МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний Авіаційний Університет

Факультет кібербезпеки, комп’ютерної та програмної інженерії

Кафедра прикладної математики

Лабораторна робота № 1

На тему: «Застосування афінного перетворення до заданого зображення на площині»

Дисципліна: «Обчислювальна геометрія та комп´ютерна графіка»

Студента групи 351:

О.Т.Архіпова

Керівник:

асистент кафедри

прикладної математики

А. В. Темніков

Оцінка:

Київ 2021

Зміст

Вступ

Формування зображення та різноманітні дії з ним вимагають від користувача відомої математичної грамотності. Геометричні поняття, формули та факти, що відносяться до плоского та тривимірного випадків, грають у завданнях комп'ютерної графіки особливу роль. Принципи аналітичної геометрії у поєднанні з можливостями обчислювальної техніки, що постійно розширюються, є невичерпним джерелом істотних поступів на шляху розвитку комп'ютерної графіки.

На сьогодні комп’ютерна графіка – це наукова область, що має безліч застосувань. Вона широко застосовується в різних сферах діяльності людини: будівництві та архітектурі, промисловості, в комп’ютерних іграх та кіноіндустрії. Крім того графіка знайшла своє місце і в медицині, астрономії, картографії, фотограмметрії і в багатьох інших областях знання. Найбільш часто в комп’ютерній графіці для перетворення об’єктів використовуються, так зване афінне перетворення. Перетворення площини називається афінним, якщо воно взаємно однозначно і відображенням будь-якої прямої є пряма. Взаємно однозначне перетворення переводить кожну точку площини P в іншу точку площині P’, таким чином, що кожній точці P відповідає якась точка P’.

Постановка задачі

Актуальність: широко застосовується у моделюванні та комп’ютерній графіці.

Мета дослідження: вивчити та реалізувати програмно застосування афінного перетворення до заданого зображення на площині.

Об’єкт дослідження: потрібно до деякого заданого зображення на площині застосувати афінне перетворення та вивести нове зображення.

Предмет дослідження: матриці повороту, зсуву, відображення та масштабування.

Постановка задачі: «реалізувати застосування афінного перетворення до заданого зображення на площині із застосуванням матриці повороту, зсуву, відображення та масштабування щоб реалізувати застосування афінного перетворення до заданого зображення.

Розробка програмного забезпечення

Афінне перетворення — таке відображення площини у себе, за якого:

різні точки переходять у різні;

образом будь-якої прямої є пряма;

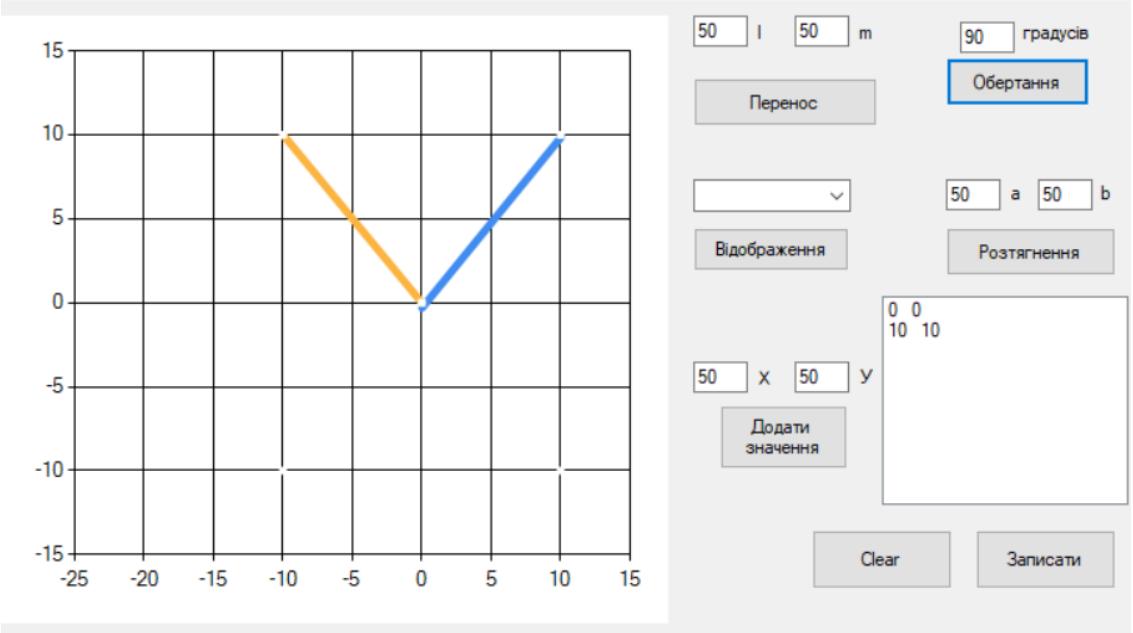
паралельні прямі переходять у паралельні прямі.

Перетворення зображення відбувається за допомогою матриць.

Для повороту зображення використовується така матриця:

$$R(\alpha) = \begin{bmatrix} Cos\alpha & Sin\alpha & 0 \\ -\ Sin\alpha & Cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

, де зображення повертається на кут α.

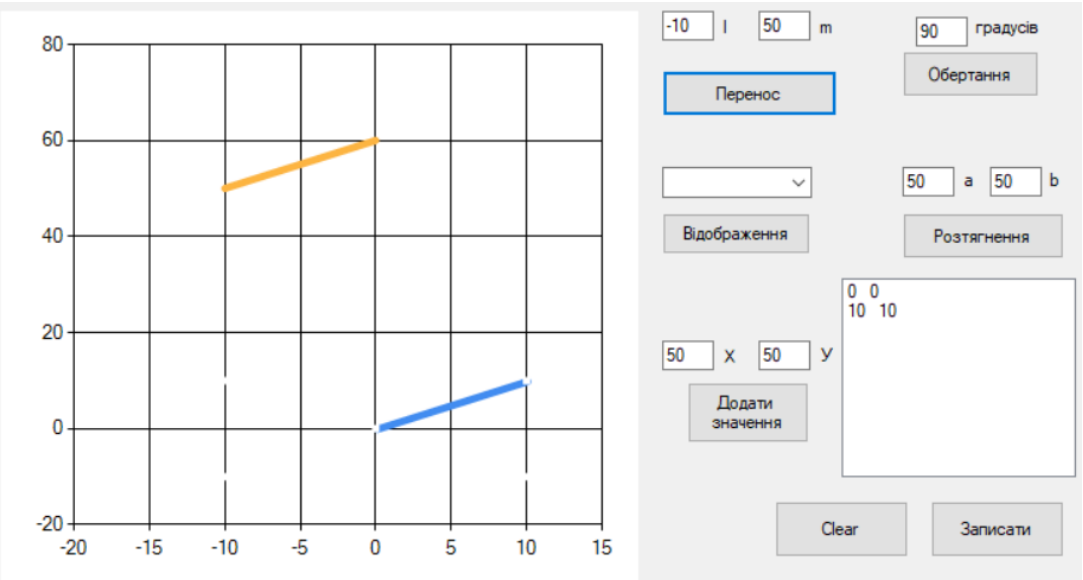


На цьому зображені ми бачимо початковий вектор(синього кольору), задано кут 90˚ та натиснули кнопку обертання і отримали вектор повернутий на 90˚.

Але ця матриця повертає тільки відносно початку координат. В такому випадку нам потрібно попередньо перемістити наше зображення в початок координат, помножити на матрицю повороту, а потім перемістити зображення в початкову точку. Для цього нам потрібна матриця зсуву:

$$[T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \lambda & \mu & 1 \end{bmatrix}$$

, де λ та μ – це координати точки.

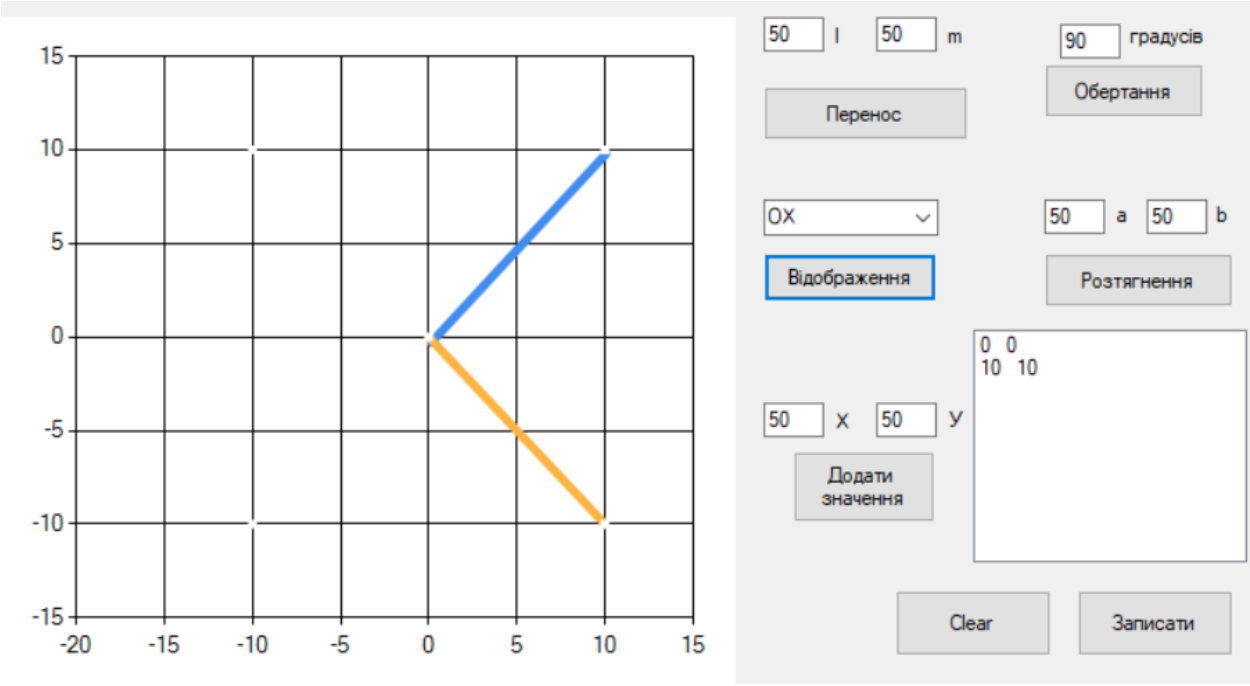


На цьому зображені видно початковий вектор(синього кольору), щоб зсунути його потрібно задати значення λ та μ та натиснути кнопку перенос. Отримали перенесений вектор(оранжевого кольору).

Також в афінних перетвореннях існує відображення відносно осей.

$$[M] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Якщо -1 стоїть на елементі (0;0), то відображення відбувається відносно осі ОУ, якщо на елементі (1;1), то відносно осі ОХ.

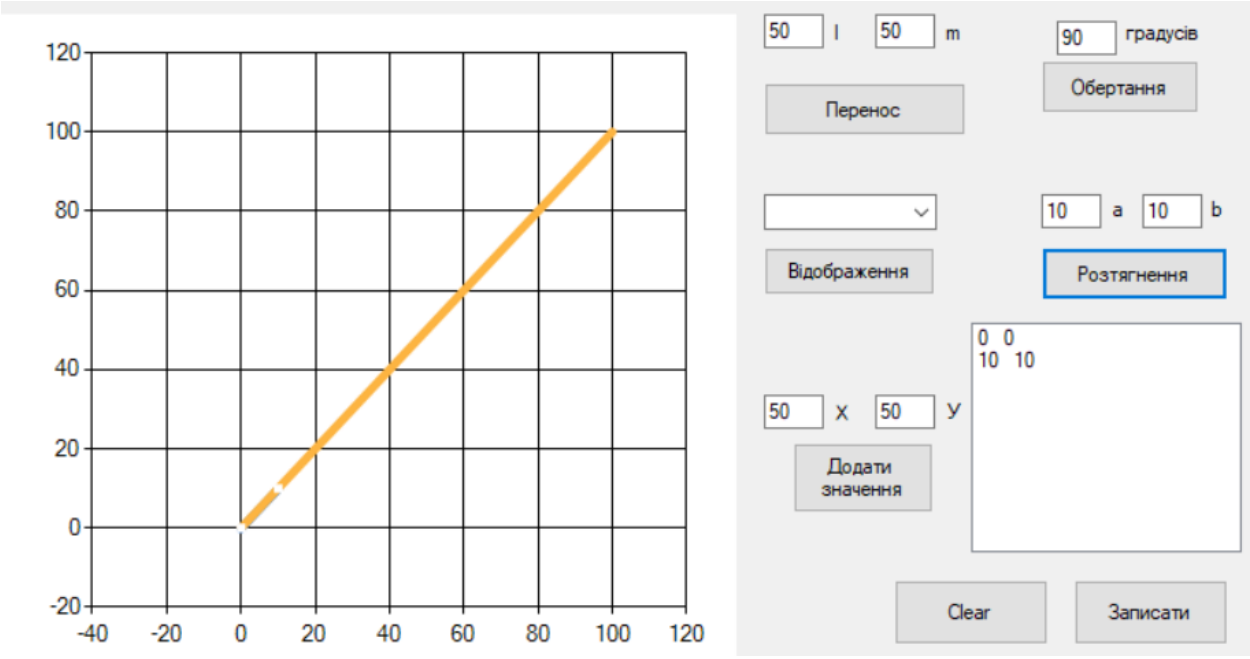


На даному зображені видно застосування відображення відносно осі ОХ. Для цього потрібно обрати в комбобоксі вісь відносно якої бажаєте зробити відображення, потім потрібно натиснути на кнопку відображення.

Останнє афінне перетворення в даній лабораторній роботі – це масштабування.

$$[D] = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \delta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ця матриця розтягує вздовж осей з коефіцієнтами α та δ.



Для масштабування потрібно ввести значення α та δ та натиснути розтягнення, після чого отримати результат – вектор оранжевого кольору.

Для застосування одного з цих перетворень до якогось зображення, потрібно однорідні координати помножити на певну матрицю й отримати нові координати перетвореного зображення.

Висновок

1. Для реалізації застосування афінного перетворення до заданого зображення на площині я використав поворот, зсув, відображення та масштабування.
2. Для реалізації повороту, зсуву, відображення та масштабування використав відповідні їм матриці для знаходження нових координатів.
3. Для реалізації афінних перетворень використав мову c# тому, що вона гарно підходить для візуалізації та зображень.

Додаток

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization;
using System.Windows.Forms.DataVisualization.Charting;

namespace lab1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
         Create_chart_dots();
      }

      double[] X;
```

```
double[] Y;
double x, y;
double[,] matrix;
int count = 0;

void Create_chart_dots()
{
   chart1.Series[1].Color = Color.White;
   chart1.Series[1].Points.AddXY(0, 0);
   chart1.Series[1].Points.AddXY(10, 10);
   chart1.Series[1].Points.AddXY(-10, -10);
   chart1.Series[1].Points.AddXY(10, -10);
   chart1.Series[1].Points.AddXY(-10, 10);
   chart1.Legends.Clear();
}

private void btnCalc_Click(object sender, EventArgs e)
{
   Matrix();
}
void Matrix()
{
   matrix = new double[3, 3];
   for(int i = 0; i < 3; i++)
   {
      for (int j = 0; j < 3; j++)
      {
         matrix[i, j] = 1;
      }
   }
   matrix[0, 0] = X[0];
   matrix[0, 1] = Y[0];
   matrix[1, 0] = X[1];
   matrix[1, 1] = Y[1];

}
private void chart1_MouseClick(object sender, MouseEventArgs e)
{
   if(count == 0)
   {
      X = new double[2];
      Y = new double[2];
   }
   chart1.Series[0].BorderWidth = 5;
   x = chart1.ChartAreas[0].AxisX.PixelPositionToValue(e.X);
   y = chart1.ChartAreas[0].AxisY.PixelPositionToValue(e.Y);
   chart1.Series[0].Points.AddXY(x, y);
   chart1.ChartAreas[0].AxisX.RoundAxisValues();
   listBox1.Items.Add(Math.Round(x) + "   " + Math.Round(y));
   //for (int i = count; i < 2; i++)
   //{
      X[count] = Math.Round(x);
      Y[count] = Math.Round(y);
```

```
   //}
   count++;
}

private void chart1_Click(object sender, EventArgs e)
{
}

private void btnAdd_Click(object sender, EventArgs e)
{
   double z = Convert.ToDouble(textBox1.Text);//x
   double k = Convert.ToDouble(textBox2.Text);//y
   chart1.Series[1].Color = Color.White;
   chart1.Series[1].Points.AddXY(z, k);
}

private void btnOb_Click(object sender, EventArgs e)
{
   Obertania();
}


void Obertania()
{
   chart1.Series[2].Points.Clear();
   double f = Convert.ToDouble(tbOb.Text);
   f = f * (Math.PI / 180);
   double[,] matrix_ob = new double[3, 3];
   double[,] matrix_00 = new double[3, 3];
   double[,] matrix_ab = new double[3, 3];
   double[,] matrix_prom = new double[3, 3];
   double S;
   for (int i = 0; i < 3; i++)
   {
      for (int j = 0; j < 3; j++)
      {
         if (i == j)
         {
            matrix_ob[i, j] = 1;
            matrix_00[i, j] = 1;
            matrix_ab[i, j] = 1;
         }
         else
         {
            matrix_ob[i, j] = 0;
            matrix_00[i, j] = 0;
            matrix_ab[i, j] = 0;
         }

      }
   }
   matrix_ob[0, 0] = Math.Round(Math.Cos(f), 2);
   matrix_ob[0, 1] = Math.Round(Math.Sin(f), 2);
```

```
matrix_ob[1, 0] = Math.Round(-Math.Sin(f), 2);
matrix_ob[1, 1] = Math.Round(Math.Cos(f), 2);


matrix_00[2, 0] = -matrix[0, 0];
matrix_00[2, 1] = -matrix[0, 1];
matrix_ab[2, 0] = matrix[0, 0];
matrix_ab[2, 1] = matrix[0, 1];

if (matrix[0,0] != 0 && matrix[0, 1] != 0)
{
   for (int i = 0; i < 3; i++)
   {
      for (int j = 0; j < 3; j++)
      {
         S = 0;
         for (int k = 0; k < 3; k++)
         {
            S = S + matrix_00[i, k] * matrix_ob[k, j];
         }
         matrix_prom[i, j] = S;
      }
   }

   for (int i = 0; i < 3; i++)
   {
      for (int j = 0; j < 3; j++)
      {
         S = 0;
         for (int k = 0; k < 3; k++)
         {
            S = S + matrix_prom[i, k] * matrix_ab[k, j];
         }
         matrix_ob[i, j] = S;
      }
   }
}


for (int i = 0; i < 3; i++)
{
   for (int j = 0; j < 3; j++)
   {
      S = 0;
      for (int k = 0; k < 3; k++)
      {
         S = S + matrix[i, k] * matrix_ob[k, j];
      }
      matrix_prom[i, j] = S;
   }
}

for (int i = 0; i < 3; i++)
```

```
    {
        for (int j = 0; j < 3; j++)
        {
            matrix[i, j] = matrix_prom[i, j];
        }
    }

    chart1.Series[2].BorderWidth = 5;
    chart1.Series[2].Points.AddXY(matrix[0, 0], matrix[0, 1]);
    chart1.Series[2].Points.AddXY(matrix[1, 0], matrix[1, 1]);
}

void Roztagnena()
{
    chart1.Series[2].Points.Clear();
    double a = Convert.ToDouble(tbA.Text);
    double b = Convert.ToDouble(tbB.Text);
    double[,] matrix_roz = new double[3, 3];
    double[,] matrix_prom = new double[3, 3];
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            if (i == j) matrix_roz[i, j] = 1;
            else matrix_roz[i, j] = 0;
        }
    }
    matrix_roz[0, 0] = a;
    matrix_roz[1, 1] = b;
    double S;

    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            S = 0;
            for (int k = 0; k < 3; k++)
            {
                S = S + matrix[i, k] * matrix_roz[k, j];
            }
            matrix_prom[i, j] = S;
        }
    }

    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            matrix[i, j] = matrix_prom[i, j];
        }
    }
    chart1.Series[2].BorderWidth = 5;
    chart1.Series[2].Points.AddXY(matrix[0, 0], matrix[0, 1]);
```

```
    chart1.Series[2].Points.AddXY(matrix[1, 0], matrix[1, 1]);
}

void Vidobr()
{
    chart1.Series[2].Points.Clear();
    double[,] matrix_prom = new double[3, 3];
    double[,] matrix_vid = new double[3, 3];
    switch (comboBox1.SelectedIndex)
    {
        case 0:
            matrix_vid[0, 0] = -1;
            matrix_vid[0, 1] = 0;
            matrix_vid[0, 1] = 0;
            matrix_vid[1, 0] = 0;
            matrix_vid[1, 1] = 1;
            matrix_vid[0, 1] = 0;
            matrix_vid[2, 0] = 0;
            matrix_vid[2, 1] = 0;
            matrix_vid[2, 2] = 1;
            break;
        case 1:
            matrix_vid[0, 0] = 1;
            matrix_vid[0, 1] = 0;
            matrix_vid[0, 2] = 0;
            matrix_vid[1, 0] = 0;
            matrix_vid[1, 1] = -1;
            matrix_vid[1, 2] = 0;
            matrix_vid[2, 0] = 0;
            matrix_vid[2, 1] = 0;
            matrix_vid[2, 2] = 1;
            break;
        case 2:
            matrix_vid[0, 0] = 0;
            matrix_vid[0, 1] = 0;
            matrix_vid[0, 2] = 1;
            matrix_vid[1, 0] = 0;
            matrix_vid[1, 1] = 1;
            matrix_vid[1, 2] = 0;
            matrix_vid[2, 0] = 1;
            matrix_vid[2, 1] = 0;
            matrix_vid[2, 2] = 0;
            break;
        case 3:
            matrix_vid[0, 0] = 0;
            matrix_vid[0, 1] = 0;
            matrix_vid[0, 2] = -1;
            matrix_vid[1, 0] = 0;
            matrix_vid[1, 1] = -1;
            matrix_vid[1, 2] = 0;
            matrix_vid[2, 0] = -1;
            matrix_vid[2, 1] = 0;
            matrix_vid[2, 2] = 0;
```

```
            break;
        }
        double S;

        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                S = 0;
                for (int k = 0; k < 3; k++)
                {
                    S = S + matrix[i, k] * matrix_vid[k, j];
                }
                matrix_prom[i, j] = S;
            }
        }
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                matrix[i, j] = matrix_prom[i, j];
            }
        }
        chart1.Series[2].BorderWidth = 5;
        chart1.Series[2].Points.AddXY(matrix[0, 0], matrix[0, 1]);
        chart1.Series[2].Points.AddXY(matrix[1, 0], matrix[1, 1]);
    }
    private void btnVid_Click(object sender, EventArgs e)
    {
        Vidobr();
    }
    private void btnRoz_Click(object sender, EventArgs e)
    {
        Roztagnena();
    }
    private void btnPer_Click(object sender, EventArgs e)
    {
        Perenos();
    }
    private void btnClear_Click(object sender, EventArgs e)
    {
        chart1.Series[0].Points.Clear();
        chart1.Series[2].Points.Clear();
        listBox1.Items.Clear();
        X = new double[2];
        Y = new double[2];
        count = 0;
    }
    private void label2_Click(object sender, EventArgs e)
    {

    }
```

```
        private void label3_Click(object sender, EventArgs e)
        {
        }
        private void textBox2_TextChanged(object sender, EventArgs e)
        {
        }
        private void textBox1_TextChanged(object sender, EventArgs e)
        {
        }
        void Perenos()
        {
            chart1.Series[2].Points.Clear();
            double l = Convert.ToDouble(tbL.Text);
            double m = Convert.ToDouble(tbM.Text);
            double[,] matrix_per = new double[3, 3];
            double[,] matrix_prom = new double[3, 3];
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    if (i == j) matrix_per[i, j] = 1;
                    else matrix_per[i, j] = 0;
                }
            }
            matrix_per[2,0] = l;
            matrix_per[2,1] = m;
            double S;
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    S = 0;
                    for (int k = 0; k < 3; k++)
                    {
                        S = S + matrix[i, k] * matrix_per[k, j];
                    }
                    matrix_prom[i, j] = S;
                }
            }
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    matrix[i, j] = matrix_prom[i, j];
                }
            }
            chart1.Series[2].BorderWidth = 5;
            chart1.Series[2].Points.AddXY(matrix[0, 0], matrix[0, 1]);
            chart1.Series[2].Points.AddXY(matrix[1, 0], matrix[1, 1]);
        }
    }
}
```