МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«Нижегородский государственный университет им. Н.И. Лобачевского»

Национальный исследовательский университет

Институт информационных технологий, математики и механики

Институт информационных технологий, математики и механики

Отчет по лабораторной работе на тему:

«Численное решение систем ДУ методом Рунге- Кутта четвертого порядка»

Выполнил:
студент группы 381706-2
Банденков Даниил Викторович

____
подпись

Преподаватель:
Ассистент кафедры дифференциальных уравнений, математического и численного анализа ИИТММ
Морозов Кирилл Евгеньевич

____
подпись

Нижний Новгород
2020

Содержание

Введение

Обыкновенным *дифференциальным уравнением n –го порядка* называется уравнение вида $F\ (x, y(x), y\ '(x), y\ ''(x), \dots , y^{(n)}(x)) = 0$, где F — известная функция (n + 2)-х переменных, x — независимая переменная из интервала (a,b), $y(x)$ — неизвестная функция. Число n называется порядком уравнения.

Функция $y(x)$ называется *решением* (или *интегралом*) дифференциального уравнения на промежутке (a, b), если она n раз дифференцируема на (a, b) и при подстановке в уравнение обращает его в тождество.

Обыкновенные дифференциальные уравнения, разрешенные относительно старшей производной, называют уравнениями в *нормальной форме*: $y^{(n)} = f(x, y, y\ ', y\ '', \dots , y^{(n-1)})$.

Дифференциальное уравнение обычно имеет бесконечно много решений. Чтобы выделить нужное решение, используют дополнительные условия.

Чтобы выделить единственное решение уравнения n–го порядка обычно задают n начальных условий $y(x0) = y0, y\ '(x0) = y1, y\ ''(x0) = y2, \dots , y^{(n-1)}(x0) = yn - 1$.

Задачей Коши (или начальной задачей) называется задача отыскания решения $y = y(x)$ уравнения $F(x, y(x), y\ '(x), y\ ''(x), \dots , y^{(n\ )}(x)) = 0$, $x>x0$, удовлетворяющего условиям $y(x0) = y0, y\ '(x0) = y1, y\ ''(x0) = y2, \dots , y^{(n-1)}(x0) = yn - 1$.

Условия $y(x0) = y0, y\ '(x0) = y1, y\ ''(x0) = y2, \dots , y^{(n-1)}(x0) = yn - 1$ называются начальными данными, начальными условиями или данными Коши.

Любое конкретное решение $y = \varphi(x)$ уравнения n –го порядка $F(x, y(x), y\ '(x), y\ ''(x), \dots , y^{(n\ )}(x)) = 0$, называется *частным решением*.

Общим решением дифференциального уравнения $F(x, y(x), y\ '(x), y\ ''(x), \dots , y^{(n)}(x)) = 0$ называется функция $y = \Phi(x, C1, C2, \dots , Cn)$ содержащая некоторые постоянные (параметры) $C1, C2, \dots , Cn$, и обладающая следующими свойствами:

1. $\Phi(x, C1, C2, \dots , Cn)$ является решением уравнения при любых допустимых значениях $C1, C2, \dots , Cm$;
2. для любых начальных данных $y(x0) = y0, y\ '(x0) = y1, y\ ''(x0) = y2, \dots , y^{(n-1)}(x0) = yn - 1$, для которых задача Коши имеет единственное решение, существуют значения постоянных $C1 = A1, C2 = A2, \dots , Cn = An$, такие что решение $y = \Phi(x, A1, A2, \dots, An)$ удовлетворяет заданным начальным условиям.

Иногда частное или общее решение уравнения удается найти только в неявной форме: $f(x, y) = 0$ или $G(x, y, C1, C2, ..., Cn) = 0$. Такие неявно заданные решения называются *частным интегралом* или *общим интегралом* уравнения.

Если задачу об отыскании всех решений дифференциального уравнения удается свести к алгебраическим операциям и к вычислению конечного числа интегралов и производных от известных функций, то уравнение называется *интегрируемым в квадратурах*. Класс таких уравнений относительно узок.

Для решения уравнений, которые не интегрируются в квадратурах, применяются приближенные или численные методы.

Наиболее эффективными и часто встречаемыми методами решениями задачи Коши являются методы Рунге - Кутта. Они основаны на аппроксимации искомой функции y(x) в пределах каждого шага многочленом, который получен при помощи разложения функции y(x) в окрестности шага h каждой i-ой точки в ряд Тейлора.

Усекая ряд Тейлора в различных точках и отбрасывая правые члены ряда, Рунге и Кутта получали различные методы для определения значений функции y(x) в каждой узловой точке. Точность каждого метода определяется отброшенными членами ряда.

Постановка задачи

В данной работе необходимо реализовать программу решения модели ФитцХью — Нагумо[1] $\begin{array}{l} \dot{u} = \mathrm{u} - \frac{\mathrm{u}^3}{3} - \mathrm{v} + \mathrm{I}_{\mathrm{ext}} \\ t\dot{\mathrm{v}} = \mathrm{u} + \mathrm{a} - \mathrm{b}v \end{array}$ методом Рунге-Кутта четвертого порядка. Получить практические навыки программирования.

[1] Модель ФитцХью — Нагумо описывает прототип возбудимой системы (например, нейрона), является примером генератора релаксационных колебаний, вследствие того, что, если внешний стимул I_{ext} превышает определенное пороговое значение, система будет демонстрировать характерное возвратно-поступательное движение (экскурсию) в фазовом пространстве, до тех пора переменные *x* и *y* не "релаксируют" до предыдущего состояния. Это поведение характерно для спайков, возбуждённых в нейроне стимуляцией внешним входным сигналом.

Реализация задачи

Метод Рунге-Кутта 4-го

Классический метод Рунге-Кутта 4-го порядка описывается следующей системой пяти равенств: $y_{i+1} = y_m + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4),$ где:

$$k_1 = f(x_i, y_i),$$

$$k_2 = f(x_i + \frac{h}{2}, y_i + \frac{hk_1}{2}),$$

$$k_3 = f(x_i + \frac{h}{2}, y_i + \frac{hk_2}{2}),$$

$$k_4 = f(x_i + h, y_i + hk_3).$$

При запуске программа решает модель ФитцХью — Нагумо

$\dot{u} = \mathrm{u} - \frac{\mathrm{u}^3}{3} - \mathrm{v} + \mathrm{I}_{\mathrm{ext}}$ методом Рунге-Кутта, а также строит графики фазовый портрет.
$t\dot{\mathrm{v}} = \mathrm{u} + \mathrm{a} - \mathrm{b}v$

Для реализации данной программы был выбран язык программирования python, из-за огромного количества библиотек для визуализации графики и универсальности языка. В качестве gui используется стандартная форма matplotlib. Коллекция команд pyplot, заставляет работать matplotlib как библиотеку MATLAB. Также благодаря возможностям данной библиотеки, можно сохранять полученные результаты менять масштаб и цвета графиков, а благодаря добавленным на формам кнопкам можно менять параметры, шаг, значение переменных и количество точек, удалять полученные графики и добавлять новые.

В программе реализованы 3 функции:

def Runge_Kutt(x0,y0,t,h)– метод Рунге-Кутта.

def fx(x,y)–функция fx, зависящая от двух переменных.

def fy(x,y)–функция fy, зависящая от двух переменных.

При запуске программы можно менять значения параметров и начальных условий в соответствующих окнах, шага и количества точек. Добавлена кнопка очищения графика. Так же есть значения по умолчанию, на случай если оставить форму пустой, или если пользователь решит ввести символы вместо числа. В консоли выскочит предупреждение и будет выведено значение по умолчанию(соответствуют рисунку 1).

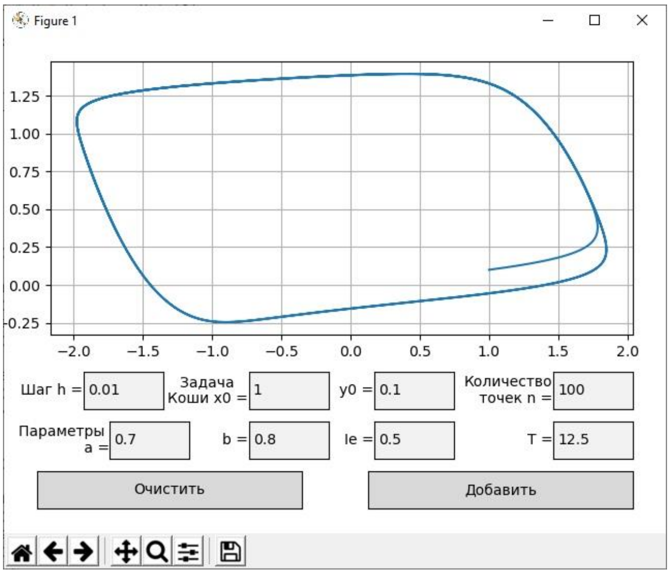

Рис1

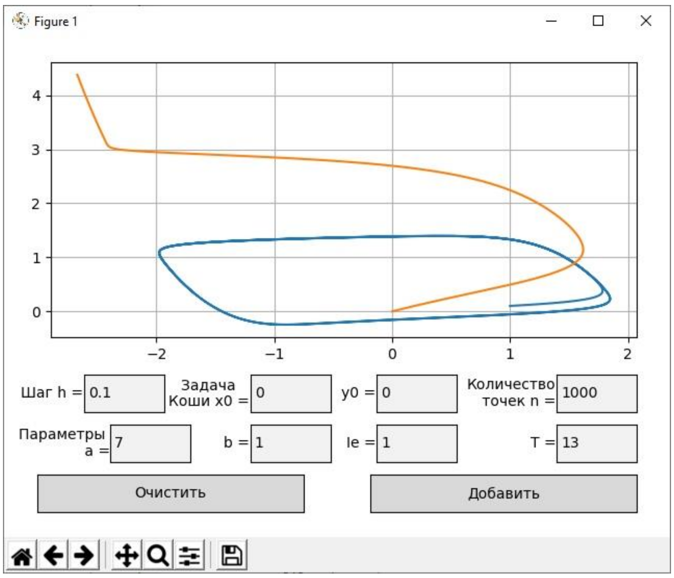

Рис 2

Заключение

В ходе работы была реализована программа решения модели ФитцХью — Нагумо методом Рунге-Кутта. Были получены практические навыки программирования.

Также в ходе работы было сделано несколько выводов: метод Рунге-Кутта не сложен в реализации, и имеет высокую точность. Этот метод имеет четвёртый порядок точности. Это значит, что ошибка на одном шаге имеет порядок $O(h^5)$, а суммарная ошибка на конечном интервале интегрирования имеет порядок $O(h^4)$.

К достоинствам этого метода можно добавить то, что он является явными, т.е. значения y_{i+1}находится по ранее найденным значениям $y_1, y_2, \ldots y_i$. Также метод допускает введения переменного шага h.

Литература

1. Степанов В. В., Курс дифференциальных уравнений, 9 изд., М.,1966
2. Ильина В. А., Силаев П. К. . Численные методы для физиков-теоретиков. Ч. 2. — Москва-Ижевск: Институт компьютерных исследований, 2004.
3. https://en.wikipedia.org/wiki/FitzHugh-Nagumo_model

Приложение

```
import matplotlib.pyplot as plt
from matplotlib.widgets import Button, TextBox

#function for X
def fx(u, v, Ie):
    return float(u - u * u * u / 3. - v + Ie)
#function for Y
def fy(u, v, a, b, T):
    return float((u + a - b * v) / T)

#runge kutta 4th order
def Runge_Kutt(x0,y0,n,h):
    x.append(x0)
    y.append(y0)
    t0 = 0.
    i = 0
    yt = 0
    xt = 0
    while(t0 < n):
        k1_y = h * fy(x[i], y[i], a, b, T)
        kx = x[i] + h / 2
        y1 = y[i] + k1_y / 2
        k2_y = h * fy(kx, y1, a, b, T)
        y1 = y[i] + k2_y / 2
        k3_y = h * fy(kx, y1, a, b, T)
        kx = kx + h / 2
        y1 = y[i] + k3_y
        k4_y = h * fy(kx, y1, a, b, T)
        yt = y[i] + (k1_y + 2 * k2_y + 2 * k3_y + k4_y) / 6
        y.append(yt)

        k1_x = h * fx(x[i], y[i], Ie)
        kx = x[i] + h / 2
        y1 = y[i] + k1_x / 2
        k2_x = h * fx(kx, y1, Ie)
        y1 = y[i] + k2_x / 2
        k3_x = h * fx(kx, y1, Ie)
        kx = kx + h / 2
        y1 = y[i] + k3_x
        k4_x = h * fx(kx, y1, Ie)
        xt = x[i] + (k1_x + 2 * k2_x + 2 * k3_x + k4_x) / 6
        x.append(xt)
        t0+=h
        i+=1

#draw graph_axes
def addGraphPhase(graph_axes, x, y):
     graph_axes.plot(x, y)
     plt.draw()

if __name__ == '__main__':
    x = list()
    y = list()

    #Event handler for the Add button
    def onButtonAddClicked(event):
        global graph_axes, x, y
        global x0, y0, n, h, a, b, Ie, T
        Runge_Kutt(x0,y0,n,h)
        addGraphPhase(graph_axes, x, y)
        x = []
        y = []
```

```
#Event handler for the Clear button
def onButtonClearClicked(event):
    global graph_axes
    graph_axes.clear()
    graph_axes.grid()
    plt.draw()

#Event handler for entering text (may be finish it)
def submitA(text):
    global a
    a = float(text)

def submitH(text):
    global h
    h = float(text)

def submitX0(text):
    global x0
    x0 = float(text)

def submitY0(text):
    global y0
    y0 = float(text)

def submitN(text):
    global n
    n = float(text)

def submitB(text):
    global b
    b = float(text)

def submitIe(text):
    global Ie
    Ie = float(text)

def submitT(text):
    global T
    T = float(text)

#Create graph
fig, graph_axes = plt.subplots()
graph_axes.grid()
fig.subplots_adjust(left=0.07, right=0.95, top=0.95, bottom=0.4)

#Create add button
axes_button_add = plt.axes([0.55, 0.05, 0.4, 0.075])
button_add = Button(axes_button_add, 'Добавить')
button_add.on_clicked(onButtonAddClicked)

#Create clear buttin
axes_button_clear = plt.axes([0.05, 0.05, 0.4, 0.075])
button_clear = Button(axes_button_clear, 'Очистить')
button_clear.on_clicked(onButtonClearClicked)

#Create textbox for h, t, x0, y0, a
axbox = plt.axes([0.12, 0.25, 0.12, 0.075])
h_box = TextBox(axbox, 'Шаг h =', initial="0.01")
h_box.on_submit(submitH)

axbox = plt.axes([0.37, 0.25, 0.12, 0.075])
x0_box = TextBox(axbox, 'Задача   \nКоши x0 =', initial="1")
x0_box.on_submit(submitX0)

axbox = plt.axes([0.56, 0.25, 0.12, 0.075])
```

```
y0_box = TextBox(axbox, 'y0 =', initial= "0.1")
y0_box.on_submit(submitY0)

axbox = plt.axes([0.83, 0.25, 0.12, 0.075])
n_box = TextBox(axbox, 'Количество\n точек n =', initial="100")
n_box.on_submit(submitN)

axbox = plt.axes([0.16, 0.15, 0.12, 0.075])
a_box = TextBox(axbox, 'Параметры \n a =', initial= "0.7")
a_box.on_submit(submitA)

axbox = plt.axes([0.37, 0.15, 0.12, 0.075])
b_box = TextBox(axbox, 'b =', initial= "0.8")
b_box.on_submit(submitB)

axbox = plt.axes([0.56, 0.15, 0.12, 0.075])
Ie_box = TextBox(axbox, 'Ie =', initial= "0.5")
Ie_box.on_submit(submitIe)

axbox = plt.axes([0.83, 0.15, 0.12, 0.075])
T_box = TextBox(axbox, 'T =', initial= "12.5")
T_box.on_submit(submitT)

 plt.show()
```