

Министерство образования и науки Российской Федерации
ФГАОУ ВПО «УрФУ имени первого Президента России Б. Н. Ельцина»
Институт радиоэлектроники и информационных технологий - РтФ
Департамент информационных технологий и автоматики

Исследование предельных циклов нелинейной
системы

ОТЧЕТ

по лабораторной работе

Преподаватель: Пименов Владимир Германович
Студент: Сухоплюев Илья Владимирович
Группа: РИ-440001

Екатеринбург
2017

Аннотация

Работа описывает исследование параметризованной нелинейной системы на наличие предельных циклов. В ходе исследования системы, рассматриваются следующие вопросы: нахождение параметра системы, при котором наблюдается предельный цикл; поиск параметра, где наблюдается бифуркация поведения системы; исследование свойств обнаруженного предельного цикла; определение характера его устойчивости; исследование влияния разных видов запаздывания на систему (постоянное, переменное и распределенное запаздывание), а также влияния стохастического воздействия на систему. Данные задачи изучаются путем проведения численных экспериментов с помощью интерпретатора Python 3.5 и математических библиотек (numpy[1], matplotlib[2]).

Содержание

1	Поиск предельного цикла	4
2	Поиск точек бифуркаций	9
3	Исследование свойств предельного цикла	12
4	Определение устойчивости через мультипликаторы системы . .	14
5	Определение устойчивости с помощью метода Пуанкаре . . .	17
6	Влияние постоянного запаздывания	19
7	Влияние переменного запаздывания	25
8	Влияние распределенного запаздывания	29
9	Влияние случайного шума	33
	Заключение	37
	Список использованных источников	38
A	Исходный код программ	39
A.1	Поиск предельного цикла	39
A.2	Исследование точек бифуркации системы	41
A.3	Исследование параметров найденного предельного цикла	43
A.4	Проверка устойчивости теоремой о мультипликаторах .	44
A.5	Проверка устойчивости цикла методом Пуанкаре . . .	46
A.6	Влияние постоянного запаздывания	48
A.7	Влияние переменного запаздывания	50
A.8	Влияние распределенного запаздывания	53
A.9	Влияние случайного шума	56
B	Дополнительные эксперименты	58
B.1	Влияние постоянного запаздывания на вторую переменную системы	58
B.2	Влияние переменного запаздывания на вторую переменную системы	62
B.3	Влияние распределенного запаздывание на вторую переменную системы	66
B.4	Влияние случайного шума на вторую переменную системы	69

1 Поиск предельного цикла

Рассмотрим исследуемую систему (уравнение (1.1), ν - параметр системы). Она описывается уравнением от одной фазовой переменной x . Уравнение дифференциальное, второго порядка и, в силу слагаемого $3\dot{x}^3$, нелинейное. Решение такого уравнения аналитическими методами является довольно сложной задачей, поэтому нашим методом исследования будет построение численных экспериментов, описывающих данную систему при определенном параметре ν .

$$\ddot{x} + 3\dot{x}^3 - \nu\dot{x} + x = 0 \quad (1.1)$$

Однако, в таком виде уравнение (1.1) не является удобным для моделирования. Поэтому приведем его к канонической форме от двух переменных, с помощью замены (1.2), получив уравнение от двух фазовых переменных y_1 и y_2 (Система уравнений (1.3)). В дальнейшем, мы будем пользоваться описанием нашей системы именно в таком виде.

$$\begin{cases} y_1 = x \\ y_2 = \dot{x} \end{cases} \quad (1.2)$$

$$\begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = -3y_2^3 + \nu y_2 - y_1 \end{cases} \quad (1.3)$$

Преобразовав систему к удобному для нас виду, перейдем к первой части работы – нахождения такого параметра ν , при котором наблюдается предельный цикл.

Для начала, дадим определение искомому объекту.

Определение 1 *Предельным циклом будем называть замкнутую изолированную траекторию в фазовом пространстве, подразумевая замкнутость в смысле периодичности поведения системы.*

Таким образом, нам нужно построить фазовый портрет нашей системы, на котором нужно будет обнаружить искомую замкнутую линию. Для этого, зная зависимость значения производных от их коорди-

нат, можно с помощью функции *streamplot*[3] построить фазовый портрет (Программа 1, в качестве параметра для начала возьмем $\nu = 1$).

Листинг 1 Построение фазового портрета

```
# Подключение используемых библиотек
# В дальнейшем является постоянным и опускается в листингах
# Полный исходный код программы можно найти в приложении
import matplotlib.pyplot as plt
import numpy as np

# Параметр системы
nu = 1

# создание сетки 100x100 точек в области [-3;3]x[-3;3]
Y, X = np.mgrid[-3:3:100j, -3:3:100j]

# вычисление фазовых векторов на сетке
Y1 = Y
Y2 = -3 * Y ** 3 + nu * Y - X

# построение фазового портрета
fig0, ax0 = plt.subplots()
plt.streamplot(X, Y, Y1, Y2)

# показать построенные графики (опускается в дальнейшем)
plt.show()
```

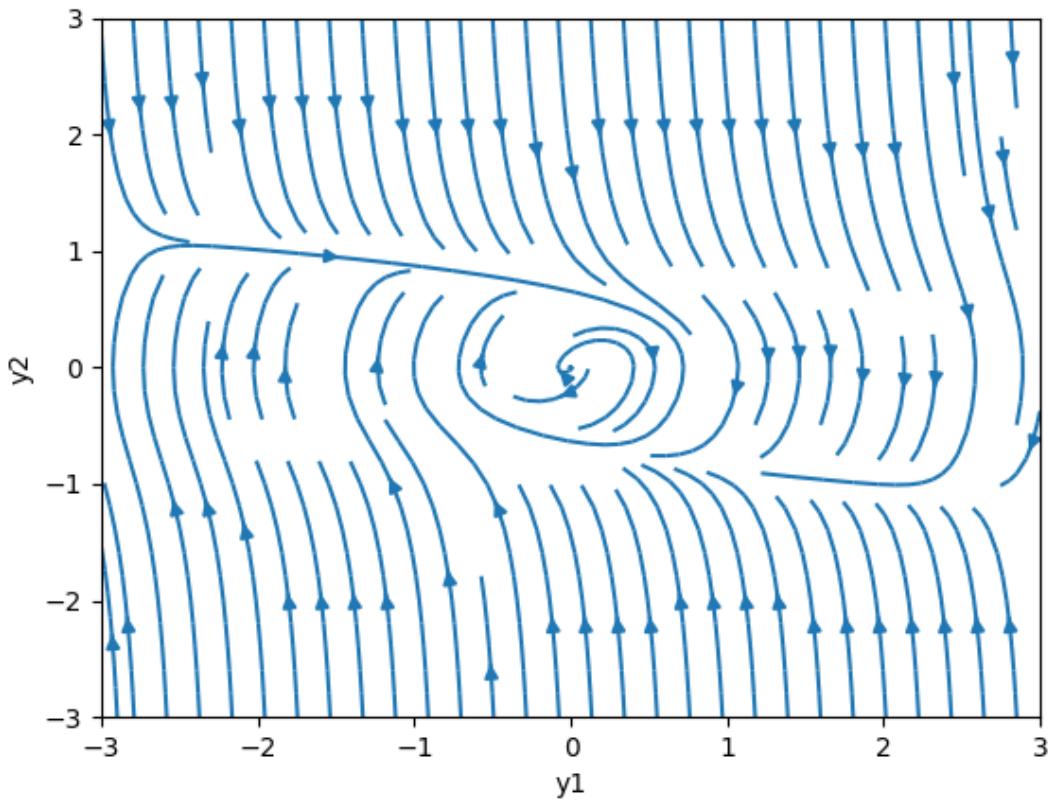


Рис. 1.1 — Поиск предельного цикла построением фазового портрета

На графике 1.1 изображен результат работы нашей программы. В данном случае значение параметра оказалось оптимальным: можно видеть, как изоклины сходятся к наклоненному прямоугольнику в центре графика.

Теперь, чтобы убедится наверняка, что траектории сходятся вокруг этого цикла и там нет разрывов, построим две линии методом Эйлера снаружи и внутри наблюдаемого цикла (Программа 2).

Листинг 2 Использование метода Эйлера для проверки предельного цикла

```
# функция построение кривой методом Эйлера
def line(y1_0, y2_0):
    y1 = [y1_0]
    y2 = [y2_0]
    h = 0.01 # длина шага
    for i in range(2000): # 2000 - количество итераций
        y1.append(y1[i] + h*(y2[i]))
        y2.append(y2[i] + h*(-3*y2[i] ** 3 + nu*y2[i] - y1[i]))
    # отображение кривой на графике
    ax0.plot(y1, y2)

# построение двух кривых, начинающихся внутри и
# вне предполагаемого предельного цикла
line(0.1, 0.1)
line(2, 2)
```

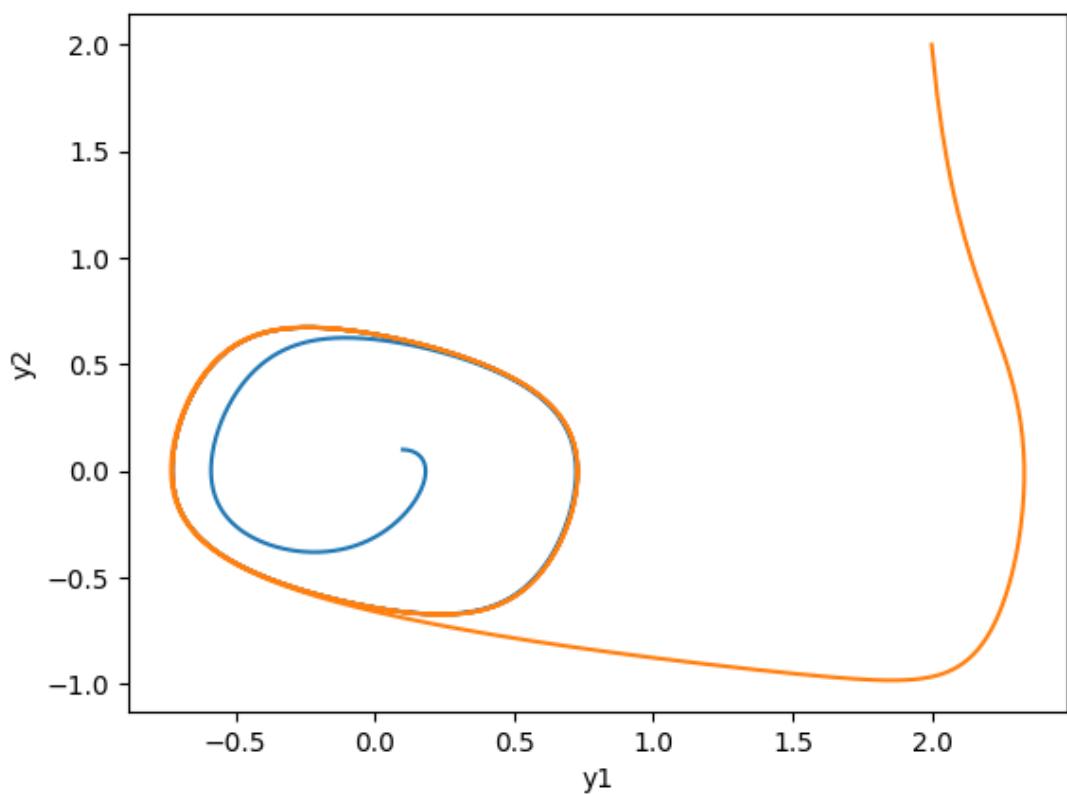


Рис. 1.2 — Обнаружение аттрактора методом Эйлера

На рисунке 1.2 мы можем видеть две линии, начинающиеся из точек $(0.1, 0.1)$ и $(2, 2)$. Эти линии сходятся сближаются к искомому предельному циклу системы.

2 Поиск точек бифуркаций

Найдя предельный цикл в системе (1.3), мы можем перейти к следующему этапу нашего исследования – определения всех значений параметра, при которых наблюдается данный цикл.

В силу того, что наша система рассматривается дифференциальным уравнением, поведение системы будет меняться плавно на промежутках, разделенных так называемыми, точками *бифуркации*(точки, в которых происходит изменение поведения системы).

Определение 2 Точка бифуркации - значение параметра системы, при котором наблюдается качественное изменение поведения системы.

Чтобы нам было удобно наблюдать изменение системы от параметра без перезапуска программы, мы обернем построения в функцию и добавим в нашу программу слайдер – бегунок, которым можно будет менять значение параметра ν .

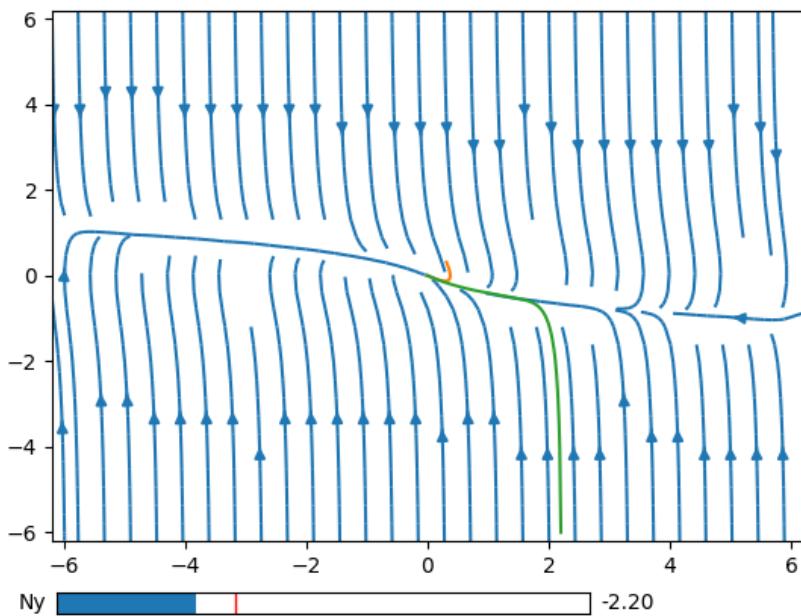


Рис. 2.1 — Стационарная точка системы при $\nu = -2.2$

Начиная с отрицательных значений (от -10) мы наблюдаем сильное стремление к центру координат – стационарной точки системы (График 2.1).

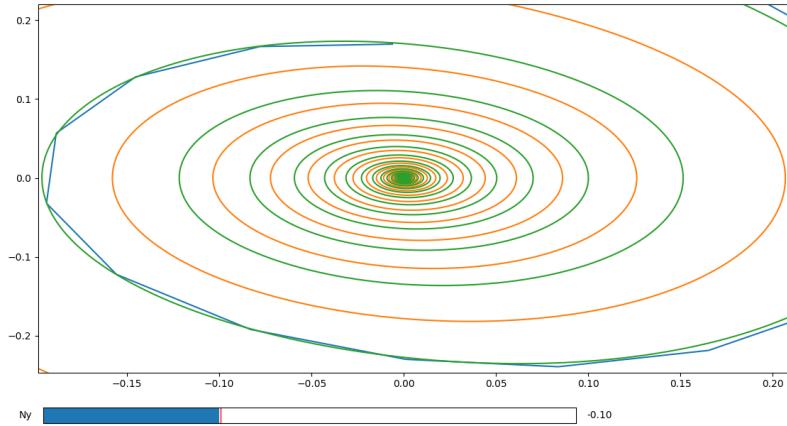


Рис. 2.2 – Стационарная точка системы при $\nu = -0.1$

При приближении параметра к нулю, поведение системы искривляется в овальную форму, но линии медленно сходятся к нулю (График 2.2).

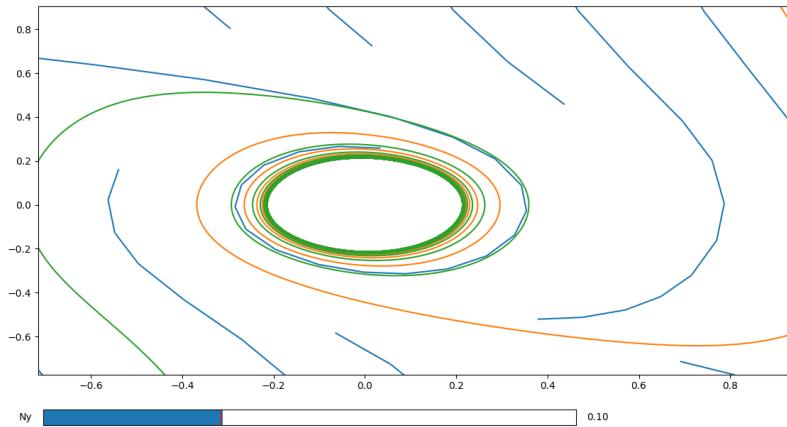


Рис. 2.3 – Появление цикла при $\nu = 0.1$

Как только мы переступаем нулевое значение параметра, наши траектории останавливаются значительно раньше – мы начинаем наблюдать знакомый нам предельный цикл, но в меньших размерах (График 2.3).

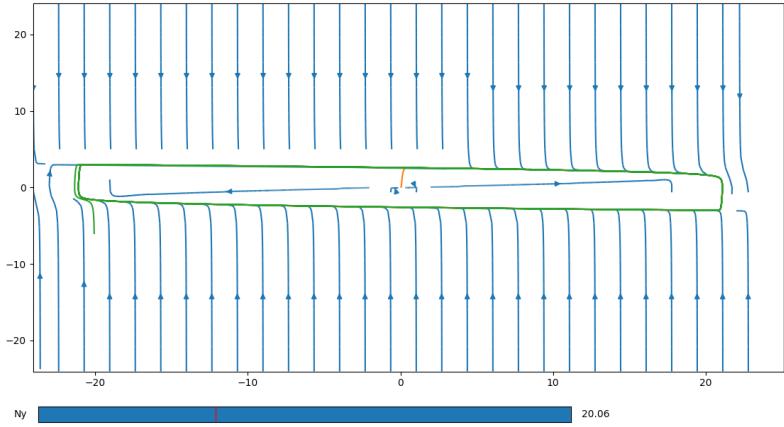


Рис. 2.4 — Расширение предельного цикла при увеличении параметра ($\nu = 20$)

Увеличивая ν дальше, остается наблюдать за ростом цикла (График 2.4).

Из полученных наблюдений можно выдвинуть гипотезу: на отрицательной полуоси исследуемая система сходится в стационарную точку; в положительной же оси наблюдается предельный цикл, который увеличивается в зависимости от параметра системы.

Стоит отметить, что наличие или отсутствие предельного цикла на границе ($\nu = 0$) мы выявить не можем, так как при приближении к параметру к нулю, чтобы быть уверенным в наличии стационарной точки или цикла, приходится увеличивать точность вычислений. В конце концов, когда точность увеличить не удается, нам остается только предполагать: толи линии сошлись к циклу, толи они не достигли нуля из-за недостаточного кол-ва шагов в методе Эйлера.

С учетом этого замечания, можно выдвинуть еще одну гипотезу: так как изменение поведения в системе происходит настолько плавно, что нам не удается уловить момент, когда мы наблюдаем стационарную точку, а когда предельный цикл. То есть мы можем говорить, что наблюдается *мягкая бифуркация системы*.

3 Исследование свойств предельного цикла

Следующим шагом в исследовании системы станет изучение свойств нашего предельного цикла при конкретном значении параметра (возьмем $\nu = 1$): нахождение его периода (от независимой переменной) и его форму. Данные свойства потребуются в следующих частях (4 и 5) для проверки характера его устойчивости.

Ставя численные эксперименты, значения могут получаться точные, но все же с погрешностью. Поэтому далее мы будем находить значение с точностью до 3-х знаков после запятой (т. е. наше значение должно расходится не более чем на $\epsilon = 0.5 * 10^{-4}$).

В программе 3 строится цикл методом точечных отображений Пуанкаре: выбирается точка на оси 0_{y_1} , от которой мы начинаем двигаться по траектории до тех пор, пока снова не пересечет ось. При приближении к нашему предельному циклу, точки будут сближаться все больше и больше. Таким образом будем считать траекторию предельным циклом, когда начальная и конечная точка сблизятся по обоим координатам ближе чем на ϵ . Периодом нашего цикла будет количество затраченных шагов ($i + 1$) помноженных на длину шага h . Как видно из работы программы, цикл имеет период $\omega = 6.663$.

Далее можно попытаться найти аналитическую форму данного цикла, но судя по графику 1.2 форма цикла не похожа на знакомые квадратичные функции и подбор аналитического вида кривой может оказаться трудной задачей. При этом мы не сможем достигнуть такой же точности, как наше поточечное решение, полученное методом Эйлера. Поэтому в следующих работах будем работать с массивами y_1 и y_2 , описывающие наш цикл.

Листинг 3 Поиск параметров системы

```
eps = 0.5 * 10 ** -4
y1_0, y2_0 = 0.72424, 0 # начальная точка

y1 = [y1_0]
y2 = [y2_0]
h = 0.0001
for i in range(100000):
    # итерация метода Эйлера
    y1.append(y1[i] + h*(y2[i]))
    y2.append(y2[i] + h*(-3*y2[i] ** 3 + ny * y2[i] - y1[i]))
    # проверка прихода в туже точку с погрешностью
    if np.abs(y1_0 - y1[i+1]) < eps and
       np.abs(y2_0 - y2[i+1]) < eps:
        # вывод результатов
        print("h={h}, i={i}, h*i={period}".format(
            h=h, i=i+1, period=h*(i+1)))
        return;
# Вывод программы
# h=0.0001, i=66633, h*i=6.663300000000004
```

4 Определение устойчивости через мультипликаторы системы

В этом разделе будет проведено исследование нашего предельного цикла на асимптотическую орбитальную устойчивость. Проверку будем проводить с помощью аналога теоремы Андронова-Витта, вычислив мультипликаторы системы первого приближения вдоль исследуемого предельного цикла.

Дадим необходимые для этого определения[4], относительно системы дифференциальных уравнений (4.1).

$$\dot{x} = f(x), \quad x \in R^n \quad f \in C[R^n] \quad (4.1)$$

Определение 3 Пусть $x = \eta(t)$ - решение системы (4.1), определенное при $t > 0$. **Положительной полутраекторией** решения назовем множество в фазовом пространстве:

$$L^+[\eta(\cdot)] = \{x \in R^n, \quad x = \eta(t), \quad t \geq 0\}.$$

Определение 4 Решение $\eta(t)$ системы (4.1) называется **орбитально устойчивым** при $t \rightarrow \infty$, если для любого $\epsilon > 0$ найдется $\delta > 0$ такое, что для всех других решений $x(t)$ системы (4.1) с условием $\|x(0) - \eta(0)\| < \delta$ выполняется $\rho(x(t), L^+[\eta(\cdot)]) < \epsilon$ для всех $t \geq 0$. Здесь $\rho(x, L)$ означает расстояние от точки x до множества L в пространстве R^n .

Определение 5 Орбитально устойчивое решение $\eta(t)$ называется **асимптотически орбитально устойчивым**, если существует $\Delta > 0$ такое, что для всех решений $x(t)$, удовлетворяющих соотношению $\|x(0) - \eta(0)\| < \Delta$, выполняется предельное соотношение $\rho(x(t), L^+[\eta(\cdot)]) \rightarrow 0$ при $t \rightarrow \infty$.

Из поставленных определений видна мотивация исследования решения на наличие асимптотической орбитальной устойчивости: подтвердив его, мы, увеличивая точность вычислений, будем уверены, что приближаемся к искомой траектории. В этом нам поможет, аналог теоремы Андронова-Витта.

Теорема 1 теорема Андронова-Витта. Если имеется периодическое решение автономной системы и его система первого приближения имеет два мультипликатора, один равный единице, а второй по-модулю меньше единицы, то полученное периодическое решение асимптотически орбитально устойчиво.

Мультипликаторами называются значения величин, полученных в результате алгоритма, изученного на лекционных занятиях:

- Рассмотрим систему $\dot{x} = F(x)$ и периодическое решение $\eta(t)$;
- Выразим линеаризованную систему $\dot{y} = F'(\eta(t))y$ вдоль данного решения;
- Вычислим ее вдоль периодического решения с н.у. $(1, 0)$ и $(0, 1)$;
- Получим матрицу монодромии $\Phi = (\phi_1, \phi_2)$, где ϕ_1, ϕ_2 - решения системы, полученные на предыдущем шаге.
- Собственные числа матрицы монодромии и будут мультипликаторами системы.

Рассмотрим нашу систему (4.2):

$$\begin{cases} \dot{y}_1 = y_2 = f_1(y_1, y_2) \\ \dot{y}_2 = -3y_2^3 + \nu y_2 - y_1 = f_2(y_1, y_2) \end{cases} \quad (4.2)$$

Посчитаем Якобиан нашей системы (уравнения (4.3)):

$$\frac{\partial f_1}{\partial y_1} = 0; \quad \frac{\partial f_1}{\partial y_2} = 1; \quad \frac{\partial f_2}{\partial y_1} = -1; \quad \frac{\partial f_2}{\partial y_2} = -9y_2^2 + \nu; \quad (4.3)$$

И строим линеаризованную систему вдоль цикла $\eta(t)$:

$$\begin{cases} \dot{Y}_1 = Y_2 \\ \dot{Y}_2 = -Y_1 + (-9\eta_2^2(t) + \nu)Y_2 \end{cases} \quad (4.4)$$

В листинге 4 мы, методом Эйлера решаем вычисленную систему и находим Собственные числа (Eigenvalues) матрицы монодромии. Как видно из вывода программы, вычисленные мультипликаторы удовлетворяют условиям теоремы с искомой точностью ($8.59 * 10^{-4}$ и 1), таким образом исследуемый цикл асимптотически орбитально устойчив.

Листинг 4 Вычисление мультиликаторов

```
def linear_form(y1_0, y2_0, cycle_y1, cycle_y2):  
    # Решение линеаризованной системы вдоль цикла  
    y1 = [y1_0]  
    y2 = [y2_0]  
    for i in range(len(cycle_y1)):  
        y1.append(y1[i] + h * (y2[i]))  
        y2.append(y2[i] + h * (-y1[i] +  
                               (-9*cycle_y2[i] ** 2 + ny) * y2[i]))  
    return [y1[-1], y2[-1]]  
  
cycle_y1, cycle_y2 = line(0.72424, 0) # вычисление цикла  
# решение линеаризированной системы с н. у. (1, 0) и (0, 1)  
f1 = linear_form(1, 0, cycle_y1, cycle_y2)  
f2 = linear_form(0, 1, cycle_y1, cycle_y2)  
f = np.array([ # Матрица монодромии  
    [f1[0], f2[0]],  
    [f1[1], f2[1]],  
])  
print("F-matrix:")  
print(f)  
# Вычисление собственных чисел матрицы монодромии  
p = np.linalg.eig(f)  
print("Eigenvalues:")  
print(p[0])  
# Вывод программы:  
# F-matrix:  
# [[ 8.56578243e-04 -3.73826069e-06]  
#  [ 5.05107751e-01  1.00012266e+00]]  
# Eigenvalues:  
# [ 8.58467858e-04  1.00012077e+00]
```

5 Определение устойчивости с помощью метода Пуанкаре

В этом разделе, рассмотрим другой метод определения асимптотической орбитальной устойчивости циклического решения, предложенный Анри Пуанкаре. Он определен только для двумерных автономных систем, что нам подходит.

Теорема 2 *Теорема Пуанкаре. периодическое решение системы будет асимптотически орбитально устойчивым тогда и только тогда, когда интеграл дивергенции вдоль решения будет меньше нуля.*

Таким образом, нам остается лишь взять вычисленные в уравнении (4.3) производные, составляющие дивергенцию системы. После чего вычислить интеграл (возьмем метод прямоугольников). Программа 5 производит данное вычисление, и результат подтверждает полученный в предыдущей работе результат: предельный цикл удовлетворяет критерию Пуанкаре, значит асимптотически орбитально устойчив.

Листинг 5 Вычисление интеграла от дивергенции системы

```
def integral_from_div(cycle_y1, cycle_y2):  
    """  
        Вычисление интеграла от дивергенции системы  
        вдоль цикла  
    """  
  
    sum = 0  
    for j in range(len(cycle_y1)):  
        sum += -9 * cycle_y2[j] ** 2 + ny  
    sum *= h  
    return sum  
  
# Основная программа  
cycle_y1, cycle_y2 = line(0.724197, 0)  
s = integral_from_div(cycle_y1, cycle_y2)  
print("Integral of the divergence: {}".format(s))  
# Вывод программы:  
# Integral of the divergence: -7.059887304427718
```

6 Влияние постоянного запаздывания

Начиная с этой главы мы познакомимся с разными видами дифференциальных уравнений с запаздыванием, называемые также функционально-дифференциальными уравнениями (ФДУ), которые являются обобщением обыкновенных дифференциальных уравнений.

Определение 6 *Дифференциальным уравнением с постоянным запаздыванием, мы будем называть уравнение вида:*

$$\begin{cases} \dot{x} = f(t, x(t), x(t - \tau)) \\ x(t_0) = x_0 \\ x_{t_0}(\cdot) = \{y_0(t); t \in [t_0 - \tau; t_0]\} \end{cases}$$

Посмотрев на него, можно выделить основные особенности от обычных дифференциальных уравнений, помимо зависимости от t и x , у нас появляется зависимость от значения фазовой переменной в момент $(t - \tau)$. Так же понадобилось задать значение фазовой переменной на $[t_0 - \tau; t_0]$ (обозначенное $x_{t_0}(\cdot)$), в противном случае само уравнение системы, например во время $t = 0$, нельзя трактовать однозначно.

Основной задачей, которую придется решить при моделировании такой системы, является то, как мы будем представлять влияние значения фазовой переменной $x(t - \tau)$. В случае постоянного запаздывания все достаточно просто: мы можем выбрать шаг h таким образом, что величина $\frac{\tau}{h}$ будет целым m . Тогда нам достаточно будет брать x_{i-m} член из нашего дискретной последовательности (6.1).

$$x^{i+1} = x^i + h * f(t^i, x^i, x^{i-m}) \quad (6.1)$$

Для того, чтобы пронаблюдать как постоянное запаздывание влияет на предельный цикл, введем в нашу систему слагаемое $\alpha * y_1(t - \tau)$ (Ур. (6.2)). При параметре $\alpha = 0$ оно не влияет на систему, следовательно мы будем наблюдать наш предельный цикл. Постепенно изменяя этот параметр, мы будем наблюдать на изменения характера системы.

$$\begin{cases} \dot{y}_1 = y_2 + \alpha * y_1(t - \tau) \\ \dot{y}_2 = -3y_2^3 + \nu y_2 - y_1 \end{cases} \quad (6.2)$$

Итерационный метод будет выглядеть соответственно (6.1):

$$\begin{cases} y_1^{i+1} = y_1^i + h(y_2^i + \alpha * y_1^{i-m}) \\ y_2^{i+1} = y_2^i + h(-3(y_2^i)^3 + \nu y_2^i - y_1^i) \end{cases} \quad (6.3)$$

В прошлых экспериментах мы рассматривали два решения: от точки $(0.1, 0.1)$ и от точки $(2, 2)$. Теперь же нам еще нужна предыстория нашей системы, поэтому мы просто возьмем ее константной (ур. (6.4) и (6.5)).

$$\begin{cases} y_{1t_0} = 0.1, t \in [-\tau, 0] \\ y_{2t_0} = 0.1, t \in [-\tau, 0] \end{cases} \quad (6.4)$$

$$\begin{cases} y_{1t_0} = 2, t \in [-\tau, 0] \\ y_{2t_0} = 2, t \in [-\tau, 0] \end{cases} \quad (6.5)$$

Стоит отметить, модернизацию нашей программы в техническом виде, с помощью слайдера оказалось неудобным отлаживать подбор необходимого параметра. Поэтому, чтобы упростить эксперименты и не перезапускать программу, вместо слайдера был поставлен TextBox - поле для ввода текста, куда мы можем ввести любое, необходимое нам, значение (пересчет графика произойдет при нажатии на клавишу Enter). При желании, с подробностями использования этого виджета можно ознакомиться в приложении.

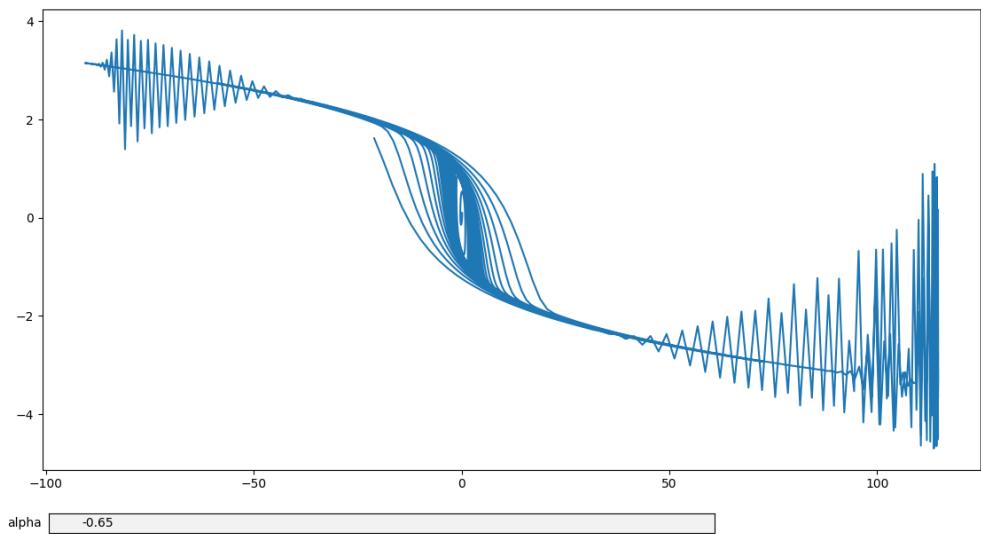


Рис. 6.1 — ($\alpha = -0.65$) Решение от точки $(0.1, 0.1)$ хаотично расходится от центра. Решение от точки $(2, 2)$ не смогло построится

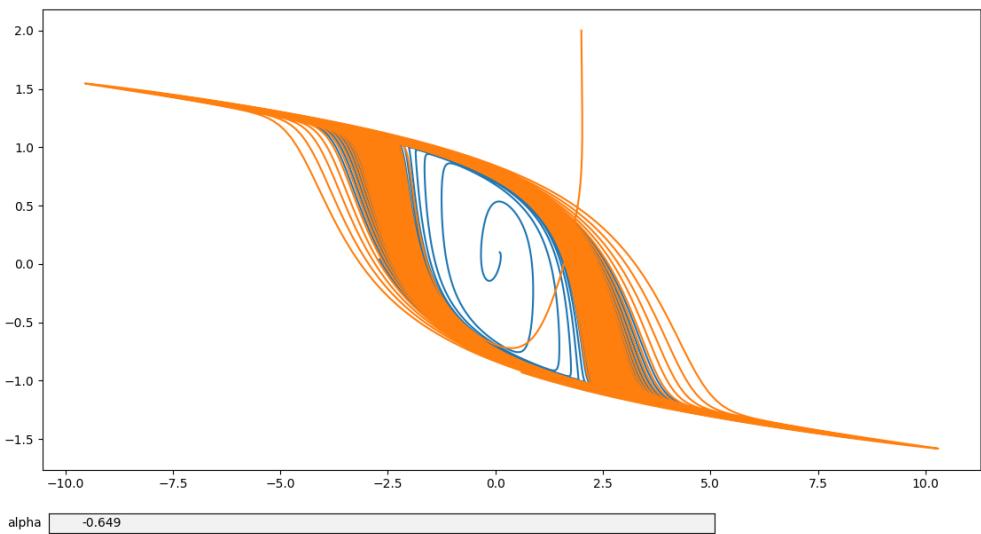


Рис. 6.2 — ($\alpha = -0.649$) Оба решения построились. Отчетливо наблюдается жесткость системы, цикл сохранился, но об изолированности говорить не приходится

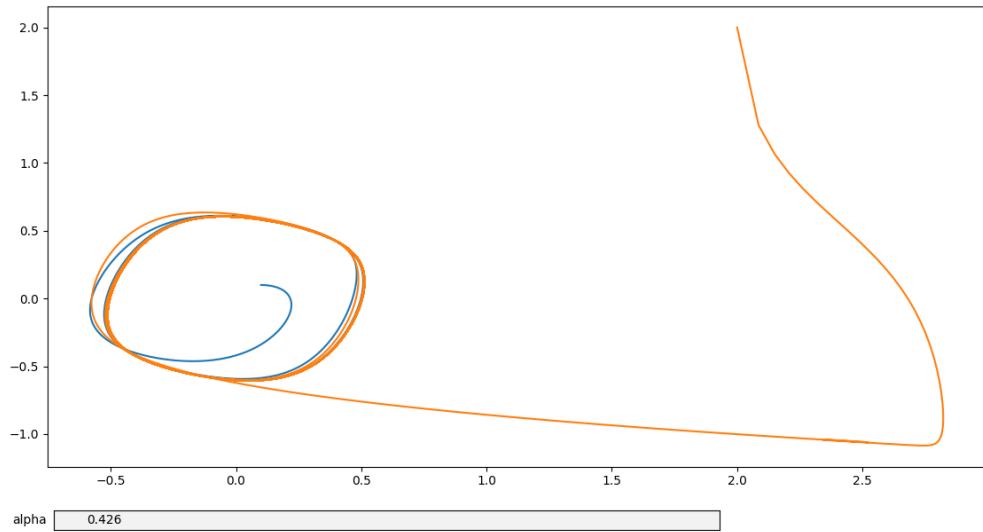


Рис. 6.3 — ($\alpha = 0.426$) Решение все еще сходится к циклу, но постепенно разбалтывается

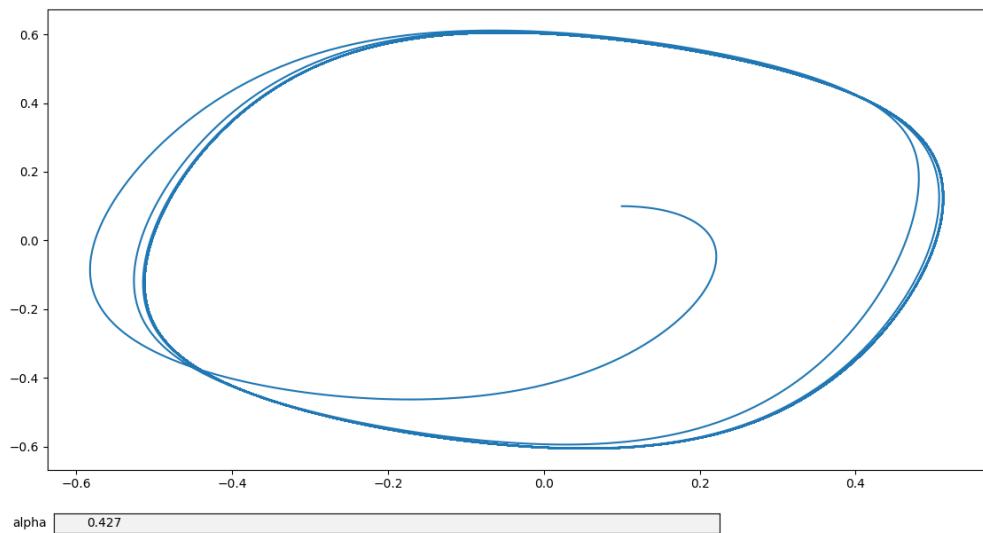


Рис. 6.4 — ($\alpha = 0.427$) Второе решение перестало моделироваться.
Первое все еще сходится к циклу

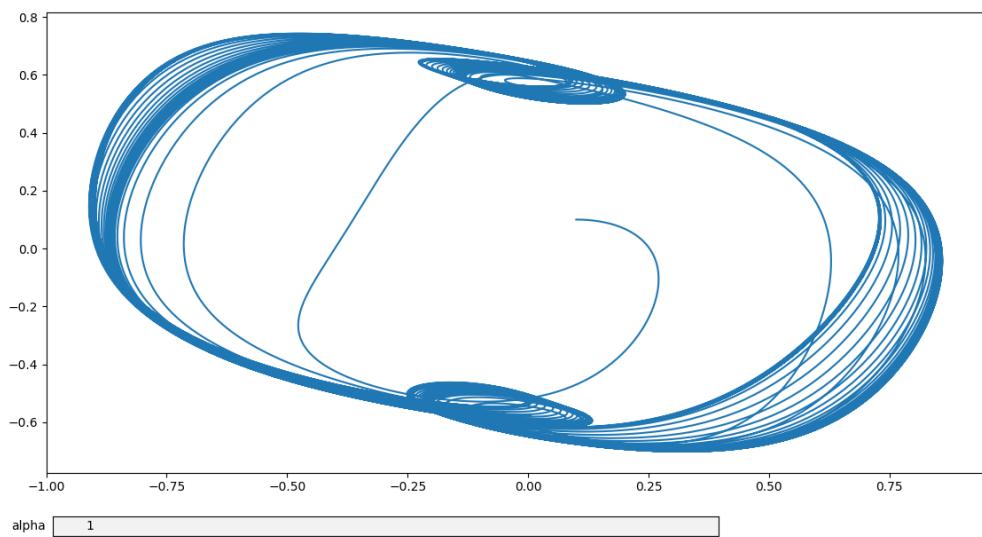


Рис. 6.5 — ($\alpha = 1$) У первого решения появляются две петельки и траектория становится более хаотична

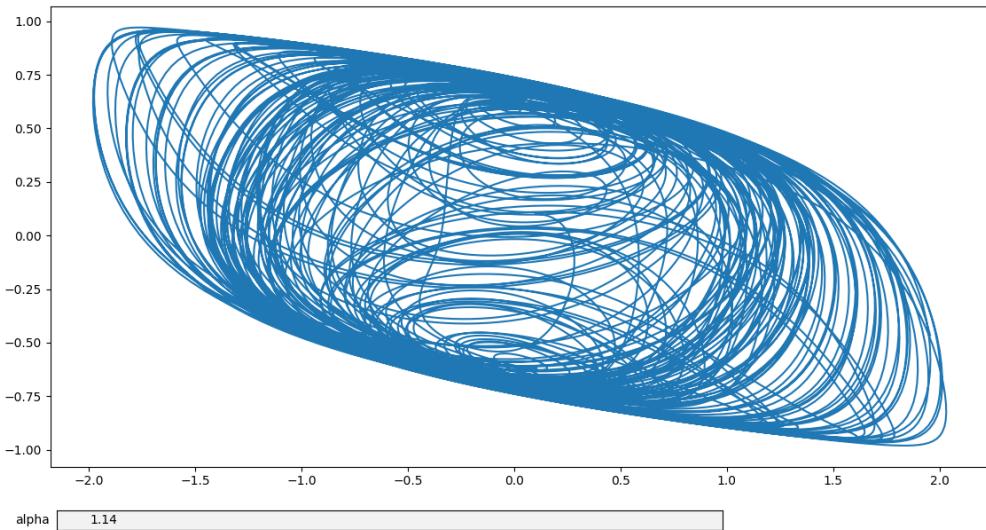


Рис. 6.6 — ($\alpha = 1.24$) Первое решение хаотично изменяется в пределах цикла

Нам не удавалось сильно изменить параметр α : сразу же начинали расходиться решения (или решение); проявлялось, своего рода, хаотичное поведение; исчезали циклы. То есть мы увидели, как сильно влияет на поведение системы даже самая простая, в плане понимания, задержка.

При добавлении подобного слагаемого во второе уравнение можно про наблюдать другие бифуркции. При желании, эти эксперименты можно посмотреть в Приложении Б.1.

7 Влияние переменного запаздывания

Следующим видом запаздывания, которое мы рассмотрим, будет переменное. Его основное отличие заключается в том, что величина задержки, на которую мы смотрим в прошлое, меняется от текущего момента времени. Например, в нашем уравнении (7.1) величина задержки будет $\tau * \sin(\frac{2t}{T})$, тут τ - это максимальная величина на которую может потребоваться заглянуть в историю, и соответственно в зависимости от $\sin(\frac{2t}{T})$ мы будем брать то близкие к текущему моменту времени значения фазовой переменной, то далекие. Коэффициент у синуса $2/T$ был выбран для того, что бы период изменения был пропорционален рассматриваемому отрезку времени.

$$\begin{cases} \dot{y}_1 = y_2 + \alpha * y_1(t - \tau * \sin(\frac{2t}{T})) \\ \dot{y}_2 = -3y_2^3 + \nu y_2 - y_1 \end{cases} \quad (7.1)$$

Для моделирования будут использоваться те же начальные значения, что и в разделе 6, а также использоваться метод Эйлера с *кусочно постоянной интерполяцией*.

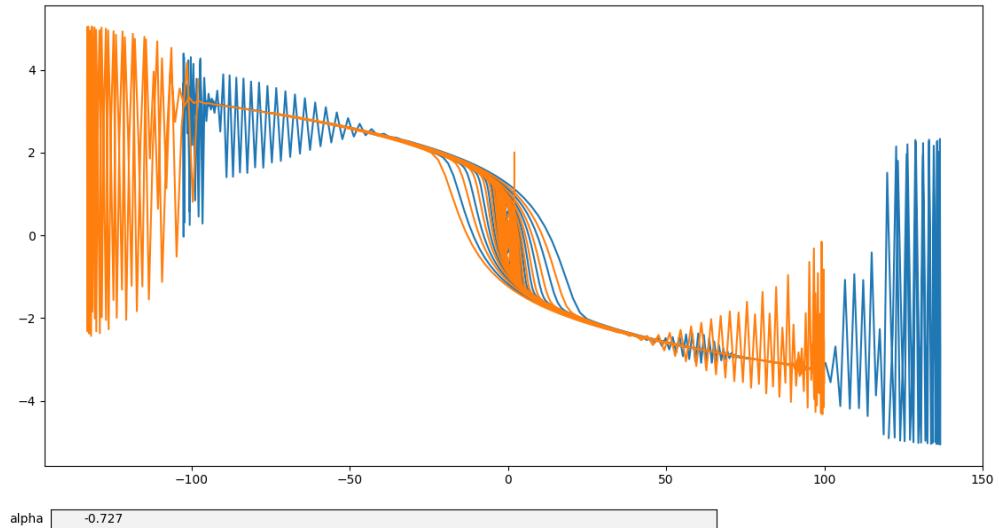


Рис. 7.1 — ($\alpha = -0.727$) Решения начинают сильно расходиться по краям

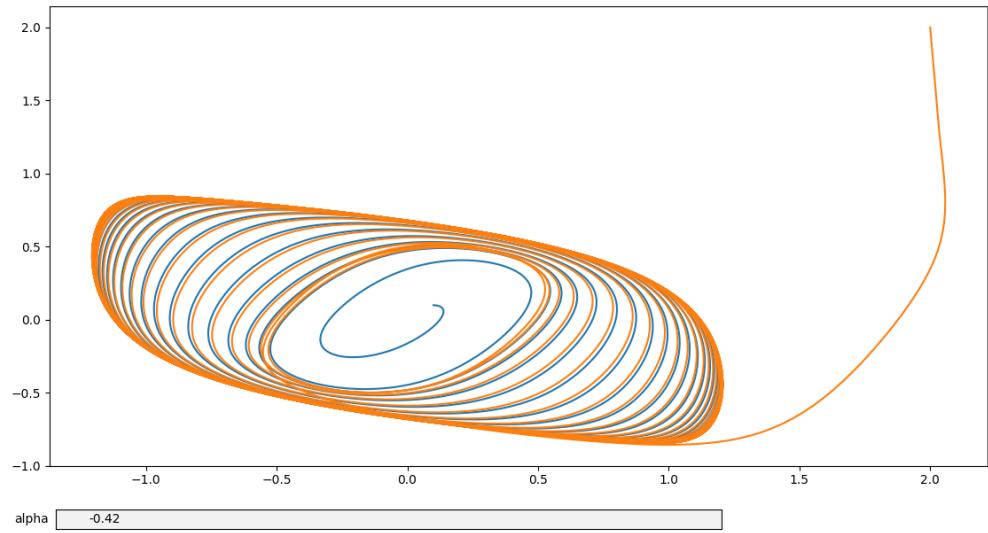


Рис. 7.2 — ($\alpha = -0.42$) Цикл искривляется в спираль

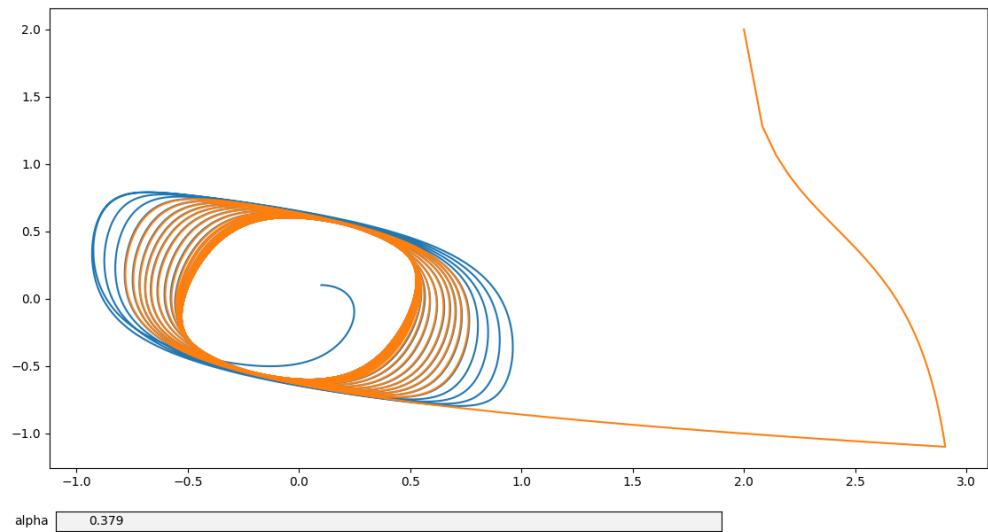


Рис. 7.3 — ($\alpha = 0.379$) Оба решения построились. Спиралевидно сходятся внутри цикла

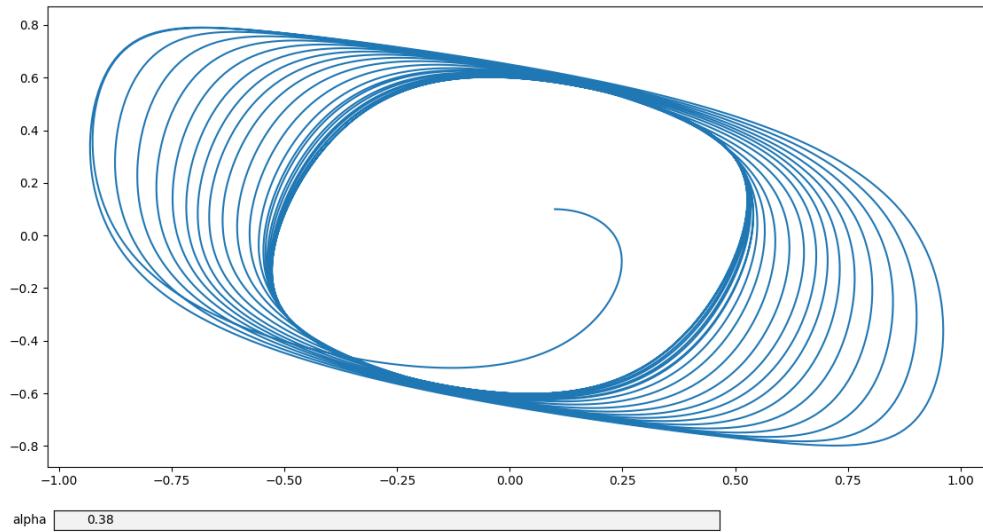


Рис. 7.4 — ($\alpha = 0.38$) Второе решение перестало строится. Первое ведет себя спиралевидно

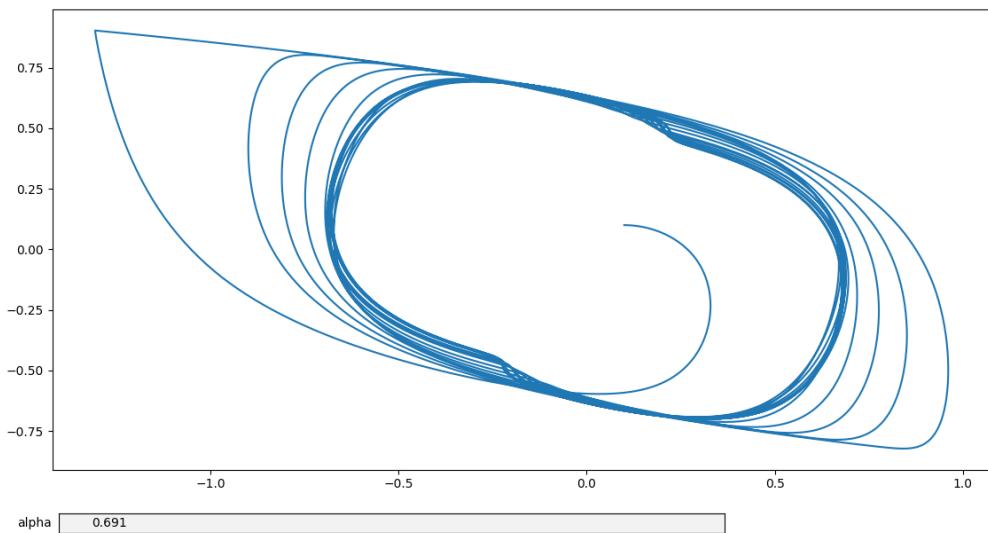


Рис. 7.5 — ($\alpha = 0.691$) У второго решения видны жесткие изменения системы. Дальше второе решение перестало строиться

Таким образом, при добавлении к системе переменного запаздывания нам даже не удалось посмотреть влияния его на систему при $\alpha \geq 1$, то есть в нашем случае его воздействие на систему оказалось гораздо сильнее, чем постоянная задержка.

В приложении Б.2 можно посмотреть, как такое запаздывание влияет при добавлении его ко второй переменной системы.

8 Влияние распределенного запаздывания

Последним видом запаздывания, которое мы рассмотрим будет распределенное запаздывание. Его характерной особенностью является то, что на поведение системы так или иначе влияет вся предыстория системы на отрезке $[t - \tau; t]$ и задается это влияние интегралом (Как например в нашем опыте, ур. (8.1)).

$$\begin{cases} \dot{y}_1 = y_2 + \alpha * \int_{t-\tau}^t y_1^2(s)ds \\ \dot{y}_2 = -3y_2^3 + \nu y_2 - y_1 \end{cases} \quad (8.1)$$

Считать интеграл мы будем методом прямоугольников, с одной дополнительной оптимизацией: так как отрезок интегрирования на каждом шаге меняется не сильно, а лишь сдвигается на шаг h , то мы можем не пересчитывать интеграл полностью: достаточно просто вычесть последнее слагаемое в сумме, и добавить новое, появившееся за итерацию. Тогда вычислительная сложность моделирования уменьшится на порядок. В остальном, начальные значения системы и метод моделирования остается таким же.

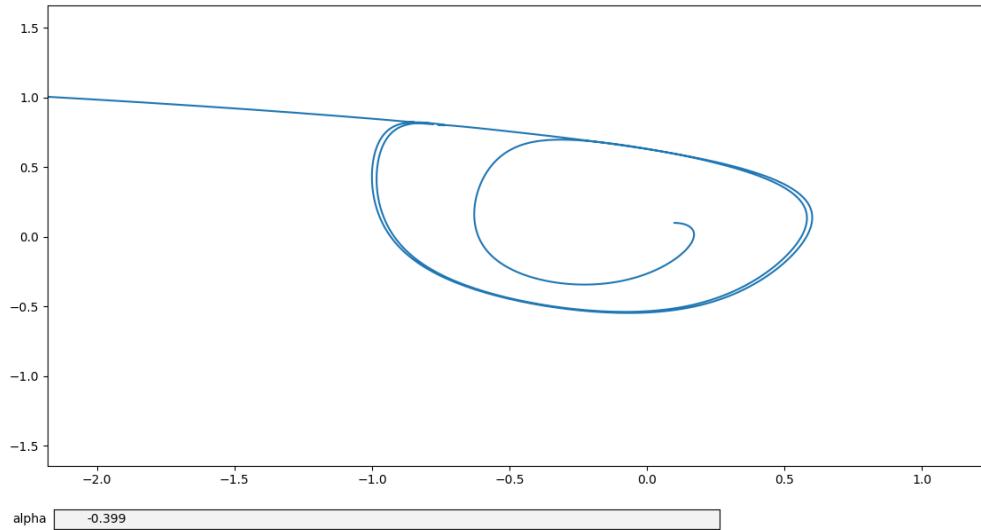


Рис. 8.1 — ($\alpha = -0.399$) Второе решение не построилось. Первое, сделав пару циклов, ушло в минус бесконечность по оси Oy_1

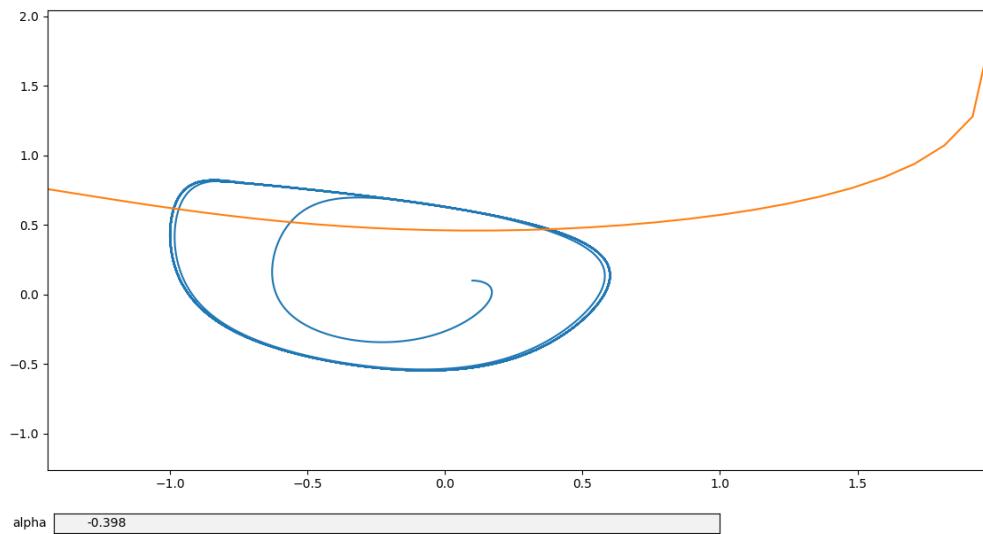


Рис. 8.2 — ($\alpha = -0.398$) Первое решение остается в пределах цикла.
Второе расходится

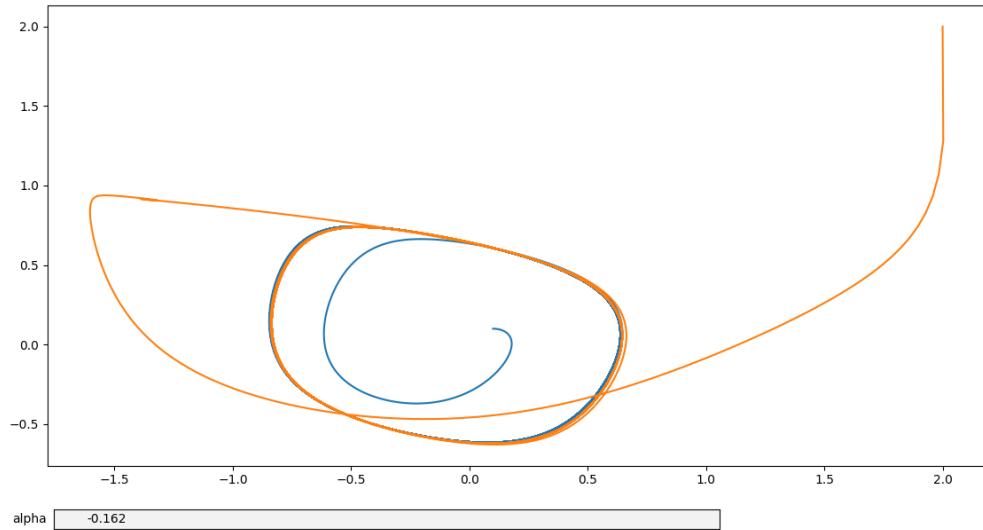


Рис. 8.3 — ($\alpha = -0.162$) Оба решения стабилизируются вокруг цикла

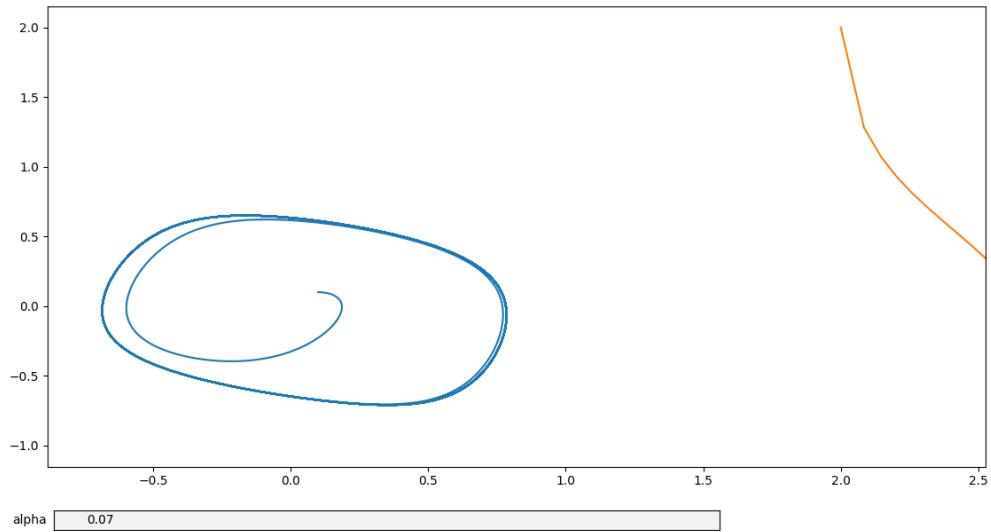


Рис. 8.4 — ($\alpha = 0.07$) Небольшое положительное влияние, а второе
решение уже разошлось

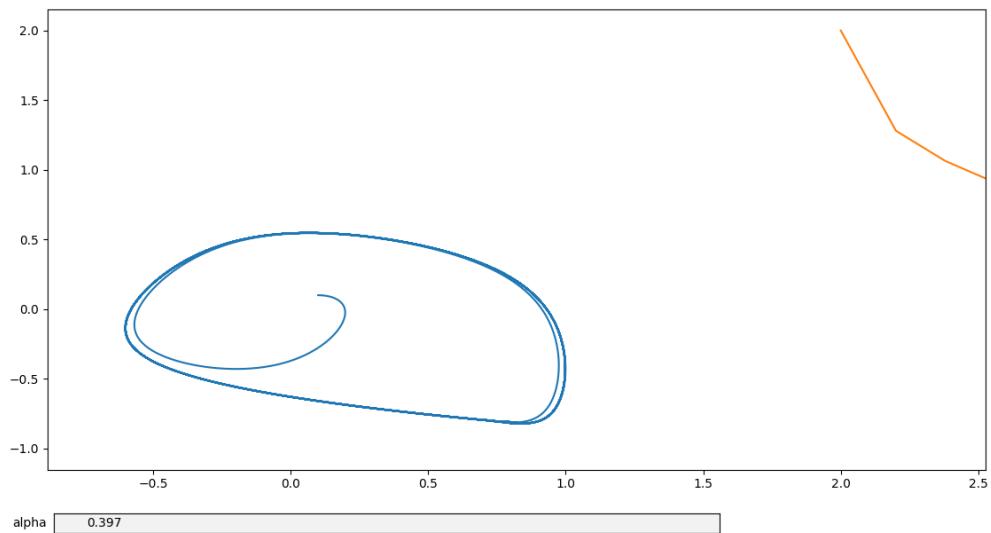


Рис. 8.5 — ($\alpha = 0.397$) Цикл первого решения все еще строится, хоть и
деформировался

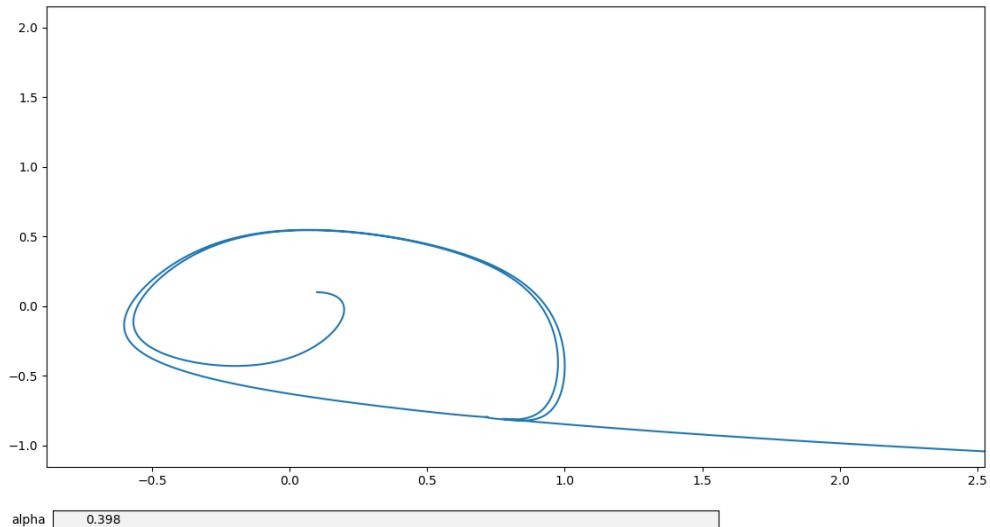


Рис. 8.6 — ($\alpha = 0.398$) Второе решение не построилось. Первое ушло в плюс бесконечность по оси Oy_1

В данном эксперименте можно увидеть одну необычную закономерность: относительно параметра α первое решение ведет себя очень похоже, что при -0.398 , что при 0.398 . За исключением того, что петелька зеркально переворачивается и начинает расходиться в противоположное направление.

Также можно увидеть, что у в этом наборе опытов распределенное запаздывание быстрее всех повлияло на предельный цикл: при постоянном запаздывании решение расходилось при $\alpha > 1.24$, при переменном $\alpha > 0.691$, при распределенном же разошлось при $\alpha > 0.398$.

Но стоит заметить, что этот вывод характерен только для этой серии экспериментов: при изменении системы или формы задержки все может поменяться непредсказуемым образом. Как пример этого, можно сравнить эти результаты с экспериментами в приложении Б: там влияние переменной задержки при отрицательных α не разрушает цикл системы очень долго.

9 Влияние случайного шума

Последним видом моделируемых систем, с которым мы познакомимся в этой работе, будет моделирование *стохастического дифференциального уравнения*[5]. Оно описывает систему, на работу которой в каждый момент времени может оказывать влияние случайное искажение (шум). Обычно, в качестве шума рассматривают реализацию выборки стандартного нормального распределения, поэтому в наших экспериментах мы будем использовать его.

В том виде, в котором описывались предыдущие дифференциальные уравнения, мы описывать данное уравнение не можем, однако такое уравнение хорошо описывается через интегральную форму (9.1).

$$x(t) = x_0 + \int_{t_0}^t f(s, x(s))ds + \int_{t_0}^t \sigma(W, x(W))dW \quad (9.1)$$

Второе слагаемое уравнения (9.1) является интегралом по Винеровскому процессу, который как раз и описывает приращение ошибки (шума) в системе. От такого вида уравнения можно перейти к СДУ в виде дифференциалов (9.2).

$$dx = f(t, x)dt + \sigma(t, x(t))dW \quad (9.2)$$

Для нашей системы, такое уравнение примет вид (9.3).

$$\begin{cases} dy_1 = y_2 dt + \sigma * dW \\ dy_2 = (-3y_2^3 + \nu y_2 - y_1)dt \end{cases} \quad (9.3)$$

В этом уравнении мы добавляем шум к изменению первой координаты системы, Виноровский процесс описывается через стандартное нормальное распределение, а за счет коэффициента σ мы сожем влиять на среднеквадратичное отклонение данного процесса, а следовательно, и на силу влияния данного процесса на систему.

Для моделирования такой системы используется другая модификация метода Эйлера - *метод Эйлера-Марайамы*. Для нашей системы данный метод будет выглядеть так:

$$\begin{cases} y_1^{i+1} = y_1^i + hy_2^i + \sigma * \sqrt{h}W^i \\ y_2^{i+1} = y_2^i + h(-3(y_2^i)^3 + \nu y_2^i - y_1^i) \end{cases} \quad (9.4)$$

W^i - реализация генератора случайных чисел стандартного нормального распределения. Начальные значения в текущем эксперименте не принципиальны. Поэтому были взяты предыдущие точки $(0.1, 0.1)$ и $(2, 2)$, к которым добавилось случайное воздействие σW^0 .

Определение 7 Метод моделирующий стохастическое уравнение сходится с порядком p в сильном смысле, если $\exists C : M(|X(T) - x^N|) \leq Ch^p$, где $X(t)$ - настоящее решение данного уравнения, а x_i - решение, полученное с помощью метода.

Определение 8 Метод моделирующий стохастическое уравнение сходится с порядком p слабо, если $\exists C : M(\int_{t_0}^T (X(t) - x^i)^2 dt)^{\left(\frac{1}{2}\right)} \leq Ch^p$

Теорема 3 Метод Эйлера-Мараийамы сходится, как в сильном, так и в слабом смысле с порядком $p = \frac{1}{2}$.

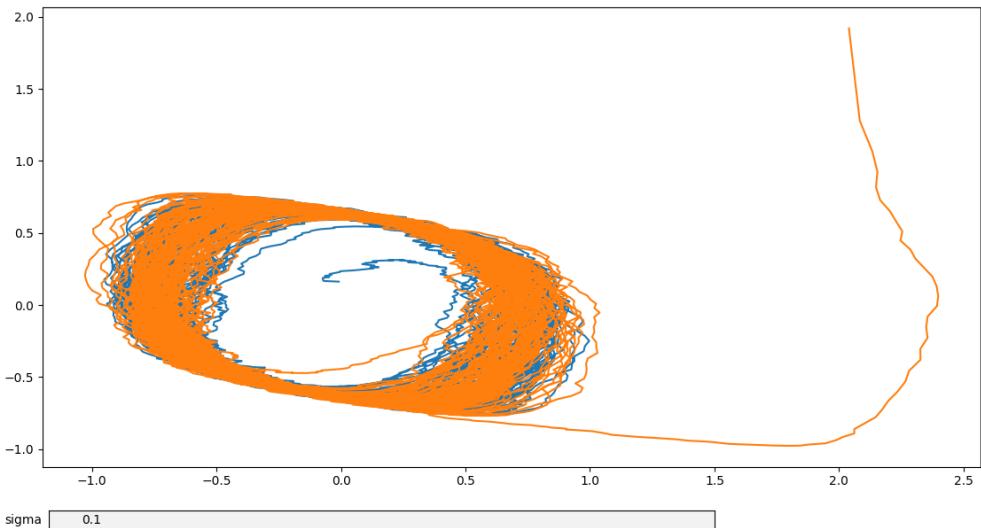


Рис. 9.1 — ($\sigma = 0.1$) видно, что цикл сохранил форму, но контур стал размываться вдоль оси Oy_1

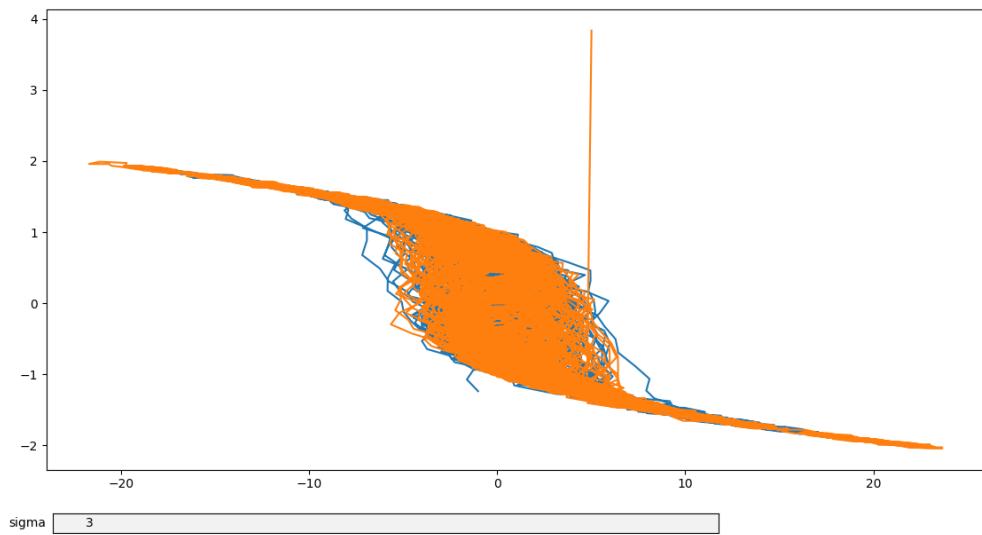


Рис. 9.2 — ($\sigma = 3$) Цикл напоминает размытое пятно, но значения решения не расходятся

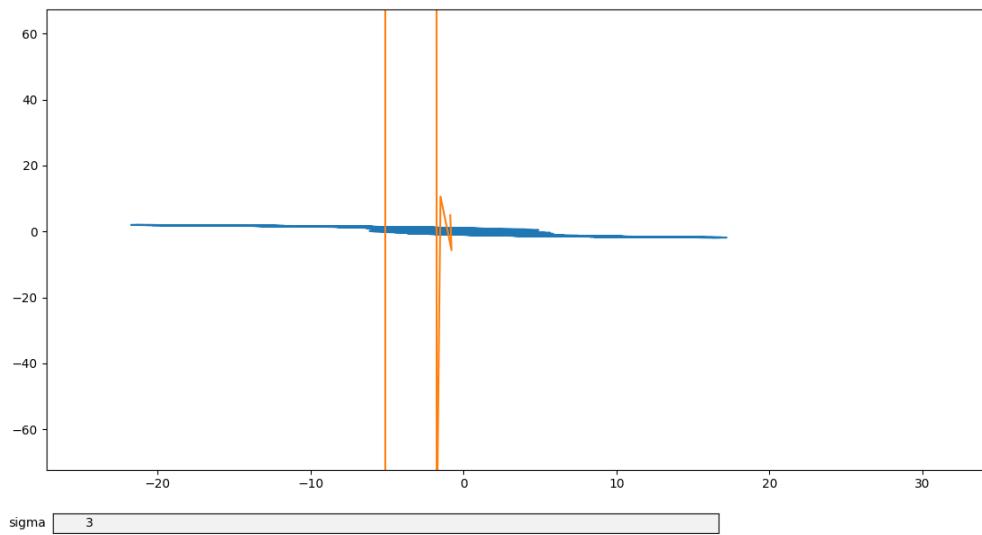


Рис. 9.3 — ($\sigma = 3$) Значение σ не поменялось, но при определенной выборке решение разошлось

Таким образом видно, что при добавлении шума в небольшом количестве, поведение системы остается в пределах известного поведения, хоть и становится слишком размытым. При сильном увеличении влияния процесс может, как разойтись, так и нет, в зависимости от реализации выборки случайной величины.

Заключение

В ходе выполнения работы, была смоделирована система обыкновенных дифференциальных уравнений(ОДУ), на примере которой мы рассмотрели понятие предельного цикла, а также его свойств, обсужденных на лекциях, построенных на пособии [4].

В разделах 4 и 5 мы рассмотрел понятие асимптотической орбитальной устойчивости и проверили это свойство на рассматриваемом предельном цикле с помощью теорем Андронова-Ватта и метода Пуанкаре соответственно.

После этого, были рассмотрено моделирование функциональных дифференциальных уравнений (ФДУ)[6], путем модификации нашей системы за счет добавления разнообразных видов задержек (постоянная, переменная, распределенная) и сравнения их влияния на предельный цикл системы, что сильно меняло его свойства и выходило за пределы теории ОДУ.

И последним видом уравнений, которые были смоделированы в этой работе, были стохастические дифференциальные уравнения, в которых на исследуемый цикл влиял случайный шум на каждой итерации.

Список использованных источников

1. Numpy - библиотека для научных вычислений на языке Python. <http://www.numpy.org/>.
2. Matplotlib - библиотека для визуализации данных. <http://matplotlib.org/>.
3. Функция streamplot. http://matplotlib.org/examples/images_contours_and_fields/streamplot_demo_features.html.
4. В.Г.Пименов. Избранные главы дифференциальных уравнений. Урал. ун-т, 2003. 83 с.
5. Д.Ф. Кузнецов. Численное моделирование стохастических дифференциальных уравнений и стохастических интегралов. С.Петербург. Наука, 1999.
6. А.В.Ким В.Г.Пименов. i-Гладкий анализ и численные методы решения функционально-дифференциальных уравнений. М., 2004. 256 с.

Приложение А Исходный код программ

A.1 Поиск предельного цикла

```
import matplotlib.pyplot as plt
import numpy as np

# Параметр системы
nu = 1

# создание сетки 100x100 точек в области [-3;3]x[-3;3]
Y, X = np.mgrid[-3:3:100j, -3:3:100j]

# вычисление фазовых векторов на сетке
Y1 = Y
Y2 = -3 * Y ** 3 + nu * Y - X

# построение фазового портрета
fig, ax = plt.subplots()
plt.streamplot(X, Y, Y1, Y2)

# подпись осей на графике
ax.set_xlabel("y1")
ax.set_ylabel("y2")

# функция построение кривой методом Эйлера
def line(y1_0, y2_0):
    y1 = [y1_0]
    y2 = [y2_0]
    h = 0.01 # длина шага
    for i in range(2000): # 2000 - количество итераций
        y1.append(y1[i] + h*(y2[i]))
        y2.append(y2[i] +
                  h * (-3*y2[i] ** 3 + nu*y2[i] - y1[i]))
```

```
# отображение кривой на графике
ax.plot(y1, y2)

# построение двух кривых, начинающихся внутри и
# вне предположенного предельного цикла
line(0.1, 0.1)
line(2, 2)

# показать построенные графики
plt.show()
```

A.2 Исследование точек бифуркации системы

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button

ny0 = 0 # Первоначальное значения параметра

fig, ax = plt.subplots()
plt.subplots_adjust(bottom=0.15)

# подпись осей на графике
ax.set_xlabel("y1")
ax.set_ylabel("y2")

# функция построение кривой методом Эйлера
def line(y1_0, y2_0, ny):
    y1 = [y1_0]
    y2 = [y2_0]
    h = 0.003
    for i in range(50000):
        y1.append(y1[i] + h * (y2[i]))
        y2.append(y2[i] +
                  h * (-3*y2[i] ** 3 + ny * y2[i] - y1[i]))
    ax.plot(y1, y2)

def streamplot(ny):
    # увеличиваем параметры сетки в зависимости от модуля ny
    b = (4 + np.abs(ny)) # границы сетки
    c = (100 + np.abs(ny)) * 1j # число точек разбиения
    Y, X = np.mgrid[-b:b:c, -b:b:c]
    Y1 = Y
    Y2 = -3 * Y ** 3 + ny * Y - X
    ax.streamplot(X, Y, Y1, Y2)
```

```

# перерисовка графика в зависимости от параметра
# выводится фазовый портрет и две линии
def update_plot(ny):
    ax.cla()
    streamplot(ny)
    line(1. / (np.abs(ny) + 1), 1. / (np.abs(ny) + 1), ny)
    line(-ny, -6, ny)
    fig.canvas.draw_idle()

# Слайдер - чтобы менять параметр без перезапуска программы
axfreq = plt.axes([0.13, 0.05, 0.55, 0.03])
slider = Slider(axfreq, 'Ny', -10.0, 20.0, valinit=ny0)
slider.on_changed(update_plot)

step = 0.1 # шаг изменения параметра по нажатии клавиши
def _keyboard_handler(event):
    # выход при нажатии escape
    if event.key == 'escape':
        plt.close('all')
    # уменьшение параметра при нажатии стрелки "вниз"
    elif event.key == 'down':
        slider.set_val(slider.val - step)
    # увеличение параметра при нажатии стрелки "вверх"
    elif event.key == 'up':
        slider.set_val(slider.val + step)

fig.canvas.mpl_connect('key_press_event', _keyboard_handler)

update_plot(ny0)
plt.show()

```

A.3 Исследование параметров найденного предельного цикла

```
import numpy as np

ny = 1

y1_0 = 0.72424
y2_0 = 0

y1 = [y1_0]
y2 = [y2_0]
eps = 0.5 * 10 ** -4

# метод Эйлера с большей точностью
h = 0.0001
for i in range(100000):
    y1.append(y1[i] + h*(y2[i]))
    y2.append(y2[i] +
               h * (-3*y2[i] ** 3 + ny * y2[i] - y1[i]))
    # считаем цикл завершенным, когда покоординатно
    # приблизились к начальной точке ближе, чем на eps
    if (np.abs(y1_0 - y1[i+1]) < eps and
        np.abs(y2_0 - y2[i+1]) < eps):
        print("h={h}, i={i}, h*i={period}".format(
            h=h, i=i+1, period=h*(i+1)))
    )
    break
```

A.4 Проверка устойчивости теоремой о мультипликаторах

```
import numpy as np

ny = 1      # Параметр системы
h = 0.0001 # Общий шаг метода Эйлера

def line(y1_0, y2_0):
    """
    Вычисление системы методом Эйлера от точки (y1_0, y2_0)
    за счет правильно подобранных в прошлой работе координат
    функция вычислит близкую к предельному циклу траекторию
    """
    y1 = [y1_0]
    y2 = [y2_0]
    eps = 0.5 * 10 ** -4
    for i in range(100000):
        y1.append(y1[i] + h * y2[i])
        y2.append(y2[i] +
                  h*(-3*y2[i] ** 3 + ny * y2[i] - y1[i]))
        if (np.abs(y1_0 - y1[i+1]) < eps and
            np.abs(y2_0 - y2[i+1]) < eps):
            return (y1, y2)
    raise Exception("Cycle not found")

def linear_form(y1_0, y2_0, cycle_y1, cycle_y2):
    """
    Решение линеаризованной системы вдоль цикла
    """
    y1 = [y1_0]
    y2 = [y2_0]
    for i in range(len(cycle_y1)):
```

```

y1.append(y1[i] + h * (y2[i]))
y2.append(y2[i] + h * (-y1[i] +
(-9*cycle_y2[i] ** 2 + ny) * y2[i]))
return [y1[-1], y2[-1]]

# Начало программы
# вычисление поточечного описания предельного цикла
cycle_y1, cycle_y2 = line(0.72424, 0)

# решение линеаризированной системы
# с начальными условиями (1, 0) и (0, 1)
f1 = linear_form(1, 0, cycle_y1, cycle_y2)
f2 = linear_form(0, 1, cycle_y1, cycle_y2)

# Матрица монодромии
f = np.array([
    [f1[0], f2[0]],
    [f1[1], f2[1]],
])
print("F-matrix:")
print(f)

# Вычисление собственных чисел матрицы монодромии
p = np.linalg.eig(f)
print("Eigenvalues:")
print(p[0])

```

A.5 Проверка устойчивости цикла методом Пуанкаре

```
import numpy as np

ny = 1      # Параметр системы
h = 0.0001 # шаг метода Эйлера
            # и основание прямоугольников при интегрировании

def line(y1_0, y2_0):
    """
    Вычисление системы методом Эйлера от точки (y1_0, y2_0)
    за счет правильно подобранных в предыдущей работе координат
    функция вычислит близкую к предельному циклу траекторию
    """
    y1 = [y1_0]
    y2 = [y2_0]
    eps = 0.5 * 10 ** -4
    for i in range(100000):
        y1.append(y1[i] + h * (y2[i]))
        y2.append(y2[i] +
                   h*(-3*y2[i] ** 3 + ny * y2[i] - y1[i]))
        if (np.abs(y1_0 - y1[i + 1]) < eps and
            np.abs(y2_0 - y2[i + 1]) < eps):
            return (y1, y2)

    raise Exception("Cycle not found")

def integral_from_div(cycle_y1, cycle_y2):
    """
    Вычисление интеграла от дивергенции системы
    вдоль цикла
    """
    pass
```

```
sum = 0
for j in range(len(cycle_y1)):
    sum += -9 * cycle_y2[j] ** 2 + ny
sum *= h
return sum

# Основная программа
cycle_y1, cycle_y2 = line(0.72424, 0)
s = integral_from_div(cycle_y1, cycle_y2)
print("Integral of the divergence: {}".format(s))
```

A.6 Влияние постоянного запаздывания

```
# Лаб. работа 6: влияние постоянного запаздывания на систему
import copy
import matplotlib.pyplot as plt
from matplotlib.widgets import TextBox

figure, ax = plt.subplots()
plt.subplots_adjust(bottom=0.15)
# подпись осей на графике
ax.set_xlabel("y1")
ax.set_ylabel("y2")

ny = 1
alpha0 = 0.01

def line(y1_0, y2_0, alpha):
    # решение задачи методом Эйлера
    y1 = copy.deepcopy(y1_0)
    y2 = copy.deepcopy(y2_0)
    h = 0.03
    for i in range(len(y1_0)-1, 20000):
        # Запаздывание влияет на y1
        y1.append(y1[i] + h * (y2[i] +
                               alpha * y1[i - len(y1_0) + 1]))
        y2.append(y2[i] +
                  h * (-3*y2[i] ** 3 + ny * y2[i] - y1[i]))

    # Запаздывание влияет на y2
    # y1.append(y1[i] + h * (y2[i]))
    # y2.append(y2[i] +
    #           h * (-3*y2[i] ** 3 + ny * y2[i] - y1[i] +
```

```

#           alpha * y1[i - len(y1_0) + 1]))
ax.plot(y1, y2)

def update_plot(strAlpha):
    # перерисовка графика в зависимости от параметра
    alpha = float(strAlpha)
    ax.cla()
    line(
        [0.1 for a in range(100)],
        [0.1 for a in range(100)],
        alpha)
    line(
        [2 for a in range(100)],
        [2 for a in range(100)],
        alpha)
    figure.canvas.draw_idle()

axfreq = plt.axes([0.13, 0.05, 0.55, 0.03])
textBox = TextBox(axfreq, 'alpha', initial=str(alpha0))
textBox.on_submit(update_plot)

def _keyboard_handler(event):
    # выход при нажатии escape
    if event.key == 'escape':
        plt.close('all')

figure.canvas.mpl_connect('key_press_event', _keyboard_handler)

update_plot(alpha0)
plt.show()

```

A.7 Влияние переменного запаздывания

```
# Лаб. работа 7: влияние переменного запаздывания на систему
import copy
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import TextBox

figure, ax = plt.subplots()
plt.subplots_adjust(bottom=0.15)
# подпись осей на графике
ax.set_xlabel("y1")
ax.set_ylabel("y2")

ny = 1
alpha0 = 0.01

N = 100
tao = 3
h = tao / N
T = 200

def line(y1_0, y2_0, alpha):
    # решение задачи методом Эйлера
    y1 = copy.deepcopy(y1_0)
    y2 = copy.deepcopy(y2_0)
    for i in range(len(y1_0)-1, int(T / h)):
        # Запаздывание влияет на y1
        y1.append(y1[i] + h * (y2[i] +
                               alpha * y1[i - int(N * np.sin(2*i*h/T))]))
    y2.append(y2[i] +
              h * (-3*y2[i] ** 3 + ny * y2[i] - y1[i]))
```

```

# Запаздывание влияет на y2
# y1.append(y1[i] + h * (y2[i]))
# y2.append(y2[i] +
#           h * (-3*y2[i]**3 + ny * y2[i] - y1[i] +
#                 alpha * y1[i - int(N * np.sin(2*i*h/T))]))
ax.plot(y1, y2)

def update_plot(strAlpha):
    # перерисовка графика в зависимости от параметра
    alpha = float(strAlpha)
    ax.cla()
    line(
        [0.1 for a in range(N)],
        [0.1 for a in range(N)],
        alpha)
    line(
        [2 for a in range(N)],
        [2 for a in range(N)],
        alpha)
    figure.canvas.draw_idle()

axfreq = plt.axes([0.13, 0.05, 0.55, 0.03])
textBox = TextBox(axfreq, 'alpha', initial=str(alpha0))
textBox.on_submit(update_plot)

def _keyboard_handler(event):
    # выход при нажатии escape
    if event.key == 'escape':
        plt.close('all')

```

```
figure.canvas.mpl_connect('key_press_event', _keyboard_handler)

update_plot(alpha0)
plt.show()
```

A.8 Влияние распределенного запаздывания

```
# Лаб. работа 7: влияние переменного запаздывания на систему
import copy
from functools import reduce
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import TextBox

figure, ax = plt.subplots()
plt.subplots_adjust(bottom=0.15)
# подпись осей на графике
ax.set_xlabel("y1")
ax.set_ylabel("y2")

ny = 1
alpha0 = 0.01

N = 100
tao = 3
h = tao / N
T = 200

def line(y1_0, y2_0, alpha):
    # решение задачи методом Эйлера
    y1 = copy.deepcopy(y1_0)
    y2 = copy.deepcopy(y2_0)
    _integral = lambda i, y1: h*reduce(lambda sum, x: sum +
                                         x ** 2, y1[i-N+1:i])
    integral = _integral(len(y1_0)-1, y1)
    try:
        for i in range(len(y1_0)-1, int(T / h)):
```

```

# Запаздывание влияет на y1
y1.append(y1[i] + h * (y2[i] +
                        alpha * integral))
y2.append(y2[i] +
           h * (-3*y2[i] ** 3 + ny * y2[i] - y1[i]))


# Запаздывание влияет на y2
# y1.append(y1[i] + h * (y2[i]))
# y2.append(y2[i] +
#           h * (-3*y2[i] ** 3 + ny * y2[i] - y1[i] +
#                 alpha * integral))

integral -= h * y1[i-N+1] ** 2
integral += h * y1[i] ** 2

finally:
    minlen = min(len(y1), len(y2))
    ax.plot(y1[0:minlen], y2[0:minlen])



def update_plot(strAlpha):
    # перерисовка графика в зависимости от параметра
    alpha = float(strAlpha)
    ax.cla()
    line(
        [0.1 for a in range(N)],
        [0.1 for a in range(N)],
        alpha)
    line(
        [2 for a in range(N)],
        [2 for a in range(N)],
        alpha)
    figure.canvas.draw_idle()

```

```
axfreq = plt.axes([0.13, 0.05, 0.55, 0.03])
textBox = TextBox(axfreq, 'alpha', initial=str(alpha0))
textBox.on_submit(update_plot)

def _keyboard_handler(event):
    # выход при нажатии escape
    if event.key == 'escape':
        plt.close('all')

figure.canvas.mpl_connect('key_press_event', _keyboard_handler)

update_plot(alpha0)
plt.show()
```

A.9 Влияние случайного шума

```
# Лаб. работа 9: влияние случайной помехи на систему
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import TextBox

figure, ax = plt.subplots()
plt.subplots_adjust(bottom=0.15)
# подпись осей на графике
ax.set_xlabel("y1")
ax.set_ylabel("y2")

ny = 1
sigma0 = 0

def line(y1_0, y2_0, sigma):
    # решение задачи методом Эйлера
    y1 = [y1_0]
    y2 = [y2_0]
    h = 0.03
    for i in range(20000):
        # y1.append(y1[i] + h * (y2[i]) +
        #           np.sqrt(h) * sigma * np.random.randn())
        # y2.append(y2[i] + h * (-3*y2[i] ** 3 +
        #           ny * y2[i] - y1[i]))
        y1.append(y1[i] + h * (y2[i]))
        y2.append(y2[i] + h * (-3*y2[i] ** 3 +
                               ny * y2[i] - y1[i]) +
                  np.sqrt(h) * sigma * np.random.randn())
    ax.plot(y1, y2)
```

```

def update_plot(strSigma):
    # перерисовка графика в зависимости от параметра
    sigma = float(strSigma)
    ax.cla()
    _y = lambda a0: a0 + sigma * np.random.randn()
    line(_y(0.1), _y(0.1), sigma)
    line(_y(2), _y(2), sigma)
    figure.canvas.draw_idle()

axfreq = plt.axes([0.13, 0.05, 0.55, 0.03])
textBox = TextBox(axfreq, 'sigma', initial=str(sigma0))
textBox.on_submit(update_plot)

def _keyboard_handler(event):
    # выход при нажатии escape
    if event.key == 'escape':
        plt.close('all')

figure.canvas.mpl_connect('key_press_event', _keyboard_handler)

update_plot(sigma0)
plt.show()

```

Приложение Б Дополнительные эксперименты

В данном приложении собраны численные эксперименты, которые не вошли в основное содержание отчета, чтобы не загромождать изложение материала, но тем не менее не стали менее важными в контексте проведенных лабораторных работ.

Б.1 Влияние постоянного запаздывания на вторую переменную системы

В данном эксперименте мы добавляем постоянное запаздывание ко второй переменной системы, получая уравнение (Б.1). Начальные значения решений и метод аналогичны разделу 6.

$$\begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = -3y_2^3 + \nu y_2 - y_1 + \alpha * y_1(t - \tau) \end{cases} \quad (\text{Б.1})$$

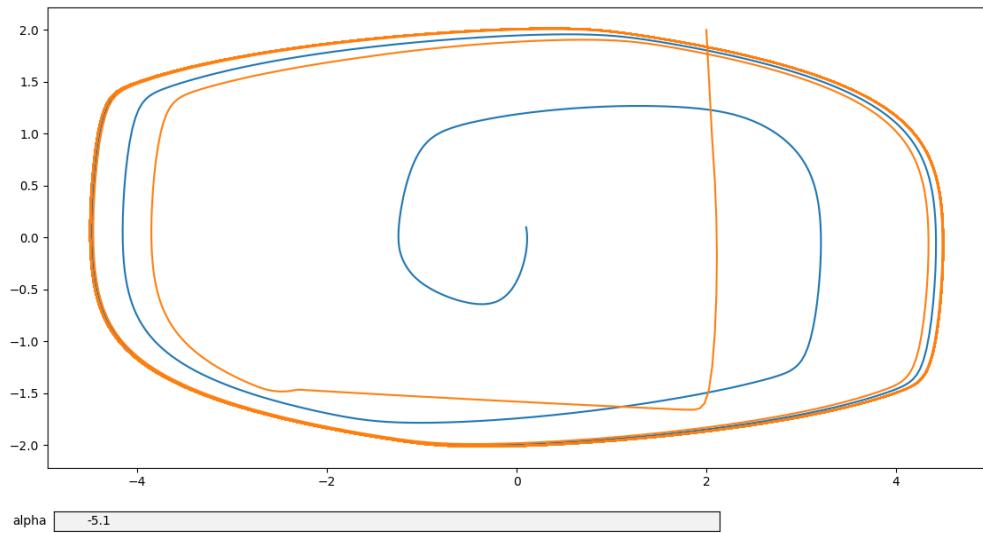


Рис. Б.1 — ($\alpha = -5.1$) При отрицательном параметре изменяется форма цикла. Можно предполагать, что предельность данного цикла остается

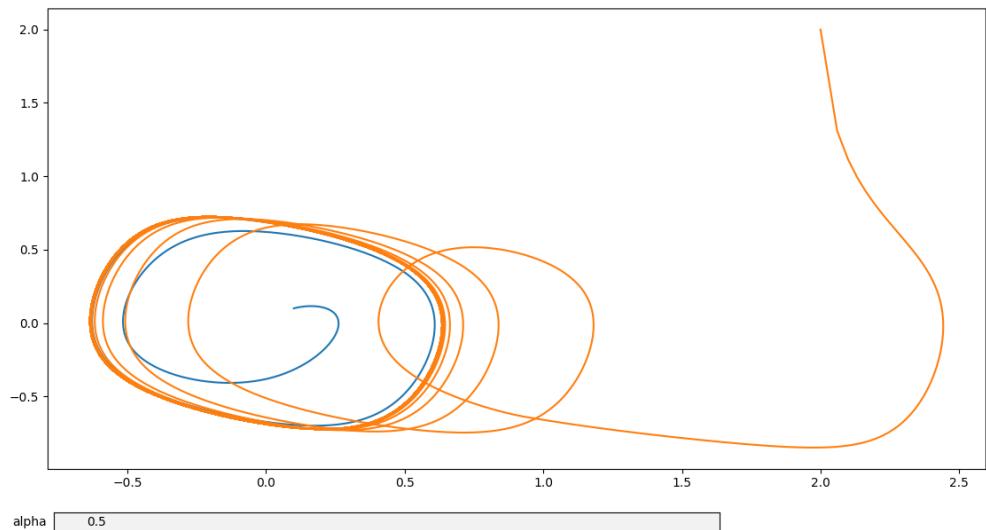


Рис. Б.2 — ($\alpha = 0.5$) Цикл начинает раскручиваться в правую сторону

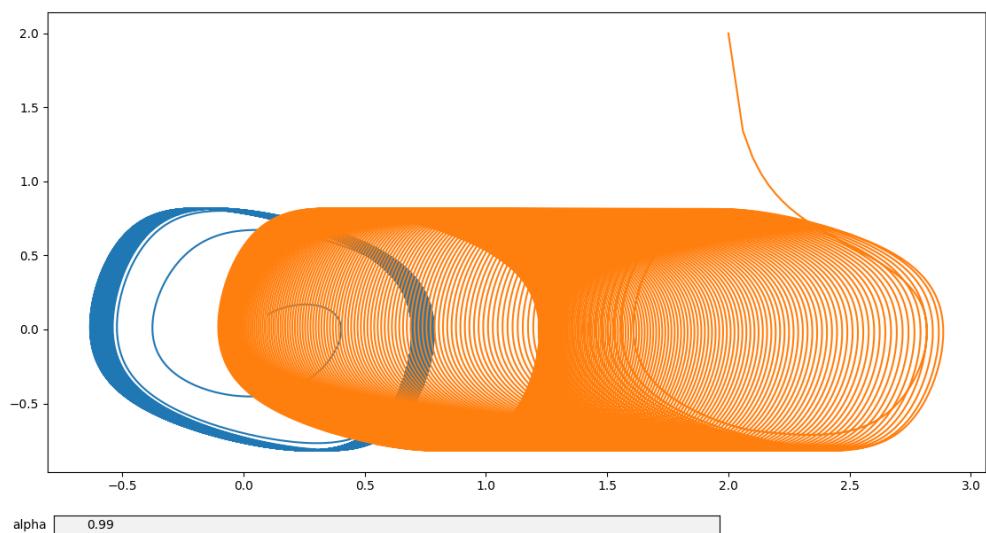


Рис. Б.3 — ($\alpha = 0.99$) Наблюдается очень плотная спираль

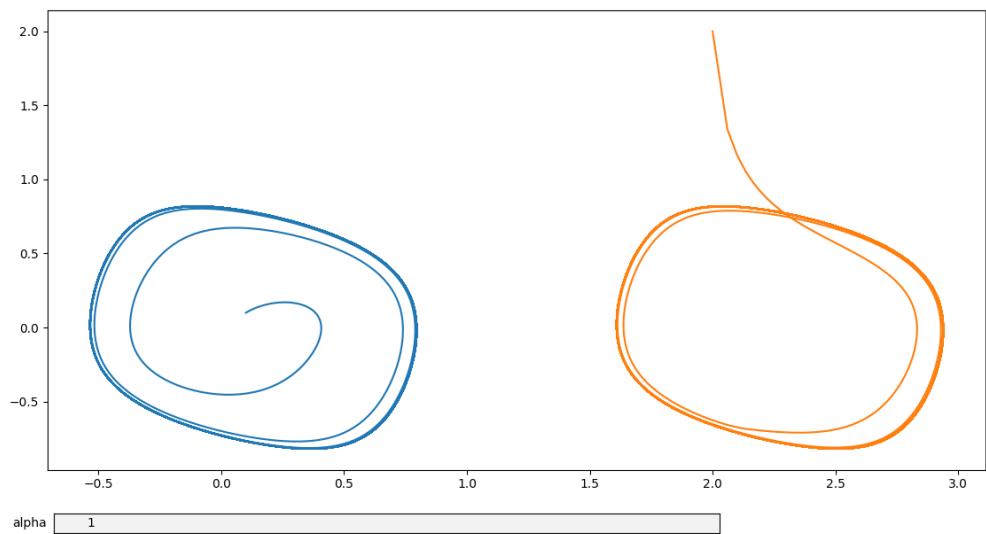


Рис. Б.4 — ($\alpha = 1$) Потенциальная точка бифуркации. Два решения сошлись к двум разным циклам

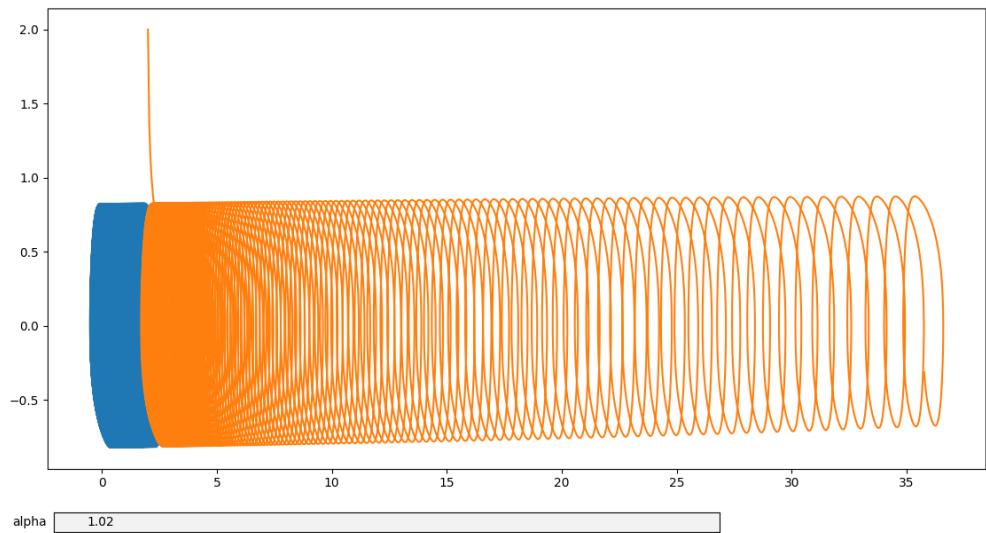


Рис. Б.5 — ($\alpha = 1.02$) Даже небольшое увеличение α изменило систему

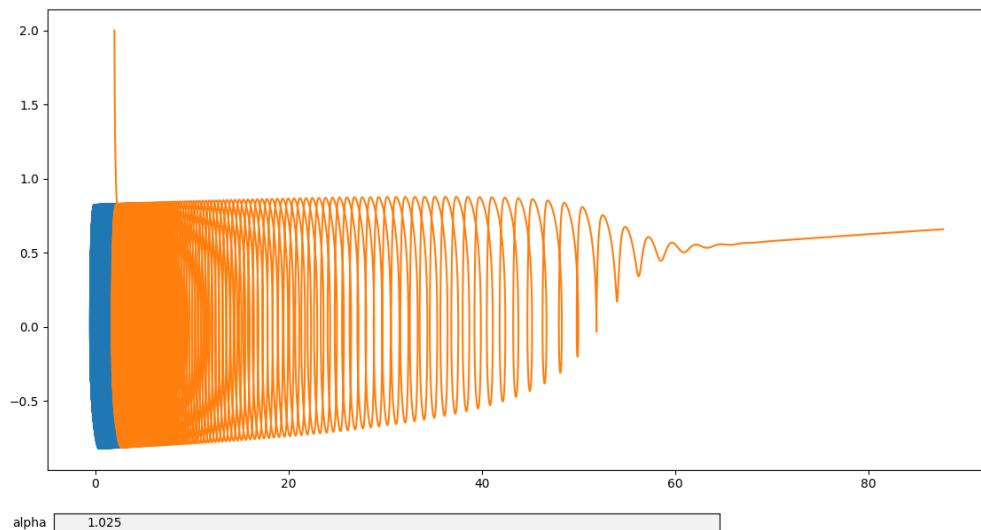


Рис. Б.6 — ($\alpha = 1.025$) Решение начинает увеличиваться по y_1

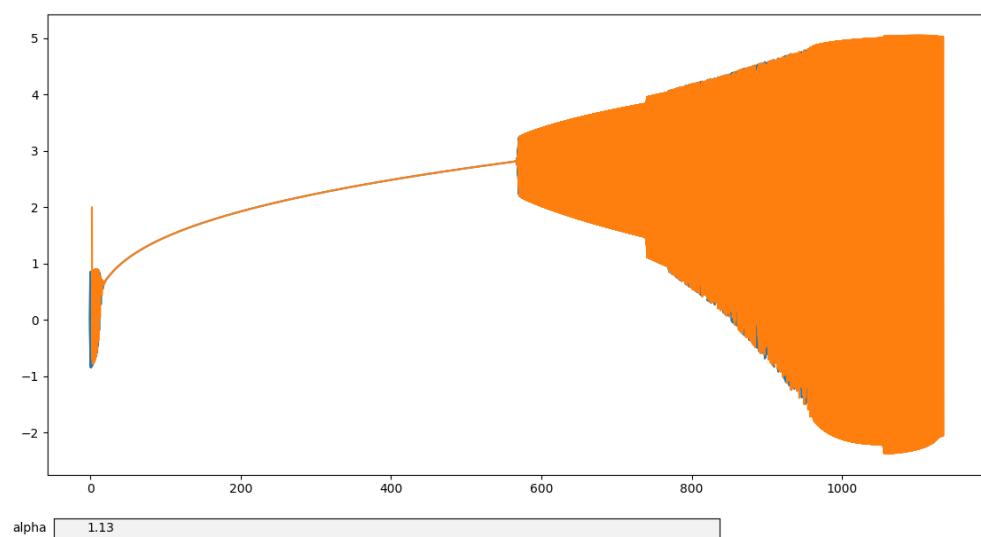


Рис. Б.7 — ($\alpha = 1.13$) В интервале от 600 до 1000 наблюдаются резкие колебания

Б.2 Влияние переменного запаздывания на вторую переменную системы

В данном эксперименте мы добавляем переменное запаздывание ко второй переменной системы, получая уравнение (Б.2). Начальные значения решений и метод аналогичны разделу 7.

$$\begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = -3y_2^3 + \nu y_2 - y_1 + \alpha * y_1(t - \tau * \sin(\frac{2t}{T})) \end{cases} \quad (\text{Б.2})$$

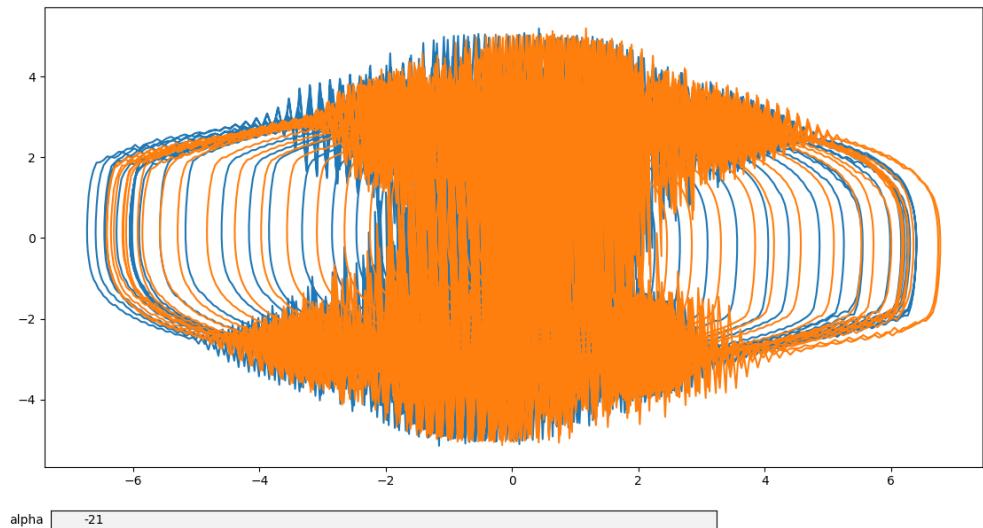


Рис. Б.8 — ($\alpha = -21$) В цикле начались резкие колебания

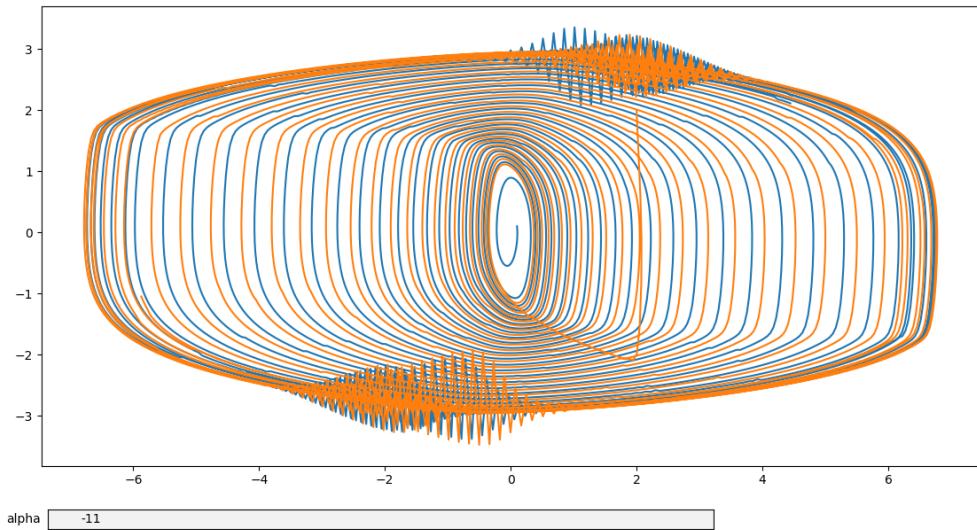


Рис. Б.9 — ($\alpha = -11$) Зарождения колебаний в цикле

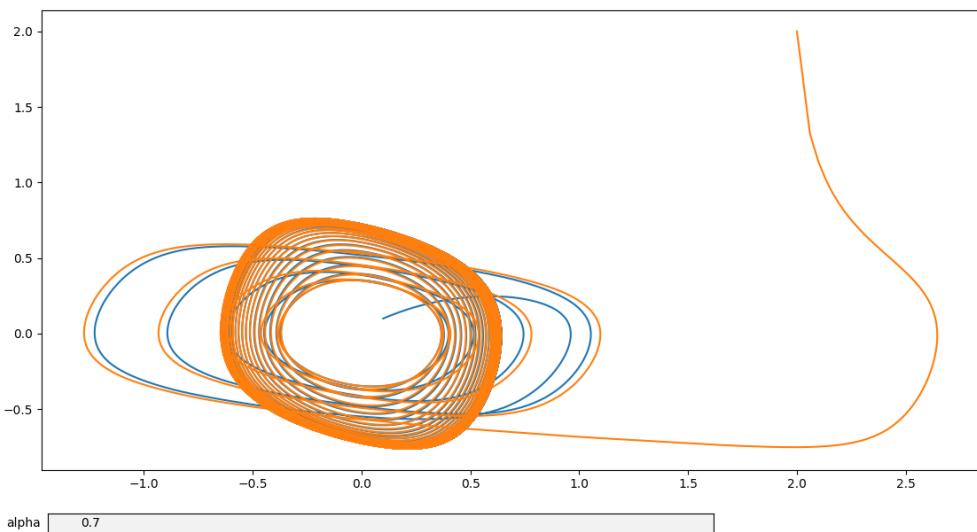


Рис. Б.10 — ($\alpha = 0.7$) Цикл меняет свою форму и напоминает виниловую пластинку

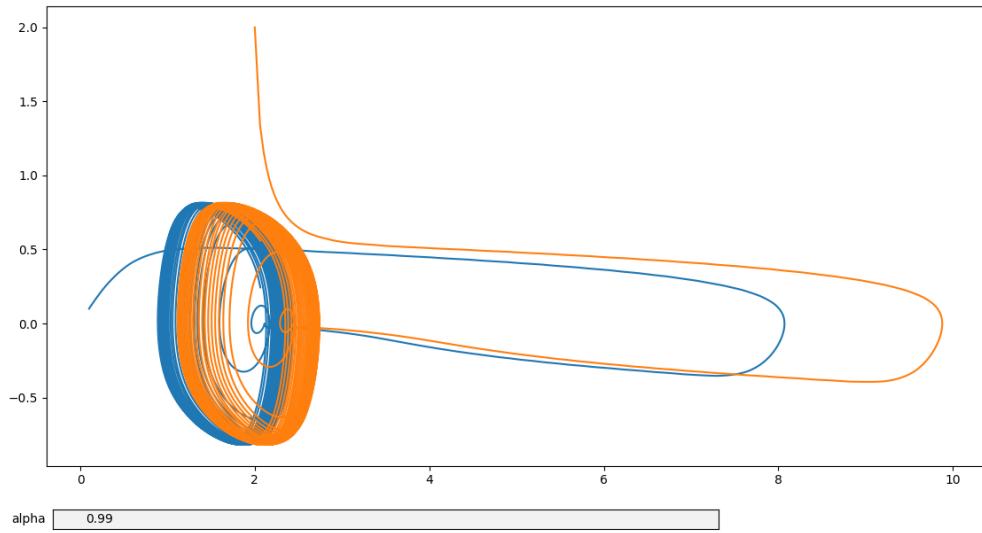


Рис. Б.11 — ($\alpha = 0.99$) Цикл начинает смещаться вправо

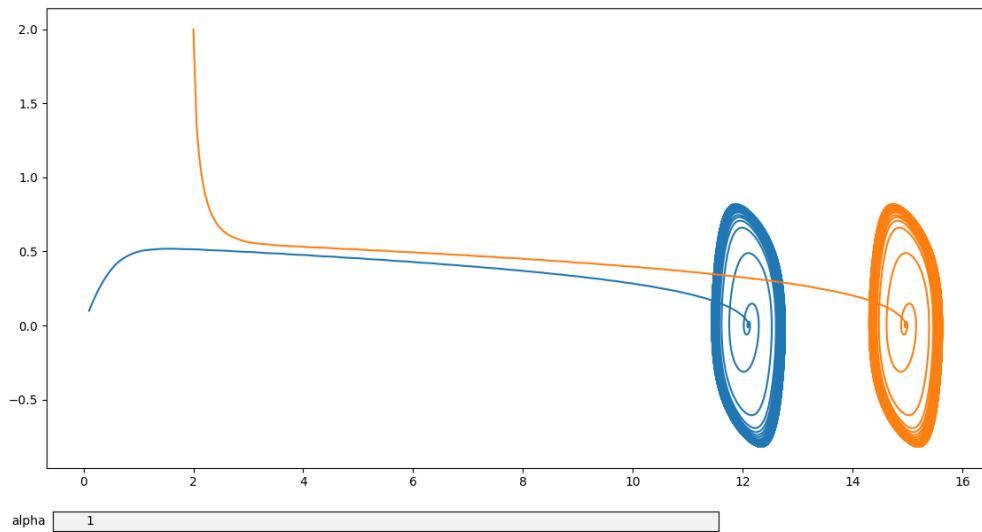


Рис. Б.12 — ($\alpha = 1$) Жесткая бифуркация: каждое решение сошлось к
своему циклу

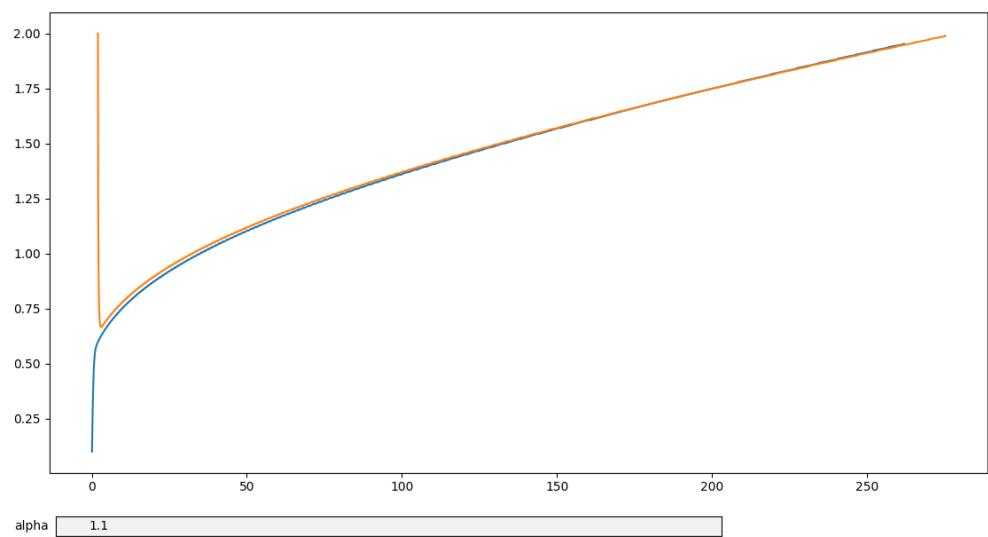


Рис. Б.13 — ($\alpha = 1.1$) Оба решения начали уходить в бесконечность
вдоль асимптоты

Б.3 Влияние распределенного запаздывание на вторую переменную системы

В данном эксперименте мы добавляем распределенное запаздывание ко второй переменной системы, получая уравнение (Б.3). Начальные значения решений и метод аналогичны разделу 8.

$$\begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = -3y_2^3 + \nu y_2 - y_1 + \alpha * \int_{t-\tau}^t y_1^2(s)ds \end{cases} \quad (\text{Б.3})$$

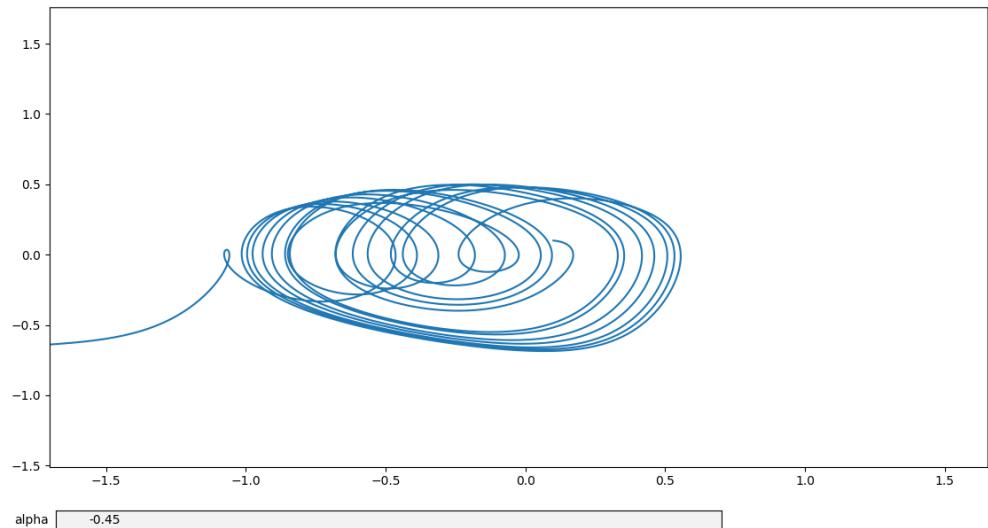


Рис. Б.14 — ($\alpha = -0.45$) Хаотично покрутившись в районе цикла, первое решение разошлось влево

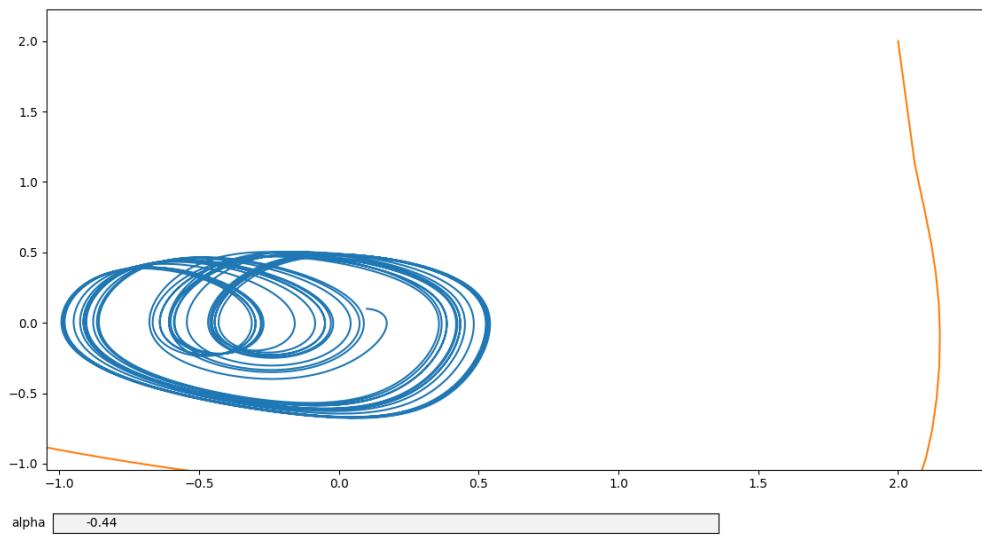


Рис. Б.15 — ($\alpha = -0.44$) Второе решение разошлось, первое циклично
двигается, заворачиваясь в петельку

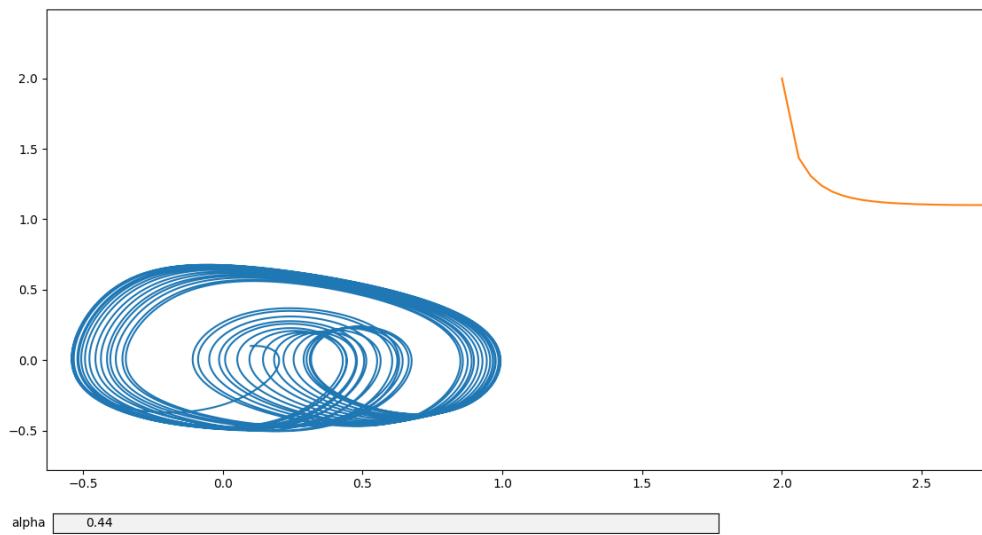


Рис. Б.16 — ($\alpha = 0.44$) Аналогичные результаты, только петелька
отразилась сверху вниз

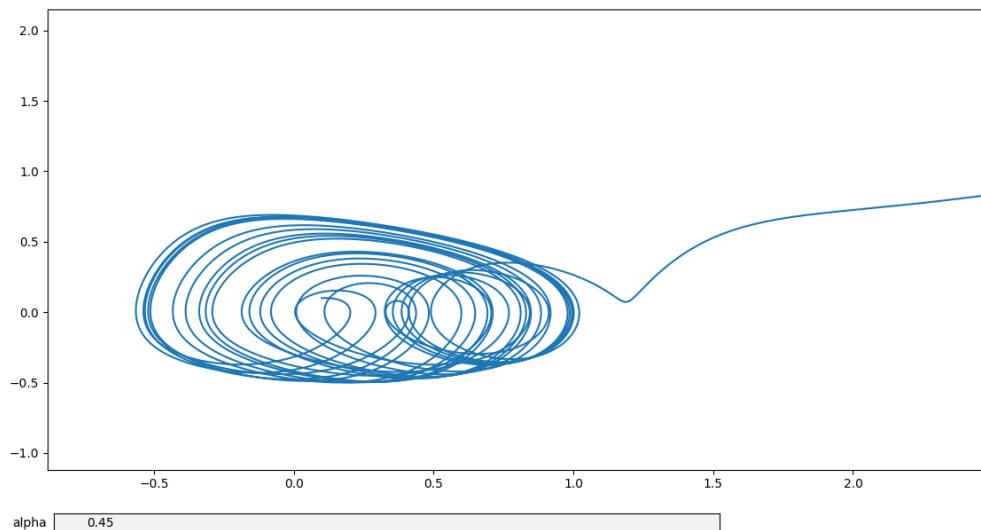


Рис. Б.17 — ($\alpha = 0.45$) Первое решение разошлось вправо

Как можно заметить, симметрия относительно значений α наблюдается даже в этом эксперименте.

Б.4 Влияние случайного шума на вторую переменную системы

В данном эксперименте мы добавляем влияние шума ко второй переменной системы, получая уравнение (Б.4). Начальные значения решений и метод аналогичны разделу 9.

$$\begin{cases} dy_1 = y_2 dt \\ dy_2 = (-3y_2^3 + \nu y_2 - y_1)dt + \sigma * dW \end{cases} \quad (\text{Б.4})$$

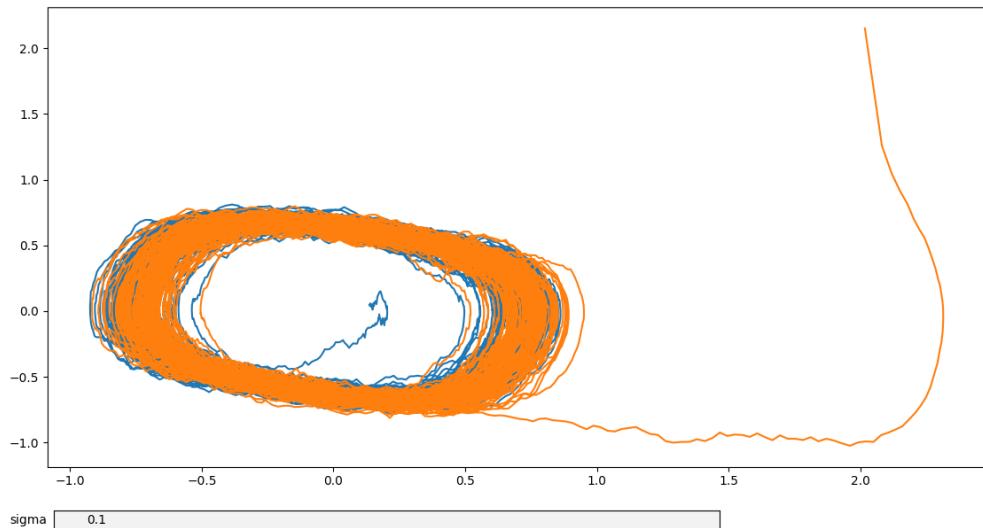


Рис. Б.18 — ($\alpha = 0.1$) Решения более зашумленные вдоль оси Oy_2

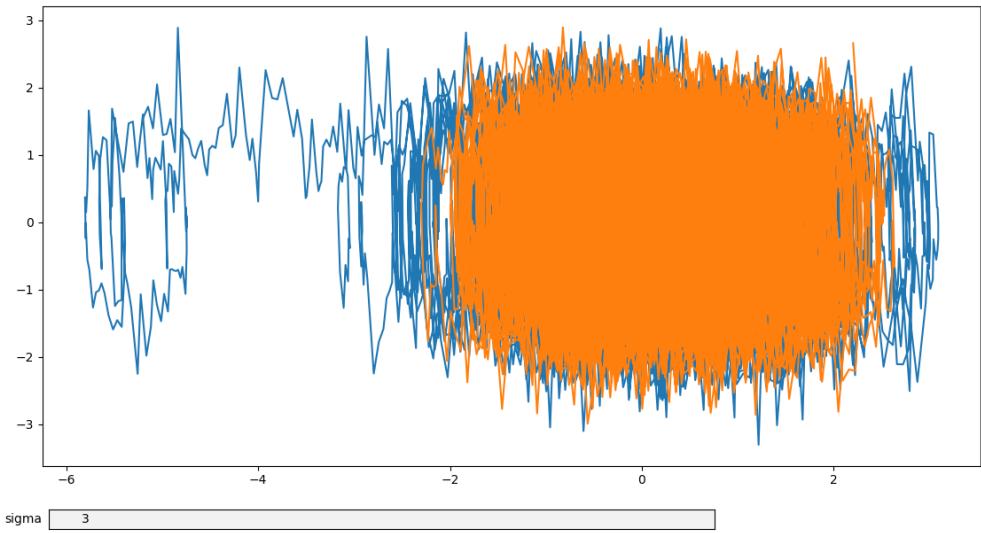


Рис. Б.19 — ($\alpha = 3$) Решения не разошлись, напоминает НЛО

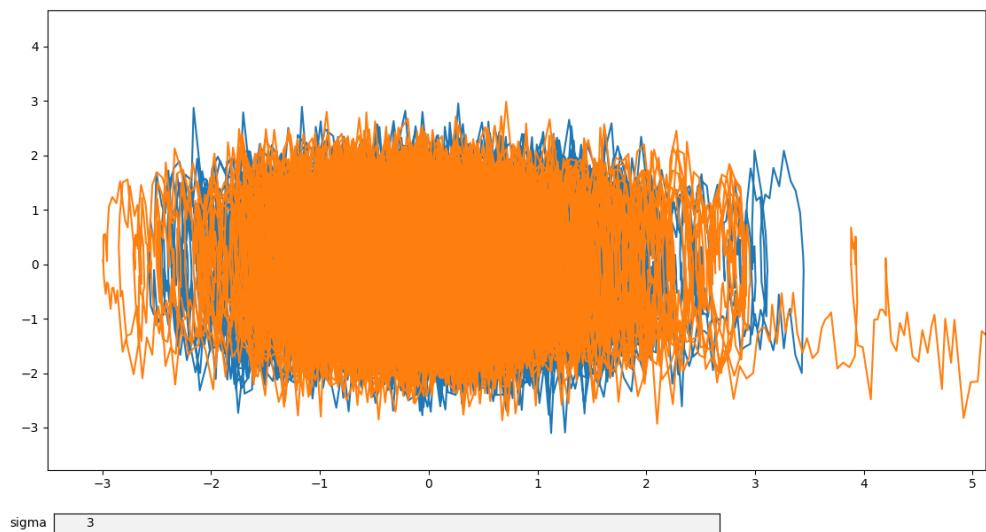


Рис. Б.20 — ($\alpha = 3$) В этой реализации случайное воздействие сильно повлияло на второе решение, и оно разошлося