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 $F(x, y(x), y'(x), y''(x), \ldots, y^{(n)}(x)) = 0$, 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:

$y^{(n)} = f(x, y, y', y'', \ldots, y^{(n-1)})$.

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: $y(x0) = y0, y'(x0) = y1, y''(x0) = y2, \ldots, y^{(n-1)}(x0) = yn - 1$.

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

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

Любое конкретное решение $y = \varphi(x)$ уравнения n –го порядка:

$F(x, y(x), y'(x), y''(x), \ldots, y^{(n)}(x)) = 0$, называется частным решением.

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

1. $\Phi(x, C1, C2, \ldots, Cn)$ является решением уравнения при любых допустимых значениях $C1, C2, \ldots, Cm$;

2. Для любых начальных данных $y(x0) = y0, y'(x0) = y1, y''(x0) = y2, \ldots, y^{(n-1)}(x0) = yn-1$, для которых задача Коши имеет единственное решение, существуют значения постоянных $C1 = A1, C2 = A2, \ldots, Cn = An$, такие что решение $y = \Phi(x, A1, A2, \ldots, An)$ 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 $\begin{matrix}\dot{x} = x - \frac{x^3}{3} - y + I_{ext} \\ T\dot{y} = x + a - by\end{matrix}$ методом Рунге-Кутта четвертого порядка. Получить практические навыки программирования.

Модель ФитцХью — Нагумо описывает прототип возбудимой системы (например, нейрона), является примером генератора релаксационных колебаний, вследствие того, что, если внешний стимул Iext 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

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:

$\dot{x} = x - \frac{x^3}{3} - y + I_{ext}$ методом Рунге-Кутта, а также строит графики фазовый портрет.
$T\dot{y} = x + a - by$

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. Обработчики событий позволяют пользователю взаимодействовать с формой при помощи клавиатуры и мыши.

Основные функции нужны для работы программы:

`def fx(x, y, Ie)` – функция Fx

`def fy(x, y, a, b, T)` – функция Fy

`def Runge_Kutt(x0, y0, n, h, t0)` – Метод Рунге-Кутта 4-ого порядка точности

`def addGraphPhase(graph_axes, x, y)` – Функция создающая поля осей

`def firstClick(event)` – Обработчик события, клик левой кнопки мыши по форме для получения начальных значений

`def tapSpace(event)` – Обработчик события, нажатие клавиши пробел для продолжения траектории

`def onButtonAddClicked(event)` – Обработчик события, нажатие на форме клавиши добавить (создает график по заданным параметрам)

`def onButtonClearClicked(event)` – Обработчик события, нажатие на форме клавиши очистить (удаление всех графиков)

`def onButtonCreateCliked(event)` - Обработчик события, вызывает отдельную функцию для работы с событиями (нажатие клавиши пробел, клик мыши)

Вспомогательные функции будут приведены в листинге программы.

При запуске программы вызывается интерфейс (Рис. 1):

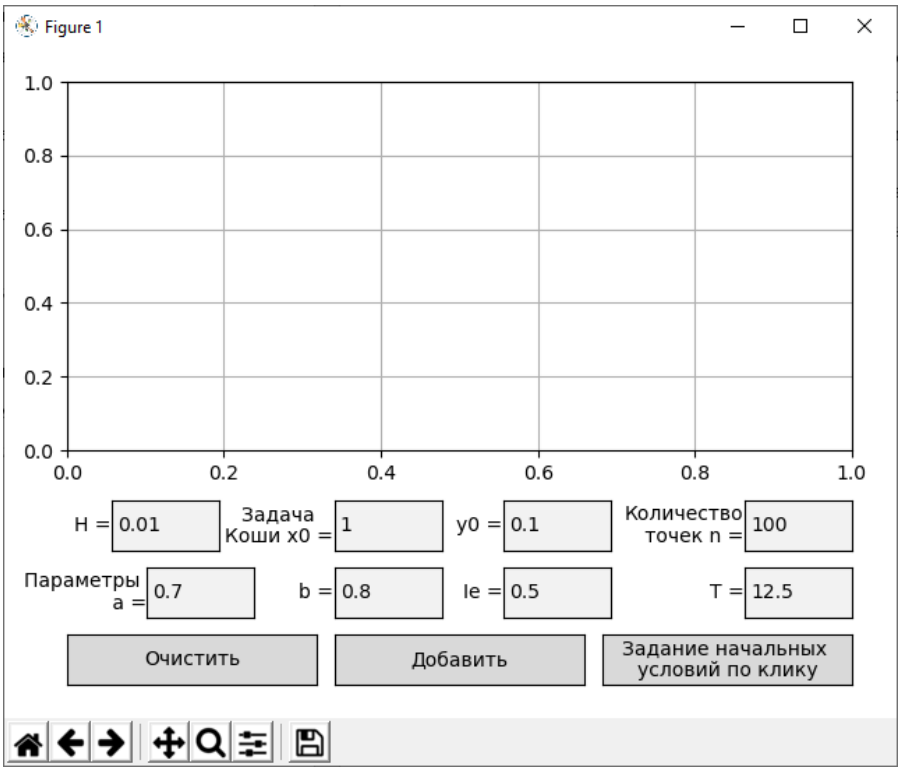

Рис1

В данной реализации возможно можно менять значения параметров и начальных условий в соответствующих окнах. Добавлена кнопка очищения графика.

Так же есть значения по умолчанию на случай, если оставить форму пустой. (Рис. 2)

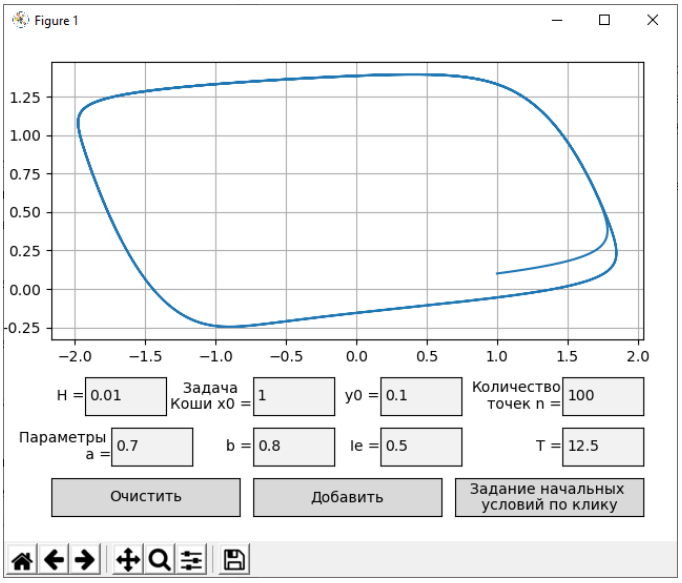

Рис2

Если пользователь решит ввести символы вместо числа, в консоли выскочит предупреждение и будет выведено значение по умолчанию. (Рис. 3)

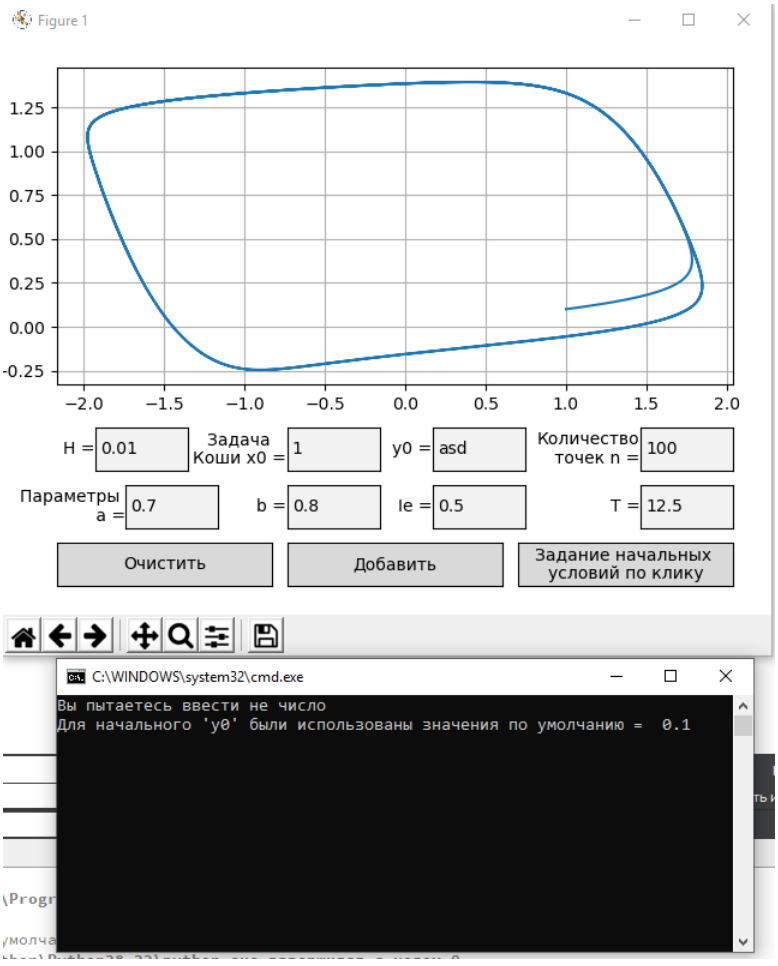

Рис. 3

При нажатии кнопки “Задание начальных условий по клику”, будет вызвана вспомогательная интерактивная панель позволяющая кликом мыши задавать начальные условия, и пробелом продолжать фазовую траекторию. (Рис. 4)

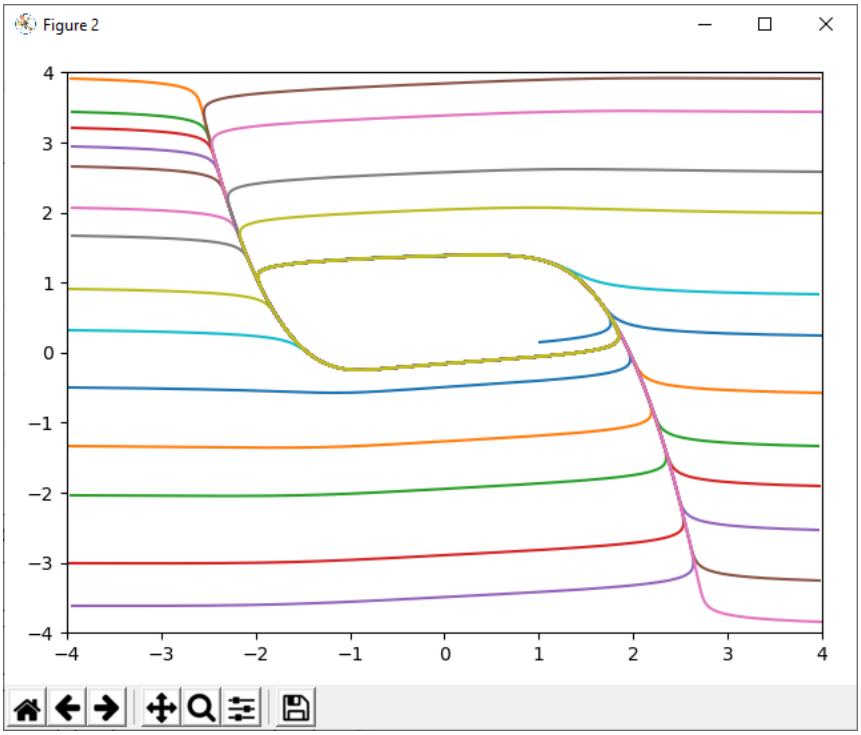

Рис. 4

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 $y_1, y_2, \dots y_i$. 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

Листинг

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

#Модель ФитцХью-Нагумо
#функция Fx
def fx(x, y, Ie):
    return float(x - x * x * x / 3. - y + Ie)
#функция Fy
def fy(x, y, a, b, T):
    return float((x + a - b * y) / T)

#Метод Рунге-Кутта 4-ого порядка точности
def Runge_Kutt(x0,y0,n,h, t0):
    x.append(x0)
    y.append(y0)
    i = 0
    yt = 0
    xt = 0
    while(t0 < n): #считаем коэффициенты
        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

#Функция создающая поля осей
def addGraphPhase(graph_axes, x, y):
     graph_axes.plot(x, y)
     plt.draw()

if __name__ == '__main__':
    x = list()
    y = list()
    t0 = 0.
```

```
#Обработчик события, клик левой кнопки мыши по форме для получения начальных значений
    def firstClick(event):
        x0 = event.xdata #получает начальные условия по клику мыши
        y0 = event.ydata
        global n, h, count, t0, x, y
        x = []
        y = []
        n = submitN(n)
        Runge_Kutt(x0,y0,n,h, t0) #вызывает метод Рунге-Кутта
        global x_, y_
        x_ = x.pop()
        x.append(x_)
        y_ = y.pop()
        y.append(y_)
        plt.plot(x, y)
        plt.draw()

#Обработчик события, нажатие клавиши пробел для продолжения траектории
    def tapSpace(event):
        global n, h, x_, y_, t0, x, y
        x0 = x_
        y0 = y_
        t0 = n
        n = n + n / 2
        x = []
        y = []
        Runge_Kutt(x0,y0,n,h, t0) #вызывает метод Рунге-Кутта
        x_ = x.pop()
        x.append(x_)
        y_ = y.pop()
        y.append(y_)
        plt.plot(x, y)
        plt.draw()

#Создает отдельную форму для работы с событиями(нажатие клавиши пробел, клик мыши)
    def identEvent():
        fig, ax = plt.subplots()
        global n, h, t0, sk
        fig.canvas.mpl_connect('button_press_event', firstClick)
        fig.canvas.mpl_connect('key_press_event', tapSpace)
        plt.tight_layout()
        plt.axis([-4, 4, -4, 4])
        plt.show()

#Обработчик события, нажатие на форме клавиши добавить(создает график по заданным
параметрам)
    def onButtonAddClicked(event):
        global graph_axes, x, y
        global x0, y0, n, h, a, b, Ie, T, t0 #получает значения написанные в полях для
ввода
        t0 = 0.
        x = []
        y = []
        n = submitN(n)
        Runge_Kutt(x0,y0,n,h, t0) #вызывает метод Рунге-Кутта
        addGraphPhase(graph_axes, x, y) #отрисовывает график
        x = []
        y = []

#Обработчик события, нажатие на форме клавиши очистить(удаление всех графиков)
    def onButtonClearClicked(event):
        global graph_axes
        graph_axes.clear()
        graph_axes.grid()
        plt.draw() #отображет созданные графики
```

```
#Обработчик события, вызывает отдельную функцию для работы с событиями(нажатие клавиши
пробел, клик мыши
    def onButtonCreateCliked(event):
        identEvent()
#Здесь и далее. Обработчики событий, которые считывают данные введенные в поля
    def submitA(text):
        global a
        try:
            a = float(text)
        except ValueError:
            print("Вы пытаетесь ввести не число")
            print("Для параметра 'a' были использованы значения по умолчанию = ", a)

    def submitH(text):
        global h
        try:
            h = float(text)
        except ValueError:
            print("Вы пытаетесь ввести не число")
            print("Для шага 'h' были использованы значения по умолчанию = ", h)

    def submitX0(text):
        global x0
        try:
            x0 = float(text)
        except ValueError:
            print("Вы пытаетесь ввести не число")
            print("Для начального 'x0' были использованы значения по умолчанию = ", x0)

    def submitY0(text):
        global y0
        try:
            y0 = float(text)
        except ValueError:
            print("Вы пытаетесь ввести не число")
            print("Для начального 'y0' были использованы значения по умолчанию = ", y0)

    def submitN(text):
        global n
        try:
            n = float(text)
            return n
        except ValueError:
            print("Вы пытаетесь ввести не число")
            print("Для количества точек'n'были использованы значения по умолчанию=", n)

    def submitB(text):
        global b
        try:
            b = float(text)
        except ValueError:
            print("Вы пытаетесь ввести не число")
            print("Для параметра 'b' были использованы значения по умолчанию = ", b)

    def submitIe(text):
        global Ie
        try:
            Ie = float(text)
        except ValueError:
            print("Вы пытаетесь ввести не число")
            print("Для параметра 'a' были использованы значения по умолчанию = ", Ie)
```

```
    def submitT(text):
        global T
        try:
            T = float(text)
        except ValueError:
            print("Вы пытаетесь ввести не число")
            print("Для параметра 'T' были использованы значения по умолчанию = ", T)
#Создает форму для графика
    fig, graph_axes = plt.subplots()
    graph_axes.grid()
    fig.subplots_adjust(left=0.07, right=0.95, top=0.95, bottom=0.4)
#Созданет кнопку добавить
    axes_button_add = plt.axes([0.37, 0.05, 0.28, 0.075])
    button_add = Button(axes_button_add, 'Добавить')
    button_add.on_clicked(onButtonAddClicked)
#Создает кнопку для задания начальных условий по клику мыши
    axes_button_create = plt.axes([0.67, 0.05, 0.28, 0.075])
    button_create = Button(axes_button_create, 'Задание начальных \nусловий по клику')
    button_create.on_clicked(onButtonCreateCliked)
#Создает кнопку очистки графиков, без перезапуска программы
    axes_button_clear = plt.axes([0.07, 0.05, 0.28, 0.075])
    button_clear = Button(axes_button_clear, 'Очистить')
    button_clear.on_clicked(onButtonClearClicked)
#Здесь и далее. Создает текстовые поля для h, n, x0, y0, a, b, T,
    axbox = plt.axes([0.12, 0.25, 0.12, 0.075])
    h_box = TextBox(axbox, 'H =', initial="0.01")
    h = 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 = 1.0
    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 = 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 = 100.0
    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 = 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 = 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 = 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 = 12.5
    T_box.on_submit(submitT)
    plt.show() #Отображет созданные графики
```