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 нелиненой системы

ОТЧЕТ

по лабораторной работе

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

Екатеринбург
2017

Отчет

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 и определение характера его устойчивости. 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

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.

$$\ddot{x} + 3\dot{x}^3 - \nu\dot{x} + x = 0 \tag{1.1}$$

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.

$$\begin{cases} y_1 = x \\ y_2 = \dot{x} \end{cases} \tag{1.2}$$

$$\begin{cases} \dot{y_1} = y_2 \\ \dot{y_2} = -3y_2^3 \ + \nu y_2 - y_1 \end{cases} \tag{1.3}$$

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: *предельным циклом* мы 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 координат, 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()
```

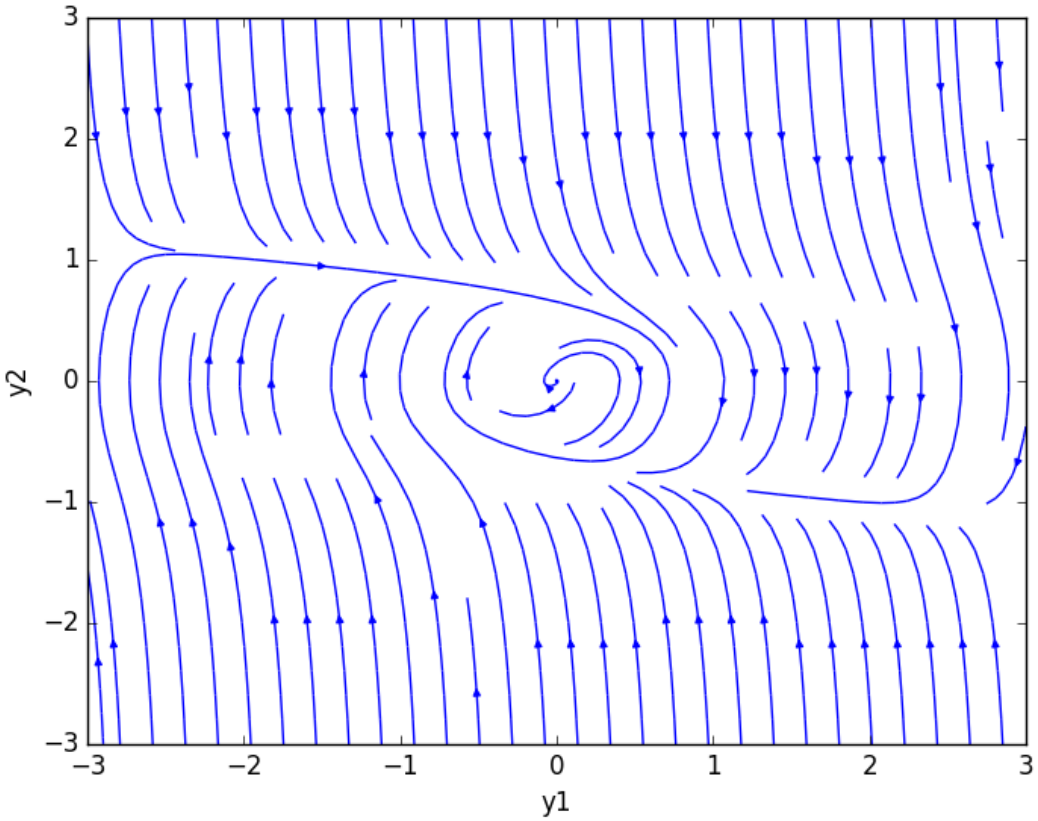

График 1.1 — Поиск предельного цикла построением фазового портрета

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): # 200о - количество итераций
        y1.append(y1[-1] + h*(y2[-1]))
        y2.append(y2[-1] + h*(-3*y2[-1] ** 3 + nu*y2[-1] - y1[-1]))
    # 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)
```

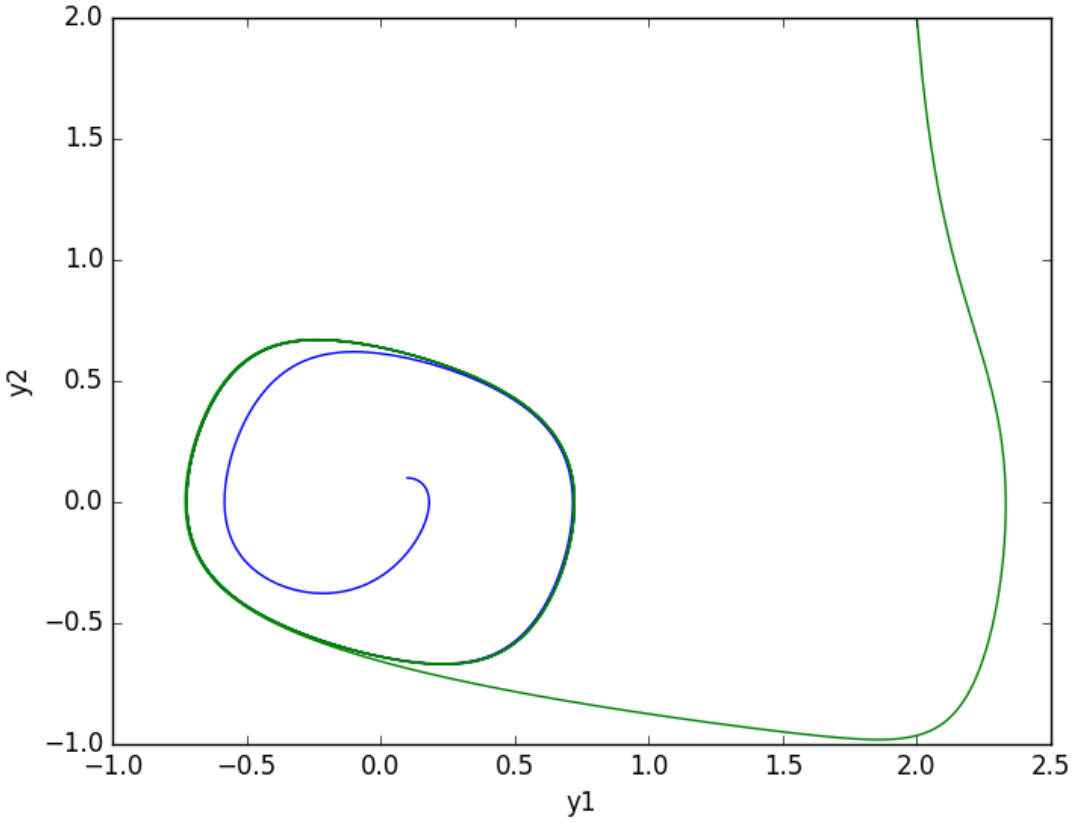

График 1.2 — Обнаружение аттрактора методом Эйлера

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 1.2, 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).

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.

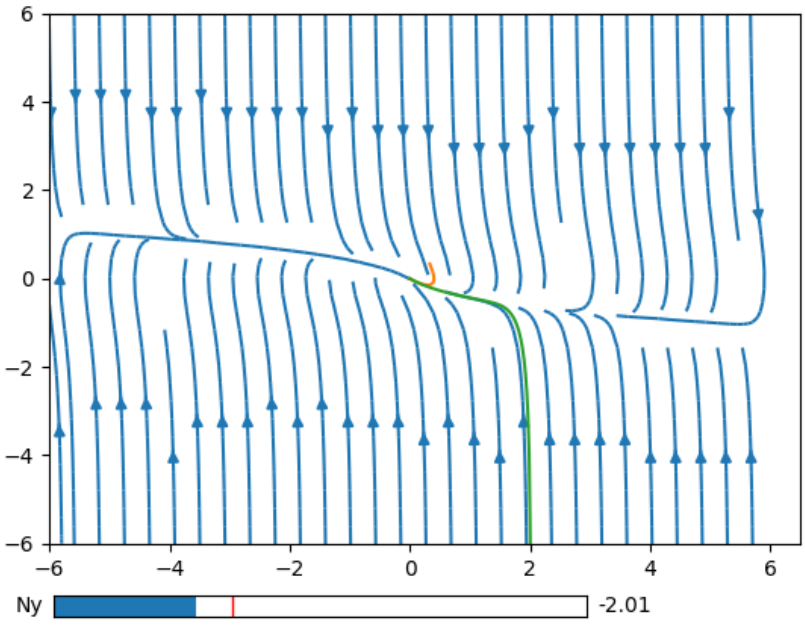

График 2.1 — Стационарная точка системы при $\nu = -2.02$

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

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

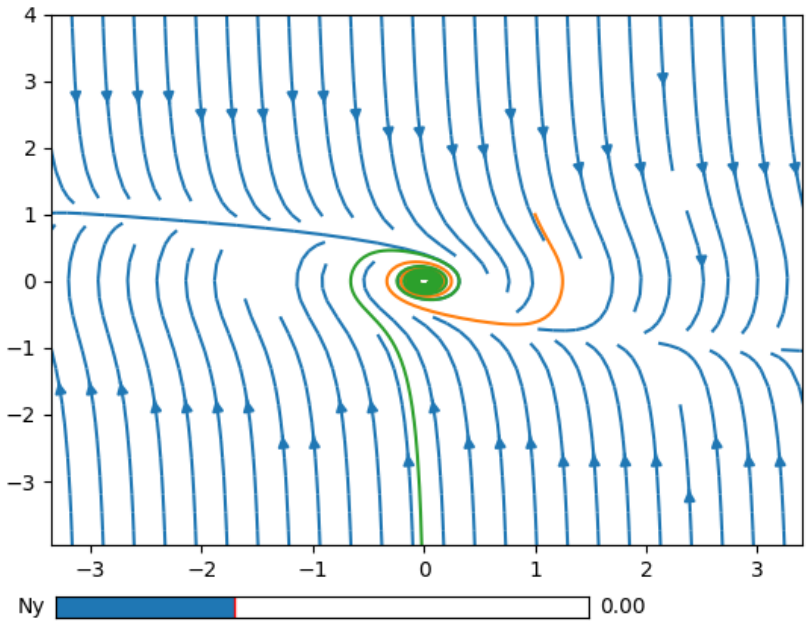

График 2.2 — Стационарная точка системы при $\nu = -0.02$

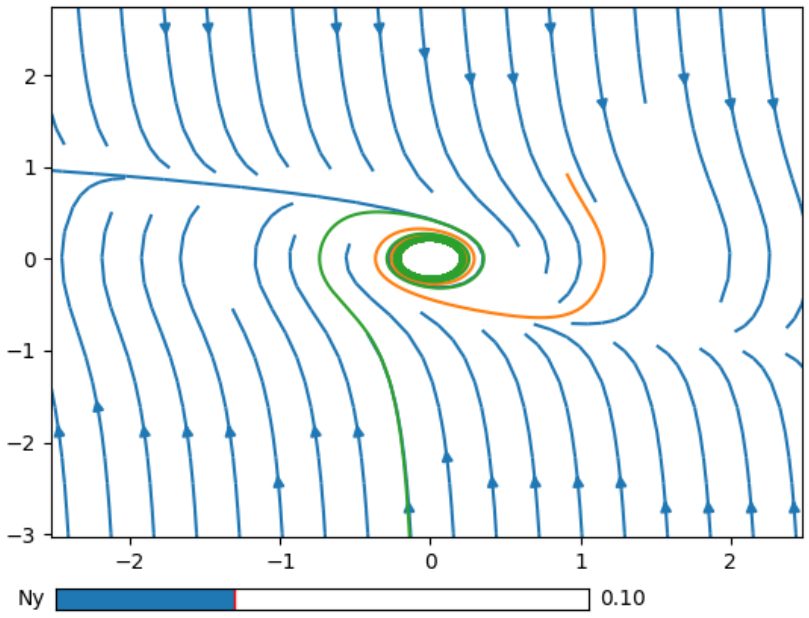

График 2.3 — Появление цикла при $\nu = 0.1$

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).

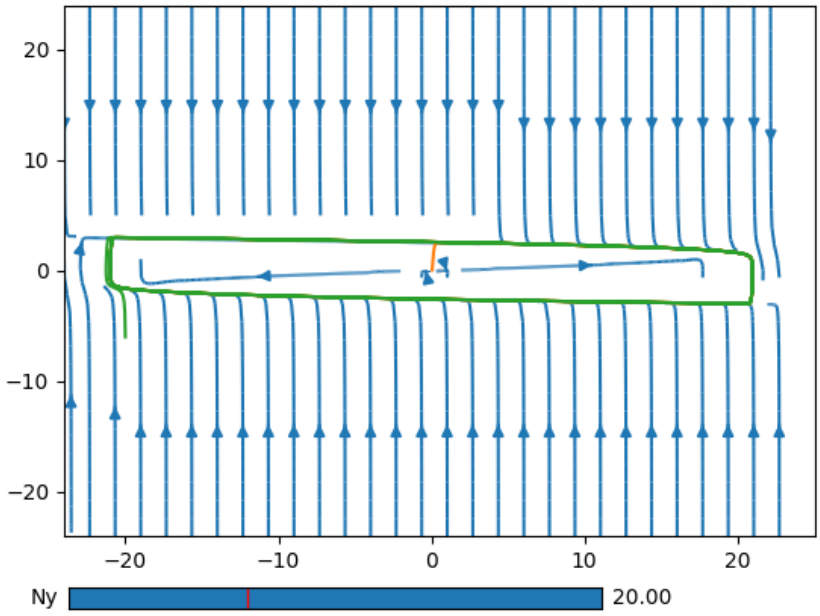

График 2.4 — Расширение предельного цикла при увеличении параметра ($\nu = 20$)

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 параметра 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 точка мы наблюдаем стационарную

точку, а когда предельный цикл, то мы можем сказать, 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 получатся 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_{y1}, 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 похож 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 $y1$ и $y2$, описывающие наш цикл.

Листинг 3 Поиск параметров системы

```
eps = 0.5 * 10 ** -4
y1_0, y2_0 = 0.724197, 0 # начальная точка

y1 = [y1_0]
y2 = [y2_0]
h = 0.0001
for i in range(100000):
    # итерация метода Эйлера
    y1.append(y1[-1] + h*(y2[-1]))
    y2.append(y2[-1] + h*(-3*y2[-1] ** 3 + ny * y2[-1] - y1[-1]))
    # проверка прихода в туже точку с погрешностью
    if  np.abs(y1_0 - y1[-1]) < eps and
        np.abs(y2_0 - y2[-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=6.6633000000000004
```

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. Под *орбитальной устойчивостью* мы будем подразумевать такое свойство периодического решения системы $\phi(t)$, если для любого $\epsilon > 0$ найдется постоянное число $\delta = \delta(\epsilon) > 0$, такое, что траектория всякого решения $X(t)$, начинающегося в δ-окрестности траектории $\phi(t)$, остается в ϵ-окрестности траектории $\phi(t)$ при всех $t \geqslant 0$.

Асимптотическим же будем считать такое орбитально устойчивое решение, что выполняется условие 4.1, то есть любая достаточно близкая траектория сходится к орбитально устойчивому решению:

$$\exists \alpha, \beta > 0 : \|\mathbf{X}(0) - \boldsymbol{\varphi}(0)\| < \delta \rightarrow \|\mathbf{X}(t) - \boldsymbol{\varphi}(t)\| \leq \alpha \|\mathbf{X}(0) - \boldsymbol{\varphi}(0)\| e^{-\beta t} \tag{4.1}$$

Из поставленных определений видна мотивация исследования решения на наличие асимптотической орбитальной устойчивости: подтвердив его, мы, увеличивая точность вычислений, можем быть уверены, что лучше приближаемся к необходимой траектории.

В этом нам поможет, аналог *теоремы 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 линеаризированную 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 мы и будем считать мультипликаторами системы.

Осталось, провести поставленные шаги. Рассмотрим нашу систему 4.2:

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

Посчитаем Якобиан нашей системы (уравнения 4.3):

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

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

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

В листинге 4 мы, методом Эйлера решаем полученную систему, находим матрицу монодромии и вычисляем ее Собственные числа (Eigenvalues). Как видно из вывода программы, вычисленные мультипликаторы удовлетворяют условиям рассмотренной теоремы с искомой точностью до 3-х знаков ($8.59 * 10^{-4}$ и 1), таким исследуемый цикл асимптотически орбитально устойчив.

Листинг 4 Вычисление мультипликаторов

```
def linear_form(y1_0, y2_0, cycle_y1, cycle_y2):
    # Решение линеаризованной системы вдоль цикла
    y1 = [y1_0]
    y2 = [y2_0]
    for j in range(len(cycle_y1)):
        y1.append(y1[-1] + h * (y2[-1]))
        y2.append(y2[-1] + h * (-y1[-1] +
                        (-9*cycle_y2[j] ** 2 + ny) * y2[-1]))
    return [y1[-1], y2[-1]]

cycle_y1, cycle_y2 = line(0.724197, 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:
# [[  8.92710796e-04   6.63982838e-05]
#  [  5.05095135e-01   1.00018677e+00]]
# Eigenvalues:
# [  8.59150781e-04   1.00022033e+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.

Звучит от следующим образом: 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: -7.058326426806031
```

Заключение

Заключение будет, когда все закончится.

Список использованных источников

1. Numpy - библиотека для научиных 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.

Приложение А Исходный код программ

А.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[-1] + h*(y2[-1]))
        y2.append(y2[-1] +
                  h * (-3*y2[-1] ** 3 + nu*y2[-1] - y1[-1]))
```

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

# построение двух кривых, начинающихся внутри и
# вне предпологаемого 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[-1] + h * (y2[-1]))
        y2.append(y2[-1] +
                h * (-3*y2[-1] ** 3 + ny * y2[-1] - y1[-1]))
    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)
    # увелечение 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 = 0.724197
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[-1] + h*(y2[-1]))
    y2.append(y2[-1] +
                h * (-3*y2[-1] ** 3 + ny * y2[-1] - y1[-1]))
    # считаем цикл завершенным, когда покоординатно
    # приблизились к начальной точке ближе, чем на eps
    if (np.abs(y1_0 - y1[-1]) < eps and
        np.abs(y2_0 - y2[-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 подобраных 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[-1] + h * y2[-1])
        y2.append(y2[-1] +
                  h*(-3*y2[-1] ** 3 + ny * y2[-1] - y1[-1]))
        if (np.abs(y1_0 - y1[-1]) < eps and
            np.abs(y2_0 - y2[-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 j in range(len(cycle_y1)):
```

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

# Начало программы
# вычисление поточечного описания предельного цикла
cycle_y1, cycle_y2 = line(0.724197, 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 подобраных 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[-1] + h * (y2[-1]))
        y2.append(y2[-1] +
                  h*(-3*y2[-1] ** 3 + ny * y2[-1] - y1[-1]))
        if (np.abs(y1_0 - y1[-1]) < eps and
            np.abs(y2_0 - y2[-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
    """
```

```
    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))
```