Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. І. Сікорського»

Кафедра автоматизації проектування енергетичних процесів і систем

Лабораторна робота №1
з дисципліни «Чисельні методи в моделюванні енергетичних процесів»
«Розв’язання систем лінійних алгебраїчних рівнянь (СЛАР) прямими методами. Звичайний метод Гауса та метод квадратних коренів.»
Варіант №1

Виконав:
студент 2-го курсу, ТЕФ
групи ТР-15
Руденко В.І.

КИЇВ-2022

Мета роботи

Розв’язати систему рівнянь з кількістю значущих цифр m = 6. Використати метод Гауса для парних варіантів шляхом зведення матриці до верхньої трикутної побудованої на побічній діагоналі, для непарних - шляхом зведення до діагональної матриці. Вивести всі проміжні результати (матриці А, що отримані в ході прямого ходу методу Гауса, матрицю зворотного ходу методу Гауса та розв’язок системи. Навести результат перевірки: вектор нев’язки r = b – Ax, де x - отриманий розв’язок.

Варіант завдання (Вихідна система рівнянь).

1-4	$\begin{pmatrix} 3{,}81 & 0{,}25 & 1{,}28 & 0{,}75+\alpha \\ 2{,}25 & 1{,}32 & 4{,}58+\alpha & 0{,}49 \\ 5{,}31 & 6{,}28+\alpha & 0{,}98 & 1{,}04 \\ 9{,}39+\alpha & 2{,}45 & 3{,}35 & 2{,}28 \end{pmatrix}$ $\alpha = 0{,}5k,\ k = №вар - 1,$	$\begin{pmatrix} 4{,}21 \\ 6{,}47+\beta \\ 2{,}38 \\ 10{,}48+\beta \end{pmatrix}$ $\beta = 0{,}5k,\quad k = №вар - 1$

Теоретична частина

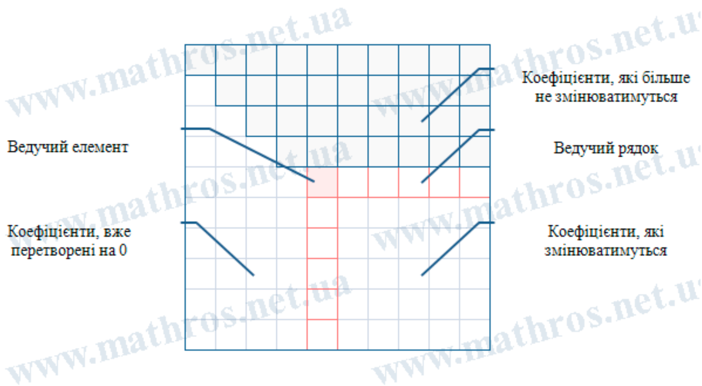
Ме́тод Га́уса — алгоритм розв'язування систем лінійних алгебраїчних рівнянь. Зазвичай під цим алгоритмом розуміють деяку послідовність операцій, що виконують над відповідною матрицею коефіцієнтів, для приведення її до трикутного вигляду, з наступним вираженням базисних змінних через небазисні. Цей метод також можливо використовувати для знаходження рангу матриці, для обчислення визначника матриці, а також для обчислення обернення невиродженої квадратної матриці.

Метод Гаусачудово підходить для вирішення систем лінійних рівнянь алгебри (СЛАУ). Він має низку переваг у порівнянні з іншими методами:

- по-перше, не потрібно попередньо досліджувати систему рівнянь на спільність;
- по-друге, методом Гауса можна вирішувати не тільки СЛАУ, в яких кількість рівнянь збігається з кількістю невідомих змінних та основна матриця системи невироджена, але й системи рівнянь, у яких число рівнянь не збігається з кількістю невідомих змінних або визначник основної матриці дорівнює нулю;
- по-третє, метод Гауса призводить до результату при порівняно невеликій кількості обчислювальних операцій

Алгоритм методу Гаусса:

Процес розв’язування систем лінійних рівнянь методом Гауса складається з двох етапів. На першому етапі (прямий хід) система рівнянь, за допомогою елементарних перетворень, зводиться до рівносильної їй системи трикутної форми.



На другому етапі (обернений хід), з отриманої *«ступінчастої»* системи, виконується послідовне знаходження значень невідомих величин.Розглянемо кожен з цих етапів більш детально. Отже, нехай дано систему лінійних алгебраїчних рівнянь виду:

$$\begin{cases} a_{11}\cdot x_1 + a_{12}\cdot x_2 + a_{13}\cdot x_3 + \cdots + a_{1n}\cdot x_n = b_1 \\ a_{21}\cdot x_1 + a_{22}\cdot x_2 + a_{23}\cdot x_3 + \cdots + a_{2n}\cdot x_n = b_2 \\ a_{31}\cdot x_1 + a_{32}\cdot x_2 + a_{33}\cdot x_3 + \cdots + a_{3n}\cdot x_n = b_3 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{n1}\cdot x_1 + a_{n2}\cdot x_2 + a_{n3}\cdot x_3 + \cdots + a_{nn}\cdot x_n = b_n \end{cases} \quad (1)$$

Припустимо, що коефіцієнт $a_{11} \neq 0$ (ведучий елемент). Цього, завжди, можна досягнути перестановкою рівнянь. Поділивши коефіцієнти першого рівняння системи *(1)* на a_{11} отримаємо:

$$x_1 + c_{12}\cdot x_2 + c_{13}\cdot x_3 + \cdots + c_{1n}x_n = d_1 \quad (2)$$

де $c_{1j} = a_{1j}/a_{11}\ (j > 1);\ d_1 = b_1/a_{11}$.

Користуючись рівнянням *(2)*, легко виключити з $2,3,4,\dots,n$ рівняння системи *(1)* невідому x_1. Для цього достатньо від другого рівняння системи *(1)* відняти рівняння *(2)*, помножене на a_{21}; від третього рівняння системи *(1)* , відняти рівняння *(2)*, помножене на a_{31}, і так далі.

Таким чином, отримаємо систему трикутної форми:

$$\begin{cases} x_1 + c_{12}\cdot x_2 + c_{13}\cdot x_3 + \cdots + c_{1n-1}\cdot x_{n-1} + c_{1n}\cdot x_n = d_1 \\ x_2 + c_{23}\cdot x_3 + \cdots + c_{2n-1}\cdot x_{n-1} + c_{2n}\cdot x_n = d_2 \\ x_3 + \cdots + c_{3n-1}\cdot x_{n-1} + c_{3n}\cdot x_n = d_3 \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ x_{n-1} + c_{n-1n}\cdot x_n = d_{n-1} \\ x_n = d_n \end{cases} \quad (3)$$

Після чого відбувається перехід до зворотного ходу розв’язання матриці. І її єдина особливість і відмінність полягає в тому, що операції прямого ходу відбуваються в порядку починаючи від кінця матриці, та зводячі (верхні) елементи діагоналі до нуля.

Проміжні результати виконання:

```
The initial view of the matrix:
        3.81000 0.25000 1.28000 0.75000  ||     4.21000
        2.25000 1.32000 4.58000 0.49000  ||     6.47000
        5.31000 6.28000 0.98000 1.04000  ||     2.38000
        9.39000 2.45000 3.35000 2.28000  ||     10.48000
-------------------------------------------------------
Sort the matrix:
Step: 1 For Row: 2 Swap colum 2 & 3
        3.81000 1.28000 0.25000 0.75000  ||     4.21000
        2.25000 4.58000 1.32000 0.49000  ||     6.47000
        5.31000 0.98000 6.28000 1.04000  ||     2.38000
        9.39000 3.35000 2.45000 2.28000  ||     10.48000
-------------------------------------------------------
Main operations on matrix:
Step: 2 Row: 1 / 3.81
        1.00000 0.33596 0.06562 0.19685  ||     1.10499
        2.25000 4.58000 1.32000 0.49000  ||     6.47000
        5.31000 0.98000 6.28000 1.04000  ||     2.38000
        9.39000 3.35000 2.45000 2.28000  ||     10.48000
-------------------------------------------------------
Step: 3 Row: 2 -= row: 1 * 2.25
        1.00000 0.33596 0.06562 0.19685  ||     1.10499
        0.00000 3.82409 1.17236 0.04709  ||     3.98378
        5.31000 0.98000 6.28000 1.04000  ||     2.38000
        9.39000 3.35000 2.45000 2.28000  ||     10.48000
-------------------------------------------------------
Step: 4 Row: 3 -= row: 1 * 5.31
        1.00000 0.33596 0.06562 0.19685  ||     1.10499
        0.00000 3.82409 1.17236 0.04709  ||     3.98378
        0.00000 -0.80394        5.93157 -0.00528        ||      -3.48748
        9.39000 3.35000 2.45000 2.28000  ||     10.48000
-------------------------------------------------------
Step: 5 Row: 4 -= row: 1 * 9.39
        1.00000 0.33596 0.06562 0.19685  ||     1.10499
        0.00000 3.82409 1.17236 0.04709  ||     3.98378
        0.00000 -0.80394        5.93157 -0.00528        ||      -3.48748
        0.00000 0.19535 1.83386 0.43157  ||     0.10417
-------------------------------------------------------
Step: 6 Row: 2 / 3.82409
        1.00000 0.33596 0.06562 0.19685  ||     1.10499
        0.00000 1.00000 0.30657 0.01231  ||     1.04176
        0.00000 -0.80394        5.93157 -0.00528        ||      -3.48748
        0.00000 0.19535 1.83386 0.43157  ||     0.10417
-------------------------------------------------------
```

```
Step: 7 Row: 3 -= row: 2 * -0.803937
        1.00000 0.33596 0.06562 0.19685  ||     1.10499
        0.00000 1.00000 0.30657 0.01231  ||     1.04176
        0.00000 0.00000 6.17804 0.00462  ||     -2.64997
        0.00000 0.19535 1.83386 0.43157  ||     0.10417
-------------------------------------------------------
Step: 8 Row: 4 -= row: 2 * 0.195354
        1.00000 0.33596 0.06562 0.19685  ||     1.10499
        0.00000 1.00000 0.30657 0.01231  ||     1.04176
        0.00000 0.00000 6.17804 0.00462  ||     -2.64997
        0.00000 0.00000 1.77397 0.42917  ||     -0.09934
-------------------------------------------------------
Step: 9 Row: 3 / 6.17804
        1.00000 0.33596 0.06562 0.19685  ||     1.10499
        0.00000 1.00000 0.30657 0.01231  ||     1.04176
        0.00000 0.00000 1.00000 0.00075  ||     -0.42893
        0.00000 0.00000 1.77397 0.42917  ||     -0.09934
-------------------------------------------------------
Step: 10        Row: 4 -= row: 3 * 1.77397
        1.00000 0.33596 0.06562 0.19685  ||     1.10499
        0.00000 1.00000 0.30657 0.01231  ||     1.04176
        0.00000 0.00000 1.00000 0.00075  ||     -0.42893
        0.00000 0.00000 0.00000 0.42784  ||     0.66158
-------------------------------------------------------
Step: 11        Row: 4 / 0.427842
        1.00000 0.33596 0.06562 0.19685  ||     1.10499
        0.00000 1.00000 0.30657 0.01231  ||     1.04176
        0.00000 0.00000 1.00000 0.00075  ||     -0.42893
        0.00000 0.00000 0.00000 1.00000  ||     1.54631
-------------------------------------------------------
Step: 12        Row: 3 -= row: 4 * 0.00074836
        1.00000 0.33596 0.06562 0.19685  ||     1.10499
        0.00000 1.00000 0.30657 0.01231  ||     1.04176
        0.00000 0.00000 1.00000 0.00000  ||     -0.43009
        0.00000 0.00000 0.00000 1.00000  ||     1.54631
-------------------------------------------------------
Step: 13        Row: 2 -= row: 4 * 0.0123131
        1.00000 0.33596 0.06562 0.19685  ||     1.10499
        0.00000 1.00000 0.30657 0.00000  ||     1.02272
        0.00000 0.00000 1.00000 0.00000  ||     -0.43009
        0.00000 0.00000 0.00000 1.00000  ||     1.54631
-------------------------------------------------------
```

```
-------------------------------------------------------
Step: 14        Row: 1 -= row: 4 * 0.19685
        1.00000 0.33596 0.06562 0.00000  ||     0.80059
        0.00000 1.00000 0.30657 0.00000  ||     1.02272
        0.00000 0.00000 1.00000 0.00000  ||     -0.43009
        0.00000 0.00000 0.00000 1.00000  ||     1.54631
-------------------------------------------------------
Step: 15        Row: 2 -= row: 3 * 0.306572
        1.00000 0.33596 0.06562 0.00000  ||     0.80059
        0.00000 1.00000 0.00000 0.00000  ||     1.15457
        0.00000 0.00000 1.00000 0.00000  ||     -0.43009
        0.00000 0.00000 0.00000 1.00000  ||     1.54631
-------------------------------------------------------
Step: 16        Row: 1 -= row: 3 * 0.0656168
        1.00000 0.33596 0.00000 0.00000  ||     0.82882
        0.00000 1.00000 0.00000 0.00000  ||     1.15457
        0.00000 0.00000 1.00000 0.00000  ||     -0.43009
        0.00000 0.00000 0.00000 1.00000  ||     1.54631
-------------------------------------------------------
Step: 17        Row: 1 -= row: 2 * 0.335958
        1.00000 0.00000 0.00000 0.00000  ||     0.44093
        0.00000 1.00000 0.00000 0.00000  ||     1.15457
        0.00000 0.00000 1.00000 0.00000  ||     -0.43009
        0.00000 0.00000 0.00000 1.00000  ||     1.54631
-------------------------------------------------------
Return X to original form:
Step: 18        For Row: 2 Swap colum 2 & 3
        1.00000 0.00000 0.00000 0.00000  ||     0.44093
        0.00000 1.00000 0.00000 0.00000  ||     -0.43009
        0.00000 0.00000 1.00000 0.00000  ||     1.15457
        0.00000 0.00000 0.00000 1.00000  ||     1.54631
-------------------------------------------------------
The final view of the matrix:
        1.00000 0.00000 0.00000 0.00000  ||     0.44093
        0.00000 1.00000 0.00000 0.00000  ||     -0.43009
        0.00000 0.00000 1.00000 0.00000  ||     1.15457
        0.00000 0.00000 0.00000 1.00000  ||     1.54631
```

Розв’язок задачі у середовищі MathCAD 14

ORIGIN := 1

$$Ar := augment(A, B) \qquad Ar = \begin{pmatrix} 3.81 & 0.25 & 1.28 & 0.75 & 4.21 \\ 2.25 & 1.32 & 4.58 & 0.49 & 6.47 \\ 5.31 & 6.28 & 0.98 & 1.04 & 2.38 \\ 9.39 & 2.45 & 3.35 & 2.28 & 10.48 \end{pmatrix}$$

$$Ag := rref(Ar) \qquad Ag = \begin{pmatrix} 1 & 0 & 0 & 0 & 0.441 \\ 0 & 1 & 0 & 0 & -0.43 \\ 0 & 0 & 1 & 0 & 1.155 \\ 0 & 0 & 0 & 1 & 1.546 \end{pmatrix}$$

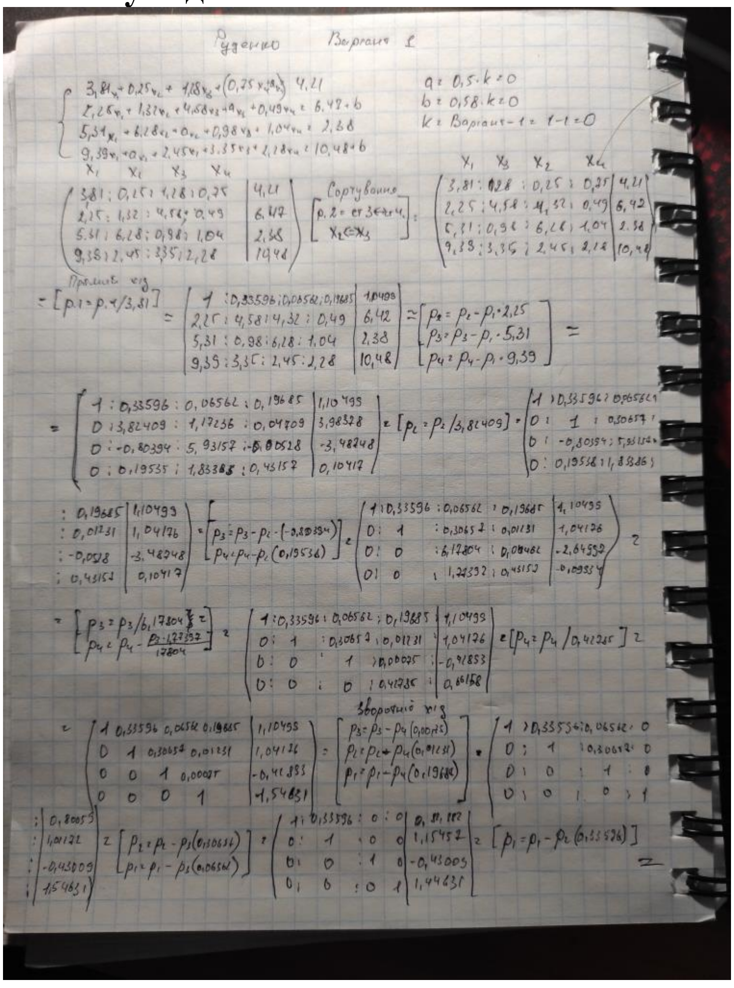
$$a := submatrix(Ag, 1, 4, 5, 5) \qquad a = \begin{pmatrix} 0.441 \\ -0.43 \\ 1.155 \\ 1.546 \end{pmatrix} \qquad P := \begin{pmatrix} 0.44093 \\ -0.43009 \\ 1.15457 \\ 1.54631 \end{pmatrix}$$

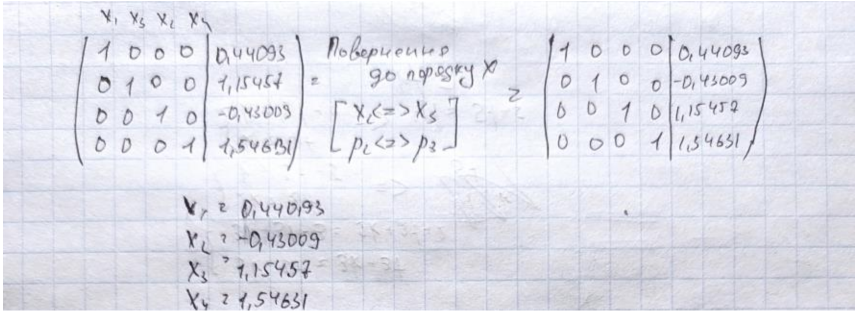
Вектор Нев'язки

Похибка

$$A \cdot P - B = \begin{pmatrix} 2.9 \times 10^{-6} \\ -3.8 \times 10^{-6} \\ 1.41 \times 10^{-5} \\ 8.5 \times 10^{-6} \end{pmatrix} \qquad stderr(a, P) = 1.209 \times 10^{-6}$$

Власний варіант розв’язку задачі





$$\begin{array}{c} \begin{array}{cccc} x_1 & x_3 & x_2 & x_4 \end{array} \\ \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 0,44093 \\ 0 & 1 & 0 & 0 & 1,15457 \\ 0 & 0 & 1 & 0 & -0,43009 \\ 0 & 0 & 0 & 1 & 1,54631 \end{array}\right) \end{array} = \text{Повернемо до порядку } x \begin{bmatrix} x_2 <=> x_3 \\ p_2 <=> p_3 \end{bmatrix} = \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 0,44093 \\ 0 & 1 & 0 & 0 & -0,43009 \\ 0 & 0 & 1 & 0 & 1,15457 \\ 0 & 0 & 0 & 1 & 1,54631 \end{array}\right)$$

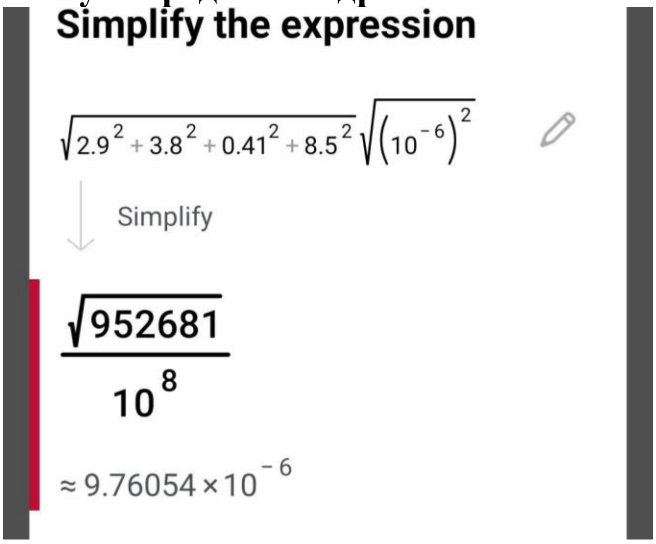
$x_1 = 0,44093$
$x_2 = -0,43009$
$x_3 = 1,15457$
$x_4 = 1,54631$

Порівняння результатів:

Метод	Власний варінат	MathCAD
X1	0,44093	0,441
X2	-0,43009	-0,43
X3	1,15457	1,155
X4	1,54631	1,546

Метод	Власний результат	MathCad
С.К Похибка (10^-6)	**9.760**	**1.209**

Пошук середньоквадратичної похибки



Лістинг програми:

```
#include <iostream>
#include <mutex>
using namespace std;

#define k 0
#define a 0.5*k
#define b 0.58*k
#define Array_rows 4
#define Array_col 5
```

```
static int step = 1;


void Sort(double (&Array)[Array_rows][Array_col], int (&Xcounter)[Array_rows]);
void MatrixTriangleLower(double (&Array)[Array_rows][Array_col]);
void MatrixTriangleUpper(double (&Array)[Array_rows][Array_col]);
void XReturn(double (&Array)[Array_rows][Array_col], int (&Xcounter)[Array_rows]);
void PrintMatrix(double (&Array)[Array_rows][Array_col]);
void ViewBetween(double (&Array)[Array_rows][Array_col], int row, int mainrow,  double multiplie);
void ViewBetween(double (&Array)[Array_rows][Array_col], int row, int col1, int col2);
void ViewBetween(double (&Array)[Array_rows][Array_col], int row, double multiplie);



void main() {
   int Xcounter[Array_rows];
   double Array[Array_rows][Array_col] =
           {
              {3.81,0.25,1.28, 0.75+a, 4.21}, //0,44
              {2.25,1.32,4.58+a,0.49, 6.47 + b}, //-0.43
              {5.31,6.28+a,0.98,1.04, 2.38},//1.15
              {9.39+a,2.45,3.35,2.28, 10.48+b} //1.54
           };

   ////////////////////
   cout << "The initial view of the matrix:" << endl;
   PrintMatrix(Array);
   ////////////////////
   cout << "Sort the matrix: " << endl;
   Sort(Array, Xcounter);
   ////////////////////
   cout << "Main operations on matrix: " << endl;
   MatrixTriangleLower(Array);
   MatrixTriangleUpper(Array);
   ////////////////////
   cout << "Return X to original form:" <<endl;
   XReturn(Array,Xcounter);
   ////////////////////
   cout << "The final view of the matrix: " << endl;
   PrintMatrix(Array);
   ////////////////////


}




void Sort(double (&Array)[Array_rows][Array_col], int (&Xcounter)[Array_rows])
{
   for(int i=0;i<Array_rows;i++)
      Xcounter[i]=i;
   for(int i=0;i<Array_rows-1;i++)
   {
```

```
        int MaxPosition=i;
        int lockKey=0;

        double MaxPositionValue = Array[i][i];
        for(int j=i;j<Array_col-1;j++)
        {
            if(Array[i][j]>MaxPositionValue)
            {
                MaxPositionValue=Array[i][j];
                MaxPosition = j;
                lockKey = 1;
            }
        }
        if(lockKey!=1) continue;
        for(int j=0;j<Array_rows;j++)
        {
            double Temp = Array[j][i];
            Array[j][i]=Array[j][MaxPosition];
            Array[j][MaxPosition]=Temp;
        }
        int XTemp;
        XTemp = Xcounter[i];
        Xcounter[i] = Xcounter[MaxPosition];
        Xcounter[MaxPosition] = XTemp;

        ViewBetween(Array,i+1,i+1,MaxPosition+1);
    }
}

void MatrixTriangleLower(double (&Array)[Array_rows][Array_col])
{
    double Temp[Array_col];
    for (int i = 0; i < Array_col - 1; i++)
    {
        double divisor = Array[i][i];
        for(int j=i;j<Array_col;j++)
        {
            Array[i][j] /= divisor;
        }
        ViewBetween(Array,i+1,divisor);
        for (int j = i + 1; j < Array_rows; j++)
        {
            double multiplier = Array[j][i] / Array[i][i];
            for (int n = 0; n < Array_col; n++)
            {
                Temp[n] = Array[i][n] * multiplier;
                Array[j][n] -= Temp[n];
            }
            ViewBetween(Array,j+1,i+1,multiplier);
        }
    }
}

void MatrixTriangleUpper(double (&Array)[Array_rows][Array_col])
{
```

```
    double Temp[Array_col];
    for (int i = Array_rows - 1; i >= 0; i--)
    {
        for (int j = i - 1; j >= 0; j--) {
            double multiplier = Array[j][i] / Array[i][i];
            for (int n = 0; n < Array_col; n++)
            {
                Temp[n] = Array[i][n] * multiplier;
                Array[j][n] -= Temp[n];
            }
            ViewBetween(Array,j+1,i+1,multiplier);
        }
    }
}

void XReturn(double (&Array)[Array_rows][Array_col], int (&Xcounter)[Array_rows])
{
    for (int i = 0; i < Array_rows-1; i++)
    {
        for (int j = i+1; j < Array_rows; j++)
            {
            if (Xcounter[i] > Xcounter[j])
                {
                int temp = Xcounter[i];
                Xcounter[i] = Xcounter[j];
                Xcounter[j] = temp;

                double Bigtemp  = Array[i][Array_col-1];
                Array[i][Array_col-1] = Array[j][Array_col-1];
                Array[j][Array_col-1]= Bigtemp;

                ViewBetween(Array,i+1,i+1,j+1);
                }
            }
    }
}

void ViewBetween(double (&Array)[Array_rows][Array_col], int row, int mainrow,  double multiplie)
{
    cout << "Step: " << step++ << "\tRow: " << row << " -= row: " << mainrow <<" * " << multiplie << endl;
    PrintMatrix(Array);
}

void ViewBetween(double (&Array)[Array_rows][Array_col], int row, int col1, int col2)
{

    cout << "Step: " << step++ << "\tFor Row: " << row << " Swap colum " << col1 <<" & " <<col2<<endl;
    PrintMatrix(Array);
}

void ViewBetween(double (&Array)[Array_rows][Array_col], int row, double multiplie)
{
    cout << "Step: " << step++ << "\tRow: " << row << " / " << multiplie << endl;
    PrintMatrix(Array);
}
```

```
void PrintMatrix(double (&Array)[Array_rows][Array_col])
{
    for(int i=0;i<Array_rows;i++)
    {
        for (int j= 0;j<Array_col-1;j++)
           printf("\t%3.5f", Array[i][j]);
        printf("\t || \t%.5f", Array[i][Array_col-1]);
        cout << endl;
    }
    cout << "-------------------------------------------------------" <<endl;
}
```