МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»
(ННГУ)
Институт информационных технологий, математики и механики
Кафедра: Дифференциальных уравнений, математического и численного анализа

ОТЧЕТ

по учебной практике
на тему:

«Визуализация типичных бифуркаций в семействе одномерных отображений»

Выполнил:
студент группы 381603-1
Королев Александр Павлович

подпись

Научный руководитель:
доцент, канд. физ.-мат. наук
Малкин Михаил Иосифович
ученая степень, ученое звание,ф.и.о.

подпись

Нижний Новгород
2020

Оглавление

Введение

Хаос — это общий термин для обозначения поведения различных сложных решений относительно достаточно просто описываемых детерминированных систем. Пионером хаотической динамики считается Анри Пуанкаре, а глубокое изучение хаотических систем началось в 70-ых годах XX века после публикации работы Эдварда Лоренца. В ней он разрабатывал модель конвекции в атмосфере и обнаружил следующую замечательную особенность: решения при незначительном изменении начальных условий будут существенно отличаться друг от друга, при этом они не покидают определенную часть пространства (чувствительная зависимость от начальных условий). Позднее было установлено, что подобное поведение характерно для многих других динамических систем. Появилась потребность в изучении хаотической динамики. В данной работе рассмотрены основные свойства хаотических динамических систем па примере различных одномерных отображений.

Определение хаотической динамической системы (R. Devaney)

Пусть X - метрическое пространство, тогда отображение $f : X \to X$ называется хаотическим на X, если :

1) Отображение f транзитивно т.е.
   Для любых двух открытых множеств U_1 , $U_2 \in X$ существует $n > 0$ такое, что $f^n(U_1) \cap U_2 \neq 0$

2) Периодические точки отображения всюду плотны $f^n \in X \ \forall \ n > 0, \ n = 1, 2, 3..$

3) У отображения f наблюдается чувствительная зависимость к начальным условиям, т.е.:
   существует $\beta > 0$ такое, что $\forall \ x_0 \in X$: $\forall$ открытого множества $U \in X$в окресности x_0, найдется $y_0 \in U$, такое, что при $n > 0 \ \ |f^n(x_0) - f^n(y_0)| > \beta$

Стоит заметить, что приведенное определение хаотической динамической системы по Devaney - не единственно (например Йорк), но все определения эквивалентны.

Хаос в дискретных динамических системах

1) Пусть $x \in R$ и $f : R \to R$ функция, которая принадлежит C^2
   Дискретная динамическая система представляет собой отображение вида:
   $x_{n+1} = f(x_n)$

2) Орбитой называется последовательность
   $x_0 = x, x_1 = f(x), \ \ldots \ , x_n = f^n(x), \ \ldots$

3) Мы называем x_0 неподвижной точкой, если $f(x_0) = x_0$

4) Мы называем x_0 периодической точкой с периодом n, если $f^n(x_0) = x_0$

Рассматриваемые семейства одномерных отображений

4.1 Tent map

Тентетивное отображение $T_r : [0,1] \to [0,1]$ - кусочно непрерывная функция, определяемая следующим образом:

$$T_r(x) = \begin{cases} rx & \text{if } 0 \leq \text{x} \leq \frac{1}{2} \\ -rx + r & \text{if } \frac{1}{2} \leq \text{x} \leq 1 \end{cases}$$

Параметр r изменяется в диапазоне $[0, 2]$

Утверждение:
T_r хаотично на интервале $[0, 1]$
Доказательство:

1) На любом интервале $[\frac{k}{2^n}, \frac{k+1}{2^n}]$ при $k = 0, 1, 2, \ \dots \ , 2^n - 1 : T_r^n \to [0, 1]$
   Таким образом, T_r^n пересекает линию $y = x$ единожды на каждом интервале.
   В результате каждый интервал содержит фиксированную точку для T_r^n, что эквивалентно наличию периодической точки для T_r с периодом n .
   Таким образом периодические точки отображения T_r плотны на $[0, 1]$.

2) Пусть U_1, U_2 являются открытыми подинтервалами в $[0, 1]$.
   Пусть для $n >> k$, U_1 включает в себя интервал вида $[\frac{k}{2^n}, \frac{k+1}{2^n}]$.
   Таким образом, отображение T_r^n с начальной точкой $x_0 \in U_1$ содержит U_2

3) Пусть $x_0 \in [0, 1]$, зададим $\beta = 1/2$, которая будет являться нашей сенсетивной константой.
   Выше мы показали, что для любого открытого интервала $U(x_0)$ отображение $T_r^n \to [0, 1]$ при достаточно больших n.
   Таким образом, существует $y_0 \in U$ такое, что $|f^n(y_0) - f^n(x_0)| \geq 1/2 = \beta$

4.2 Logistic map

"Логистическое отображение" - это отображение вида $L_r(x_n) = x_{n+1} = rx_n(1 - x_n)$
Приведем ряд утверждений:

1) Решение уравнения $rx(1-x) = x$ показывает, что L_r имеет фиксированные точки в 0 и $p_r = \frac{r-1}{r}$. Так же точки 1 и $\frac{1}{r}$ являются фиксированными т.к. $L_r(1) = 0, L_r(\frac{1}{r}) = p_r$
2) Если $0 < r < 1$, тогда $p_r < 0$, p_r - отталкивающая фиксированння точка, а 0 - притягивающая
3) $r = 1$ биффуркационная точка.
4) При $1 < r < 3$, 0 - отталкивающая фиксированная точка, а p_r - притягивающая.
5) При $r = 3$ происходит биффуркация удвоения. 0 - все еще отталкивающая фиксированная точка, а p_r - притягивающая
6) При $3 < r < 3.4$ обе точки 0 и p_r являются отталкивающими
7) При $r \approx 3.45$ происходит очередная биффуркация удвоения.
8) При $3.4 < r < 4$ изменения происходят очень быстро.

4.3 Sin map

"Синусоидальное отображение" - это отображение вида $S_r(x_n) = x_{n+1} = r \sin \pi x_n$
$S_r : [0, r] \to [0, r]$.
Параметр r принимает значения $[0, \infty]$.
Для значения параметра $r = 4$ при решении уравнения $4 \sin \pi x = x$

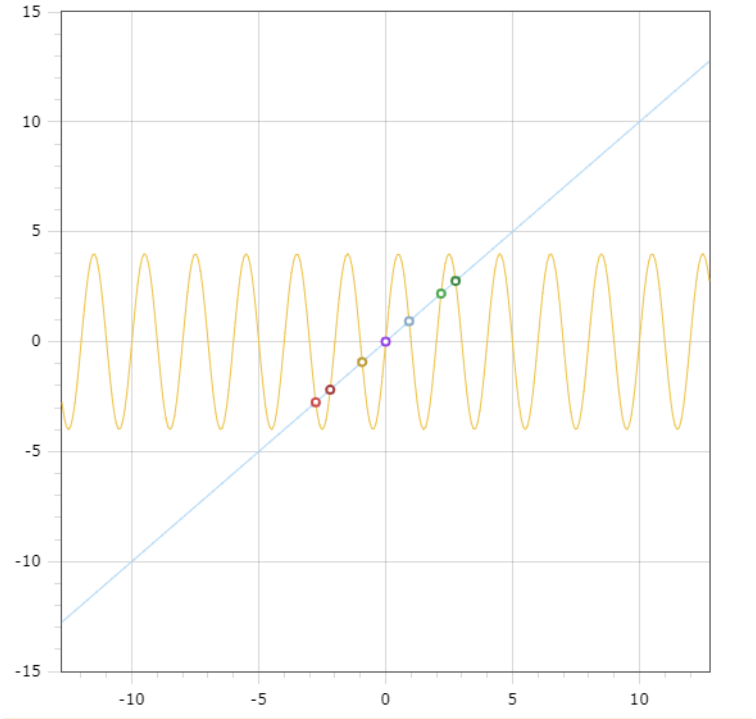

Рис. 4.1: Численное решение уравнения $4 \sin \pi x = x$

мы получим следующие фиксированные точки : $p_1 = 0, p_2 = 0.95266, p_3 = 2.1838, p_4 = 2.75784$

Цель работы

Реализовать программное обеспечение (ПО) для рассмотрения основных свойств хаотической динамики.
Данное ПО должно обладать следующим функцианалом:

1) Возможность построения орбит для одномерных отображений
2) Возможность построения бифуркационных диаграмм
3) Возможность отыскания точек удвоения периода

Структура программы

Приложение для решение данной задачи было реализовано на языке $C\#$ в среде Microsoft Visual Studio 2019 с использованием фреймворка "WindowsForm".

Calculate

Calculate - абстрактный класс, при помощи которого высчитываются точки для орбит и бифуркационных диаграмм.
Содержит в себе информацию о типе одномерного отображения, значениях параметров r, x_0 и функцию, для нахождения следующей точки

MainWindow

MainWindow - класс, который отвечает за взаимодействие между пользователем и программой(собирает исходные данные, отображает графики и таблицы)

OutputWindow

OutputWindow - класс, отвечающий за сообщения об ошибках, результатах выполнения программы

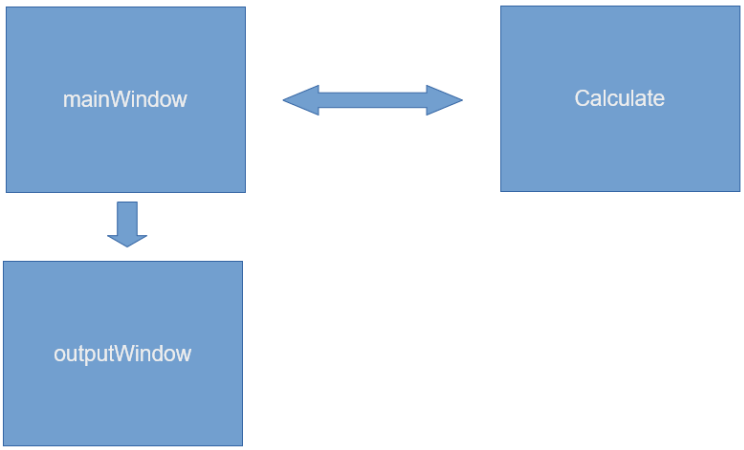

Рис. 6.1: Общая структура программы

Руководство пользователя

При запуске приложения появляется окно, в котором предлагается выбрать тип исследуемого одномерного отображения, начальную точку x_0 и значение параметра r

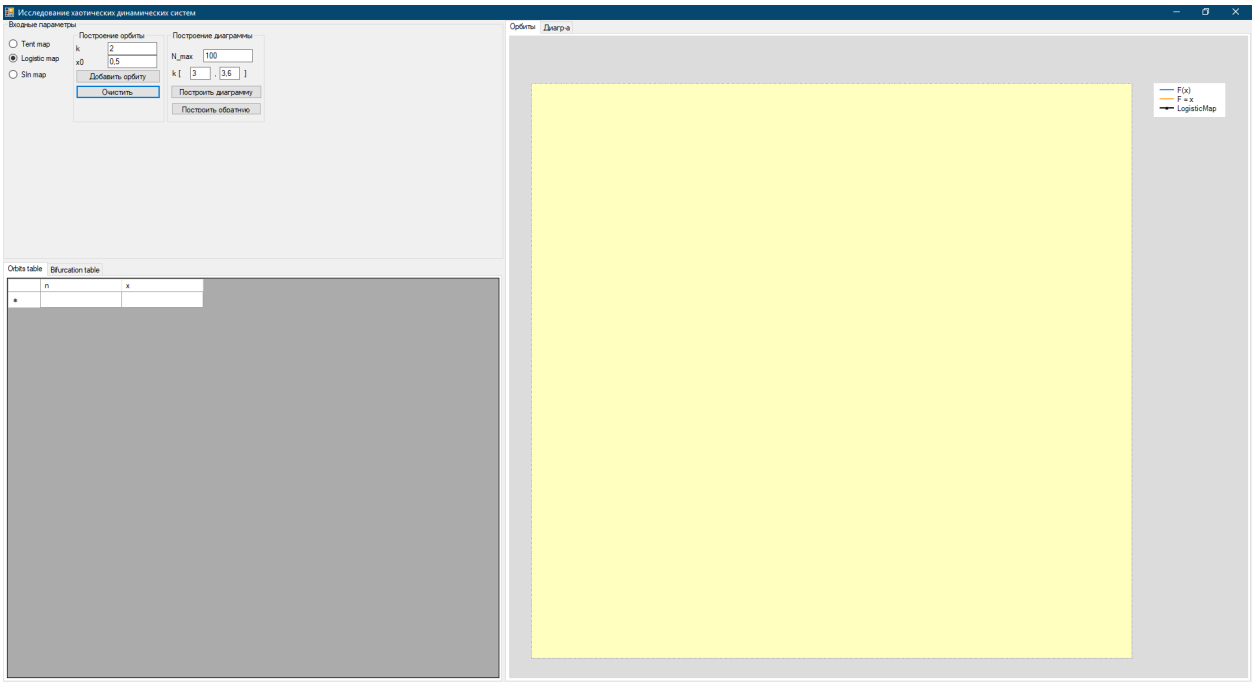

Рис. 7.1: Начальное окно программы

После, при нажатии кнопки "Добавить орбиту" , в правой части праграммы мы увидим орбиту одномерного отображения при заданных начальных условиях.
В левом нижнем углу будет показана траектория нашей орбиты.
После выведется сообщение о том, приводят ли данные н.у. в фиксированную точку

При нажатии кнопки "Построить диаграмму" - в правой части во вкладке "Диаграмма"будет построена бифуркационная диаграма для заданного диапазона r

7.1 Пример построения орбит

7.1.1 Tent Map

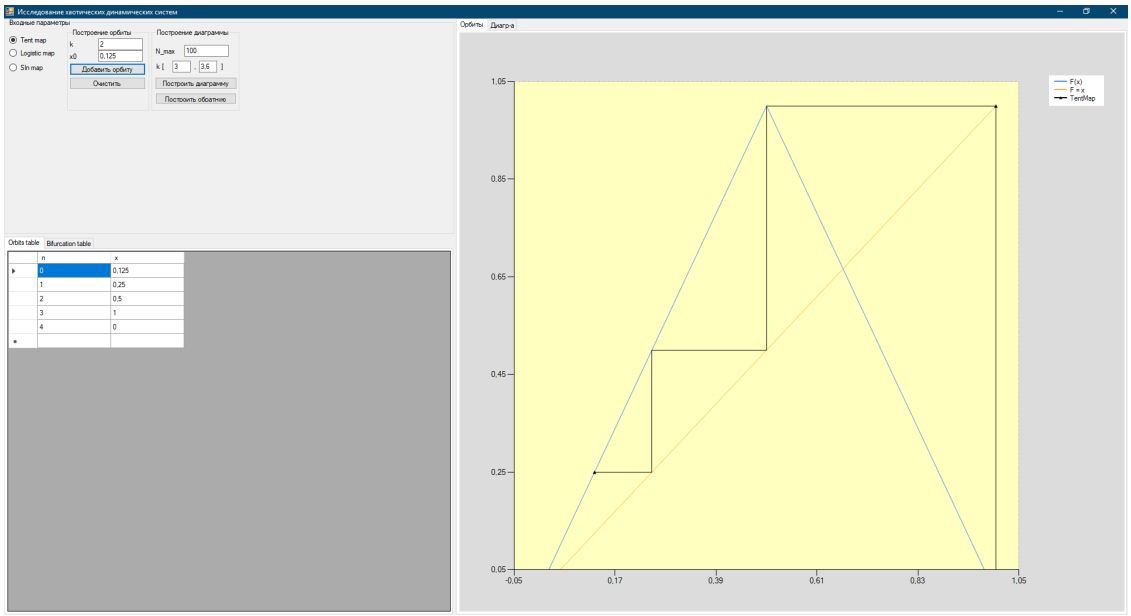

Рис. 7.2: Орбиты Tent map при $r = 2, x_0 = 0.125$

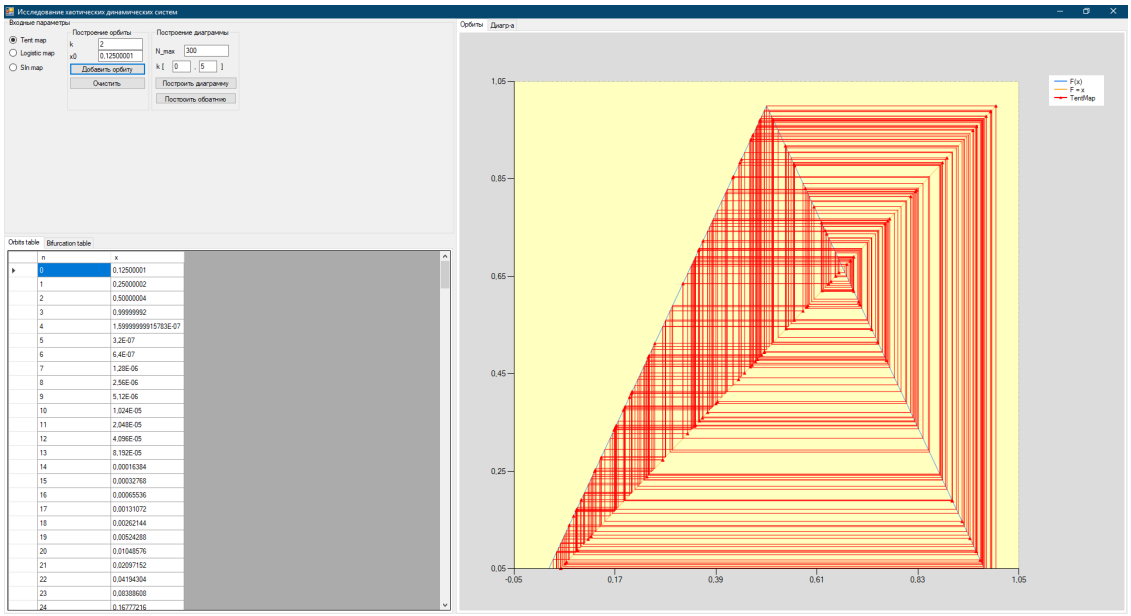

Рис. 7.3: Орбиты Tent map при $r = 2, x_0 = 0.12500001$

В данном примере на Рис.7.2 показана динамика орбиты при следующих начальных условиях : $r = 2, x_0 = 0.125$. Мы можем наблюдать, что за 5 итераций, начальная точка перемещается в фиксированную точку 0 за 4 итерации. На Рис.7.3 показано, как сильно изменяется поведение динамики орбит при незначительном изменении начальных условий. Изменение составляет $10^{-}8$, устойчивость системы полностью пропадает. Такое поведение объясняется жесткой чувствительностью к начальным условиям.

7.1.2 Logistic map

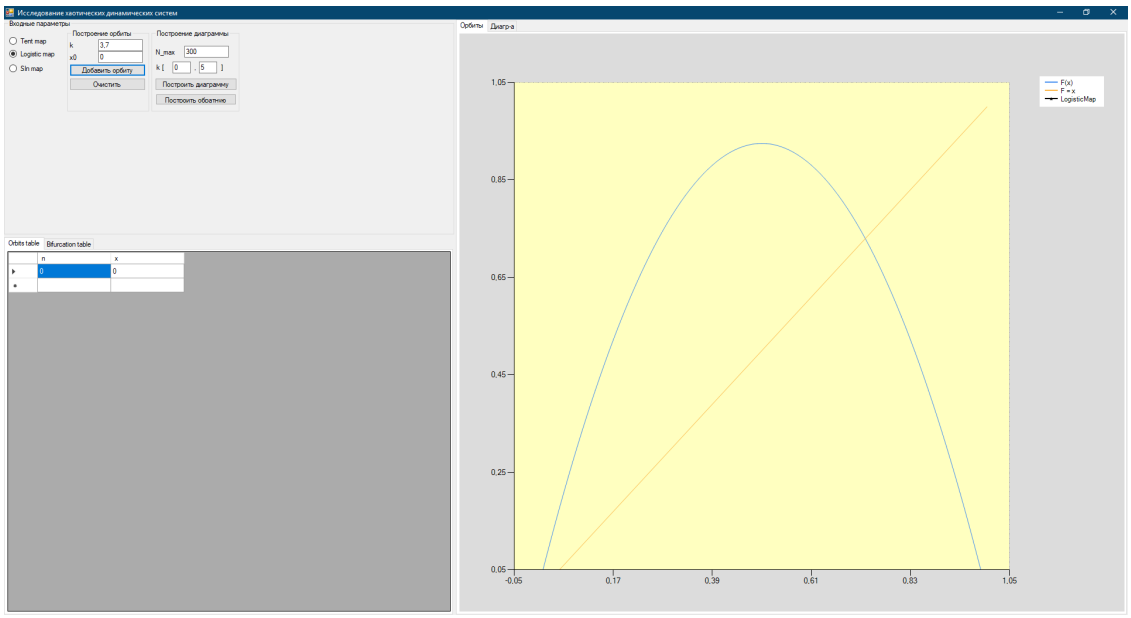

Рис. 7.4: Орбиты Logistic map при $r = 3.7, x_0 = 0$

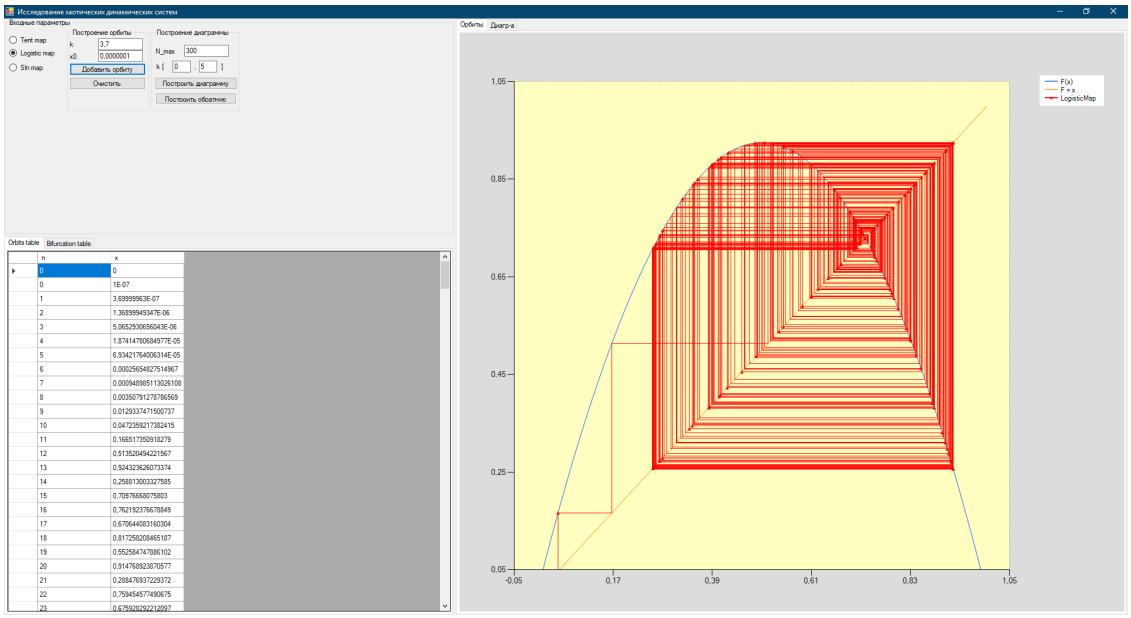

Рис. 7.5: Орбиты Logistic map при $r = 3.7, x_0 = 0.0000001$

Аналогичный пример. На Рис.7.4 показана динамика орбиты при следующих начальных условиях : $r = 3.7, x_0 = 0$. x_0 - фиксированная точка. Ожидаемо, что изменение динамики орбит наблюдаться не будет исходя из определения фиксированной точки. В предыдущем примере мы наблюдали свойство сенсетивности в действии. Точно так же мы убедились в том, что это свойство справедливо и для логистического отображения.

7.1.3 Sin map

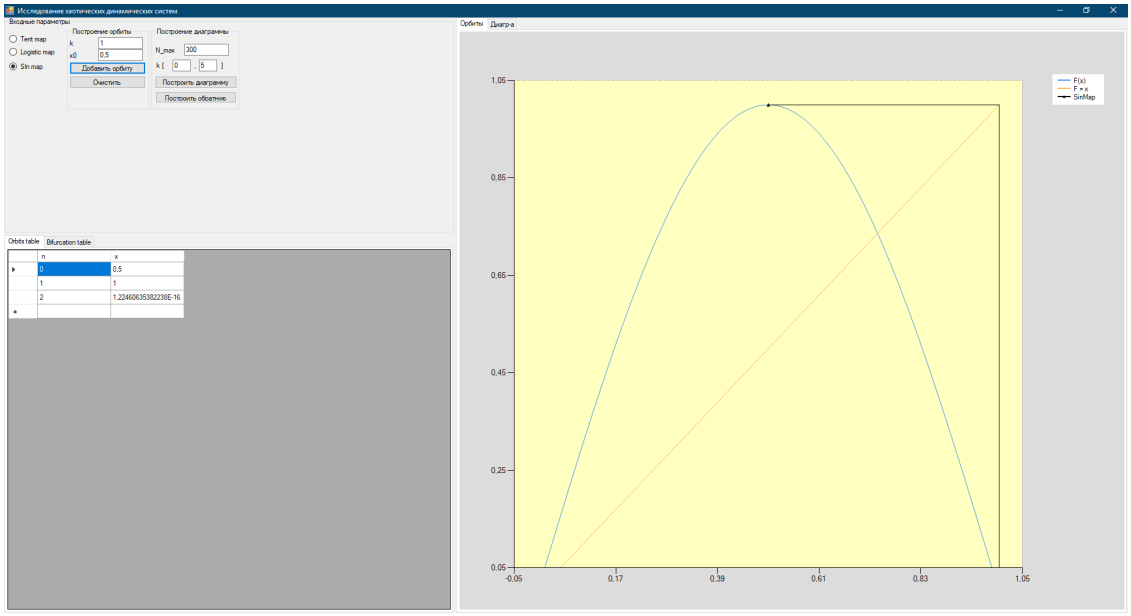

Рис. 7.6: Орбиты Logistic map при $r = 1, x_0 = 0.5$

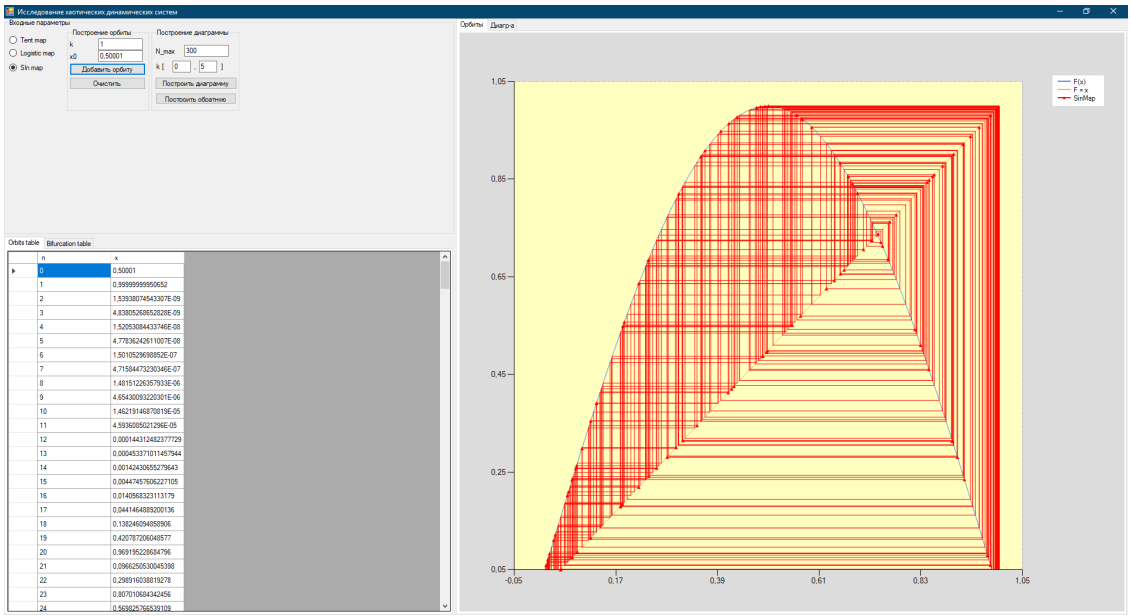

Рис. 7.7: Орбиты Logistic map при $r = 1, x_0 = 0.50001$

На рис 7.6 и 7.7 продемонстрирована жесткая чувствительность к начальным условиям для синусоидального отображения.

7.2 Пример построения бифуркационных диаграмм

7.2.1 Tent map

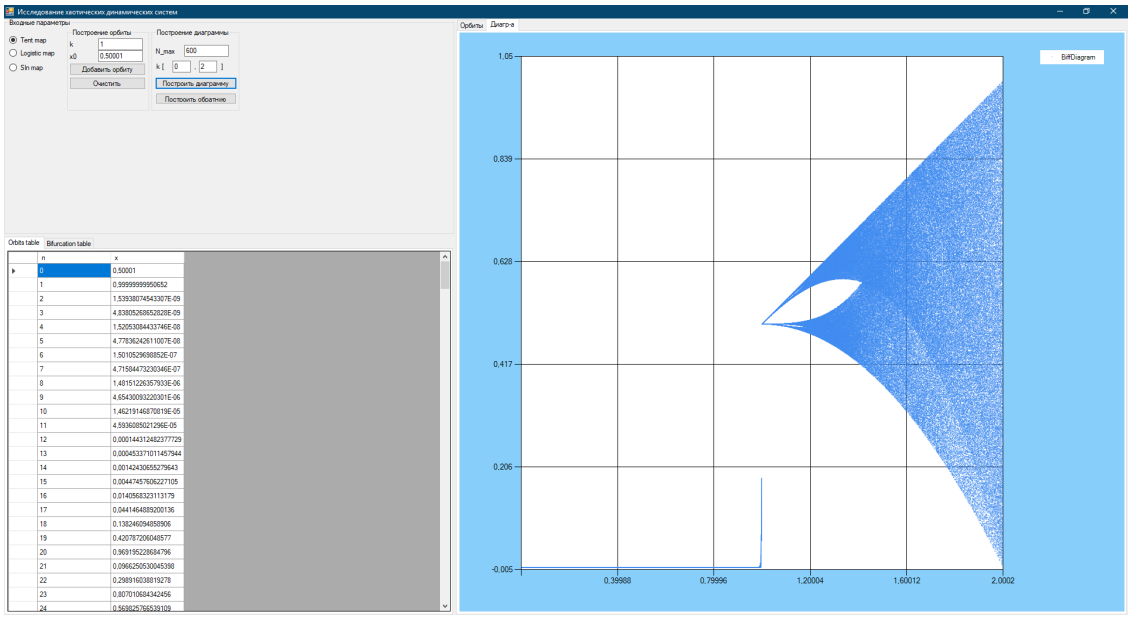

Рис. 7.8: Бифуркационная диаграмма tent map при $r \in [0, 2]$

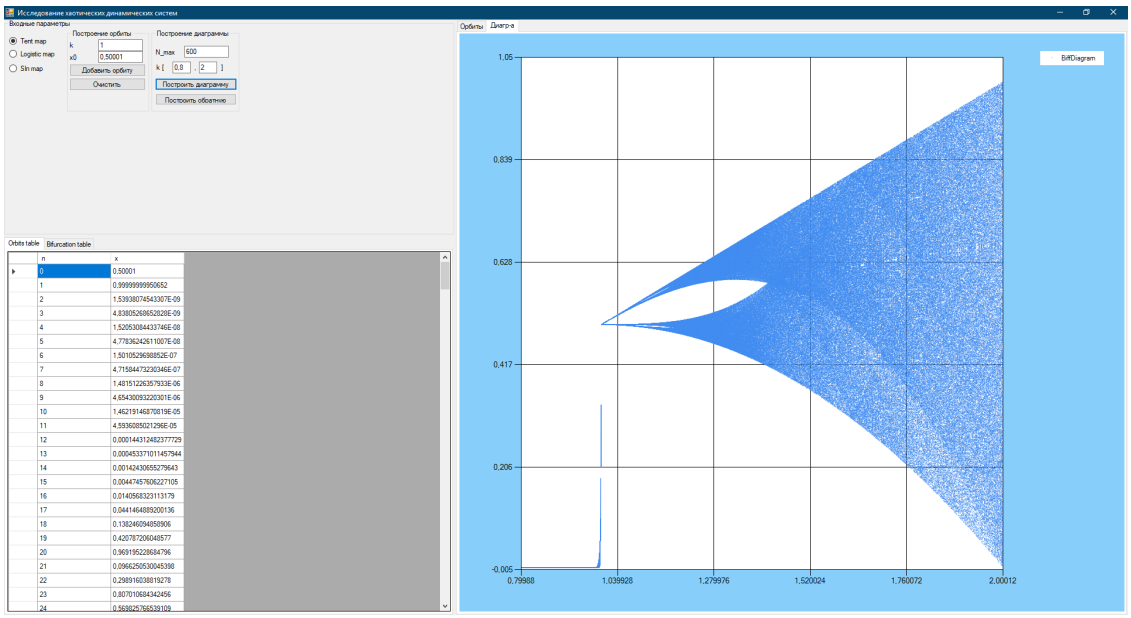

Рис. 7.9: Бифуркационная диаграмма tent map при $r \in [0.8, 2]$

На Рис. 7.8 изображена бифуркационная диаграмма тентетивного отображения при $r \in [0, 2]$. Видно, что при значениях параметра $r \in [0, 1]$ точка 0 является притягивающей. После происходит нечто, отдаленно напоминающее биффуркацию удвоения

7.2.2 Logistic map

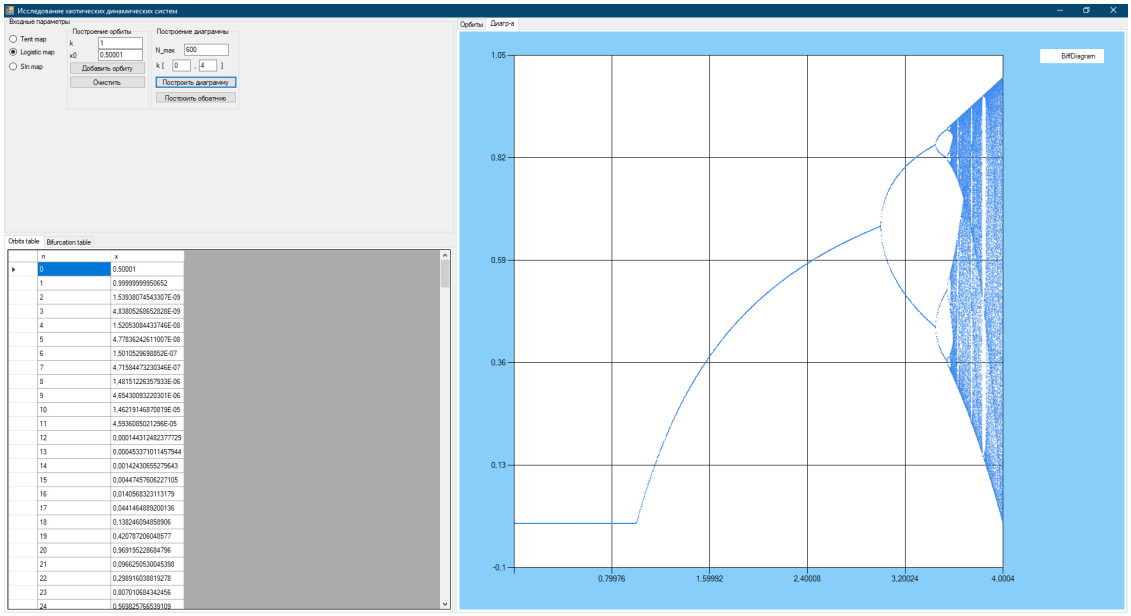

Рис. 7.10: Бифуркационная диаграмма logistic map при $r \in [0, 4]$

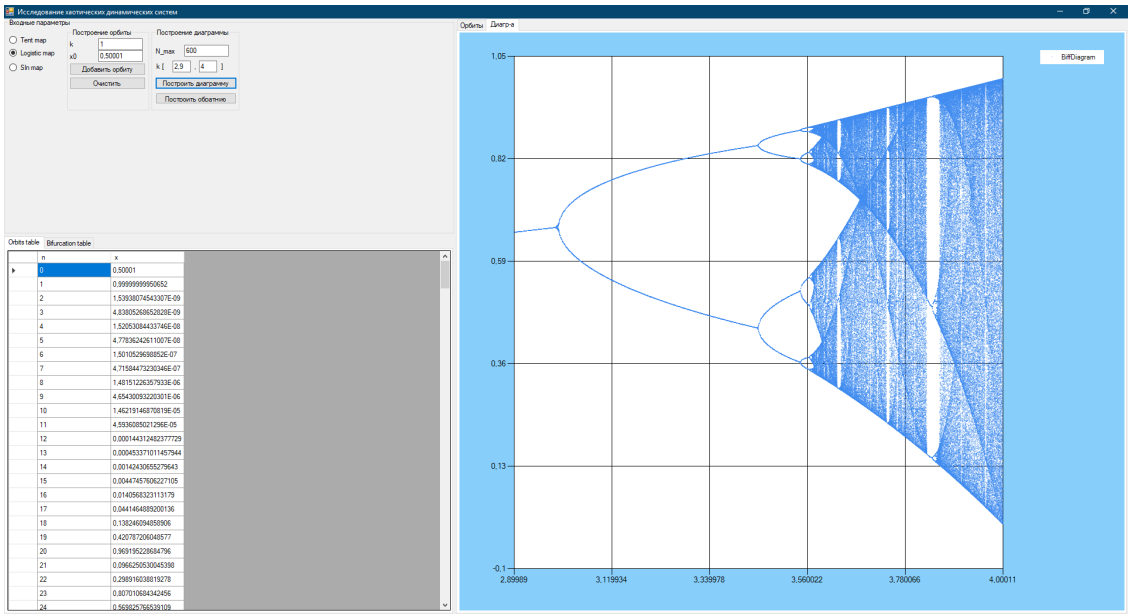

Рис. 7.11: Бифуркационная диаграмма tent map при $r \in [2.9, 4]$

7.2.3 Sin map

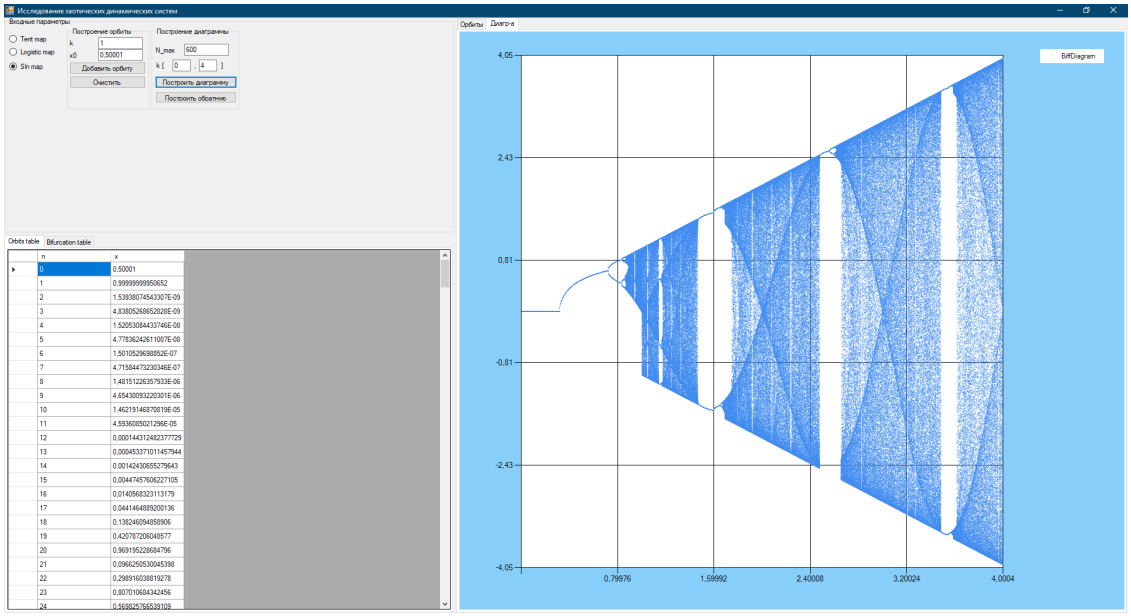

Рис. 7.12: Бифуркационная диаграмма Sin map при $r \in [0, 4]$

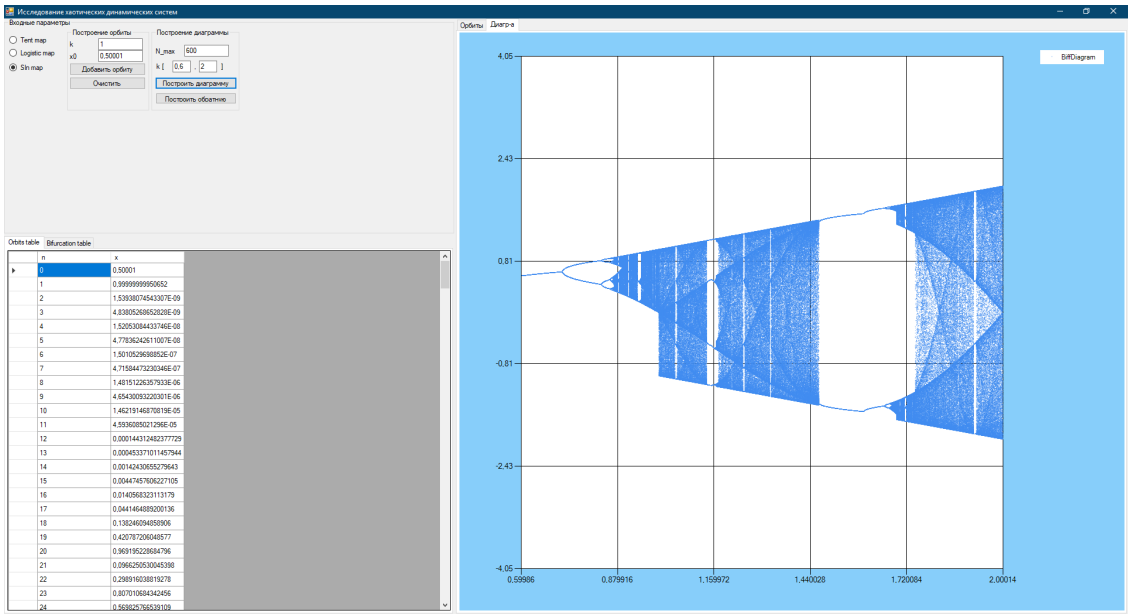

Рис. 7.13: Бифуркационная диаграмма tent map при $r \in [0.6, 2]$

Заключение

В данной работе были рассмотрены основные свойства хаотической динамики. Была написана программа на языке программирования $C\#$, позволяющая визуализировать орбиты различных одномерных отображений и строить для них бифуркационные диаграммы. При проведении эксперементов с орбитами мы убедились в чувствительной зависимости к начальным условиям в хаотических дискретных динамических системах. Так же при изучении бифуркационных диаграмм мы наблюдали точки удвоения периода, которые нам удалось отловить программно.

Литература

1) А.Каток, Б. Хасселблат — введение в современную теорию динамических систем. Г. Москва: Факториал, 1999. - 768 с.

2) R.L.Devaney - An Introduction to Chaotic Dynamical Systems. 1989 by Addison-Wesley Publishing Company

3) R.A.Holmgren - A First Course in Discrete Dynamical Systems. 1994 Springer-Verlag New York, Inc.

Приложение

10.1 Код программы

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

namespace chaos
{
public abstract class BiffurcationMap
{
private double _minCoeffValue;
public double minCoeffValue
{
protected get { return _minCoeffValue; }
set { _minCoeffValue = value; }
}

private double _maxCoeffValue;
public double maxCoeffValue
{
protected get { return _maxCoeffValue; }
set { _maxCoeffValue = value; }
}

private double _coefficient;
public double coefficient
{
get {
return _coefficient; }
set
{
if (value <= maxCoeffValue && value >= minCoeffValue)
{
```

```
_coefficient = value;
}
else _coefficient = maxCoeffValue;
}
}
private string _mapName;
public string mapName
{
get { return _mapName; }
protected set { _mapName = value; }
}

protected void setMapName(string value)
{
mapName = value;
}
public abstract double nextPoint(double point);
public abstract double nextRevertingPoint(double point);
}

public class TentMap : BiffurcationMap
{
public TentMap() : base ()
{
minCoeffValue = 0;
maxCoeffValue = 2;
coefficient = 0d;
mapName = "TentMap";

}

public TentMap(double coeff) : base()
{
minCoeffValue = 0;
maxCoeffValue = 2;
coefficient = coeff;
mapName = "TentMap";

}

public override double nextPoint(double point)
{
if (point <= 0.5)
{
return coefficient * point;
}
return -coefficient * point + coefficient;
```

```
}

public override double nextRevertingPoint(double point)
{
throw new NotImplementedException();
}
}

public class LogisticMap : BiffurcationMap
{
public LogisticMap() : base()
{
minCoeffValue = 0;
maxCoeffValue = 4;
coefficient = 0d;
mapName = "LogisticMap";

}

public LogisticMap(double coeff) : base()
{
minCoeffValue = 0;
maxCoeffValue = 4;
coefficient = coeff;
mapName = "LogisticMap";
}
public override double nextPoint(double point)
{
return coefficient * point * (1 - point);
}

public override double nextRevertingPoint(double point)
{
throw new NotImplementedException();
}
}

public class SinMap : BiffurcationMap
{
public SinMap()
{
minCoeffValue = 0;
maxCoeffValue = double.PositiveInfinity;
coefficient = 0;
mapName = "SinMap";
}
```

```
public SinMap(double coeff)
{
minCoeffValue = 0;
maxCoeffValue = double.PositiveInfinity;
coefficient = coeff;
mapName = "SinMap";
}
public override double nextPoint(double point)
{
return coefficient * Math.Sin(Math.PI * point);
}

public override double nextRevertingPoint(double point)
{
throw new NotImplementedException();
}
}

}

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.Charting;

namespace chaos
{
public partial class MainWindow : Form
{
public MainWindow()
{
InitializeComponent();
}
bool IsConvertibleToDouble(char elem)
{
if (elem == ',' || (int)elem >= '0' && (int)elem <= '9')
return true;
else
return false;
}
```

```
int CountUnconvertibleItems (string str)
{
int countBrokenIndexes = 0;
Parallel.ForEach(str, (elem) => {
if (!IsConvertibleToDouble(elem))
{
countBrokenIndexes++;
}
});
return countBrokenIndexes;
}
int CountBrokenInputSymbols()
{
int countBrokenIndexes = 0;
countBrokenIndexes += CountUnconvertibleItems(textBox_startingPoint.Text);
countBrokenIndexes += CountUnconvertibleItems(textBox_maxIter.Text);
countBrokenIndexes += CountUnconvertibleItems(textBox_coeff.Text);
countBrokenIndexes += CountUnconvertibleItems(textBox_endingCoeff.Text);
countBrokenIndexes += CountUnconvertibleItems(textBox_startingCoeff.Text);
return countBrokenIndexes;
}
private InputOrbitsParams inputParams;
private BiffurcationMap createBiffurcationMap(double coef)
{
BiffurcationMap map;

if (rb_IsTent.Checked)
{
return new TentMap(coef);
}
else if (rb_IsLog.Checked)
{
return new LogisticMap(coef);
}
else
{
map = new SinMap(coef);
}
return map;
}

delegate double funcWrapper(double point);
private void drowDefaultGraphics(funcWrapper func)
{
double x = 0;
double h = 1.0 / 100;
for (int j = 0; j <= 100; j++)
```

```
{
chart_orbits.Series[0].Points.AddXY(x, func(x));
x += h;
}
chart_orbits.Series[1].Points.AddXY(0, 0);
chart_orbits.Series[1].Points.AddXY(1, 1);
}
private void setAxisParams(BiffurcationMap map)
{
chart_orbits.Series[0].Name = "F(x)";
chart_orbits.Series[1].Name = "F = x";
chart_orbits.ChartAreas[0].AxisX.MajorGrid.Enabled = false;
chart_orbits.ChartAreas[0].AxisY.MajorGrid.Enabled = false;
chart_orbits.ChartAreas[0].AxisY.Minimum = 0.05;
chart_orbits.ChartAreas[0].AxisY.Maximum = 1.05;
chart_orbits.ChartAreas[0].AxisX.Minimum = -0.05;
chart_orbits.ChartAreas[0].AxisX.Maximum = 1.05;
}
void setInputParamsorbitsDrow()
{
inputParams.startingPoint = Convert.ToDouble(textBox_startingPoint.Text);
inputParams.itersNum =  Convert.ToInt32(textBox_maxIter.Text);
inputParams.coefficient = Convert.ToDouble(textBox_coeff.Text);
}
private void calculateAndDrowMap(BiffurcationMap map)
{
Queue<double> que = new Queue<double>();
double x_0 = inputParams.startingPoint;
double itersNum = inputParams.itersNum;
que.Enqueue(x_0);

chart_orbits.Series[2].Name = map.mapName;
dataGridView_orbitsTable.Rows.Add(0, x_0);

bool IsFixed = false;
chart_orbits.Series[2].Color = Color.Red;
int i = 1;
for (; i < itersNum && x_0 != 0; i++)
{
if (IsFixed == true) break;
double x_1 = map.nextPoint(x_0);
chart_orbits.Series[2].Points.AddXY(x_0, x_1);
chart_orbits.Series[2].Points.AddXY(x_1, x_1);
dataGridView_orbitsTable.Rows.Add(i, x_1);
x_0 = Math.Round(x_1, 11);
if (que.Count > 10000) que.Dequeue();
```

```
foreach (var s in que)
{
if (s == x_0)
{
IsFixed = true;
chart_orbits.Series[2].Color = Color.Black;
break;
}
}
que.Enqueue(x_0);
}
if (x_0 == 0)
{
IsFixed = true;
chart_orbits.Series[2].Color = Color.Black;
}

string s1 = $"Рассматривали {map.mapName} при x_0 = {inputParams.startingPoint}, k = {map.coefficient}  ";
if (IsFixed)
{
string s2 = $"Точка является периодической с периодом {i}";
Form form = new ResultWindow(s1, s2);
form.Show();
}
if (!IsFixed)
{
string s2 = $"Точка не является периодической";
Form form = new ResultWindow(s1, s2);
form.Show();
}
}
void CheckInputData()
{
while (CountBrokenInputSymbols() != 0)
{
var errorMessage = new ExceptionWindow($"\t Ошибка !!! \n Количество недопустимых символов = {CountBroken
$"Входные параметры должны содержать только цифры или запятую");
errorMessage.Show();
return;
}
}
private void button1_Click(object sender, EventArgs e)
{
CheckInputData();
setInputParamsorbitsDrow();
var map = createBiffurcationMap(inputParams.coefficient);
setAxisParams(map);
```

```
drowDefaultGraphics(map.nextPoint);
calculateAndDrowMap(map);
}


private void button2_Click(object sender, EventArgs e)
{
chart_orbits.Series[0].Points.ClearQuick();
chart_orbits.Series[1].Points.ClearQuick();
chart_orbits.Series[2].Points.ClearQuick();
chart_biffurcation.Series[0].Points.ClearQuick();
dataGridView_orbitsTable.Rows.Clear();
}

private void groupBox2_Enter(object sender, EventArgs e)
{

}


void setInputParamsBiffMap()
{
inputParams.coefficient = Convert.ToDouble(textBox_startingCoeff.Text);
inputParams.startingPoint = Convert.ToDouble(textBox_startingPoint.Text);
inputParams.coefficientEnd = Convert.ToDouble(textBox_endingCoeff.Text);
inputParams.itersNum = Convert.ToInt32(textBox_maxIter.Text);

}

private void button3_Click(object sender, EventArgs e)
{
CheckInputData();
setInputParamsBiffMap();
var flag = 0;
if (rb_IsTent.Checked)
{
flag = 1;
chart_biffurcation.ChartAreas[0].AxisY.Minimum = -0.005;
chart_biffurcation.ChartAreas[0].AxisY.Maximum = 1.05;
}
if (rb_IsLog.Checked)
{
flag = 2;
chart_biffurcation.ChartAreas[0].AxisY.Minimum = -0.1;
chart_biffurcation.ChartAreas[0].AxisY.Maximum = 1.05;
}
if (rb_IsSin.Checked)
```

```
{
flag = 3;
chart_biffurcation.ChartAreas[0].AxisY.Minimum = -4.05;
chart_biffurcation.ChartAreas[0].AxisY.Maximum = 4.05;
}
double x_0 = inputParams.startingPoint;
double coef = inputParams.coefficient;
double coef_end = inputParams.coefficientEnd;
double N_max = inputParams.itersNum;
double h = (-coef + coef_end) / 1000;
var map = createBiffurcationMap(coef);

var eps = h / 10;
chart_biffurcation.ChartAreas[0].AxisX.Minimum = coef - eps;
chart_biffurcation.ChartAreas[0].AxisX.Maximum = coef_end + eps;
chart_biffurcation.Series[0].Points.ClearQuick();
chart_biffurcation.Series[0].Points.SuspendUpdates();
List<double> check_biff = new List<double>();
List<double> arr_biff = new List<double>();
var for_me = new List<int>();
int check_count = 1;
var epsilon = 1e-2;
for (; coef <= coef_end; coef += h)
{
check_biff.Clear();
var x = x_0;
for (int i = 0; i < 500; i++)
{
x = next_x.next(flag, coef, x);
}
check_biff.Add(x);
for (int i = 0; i < N_max; i++)
{
bool checker = false;
x = next_x.next(flag, coef, x);
chart_biffurcation.Series[0].Points.AddXY(coef, x);
if (check_biff.Count()<16) foreach(double elem in check_biff)
{
if (Math.Abs(elem - x) < epsilon) { checker = true; break; }
}
if (!checker && (check_biff.Count() < 16))
{
check_biff.Add(x);
}
//chart2.Series[0].Points.AddXY(coef, x);// next_x.next(flag, coef)(x));
}
if (check_biff.Count() != check_count)
```

```
{
arr_biff.Add(coef);
for_me.Add(check_biff.Count());
}
check_count = check_biff.Count();

}

chart_biffurcation.Series[0].Points.ResumeUpdates();
dataGridView_bifurcationTable.Rows.Clear();
for (int i = 0; i < arr_biff.Count(); i++)
{
dataGridView_bifurcationTable.Rows.Add(i, arr_biff[i], for_me[i]);
}
}

private void bindingNavigator1_RefreshItems(object sender, EventArgs e)
{

}

private void button4_Click(object sender, EventArgs e)
{
var flag = 0;
if (rb_IsTent.Checked)
{
flag = 1;
chart_biffurcation.ChartAreas[0].AxisY.Minimum = -0.005;
chart_biffurcation.ChartAreas[0].AxisY.Maximum = 1.05;
}
if (rb_IsLog.Checked)
{
flag = 2;
chart_biffurcation.ChartAreas[0].AxisY.Minimum = -0.1;
chart_biffurcation.ChartAreas[0].AxisY.Maximum = 1.05;
}
if (rb_IsSin.Checked)
{
flag = 3;
chart_biffurcation.ChartAreas[0].AxisY.Minimum = -4.05;
chart_biffurcation.ChartAreas[0].AxisY.Maximum = 4.05;
}

flag = 4;
double x_0 = Convert.ToDouble(textBox_startingPoint.Text);
double coef = Convert.ToDouble(textBox_startingCoeff.Text);
double coef_end = Convert.ToDouble(textBox_endingCoeff.Text);
```

```
double N_max = Convert.ToDouble(textBox_maxIter.Text);
double h = (-coef + coef_end) / 1000;
var eps = h / 10;
chart_biffurcation.ChartAreas[0].AxisX.Minimum = coef - eps;
chart_biffurcation.ChartAreas[0].AxisX.Maximum = coef_end + eps;
chart_biffurcation.Series[0].Points.ClearQuick();
chart_biffurcation.Series[0].Points.SuspendUpdates();
List<double> check_biff = new List<double>();
List<double> arr_biff = new List<double>();
var for_me = new List<int>();
int check_count = 1;
var epsilon = 1e-2;
bool checker = false;
for (; coef <= coef_end; coef += h)
{
check_biff.Clear();
var x = x_0;
for (int i = 0; i < 500; i++)
{
x = next_x.next(flag, coef, x);
}
check_biff.Add(x);
for (int i = 0; i < N_max; i++)
{
checker = false;
x = next_x.next(flag, coef, x);
chart_biffurcation.Series[0].Points.AddXY(coef, x);
if (check_biff.Count() < 16) foreach (double elem in check_biff)
{
if (Math.Abs(elem - x) < epsilon) { checker = true; break; }
}
if (!checker && (check_biff.Count() < 16))
{
check_biff.Add(x);
}
//chart2.Series[0].Points.AddXY(coef, x);// next_x.next(flag, coef)(x));
}
if (check_biff.Count() != check_count)
{
arr_biff.Add(coef);
for_me.Add(check_biff.Count());
}
check_count = check_biff.Count();

}

chart_biffurcation.Series[0].Points.ResumeUpdates();
```

```
dataGridView_bifurcationTable.Rows.Clear();
for (int i = 0; i < arr_biff.Count(); i++)
{
dataGridView_bifurcationTable.Rows.Add(i, arr_biff[i], for_me[i]);
}
}
}
}

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;

namespace chaos
{
public partial class ResultWindow : Form
{
public ResultWindow(string s1,string s2)
{

InitializeComponent();
label1.Text = s1;
label2.Text = s2;
}


}
}
```