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 ифференциальных уравнений, математического и численного анализа
Направление подготовки: «Прикладная математика и информатика»
Профиль подготовки: «Прикладная математика и информатика (общий профиль)»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему:

«Визуализация типичных бифуркаций в семействах одномерных отображений»

Выполнил:
студент группы 381603-1
Королев Александр Павлович
ф.и.о.

_____
подпись

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

_____
подпись

Нижний Новгород
2020

Оглавление

1 Введение 3

2 Основные определения и теоремы 4
- 2.1 Определение бифуркации динамической системы . . . 4
- 2.2 Теорема Шарковского . . . 4
- 2.3 Определение дискретной динамической системы . . . 4
- 2.4 Определение орбиты . . . 5
- 2.5 Определение неподвижной точки . . . 5
- 2.6 Определение периодической точки . . . 5
- 2.7 Определение хаоса для дискретной динамической системы по R. Devaney . . . 5

3 Рассматриваемые семейства одномерных отображений 6
- 3.1 Tent map . . . 6
- 3.2 Logistic map . . . 7
- 3.3 Sin map . . . 8

4 Структура программы 9

5 Руководство пользователя 10
- 5.1 Пример построения орбит . . . 11
  - 5.1.1 Tent Map . . . 11
  - 5.1.2 Logistic map . . . 12
  - 5.1.3 Sin map . . . 13
- 5.2 Пример построения бифуркационных диаграмм . . . 14
  - 5.2.1 Tent map . . . 14
  - 5.2.2 Logistic map . . . 15
  - 5.2.3 Sin map . . . 18

6 Заключение 19

7 Литература 20

8 Приложение 21
- 8.1 Код программы . . . 21

Введение

Впервые о теории хаоса заговорили в 19 веке, но первое научное изучение этой теории началось во второй половине 20 века, вместе с работами Эдварда Лоренца из Массачусетского технологического института и франкоамериканского математика Бенуа Б. Мандельброта.

Эдвард Лоренц в начале 60-х годов 20 века изучал возникновение трудностей при прогнозировании погоды. Существовало два основных подхода прогнозирования погоды на длительный промежуток времени. Подход, сформулированный в 1776 году французским математиком П. С. Лапласом., гласил, что «... если мы представим себе разум, который в данное мгновение постиг все связи между объектами во Вселенной, то он сможет установить соответствующее положение, движения и общие воздействия всех этих объектов в любое время в прошлом или в будущем». Другими словами, Лаплас и сторонники его идей утверждали, что для точного прогнозирования погоды необходимо только собрать больше информации обо всех частицах во Вселенной, их местоположении, скорости, массе, направлении движения, ускорении и т.п. Лаплас считал, что чем больше человек будет знать, тем более точный прогноз возможно будет совершить относительно будущего.

Второй подход был сформулирован в 1903 году другим известным французским математиком Жюлем Анри Пуанкаре. «Если бы мы точно знали законы природы и положение Вселенной в начальный момент, мы могли бы точно предсказать положение той же Вселенной в последующий момент. Но даже если бы законы природы открыли нам все свои тайны, мы и тогда могли бы знать начальное положение только приближенно. Если бы это позволило нам предсказать последующее положение с тем же приближением, это было бы все, что нам требуется, и мы могли бы сказать, что явление было предсказано, что оно управляется законами. Но это не всегда так; может случиться, что малые различия в начальных условиях вызовут очень большие различия в конечном явлении. Малая ошибка в первых породит огромную ошибку в последнем. Предсказание становится невозможным, и мы имеем дело с явлением, которое развивается по воле случая».

Из этих слов Пуанкаре был сформулирован постулат теории хаоса о существенной зависимости от начальных условий. Последующее развитие науки, в частности квантовой механики, опровергло детерминизм Лапласа. В 1927 году немецкий физик Вернер Гейзенберг открыл и сформулировал принцип неопределенности. Данный принцип объясняет, почему некоторые случайные явления не подчиняются лапласовому детерминизму. Гейзенберг показал принцип неопределенности на примере радиоактивного распада ядра. Так, из-за очень малых размеров ядра невозможно знать все процессы, происходящие внутри него. Поэтому, сколько бы информации мы не собирали о ядре, точно предсказать, когда это ядро распадется невозможно.

В данной работе рассмотрены основные свойства хаотических динамических систем на примере различных семейств одномерных отображений(Tent map, Logistic map, Sin map) и показана динамика изменения поведения этих семейств при различных значениях параметров системы. Реализована программа с возможностью построения орбит, построения бифуркационных диаграмм, отыскания бифуркационных точек удвоения периода.

Основные определения и теоремы

2.1 Определение бифуркации динамической системы

Бифуркация – это качественная перестройка картины движения. Значения управляющего параметра, при которых происходят бифуркации, называются критическими или бифуркационными значениями.

Данное определение нам понадобится при рассмотрении бифуркационных диаграмм. Особый интерес будут представлять бифуркации удвоения периода.

2.2 Теорема Шарковского

Если непрерывное отображение одномерного интервала в себя имеет цикл периода m, то оно имеет также и циклы со всевозможными периодами m', предшествующими числу m в перечне всех целых чисел, выписанных в порядке Шарковского:

$$1 \lhd 2 \lhd 2^2 \lhd 2^3 \lhd 2^4 \lhd \ldots \lhd 2^3 \cdot 9 \lhd 2^3 \cdot 7 \lhd 2^3 \cdot 5 \lhd 2^3 \cdot 3 \lhd \ldots$$
$$\ldots \lhd 2^2 \cdot 9 \lhd 2^2 \cdot 7 \lhd 2^2 \cdot 5 \lhd 2^2 \cdot 3 \lhd \ldots$$
$$\ldots \lhd 2 \cdot 9 \lhd 2 \cdot 7 \lhd 2 \cdot 5 \lhd 2 \cdot 3 \lhd \ldots \lhd 9 \lhd 7 \lhd 5 \lhd 3.$$

Рис. 2.1: Порядок Шарковского

Данная теорема нам понадобится при работе с логистическим отображением при значении параметра $r > 3.8$. Порядок Шарковского мы проверяем на различных семействах одномерных отображений, в частности для логистического семейства имеется орбита периода 3 , а значит(по теореме Шарковского) существует орбита любых периодов, т.е. наблюдается хаотическое поведение.

2.3 Определение дискретной динамической системы

Пусть (X, ρ) - метрическое пространство, на котором определено непрерывное отображение $f : X \to X$

Дискретная динамическая система задается итерационным процессом вида $x_{n+1} = f(x_n)$

2.4 Определение орбиты

Орбитой (или траекторией) точки x_0 отображения f называется последовательность $x_0, x_1 = f(x_0), x_2 = f(x_1) = f^2(x_0)\ \dots\ , x_n = f(x_{n-1}) = f^n(x_0),\ \dots; n = 0, 1, 2 \dots$

2.5 Определение неподвижной точки

x_0 называется неподвижной точкой отображения f, если $f(x_0) = x_0$

2.6 Определение периодической точки

x_0 называется периодической точкой отображения f периода n, если $f^n(x_0) = x_0$

Определения 2.3 - 2.6 понадобятся при дальнейшем описании поведеня хаотических динамических систем и программной реализации задачи визуализации орбит и бифуркационных диаграмм.

2.7 Определение хаоса для дискретной динамической системы по R. Devaney

Пусть (X, ρ) — метрическое пространство, тогда отображение $f : X \to X$ называется хаотическим, если :

1) Отображение f транзитивно, т.е.
   В X есть всюду плотная орбита $\iff$ $\forall U_1$, $U_2 \subset X\ \exists x_0 \in U_1$, $\exists n \in \mathbb{N}$ такие,
   что $f^n(x_0) \in U_2; n \in \mathbb{N}$

2) Периодические точки отображения f всюду плотны в X.

3) У отображения f наблюдается чувствительная зависимость от начальных условий, т.е.:
   существует $\beta > 0$ такое, что для любой точки $x_0 \in X$
   и для любой окресности U точки x_0, найдутся $y_0 \in U$ и $n > 0$,
   для которых выполняется $\rho(f^n(x_0), f^n(y_0)) > \beta$

Стоит заметить, что приведенное определение хаотической динамической системы по Devaney — не единственно (например существует определение по Ли - Йорку). Хотя определения и различны, но для рассматриваемых нами одномерных отображений они фактически эквивалентны: выполняются (или не выполняются) одновременно.

Рассматриваемые семейства одномерных отображений

3.1 Tent map

Тент-отображение $T_r : [0,1] \to [0,1]$ - кусочно непрерывное отображение, 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 неподвижную 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 $|f^n(y_0) - f^n(x_0)| \geq 1/2 = \beta$

3.2 Logistic map

Логистическое отображение (Logistic map) — это отображение вида $L_r(x_n) = x_{n+1} = rx_n(1-x_n)$.

Оно было введено в 1845 г. П.Ф. Ферхюльстом. При помощи этого отображения описывалась динамика популяции в замкнутой среде, где x_n — численность особей в n-ый год. Из уравнения видно, что величина x_{n+1} пропорциональна численности особей в предыдущий год и свободной части жизненного пространства $(1-x_n)$. Параметр r, в свою очередь, зависит от реальной площади для жизни, плодовитости земли и проч.

Также можно рассмотреть задачу о банковских сбережениях при стабилизирующемся росте процента(Ричтер, Пейтген, 1984). Пусть x_0 — денежный вклад, который растет в зависимости от некоторого процента α следующим образом: $x_{n+1} = (1+\alpha)x_n = \cdots = (1+\alpha)^{n+1}x_0$. Чтобы помешать бесконечному обогащению, было решено, что процент будет уменьшаться пропорционально x_n т.е. $\alpha \to \alpha_0(1-\frac{x_n}{x_{\max}})$. Тогда мы получим следующий закон: $x_{n+1} = [1+\alpha(1-\frac{x_n}{x_{\max}})]x_n$, который при замене переменных является $L_r(x_n)$.

При взгляде на данное отображение можно прийти к выводу, что из-за механизма обратной связи численность особей или величина банковского счета будут стремиться к каким-то средним значениям. Но при изменении параметра r поведение становится достаточно сложным и при больших значениях параметра будет хаотическим (как показано Гроссманом и Томэ).

Отметим, что хаотическое поведение не вызвано своеобразием логистического семейства. Как было показано Фейгенбаумом, при некотоых ограничениях переход к хаосу, найденный для логистического отображения, встречается во всех разностных уравнениях первого порядка $x_{n+1} = f(x_n)$.

Приведем ряд свойств логистического отображения :

1) Решение уравнения $rx(1-x) = x$ показывает, что L_r имеет неподвижные точки в 0 и $p_r = \frac{r-1}{r}$. Кроме того замечаем, что точки 1 и $\frac{1}{r}$ отображаются в неподвижные, т.к. $L_r(1) = 0, L_r(\frac{1}{r}) = p_r$

2) Если $0 < r < 1$, то неподвижная точка $p_r < 0$ и мы ее не рассматриваем т.к. она не попадает в исследуемое нами пространство. Точка 0 является притягивающей.

3) Параметр $r = 1$ — бифуркационное значение(см. п. 4).

4) При $1 < r < 3$ точка 0 — отталкивающая неподвижная точка, а p_r — притягивающая.

5) При $r = 3$ происходит бифуркация удвоения: p_r перестает быть притягивающей точкой, а при дальнейшем увеличении r она становится отталкивающей, вместо нее появляется устойчивая периодическая орбита T_r периода 2.

6) При $3 < r < 3.4$ обе точки 0 и p_r являются отталкивающими,а орбита T_r — притягивающая .

7) При $r \approx 3.45$ происходит очередная бифуркация удвоения: орбита T_r становится отталкивающей, вместо нее появляется устойчивая орбита T_r' периода 4, и т.д. (см. п. 8)

8) При $3.44 < r < 3.56$ происходит бесконечный каскад бифуркаций удвоения.

9) При $r > 3.6$ происходят различные бифуркации(не только бифуркации удвоения), наблюдается (в основном) хаотическое поведение, но также присутствуют окна регулярности.

3.3 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 $4 \sin \pi x = x$

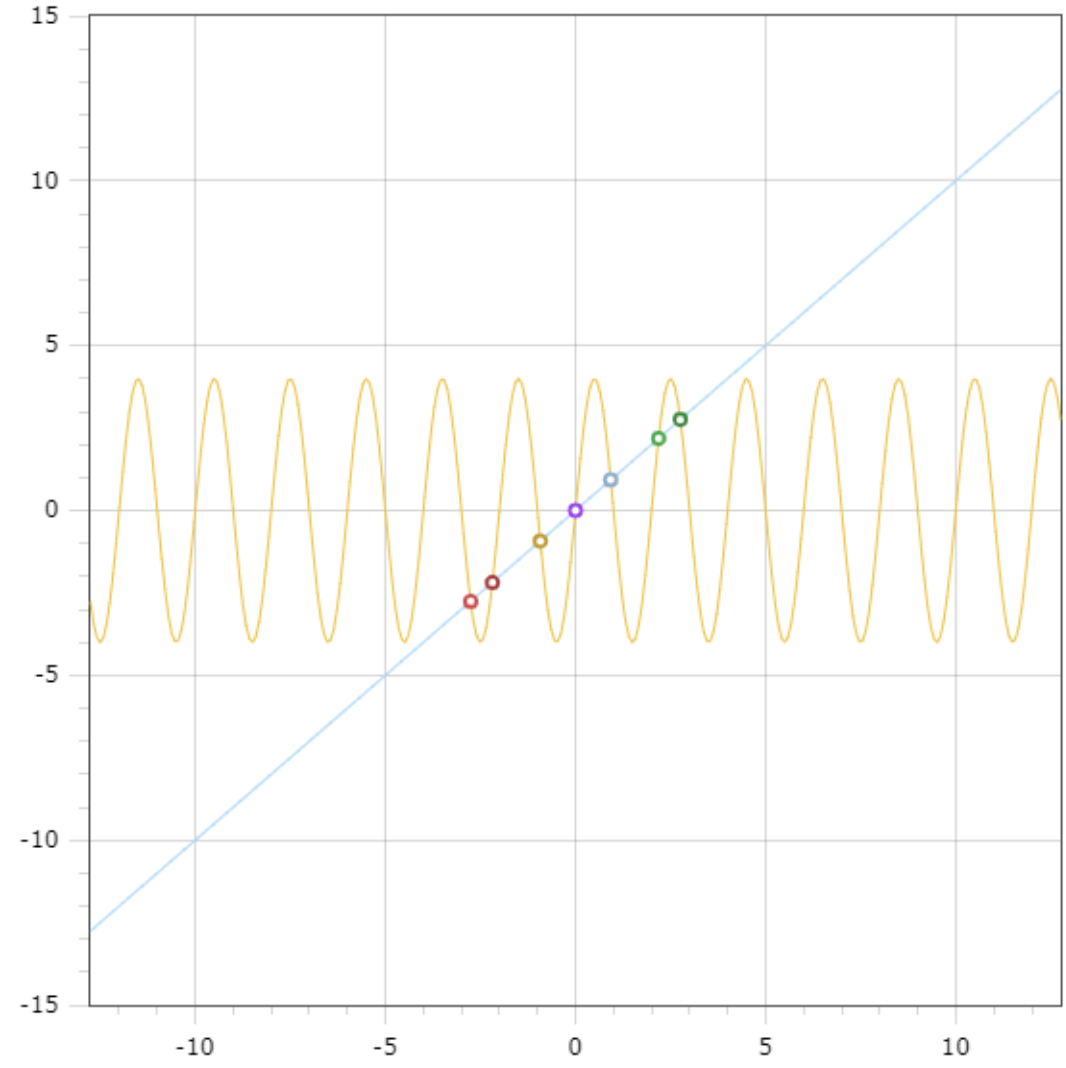

Рис. 3.1: Численное решение уравнения $4 \sin \pi x = x$

мы получим следующие неподвижные точки : $x_1 = 0, x_2 = 0.95266, x_3 = 2.1838, x_4 = 2.75784$

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

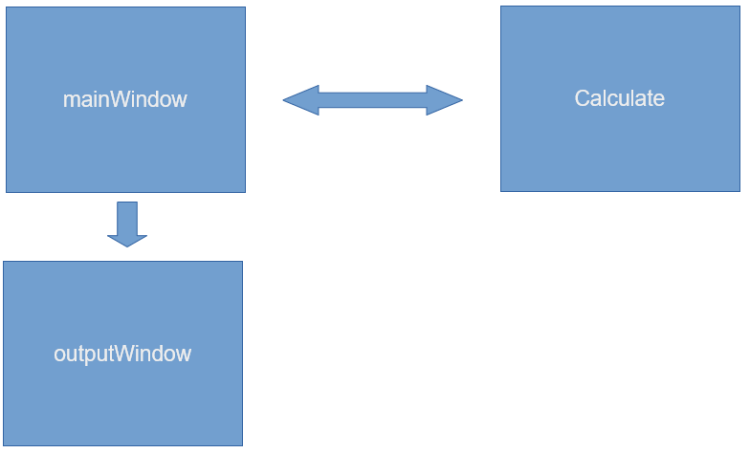

Рис. 4.1: Общая структура программы

Руководство пользователя

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

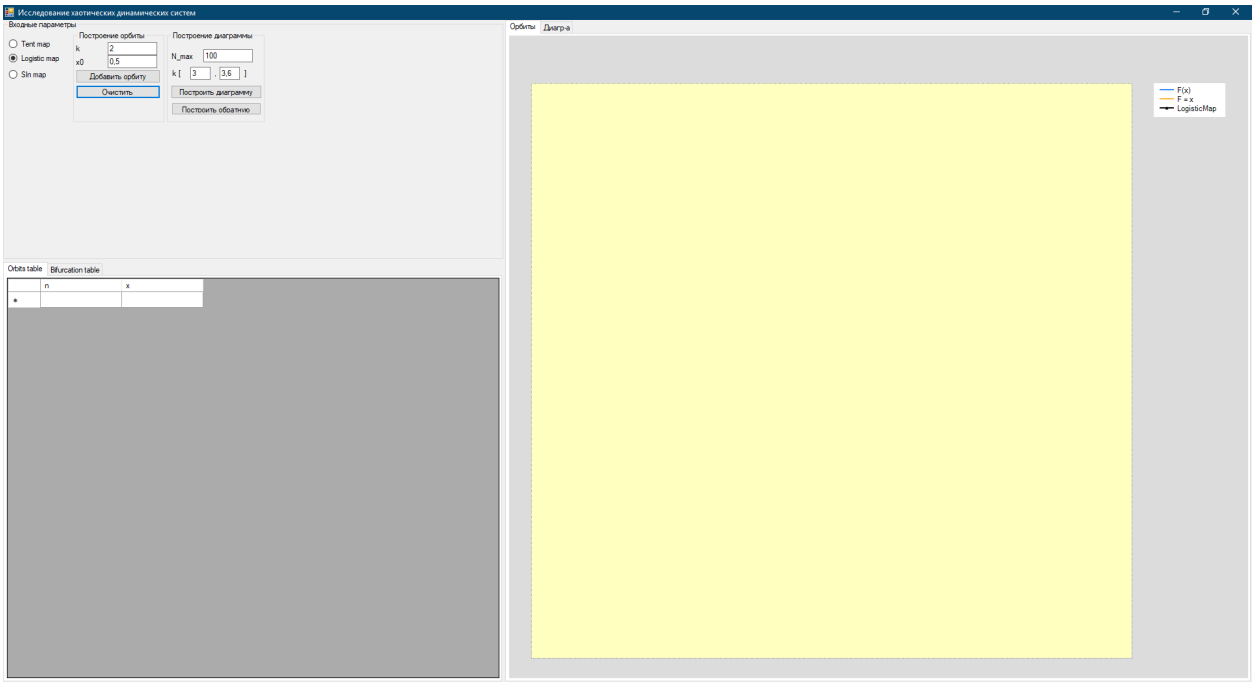

Рис. 5.1: Начальное окно программы

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 неподвижную точку

При нажатии кнопки "Построить диаграмму" — в правой части во вкладке с одноименным названием будет построена бифуркационная диаграмма для заданного диапазона r

5.1 Пример построения орбит

5.1.1 Tent Map

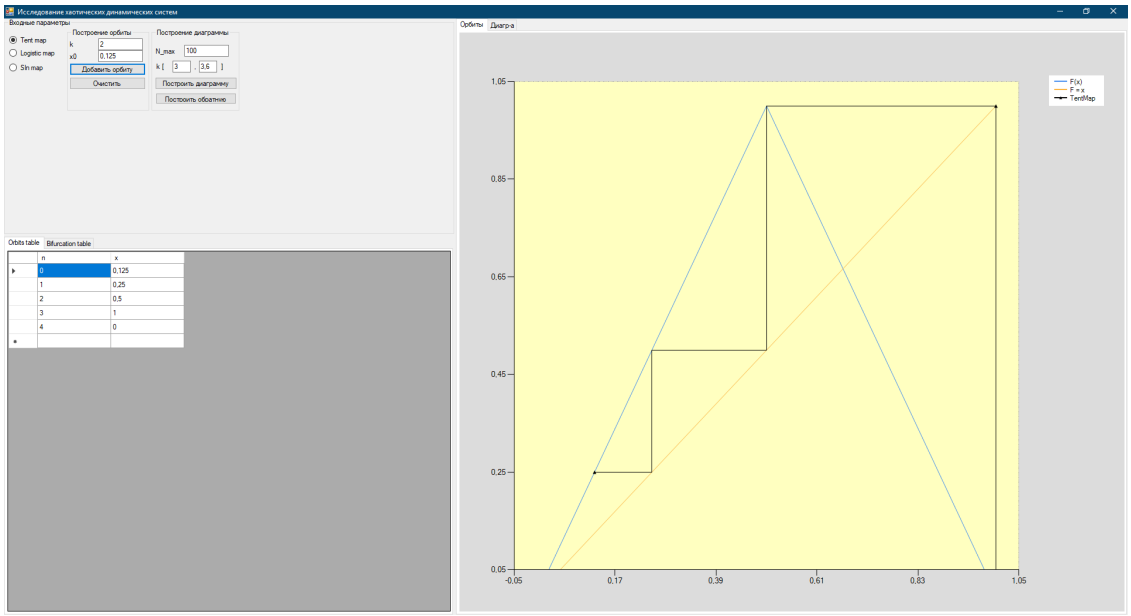

Рис. 5.2: Орбиты Tent map при $r = 2, x_0 = 0.125$

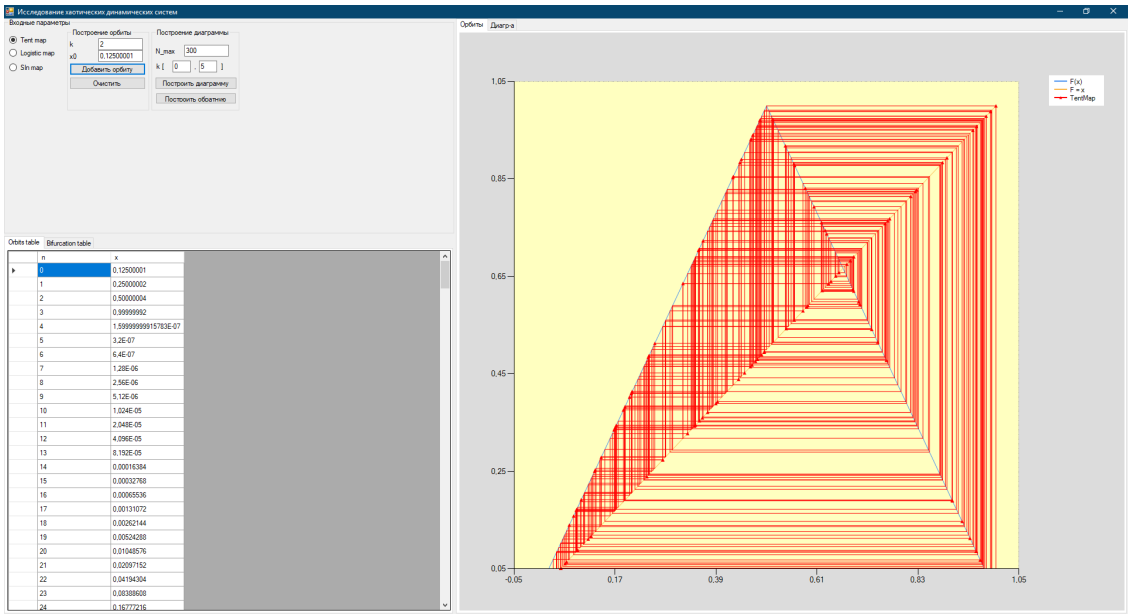

Рис. 5.3: Орбиты Tent map при $r = 2, x_0 = 0.12500001$

В данном примере на рис. 5.2 показана динамика орбиты при следующих начальных условиях $r = 2, x_0 = 0.125$. Мы можем наблюдать, что за 5 итераций, начальная точка перемещается в неподвижную точку 0. На рис.5.3 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.

5.1.2 Logistic map

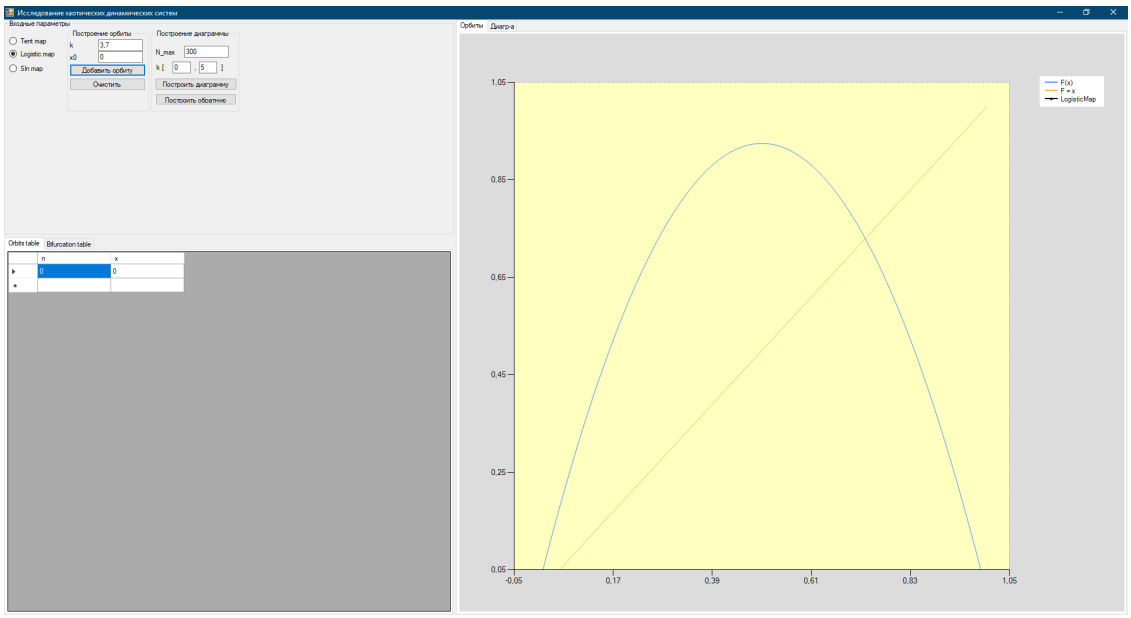

Рис. 5.4: Орбиты Logistic map при $r = 3.7, x_0 = 0$

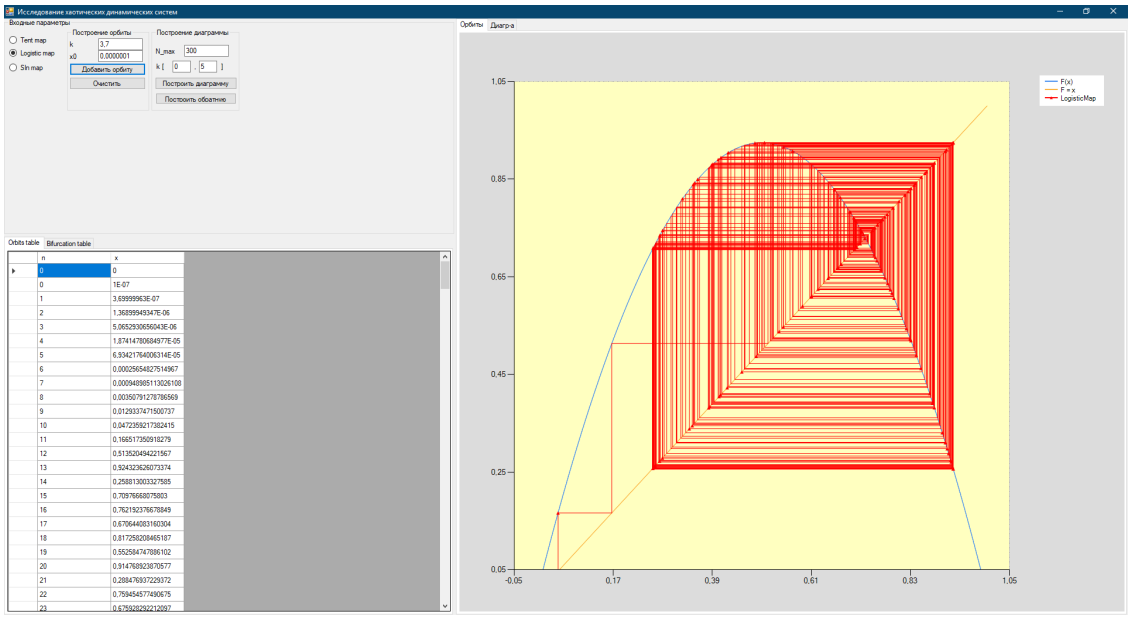

Рис. 5.5: Орбиты Logistic map при $r = 3.7, x_0 = 0.0000001$

Перейдем к рассмотрению лиогистического отображения : на рис. 5.4 показана динамика орбиты при следующих начальных условиях : $r = 3.7, x_0 = 0$. x_0 - неподвижная точка. Ожидаемо, что изменение динамики орбит наблюдаться не будет исходя из определения неподвижной 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.

5.1.3 Sin map

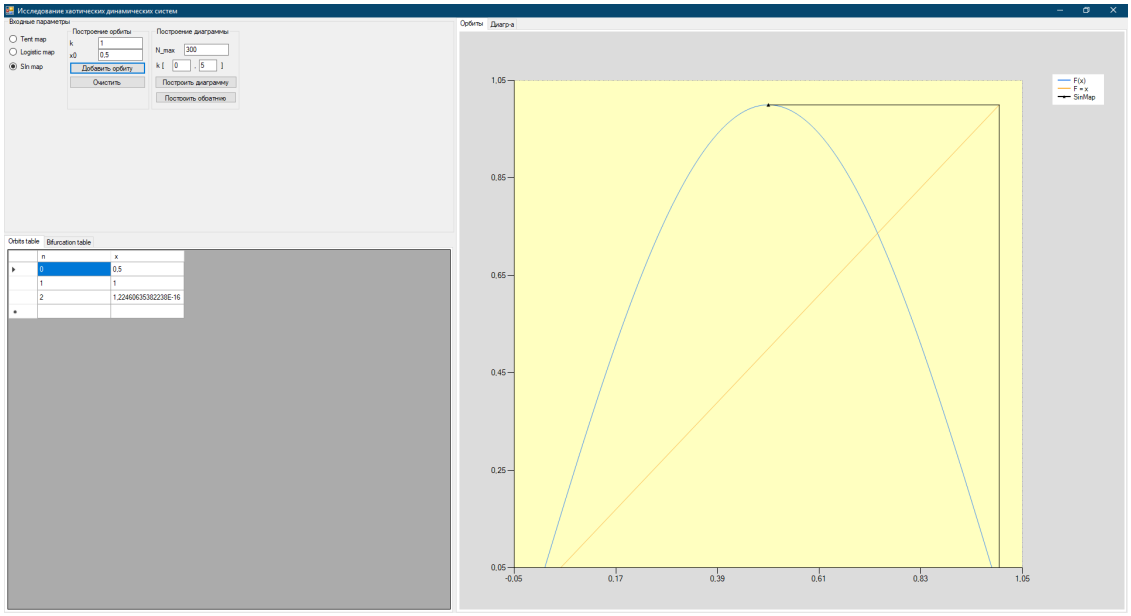

Рис. 5.6: Орбиты Logistic map при $r = 1, x_0 = 0.5$

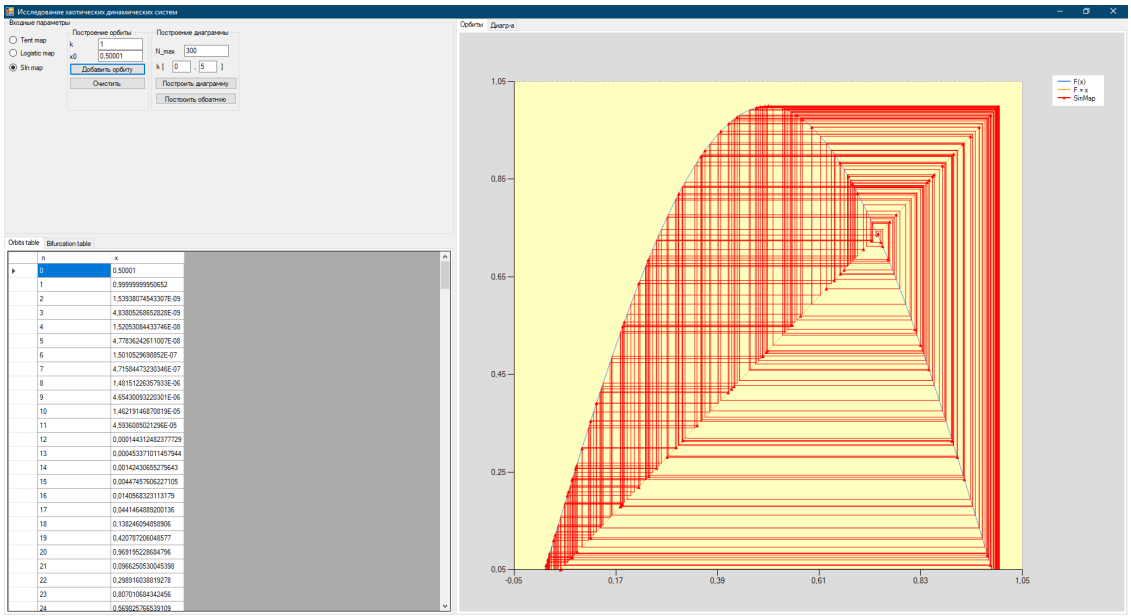

Рис. 5.7: Орбиты Logistic map при $r = 1, x_0 = 0.50001$

Теперь перейдем к рассмотрению синусоидального отображения. Так же, как и для предыдущих отображений на рис. 5.6 и рис. 5.7 продемонстрирована жесткая чувствительность к начальным условиям.

5.2 Пример построения бифуркационных диаграмм

5.2.1 Tent map

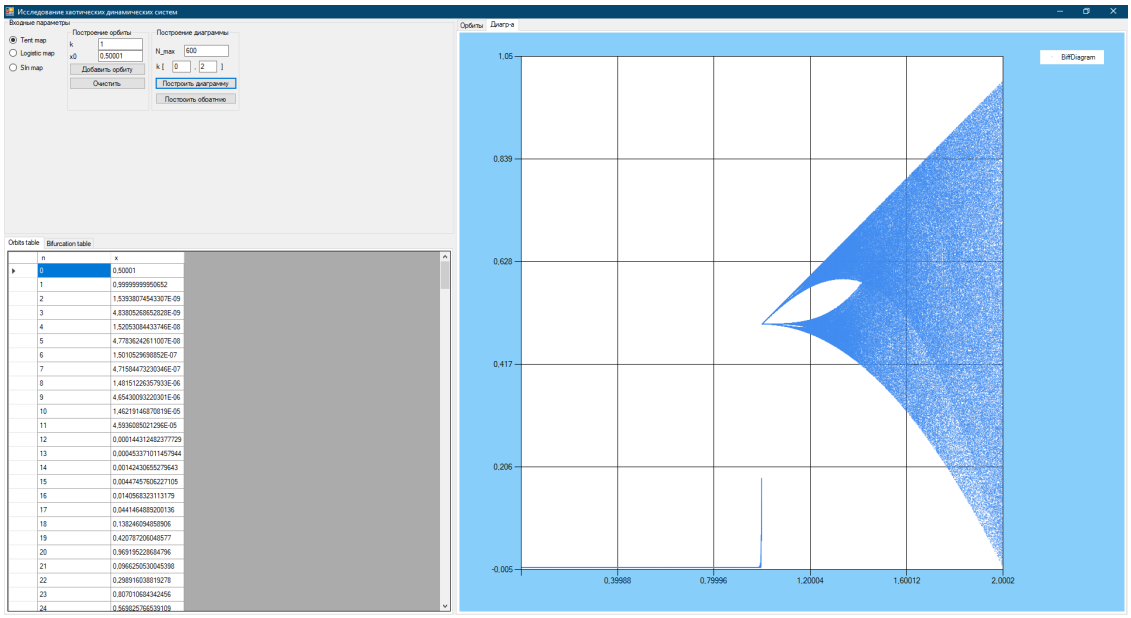

Рис. 5.8: Бифуркационная диаграмма tent map при $r \in [0, 2]$

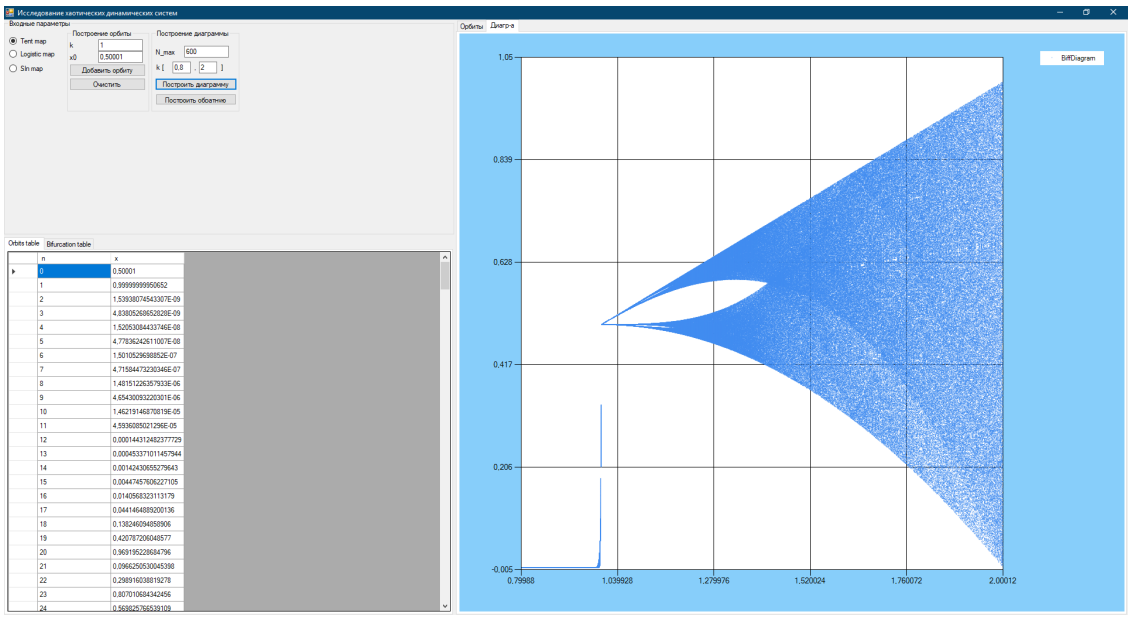

Рис. 5.9: Бифуркационная диаграмма tent map при $r \in [0.8, 2]$

На рис. 5.8 изображена бифуркационная диаграмма тент-отображения при $r \in [0, 2]$. Видно, что при значениях параметра $r \in [0, 1]$ точка 0 является притягивающей. На интервале параметров $[1, \sqrt{2}]$ мы наблюдаем частично хаотическое поведение, имеются так называемые лакуны и атрактор, представляющих собой конечное объединение интервалов. Точки лакун (кроме конечного числа отталкивающих периодических точек) через несколько итераций попадают в атрактор. Атрактор представляет собой инвариантное множество, на котором тент-отображение хаотично.

При $r > \sqrt{2}$ наступает сильный хаос.

5.2.2 Logistic map

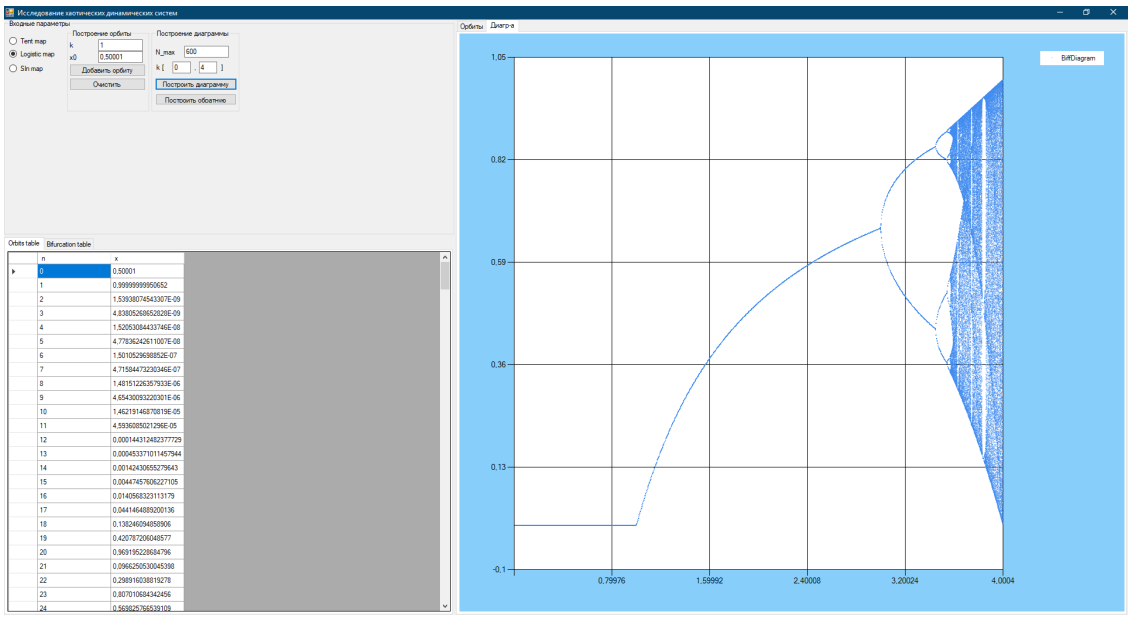

Рис. 5.10: Бифуркационная диаграмма logistic map при $r \in [0, 4]$

На рис. 5.10 показана бифуркационная диаграмма логистического отображения при $r \in [0, 4]$. Проверим свойства описанные в главе 4.2.

1) При $r \in (0, 1)$ точка 0 — притягивающая неподвижная точка, а p_r — отталкивающая.

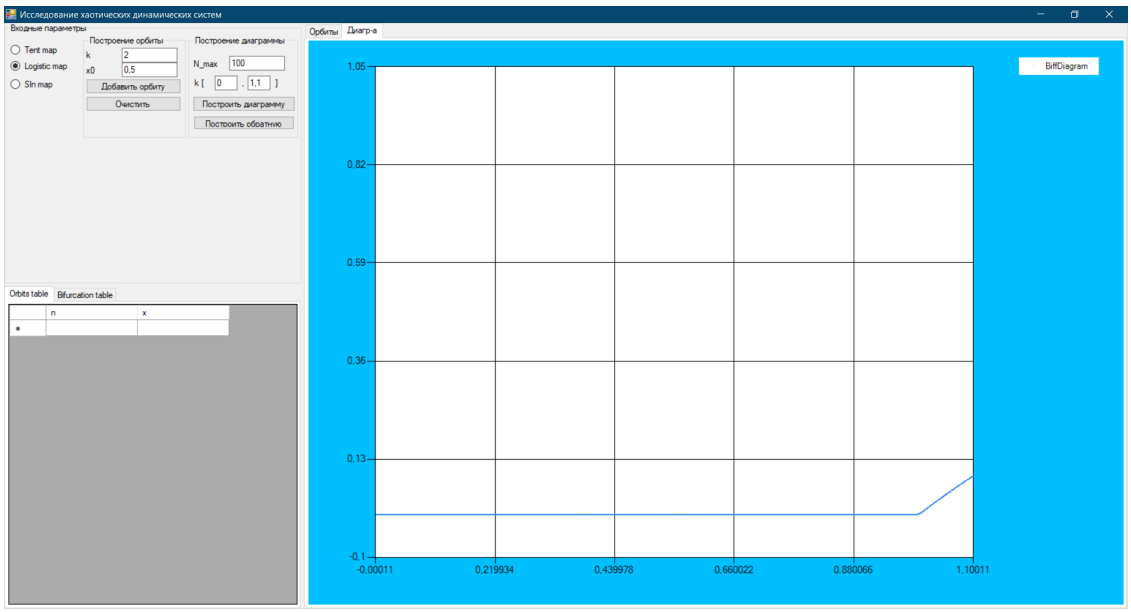

Рис. 5.11: Бифуркационная диаграмма logistic map при $r \in [0, 1.1]$

На рис. 5.11 видно, что свойство, выдвинутое нами, справедливо. При заданных значениях параметра $r \in (0, 1)$ притягивающей неподвижной точкой является 0. p_r - отталкивающая и не отображается на данном графике.

2) $r = 1$ - бифуркационная точка.

На рис. 5.11 наглядно видно, что при переходе параметра rчерез 1 меняется поведение нашей динамической системы. Следовательно $r = 1$ действительно является бифуркационной точкой и наше свойство справедливо.

3) При $1 < r < 3$, 0 - отталкивающая неподвижная точка, а p_r - притягивающая.

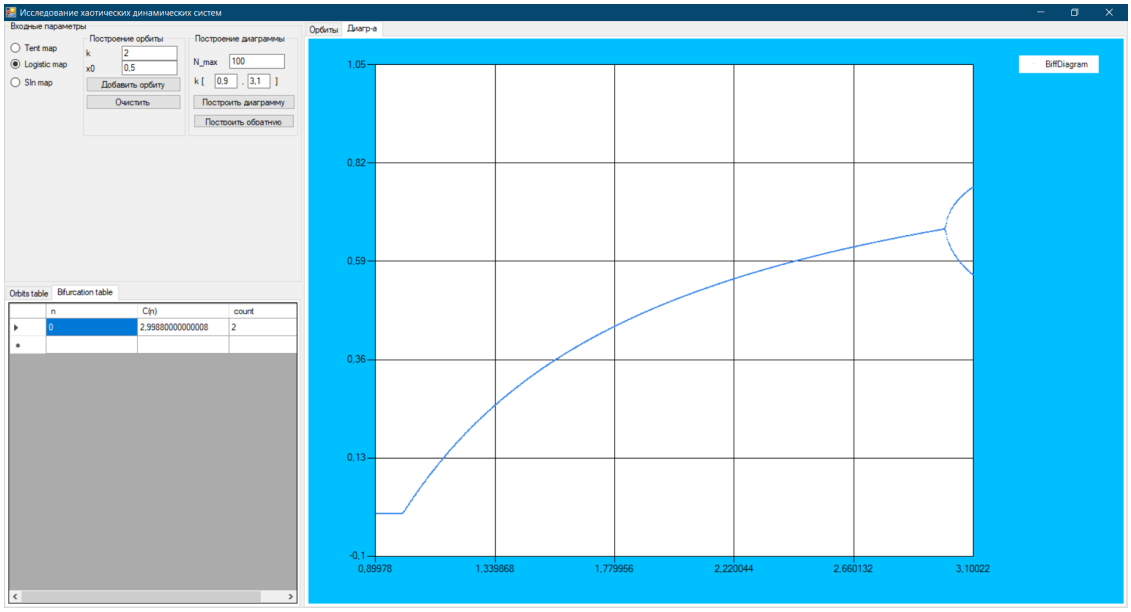

Рис. 5.12: Бифуркационная диаграмма tent map при $r \in [1, 3.1]$

На рис. 5.12 видно, что данное свойство справедливо. Напомним, что $p_r = \frac{r-1}{r}$. Поэтому график показывает изменение этой точки в соответствии с увеличением r.

4) $r = 3$ - точка бифуркации удвоения периода.
На рис. 5.12 показано, что значеник параметра $r = 3$ действительно является значением бифуркации удвоения периода.

5) При $3 < r < 3.4$ обе точки 0 и p_r являются отталкивающими
Как видно на рис. 5.13 после бифуркации удвоения у нас появилась орбита с периодом два. Найти координаты этих точек можно решив следующую систему уравнений:

$$\begin{cases} rx_n(1 - x_n) = x_{n+1} \\ rx_{n+1}(1 - x_{n+1}) = x_n \end{cases}$$

Получим следующее решение:

$$\begin{cases} x_n = \frac{\sqrt{r^2-2r-3}+r+1}{2r} \\ x_{n+1} = -\frac{(\sqrt{r^2-2r-3}-r+1)(\sqrt{r^2-2r-3}+r+1)}{4r} \end{cases}$$

Именно эти величины мы можем наблюдать на рис. 5.13.

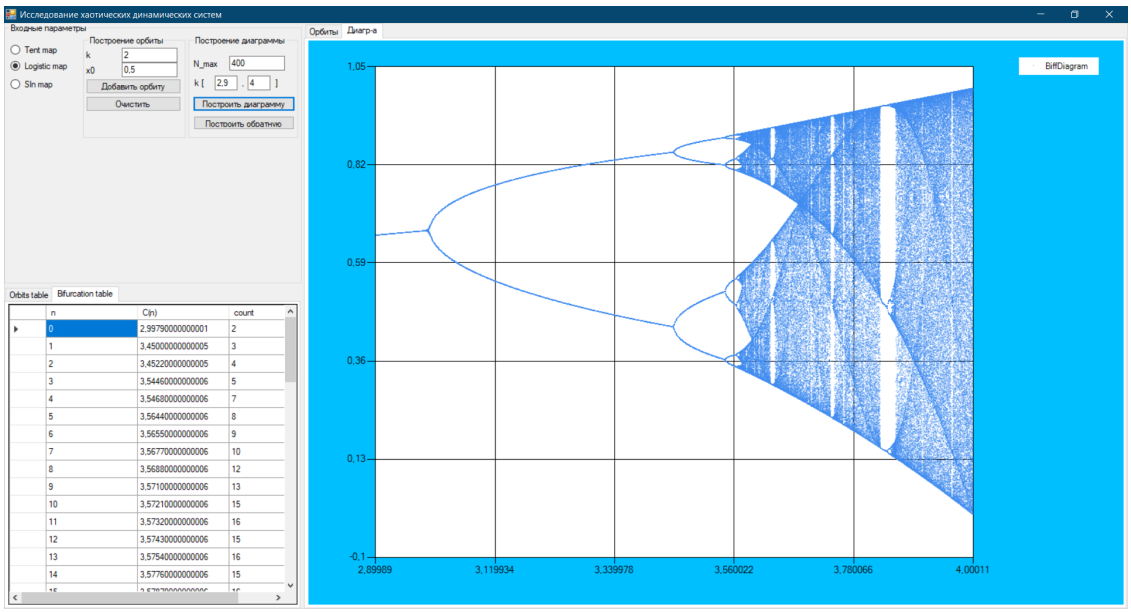

Рис. 5.13: Бифуркационная диаграмма tent map при $r \in [1, 3.1]$

6) При $r \approx 3.45$ происходит очередная бифуркация удвоения.
Опять же в этом не трудно убедиться взглянув на рис. 5.13 при $r \in [3, 3.45]$

Дополнительно стоит отметить, что на рис. 5.13 можно наблюдать как хаотическое поведение сменяется так называемыми "окнами регулярности".

5.2.3 Sin map

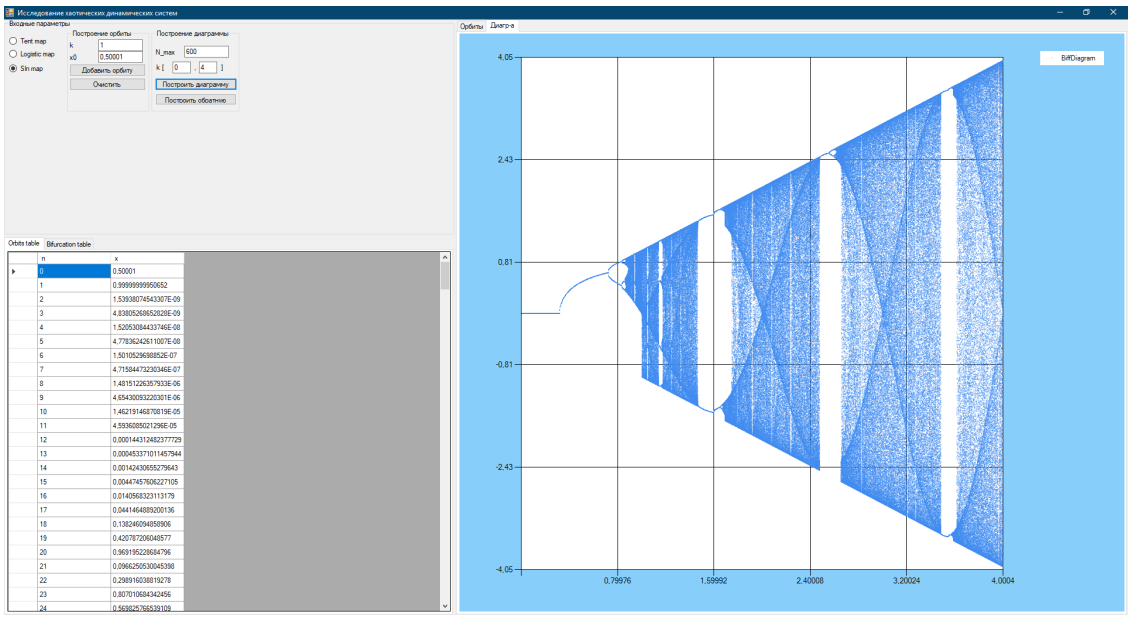

Рис. 5.14: Бифуркационная диаграмма Sin map при $r \in [0, 4]$

На рис. 5.14 изображена бифуркационная диаграмма синусоидального отображения при $r \in [0, 4]$.

При $r \in [0, 0.3]$ 0 является притягивающей неподвижной точкой.

Значение параметра $r \approx 0.3$ является бифуркационной точкой.

При $r \approx 0.71$ происходит бифуркация удвоения периода.

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. Также 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.

4) M.I. Malkin, Ming-Chia Li, “Smooth symmetric and Lorenz models for unimodal maps”, International Journal of Bifurcation and Chaos, 13:11 (2003), 3353–3372

Приложение

8.1 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;
}


}
}
```