                d = t;
                js[k] = j;
                _i = i;
            }
        }
    }

    if (Math.Abs(d) < 1e-151)
        return null;


    if (js[k] != k)
    {
        for (int i = 0; i <= size - 1; ++i)
        {
            swap<double>(ref a[i, k], ref a[i, js[k]]);
        }
    }

    if (_i != k)
    {
        for (int j = k; j <= size - 1; ++j)
        {
            swap(ref a[k, j], ref a[_i, j]);
        }

        swap(ref x[k], ref x[_i]);
    }

    d = a[k, k];

    for (int j = k + 1; j <= size - 1; ++j)
    {
        a[k, j] /= d;
    }

    x[k] /= d;

    for (int i = k + 1; i <= size - 1; ++i)
    {
        for (int j = k + 1; j <= size - 1; ++j)
        {
            a[i, j] -= a[i, k] * a[k, j];
        }

        x[i] -= a[i, k] * x[k];
```

```
            }
        }

        if (!ok)
            return null;

        d = a[size - 1, size - 1];
        if (Math.Abs(d) < 1e-151)
            return null;

        x[size - 1] /= d;

        for (int i = size - 2; i >= 0; --i)
        {
            double t = 0;

            for (int j = i + 1; j <= size - 1; ++j)
            {
                t += a[i, j] * x[j];
                x[i] -= t;
            }
        }

        js[size - 1] = size - 1;

        for (int k = size - 1; k >= 0; --k)
        {
            if (js[k] != k)
            {
                swap(ref x[k], ref x[js[k]]);
            }
        }

        return x;
    }

    public static double[,] inverse(double[,] mtx, int size)
    {
        double[,] inverse = new double[size, size];

        double[,] U = fillMtxU(mtx, size);
        double[,] UT = new double[size, size];

        for(int i = 0; i < size; ++i)
        {
            for(int j = 0; j < size; ++j)
            {
                if (i < j)
                    UT[i, j] = 0;
                else
                    UT[i, j] = U[j,i];
            }
        }

        int[,] E = new int[size, size];

        for (int i = 0; i < size; ++i)
            E[i, i] = 1;
```

```
    double[][] vectorsY = new double[size][];

    for(int i = 0; i < size; ++i)
    {
        double[] vectorE = new double[size];
        vectorE[i] = 1;

        vectorsY[i] = solveEquation(UT, vectorE, size);
    }

    double[][] vectorsX = new double[size][];
    for(int i = 0; i < size; ++i)
    {
        vectorsX[i] = solveEquation(U, vectorsY[i], size);
    }

    double[,] result = new double[size, size];
    for (int i = 0; i < size; ++i)
    {
        for (int j = 0; j < size; ++j)
        {
            result[i, j] = vectorsX[j][i];
        }
    }

    return result;
}


public static double maxNorm(double[,] mtx, int size)
{
    double max = 0;
    double sum = 0;

    for(int i =0; i < size; ++i)
    {
        for(int j = 0; j < size; ++j){
            sum += Math.Abs(mtx[i, j]);
        }

        if(sum > max){
            max = sum;
        }

        sum = 0;
    }

    return max;
}

public static void printA(double[,] mtx, int size)
{
    Console.Write("\n\nMatrix A = \n");
    for (int i = 0; i < size; ++i)
    {
        for (int j = 0; j < size; ++j)
        {
            Console.Write(mtx[i, j] + "   ");
        }
```

```
            Console.Write("\n\n");
        }
    }

    public static void printD(double[,] mtx, int size)
    {
        Console.Write("\n\nMatrix D = \n");
        for (int i = 0; i < size; ++i)
        {
            for (int j = 0; j < size; ++j)
            {
                if (i == j)
                    Console.Write(mtx[i, j] + "  ");
                else
                    Console.Write("0  ");
            }
            Console.Write("\n\n");
        }
    }

    static void Main(string[] args)
    {
        double[,] A3 = { { 81 ,   -3240 ,  41580 ,  -249480, 810810 , -1513512   , 1621620, -926640,
218790},
                         {-3240 , 172800  ,-2494800  ,  15966720   , -54054000   ,103783680  , -113513400
,65894400   , -15752880 },
                         { 41580 ,-2494800 ,   38419920   , -256132800 , 891891000  , -1748106360
,1942340400 , -1141620480, 275675400},
                         {-249480  ,  15966720   , -256132800 , 1756339200 , -6243237000, 12430978560, -
13984850880   , 8302694400 , -2021619600},
                         { 810810 ,   -54054000 ,  891891000 ,  -6243237000 ,22545022500 ,-45450765360   ,
51648597000 ,-30918888000 ,   7581073500},
                         {-1513512  , 103783680 ,  -1748106360 ,12430978560, -45450765360  ,  92554285824,
-106051785840   ,63930746880 ,-15768632880 },
                         { 1621620 ,  -113513400,  1942340400 , -13984850880  ,  51648597000 ,-
106051785840 ,  122367445200   , -74205331200  ,  18396738360},
                         { -926640  , 65894400   , -1141620480, 8302694400 , -30918888000  ,  63930746880
,-74205331200 ,   45229916160, -11263309200},
                         {218790 ,-15752880,   275675400 ,  -2021619600 ,7581073500 , -15768632880   ,
18396738360 ,-11263309200    ,2815827300 }
                       };

        double[,] A2 = { {25 ,0 ,-5, 5, 25 ,-10 ,-10 ,-20 ,5 },
                         {0 ,9 ,-6, 3 ,-6 ,9 ,3 ,9 ,6 },
                         {-5 ,-6 ,21 ,-15 ,19 ,-20 ,-8 ,18 ,-9 },
                         {5 ,3 ,-15 ,12 ,-8 ,12, 1 ,-20 ,7 },
                         { 25 ,-6, 19 ,-8 ,34 ,-58 ,-38 ,-11 ,-12},
                         {-10 ,9 ,-20 ,12 ,-58 ,-15 ,43 ,10 ,-5 },
                         {-10 ,3 ,-8 ,1 ,-38 ,43 ,-7, 13 ,-10 },
                         { -20 ,9 ,18 ,-20 ,-11 ,10 ,13 ,100 ,8},
                         {5 ,6 ,-9 ,7 ,-12 ,-5 ,-10 ,8 ,-3 }
                        };

        double[,] A = { {9, 9 ,-12 ,12, 15},
                        { 9 ,18 ,-27, 0, 30},
                         {-12 ,-27, 25 ,-8 ,-57 },
                        {12 ,0 ,-8 ,19 ,-9 },
                        {15, 30 ,-57, -9 ,66 }
                       };
```

```
            printA(A, 5);

            printD(A, 5);

            double norm = maxNorm(A, 5);

            double[,] inverse = Program.inverse(A, 5);

            double normInvA = maxNorm(inverse, 5);

            Console.WriteLine("\nMatrix A^1 = ");

            for (int i = 0; i < 5; ++i)
            {
                for(int j =0; j < 5; ++j)
                {
                    Console.Write(inverse[i, j] + " | ");
                }
                Console.Write("\n-------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------
----\n");
            }

            Console.WriteLine("Conditionality number = " + norm * normInvA);

            Console.ReadLine();
        /*
            Graph graph = new Graph();
            graph.ShowDialog();

         */
        }
    }
}
```