ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №3

За курсом «Елементи хаотичної динаміки»

Студента групи ПА-19-2

Ільяшенко Єгора Віталійовича

Кафедра комп’ютерних технологій, ДНУ

Варіант 7

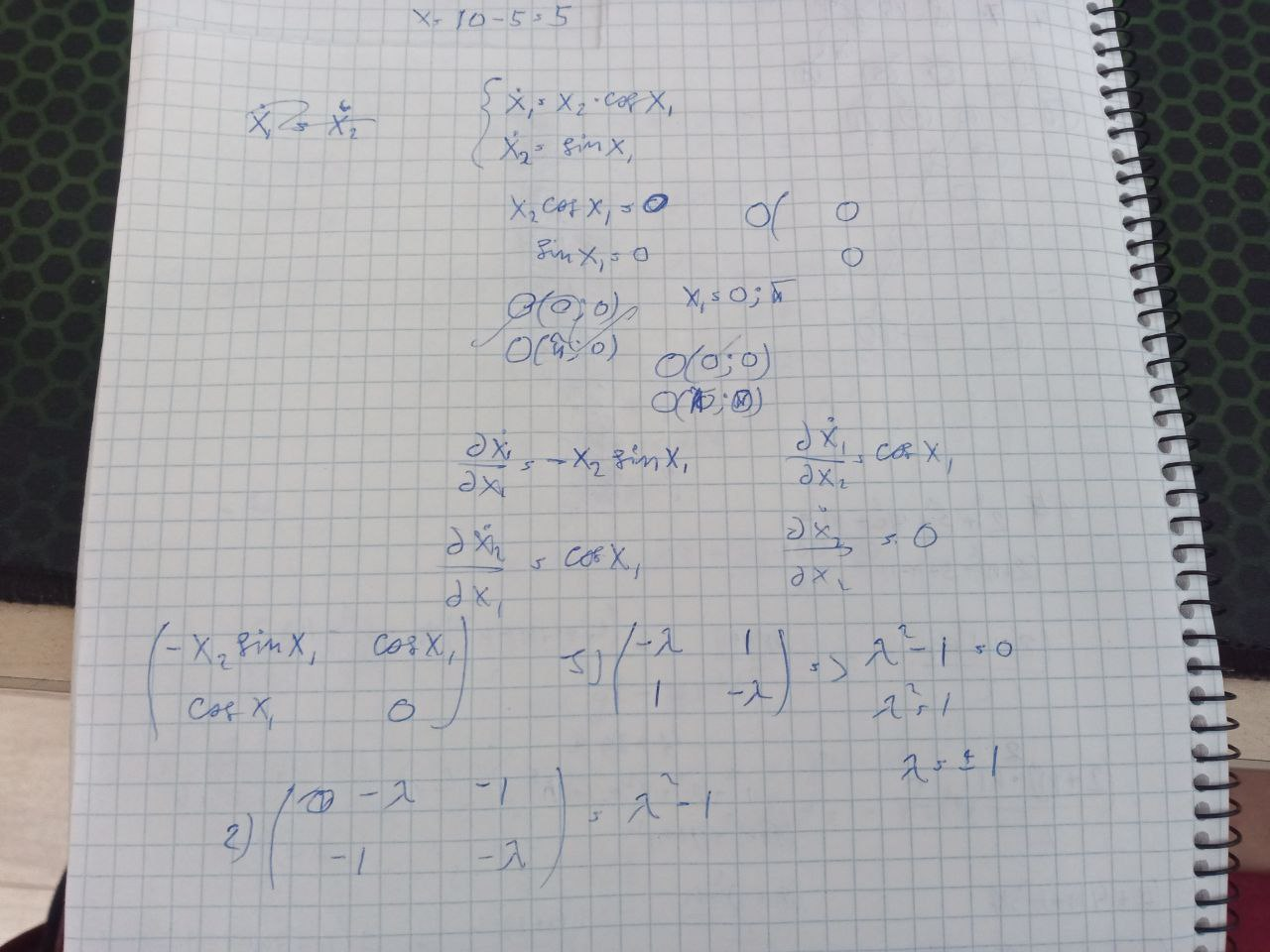
Text

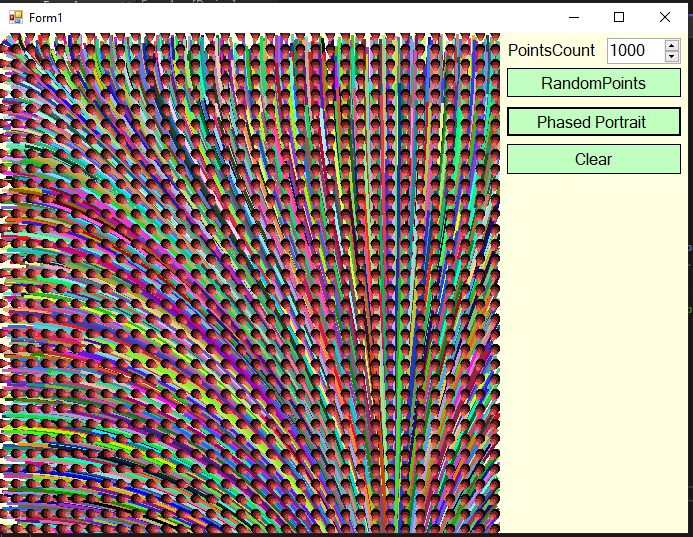
Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with low confidence

λ = +-1  
Точка – сідло





Background pattern

Description automatically generated

A picture containing chart

Description automatically generated

Chart, diagram

Description automatically generated

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Numerics;

namespace Lab\_1

{

public partial class Form1 : Form

{

const int canvas\_width = 500;

const int canvas\_height = 500;

int x\_shift = 0;

int y\_shift = 0;

double scale = 1;

Graphics gr;

Pen pen;

List<SolidBrush> arrowBrush = null;

Random rnd;

public Form1()

{

InitializeComponent();

gr = pictureBox1.CreateGraphics();

gr.Clear(Color.White);

rnd = new Random();

}

private void button1\_Click(object sender, EventArgs e) //Draw Dynamics

{

for (int i = 0; i < (int)numericUpDown1.Value; ++i)

DrawDynamics();

}

private void button2\_Click(object sender, EventArgs e) => DrawPhasedPortrait(); //Draw Phased Portrait

void DrawPhasedPortrait()

{

const int cellSize = 15;

for (int x = 0; x <=canvas\_width / cellSize; ++x)

for (int y = 0; y <= canvas\_height / cellSize; ++y)

{

Vector2 val = MyDerivative(1f \* x \* cellSize / 250, 1f \* y \* cellSize / 250);

DrawArrow(new Vector2(x \* cellSize, y \* cellSize), val, gr);

}

}

void DrawDynamics()

{

Color penColor = Color.FromArgb(rnd.Next(0, 255), rnd.Next(0, 255), rnd.Next(0, 255));

pen = new Pen(penColor, 3);

const int linePoints = 200;

PointF[] line = new PointF[linePoints];

line[0] = new PointF((float)rnd.NextDouble() \* 2, (float)rnd.NextDouble() \* 2);

for (int i = 1; i < linePoints; i++)

{

Vector2 val = Vector2.Normalize(MyDerivative(new PointF(line[i - 1].X, line[i - 1].Y))) / 50;

line[i] = new PointF(line[i - 1].X + val.X, line[i - 1].Y + val.Y);

}

for (int i = 0; i < linePoints; i++)

line[i] = new PointF(line[i].X \* 250, line[i].Y \* 250);

gr.DrawLines(pen, line);

}

Vector2 MyDerivative(float x, float y) => MyDerivative(new Vector2(x, y));

Vector2 MyDerivative(PointF coord) => MyDerivative(new Vector2(coord.X, coord.Y));

Vector2 MyDerivative(Vector2 coord) => new Vector2(coord.Y \* (float)Math.Cos(coord.X), (float)Math.Sin(coord.X));

//Vector2 MyDerivative(Vector2 coord) => new Vector2(coord.Y / 250 \* (float)Math.Cos(coord.X / 250), (float)Math.Sin(coord.X / 250));

//Vector2 MyDerivative(Vector2 coord) => new Vector2(coord.X \* coord.X + coord.Y \* coord.Y - coord.X \* coord.Y, (coord.Y + coord.X) \* (1 + coord.X \* coord.X + coord.Y + coord.Y));

//Vector2 MyDerivative(Vector2 coord) => new Vector2((coord.X - 250) \* (coord.X - 250), (coord.Y - 250) \* (coord.Y - 250));

void DrawArrow(Vector2 point, Vector2 direction, in Graphics target)

{

if (arrowBrush == null)

arrowBrush = new List<SolidBrush>()

{

new SolidBrush(Color.FromArgb(30, 20, 0, 0)),

new SolidBrush(Color.FromArgb(30, 100, 30, 30)),

new SolidBrush(Color.FromArgb(30, 180, 60, 60)),

new SolidBrush(Color.FromArgb(30, 255, 90, 90))

};

Vector2 directionNormalized = Vector2.Normalize(direction);

for (int i = 0; i < 4; i++)

target.FillEllipse(arrowBrush[i],

new RectangleF(directionNormalized.X \* i\*2 + point.X - (5 - i),

directionNormalized.Y \* i\*2 + point.Y - (5 - i),

10 - 2 \* i, 10 - 2 \* i));

}

void CanvasRefresh()

{

}

protected override bool ProcessCmdKey(ref Message msg, Keys keyData)

{

if (keyData == Keys.Q)

{

scale /= 1.5;

CanvasRefresh();

}

if (keyData == Keys.E)

{

scale \*= 1.5;

CanvasRefresh();

}

if (keyData == Keys.W)

{

y\_shift -= 50;

CanvasRefresh();

}

if (keyData == Keys.A)

{

x\_shift -= 50;

CanvasRefresh();

}

if (keyData == Keys.S)

{

y\_shift += 50;

CanvasRefresh();

}

if (keyData == Keys.D)

{

x\_shift += 50;

CanvasRefresh();

}

return base.ProcessCmdKey(ref msg, keyData);

}

private void button3\_Click(object sender, EventArgs e) => gr.Clear(Color.White);

}

}