



Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет имени
Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехники и комплексной автоматизации»
КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ
по дисциплине «Вычислительная математика»

Студент:	Зотов Даниил Александрович
Группа:	РК6-53Б
Тип задания:	лабораторная работа
Тема:	Модель биологического нейрона

Студент _____
подпись, дата

Зотов Д. А.
Фамилия, И.О.

Преподаватель _____
подпись, дата

Фамилия, И.О.

Москва, 2021

Содержание

Модель биологического нейрона	3
1 Задание	3
2 Цель выполнения лабораторной работы	5
3 Выполненные задачи	5
4 Численное решение задачи Коши для системы двух ОДУ	6
4.1. Решение системы ОДУ с помощью явного метода Эйлера	7
4.2. Решение системы ОДУ с помощью неявного метода Эйлера	9
4.3. Решение системы ОДУ с помощью метода Рунге-Кутты 4-го порядка	10
5 Численное нахождение траекторий динамической системы	12
6 Анализ методов получения численного решения ОДУ	16
7 Моделирование импульсной нейронной сети по модели Ижикевича	17
8 Заключение	20

Модель биологического нейрона

1 Задание

Численные методы решения задачи Коши для систем обыкновенных дифференциальных уравнений (ОДУ) 1-го порядка активно используются далеко за пределами стандартных инженерных задач. Примером области, где подобные численные методы крайне востребованы, является нейробиология, где открытые в XX веке модели биологических нейронов выражаются через дифференциальные уравнения 1-го порядка. Математическая формализация моделей биологических нейронов также привела к появлению наиболее реалистичных архитектур нейронных сетей, известных как спайковые нейронные сети (Spiking Neural Networks). В данной лабораторной работе мы исследуем одну из простейших моделей подобного типа: модель Ижикевича.

Задача 1 (Модель Ижикевича)

Дана система из двух ОДУ 1-го порядка:

$$\begin{cases} \frac{dv}{dt} = f_1(u, v) = 0.04v^2 + 5v + 140 - u + I; \\ \frac{du}{dt} = f_2(u, v) = a(bv - u); \end{cases} \quad (1)$$

И дополнительные условия, определяющие возникновение импульса в нейроне:

$$v \geq 30, \begin{cases} v \leftarrow c; \\ u \leftarrow u + d; \end{cases} \quad (2)$$

где v – потенциал мембранны (мВ), u – переменная восстановления мембранны (мВ), t – время (мс), I – внешний ток, приходящий через синапс в нейрон от всех нейронов, с которыми он связан.

Описания параметров представленной системы:

a – задает временной масштаб для восстановления мембранны (чем больше a , тем быстрее происходит восстановление после импульса);

b – чувствительность переменной восстановления к флюктуациям разности потенциалов.

Таблица 1: Характерные режимы заданной динамической системы и соответствующие значения ее параметров

Режим	a	b	c	d
Tonic spiking (TS)	0.02	0.2	-65	6
Phasic spiking (PS)	0.02	0.25	-65	6
Chattering (C)	0.02	0.2	-50	2
Fast spiking (FS)	0.1	0.2	-65	2

c – значение потенциала мембранны сразу после импульса;

d – значение переменной восстановления мембранны сразу после импульса.

Требуется (базовая часть).

1. Реализовать следующие функции, каждая из которых возвращает дискретную траекторию системы ОДУ с правой частью, заданной функцией f , начальным условием x_0 , шагом по времени h и конечным временем t_n :
 - euler(x_0 , t_n , f , h), где дискретная траектория строится с помощью метода Эйлера;
 - implicit_euler(x_0 , t_n , f , h), где дискретная траектория строится с помощью неявного метода Эйлера;
 - runge_kutta(x_0 , t_n , f , h), где дискретная траектория строится с помощью метода Рунге–Кутта 4-го порядка.
2. Для каждого из реализованных методов численно найти траектории заданной динамической системы, используя шаг $h = 0.5$ и характерные режимы, указанные в таблице (1). В качестве начальных условий можно использовать $v(0) = c$ и $u(0) = bv(0)$. Внешний ток принимается равным $I = 5$
3. Вывести полученные траектории на четырех отдельных графиках как зависимости потенциала мембранны v от времени t , где каждый график должен соответствовать своему характерному режиму работы нейрона.
4. По полученным графикам кратко описать особенности указанных режимов.

Задача 2 Требуется (продвинутая часть).

1. Ответьте на следующие вопросы.
 - а) В чем принципиальные отличия реализованных методов друг от друга? В чем они схожи?
 - б) Для каждой из схем каково значение шага, при котором она становится неустойчивой?

- в) Какая из схем по вашему мнению предоставляет оптимальный баланс между устойчивостью решения и временем вычислений?
2. Произвести интегрирование во времени до 1000 мс нейронной сети с помощью метода Эйлера, используя следующую информацию.
- а) Динамика каждого нейрона в нейронной сети описывается заданной моделью Ижикевича. В нейронной сети имеется 800 возбуждающих нейронов и 200 тормозных. Возбуждающие нейроны имеют следующие значения параметров: $a, b = 0.2$, $c = -65 + 15\alpha^2$, $d = 8 - 6\beta^2$, и внешний ток в отсутствие токов от других нейронов равен $I = I_0 = 5\xi$, где α, β и ξ - случайные числа от 0 до 1(распределение равномерное).
- Тормозные нейроны имеют следующие значения параметров: $a = 0.02 + 0.08\gamma$, $b = 0.25 - 0.05\delta$, $c = -65$, $d = 2$ и внешний ток в отсутствие токов от других нейронов равен $I = I_0 = 2\zeta$, где γ, δ и ζ - случайные числа от 0 до 1. В качестве начальных условий используются значения $v(0) = -65$ и $u(0) = v(0)b$
- б) Нейронная сеть может быть смоделирована с помощью полного графа. Матрица смежности W этого графа описывает значения токов, передаваемых от нейрона i к нейрону j в случае возникновения импульса. То есть, при возникновении импульса нейрона j внешний ток связанного с ним нейрона i единовременно увеличивается на величину W_{ij} и затем сразу же падает до нуля, что и моделирует передачу импульса по нейронной сети. Значение W_{ij} равно 0.5θ , если нейрон j является возбуждающим, и $-\tau$, если тормозным, где θ и τ - случайные числа от 0 до 1.
3. Вывести на экран импульсы всех нейронов как функцию времени и определить частоты характерных синхронных (или частично синхронных) колебаний нейронов в сети.

2 Цель выполнения лабораторной работы

Цель выполнения лабораторной работы – изучение методов численного решения задачи Коши для систем ОДУ, таких, как метод Эйлера и метод Рунге-Кутты. Ознакомление с базовыми понятиями импульсной нейронной сети и моделью Ижикевича.

3 Выполненные задачи

1. Реализованы функции для нахождения численного решения системы ОДУ с помощью метода Эйлера (явного и неявного).

2. Реализована функция для нахождения численного решения системы ОДУ с помощью метода Рунге-Кутты.
3. Для реализованных функций численно найдены тректории заданной динамической системы.
4. По результатам численных решений системы, определены отличия и особенности режимов, указанных в таблице 1.
5. Смоделирована импульсная нейронная сеть на основе модели Ижикевича и выведен график зависимости возникновения импульса в нейроне от дискретизированного времени t_i .

4 Численное решение задачи Коши для системы двух ОДУ

В ходе лабораторной работы была рассмотрена система нормальных ОДУ (1), т. е. система, состоящая из ОДУ, разрешенных относительно производной:

$$y^{(n)}(t) = f(t, y, y', \dots, y^{(n-1)}), t \in [a; b]. \quad (3)$$

Нахождение решения ОДУ получено с помощью задачи Коши, т. е., задания начальных условий для переменных:

$$\begin{aligned} y(a) &= y_0, \\ y'(a) &= y_{01}, \\ &\dots \\ y^{(n-1)}(a) &= y_{0,n-1}. \end{aligned}$$

Подобное ОДУ n -го порядка (3) можно представить в виде системы ОДУ:

$$\frac{d}{dt} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} y_2 \\ y_3 \\ \vdots \\ f(t, y_1, \dots, y_n) \end{bmatrix}. \quad (4)$$

И, учитывая, что задача не одномерна, преобразуем (3) в n -мерную систему 1-го порядка в общем виде:

$$\frac{d}{dt} \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_n(t) \end{bmatrix} = \begin{bmatrix} f_1(t, y_1, \dots, y_n) \\ f_2(t, y_2, \dots, y_n) \\ \vdots \\ f_n(t, y_1, \dots, y_n) \end{bmatrix}. \quad (5)$$

Или, (6) в векторном виде:

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}). \quad (6)$$

Система вида (5) с начальными условиями $y_1(a) = \alpha_1, \dots, y_n(a) = \alpha_n$ имеет единственное решение для $t \in [a; b]$.

Все методы, использованные в ходе получения численного решения, предполагают собой дискретизацию координаты $t, t \in [a; b]$, как:

$$t_i = a + ih, \quad (7)$$

где $i = 1, \dots, m$; $h = \frac{b-a}{m} = t_{i+1} - t_i$ - шаг дискретизации. Это дает дискретизацию решения $y(t)$ в виде $y_i = y(t_i)$.

4.1. Решение системы ОДУ с помощью явного метода Эйлера

Рассмотрев ОДУ вида

$$\frac{dy}{dt} = f(t, y), \quad (8)$$

где $t \in [a; b]$ и начальное условие $y(a) = \alpha$ и учитывая дискретизацию координаты t (см. (7)), произведем разложение $y(t)$ в ряд Тейлора в точке t_i :

$$y(t) \approx y(t_i) + y'(t_i)(t - t_i). \quad (9)$$

Тогда, значение в точке t_{i+1} с учетом (9):

$$y(t_{i+1}) \approx y(t_i) + hf(t_i, y(t_i)). \quad (10)$$

Выражение (10) дает формулировку метода Эйлера:

$$w_0 = \alpha, \quad (11)$$

$$w_{i+1} = w_i + hf(t_i, w_i), \quad (12)$$

где $w_i \approx y(t_i)$. В данной лабораторной работе w_i - не одномерная точка, поэтому, преобразовав (11), (12) получим метод Эйлера для n -мерной системы:

$$\mathbf{w}_0 = \boldsymbol{\alpha}, \quad (13)$$

$$\mathbf{w}_{i+1} = \mathbf{w}_i + h\mathbf{f}(t_i, \mathbf{w}_i), \quad (14)$$

где $w_0 = [y_1(t_0), \dots, y_n(t_0)]$, $w_i = [y_1(t_i), \dots, y_n(t_i)]$.

Для получения численного решения системы ОДУ (1) методом Эйлера была реализована функция `euler(x_0, t_0, t_n, f, h, coeff)`. На вход функция принимает список x_0 , состоящий из начальных условий для функций $f_1(u, v), f_2(u, v)$ системы ОДУ (1), начальное и конечное время t_0 и t_n соответственно, список f , состоящий из $f_1(u, v), f_2(u, v)$ системы ОДУ (1), шаг дискретизации h и словарь `coeff`, содержащий в себе коэффициенты a, b, c, d из таблицы 1.

Алгоритм получения численного решения системы ОДУ с помощью метода Эйлера выглядит следующим образом:

- С помощью функции `numpy.arange` с началом в точке t_0 и концом в точке t_n с шагом h создается `numpy.array` t , т. е. дискретизированная по времени сетка вида (7).
- С помощью функции `numpy.zeros` создается `numpy.array` размерности $(\frac{t_n-t_0}{h}, n)$, где n - размерность вектора начальных условий x_0 . Таким образом, получен `numpy.array` для w , где $w_i = [y_1(t_i), \dots, y_n(t_i)]$. Так как в начальном состоянии данный массив пуст, то необходимо заполнить его элемент $w[0]$ начальными условиями x_0 .
- Для получения численного решения с помощью явного метода Эйлера, в цикле по дискретизированной временной сетке t заполняются элементы $w[i+1]$, $i = 0, \dots, n-1$ в соответствие с формулой (14).

Прим. автора: для удобства была реализована функция `derivative(func, w, t)`, принимающая на вход список из функций, текущий элемент $w[i]$ и текущее t . Функция расчитывает правую часть $\frac{dv}{dt}, \frac{du}{dt}$ системы ОДУ (1) и возвращает список из вычисленных значений. Данная функция реализована во избежание повторений участка кода в каждом из последующих методов и для удобства занесения значений в `numpy.array` w .

- После вычисления значений для $w[i+1]$ производится проверка на дополнительные условия (2).

Прим. автора: для удобства была реализована функция `condition_check(w, coeff)`, принимающая на вход текущий элемент $w[i+1]$ и словарь коэффициентов `coeff`. Данная функция проверяет элемент на дополнительные условия (2) и реализована во избежание повторений участка кода в каждом из последующих методов.

- После выполнения всех итераций цикла, функция возвращает `numpy.array` w . Данный массив содержит в себе численное решение системы ОДУ, полученное с помощью явного метода Эйлера.

Полный листинг функции получения численного решения системы ОДУ с помощью явного метода Эйлера приведен ниже (см. листинг 1).

Листинг 1. Реализация функции получения численного решения с помощью явного метода Эйлера.

```

1 def euler(x_0, t_0, t_n, f, h, coeff):
2     t = np.arange(t_0, t_n, h)
3
4     row = int(len(t))
5     col = int(len(x_0))
6
7     w = np.zeros((row, col))

```

```

8     w[0] = x_0
9     w[0] = condition_check(w[0], coeff)
10
11    i = 0
12    for ti in t[:-1]:
13        result = w[i] + h * derivative(f, w[i], ti)
14        w[i + 1] = result
15        w[i + 1] = condition_check(w[i + 1], coeff)
16        i += 1
17
18    return w

```

4.2. Решение системы ОДУ с помощью неявного метода Эйлера

Рассмотрим уравнение (14): $\mathbf{w}_{i+1} = \mathbf{w}_i + h\mathbf{f}(t_i, \mathbf{w}_i)$. Перенеся слагаемые из правой части уравнения в левую, получим уравнение вида:

$$\mathbf{w}_{i+1} - h\mathbf{f}(t_i, \mathbf{w}_i) - \mathbf{w}_i = 0. \quad (15)$$

Для нахождения решения с помощью неявного метода Эйлера значение функции берется на последующем временном слое t_{i+1} [2]. Тогда из (15) получим:

$$\mathbf{w}_{i+1} - h\mathbf{f}(t_{i+1}, \mathbf{w}_{i+1}) - \mathbf{w}_i = 0. \quad (16)$$

Приближенное значение на новом временном слое, полученное с помощью решения нелинейного уравнения (16) относительно \mathbf{w}_{i+1} , называется неявным методом Эйлера.

Для получения численного решения системы ОДУ (1) неявным методом Эйлера была реализована функция *implicit_euler(x_0, t_0, t_n, f, h, coeff)*. Параметры, принимаемые функцией на вход, аналогичны параметрам функции, представленной на листинге 1.

Алгоритм получения численного решения системы ОДУ с помощью неявного метода Эйлера выглядит следующим образом:

1. С помощью функции *numpy.arange* с началом в точке t_0 и концом в точке t_n с шагом h создается *numpy.array* t , т. е. дискретизированная по времени сетка вида (7).
2. С помощью функции *numpy.zeros* создается *numpy.array* размерности $(\frac{t_n-t_0}{h}, n)$, где n - размерность вектора начальных условий x_0 . Таким образом, получен *numpy.array* для \mathbf{w} , где $w_i = [y_1(t_i), \dots, y_n(t_i)]$. Так как в начальном состоянии данный массив пуст, то необходимо заполнить его элемент $w[0]$ начальными условиями x_0 .
3. Для получения численного решения с помощью неявного метода Эйлера, в цикле по дискретизированной временной сетке t заполняются элементы $w[i+1]$, $i =$

$0, \dots, n - 1$. С помощью функции `numpy.optimize.root` решается нелинейное уравнение (16) относительно элемента $w[i+1]$ и вычисленное значение записывается в $w[i+1]$.

4. После заполнения очередного элемента $w[i+1]$ для проверки дополнительных условий (2) вызывается функция `condition_check`.
5. После выполнения всех итераций цикла, функция возвращает `numpy.array w`. Данный массив содержит в себе численное решение системы ОДУ, полученное с помощью неявного метода Эйлера.

Полный листинг функции получения численного решения системы ОДУ с помощью неявного метода Эйлера приведен ниже (см. листинг 2).

Листинг 2. Реализация функции получения численного решения с помощью неявного метода Эйлера.

```

1 def implicit_euler(x_0, t_0, t_n, f, h, coeff):
2     t = np.arange(t_0, t_n, h)
3
4     row = int(len(t))
5     col = int(len(x_0))
6
7     w = np.zeros((row, col))
8     w[0] = x_0
9     w[0] = condition_check(w[0], coeff)
10
11    i = 0
12    for ti in t[:-1]:
13        f_t = lambda w1: w1 - h * derivative(f, w1, ti) - w[i]
14        sol = root(f_t, w[i])
15        w[i + 1] = sol.x
16        w[i + 1] = condition_check(w[i + 1], coeff)
17        i += 1
18
19    return w

```

4.3. Решение системы ОДУ с помощью метода Рунге-Кутты 4-го порядка

Методы Рунге-Кутты основаны на получении аппроксимации, позволяющей получить метод n -го порядка с такой функцией $\phi(t, y)$, что в ней присутствуют только вычисления значений функции $f(t, y)$ без каких-либо ее производных. Это достигается за счет использования некоторой обобщенной формы с неопределенными коэффициентами, которые затем находятся методом неопределенных коэффициентов.

Вывод метода Рунге-Кутты 4-го порядка алгебраически трудоемок, поэтому формулировка метода Рунге-Кутты 4-го порядка для систем ОДУ (с учетом того, что $\mathbf{w}_i =$

$[f_1(t_i), \dots, f_n(t_i)]$) выглядит следующим образом [1]:

$$\mathbf{w}_0 = \boldsymbol{\alpha}, \quad (17)$$

$$\mathbf{k}_1 = h\mathbf{f}(t_i, \mathbf{w}_i), \quad (18)$$

$$\mathbf{k}_2 = h\mathbf{f}\left(t_i + \frac{h}{2}, \mathbf{w}_i + \frac{1}{2}\mathbf{k}_1\right), \quad (19)$$

$$\mathbf{k}_3 = h\mathbf{f}\left(t_i + \frac{h}{2}, \mathbf{w}_i + \frac{1}{2}\mathbf{k}_2\right), \quad (20)$$

$$\mathbf{k}_4 = h\mathbf{f}(t_i + h, \mathbf{w}_i + \mathbf{k}_3), \quad (21)$$

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4), i = 0, \dots, n - 1. \quad (22)$$

Для получения численного решения системы ОДУ (1) методом Рунге-Кутты 4-го порядка была реализована функция `runge_kutta(x_0, t_0, t_n, f, h, coeff)`. Параметры, принимаемые функцией на вход, аналогичны параметрам функции, представленной на листинге 1.

Алгоритм получения численного решения системы ОДУ с помощью метода Рунге-Кутты 4-го порядка выглядит следующим образом:

1. С помощью функции `numpy.arange` с началом в точке t_0 и концом в точке t_n с шагом h создается `numpy.array` t , т. е. дискретизированная по времени сетка вида (7).
2. С помощью функции `numpy.zeros` создается `numpy.array` размерности $(\frac{t_n-t_0}{h}, n)$, где n - размерность вектора начальных условий x_0 . Таким образом, получен `numpy.array` для \mathbf{w} , где $w_i = [y_1(t_i), \dots, y_n(t_i)]$. Так как в начальном состоянии данный массив пуст, то необходимо заполнить его элемент $w[0]$ начальными условиями x_0 .
3. Определяются четыре лямбда-функции для вычисления значений $k_i, i = 1, \dots, 4$, соответствующие формулам (18)-(21).
4. Для получения численного решения с помощью метода Рунге-Кутты 4-го порядка, в цикле по дискретизированной временной сетке t заполняются элементы $w[i+1], i = 0, \dots, n - 1$. Для получения очередного значения $w[i+1]$ производятся вычисления, соответствующие формуле (22) с вызовами лямбда функций, определенных в п. 3 алгоритма, для вычисления значений k_i .
5. После заполнения очередного элемента $w[i+1]$ для проверки дополнительных условий (2) вызывается функция `condition_check`.
6. После выполнения всех итераций цикла, функция возвращает `numpy.array` w . Данный массив содержит в себе численное решение системы ОДУ, полученное с помощью метода Рунге-Кутты 4-го порядка.

Полный листинг функции получения численного решения системы ОДУ с помощью метода Рунге-Кутты 4-го порядка приведен ниже (см. листинг 3).

Листинг 3. Реализация функции получения численного решения с помощью метода Рунге-Кутты 4-го порядка.

```

1 def runge_kutta(x_0, t_0, t_n, f, h, coeff):
2     t = np.arange(t_0, t_n, h)
3
4     row = int(len(t))
5     col = int(len(x_0))
6
7     w = np.zeros((row, col))
8     w[0] = x_0
9     w[0] = condition_check(w[0], coeff)
10
11    k1 = lambda w, ti: h * derivative(f, w, ti)
12    k2 = lambda w, ti: h * derivative(f, w + 0.5 * k1(w, ti), ti + h/2)
13    k3 = lambda w, ti: h * derivative(f, w + 0.5 * k2(w, ti), ti + h/2)
14    k4 = lambda w, ti: h * derivative(f, w + k3(w, ti), ti + h)
15
16    i = 0
17    for ti in range(len(t) - 1):
18        w[i + 1] = w[i] + (1/6) * (k1(w[i], ti) + 2 * k2(w[i], ti) + 2 * k3(w[i],
19        ti) + k4(w[i], ti))
20        w[i + 1] = condition_check(w[i + 1], coeff)
21        i += 1
22
23    return w

```

5 Численное нахождение траекторий динамической системы

Для численного нахождения траекторий заданной динамической системы (1) используются режимы, описанные в таблице 1, значение шага $h = 0.1$ (значение шага изменено для прослеживания динамики возникновения импульсов), начальное значение времени t_0 принимается равным нулю, конечное значение времени $t_n = 300$. Внешний ток принимается равным $I = 5$. Начальные условия для системы ОДУ (1) определены как

$$v(0) = c, \quad (23)$$

$$u(0) = bv(0). \quad (24)$$

Таким образом, с учетом начальных условий (23), (24), $w_0 = [v_0, u_0] = [c, bv_0]$. Программно определим параметры системы (см. листинг 4).

Численное решение системы ОДУ (1) получено с помощью вызова функций *euler*, *implicit_euler*, *runge_kutta*, соответствующих методам Эйлера (явный и неявный) и методу Рунге-Кутты 4-го порядка (см. листинг 5).

Листинг 4. Определение параметров системы.

```
1 I = 5
2 t0 = 0
3 tn = 300
4 h = 0.1
5 table = [
6     {'name': 'Tonic Spiking', 'a': 0.02, 'b': 0.2, 'c': -65, 'd': 6},
7     {'name': 'Phasic Spiking', 'a': 0.02, 'b': 0.25, 'c': -65, 'd': 6},
8     {'name': 'Chattering', 'a': 0.02, 'b': 0.2, 'c': -50, 'd': 2},
9     {'name': 'Fast Spiking', 'a': 0.1, 'b': 0.2, 'c': -65, 'd': 2}
10    ]
11    t = np.arange(t0, tn, h)
12    for item in table:
13        a = item['a']
14        b = item['b']
15        dvdt = lambda w: 0.04 * (w[0] ** 2) + 5 * w[0] + 140 - w[1] + I
16        dudt = lambda w: a * (b * w[0] - w[1])
17
18        func = [dvdt, dudt]
19        start_conditions = np.array([item['c'], item['c'] * item['b']])
```

Листинг 5. Получение численного решения системы ОДУ.

```
1 w1 = euler(start_conditions, t0, tn, func, h, item)
2 w2 = implicit_euler(start_conditions, t0, tn, func, h, item)
3 w3 = runge_kutta(start_conditions, t0, tn, func, h, item)
```

Вывод полученных траекторий для четырех режимов из таблицы 1 представлен на рис. 1-4.

Проанализировав графики на рис. 1-4, можно сделать некоторые выводы об особенностях каждого из режимов.

1. Для режима «Tonic Spiking» характерна более низкая частота пульсаций относительно других режимов. Этому способствует специфичный набор параметров: низкое значение параметра a , из-за которой восстановление после импульса происходит медленнее, низкая чувствительность к восстановлению вследствие небольшого значения параметра b и довольно большое значение параметра d , которое описывает значение переменной восстановления после импульса.
2. Для режима «Phasic Spiking» характерна более высокая частота пульсаций по сравнению с режимом «Tonic Spiking» вследствие увеличения чувствительности к восстановлению.

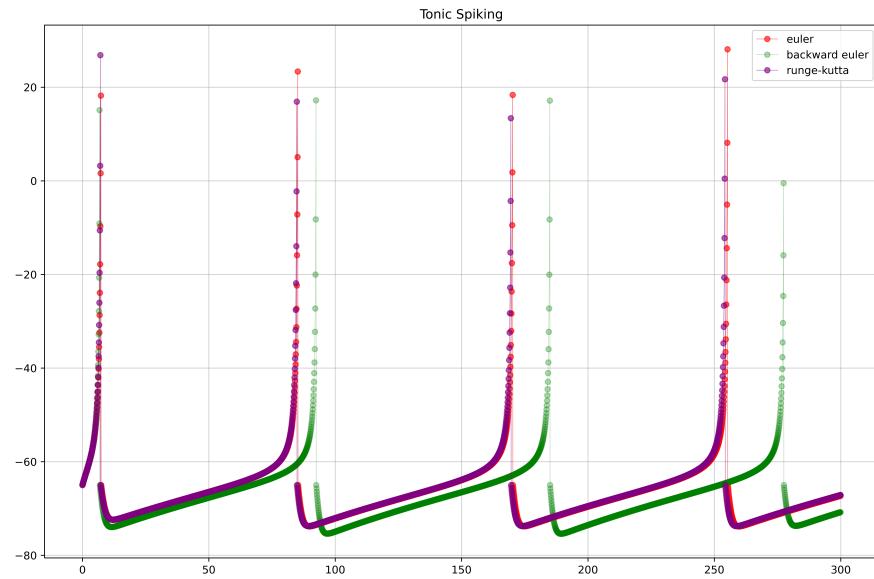


Рис. 1. Численно полученная траектория для режима Tonic Spiking

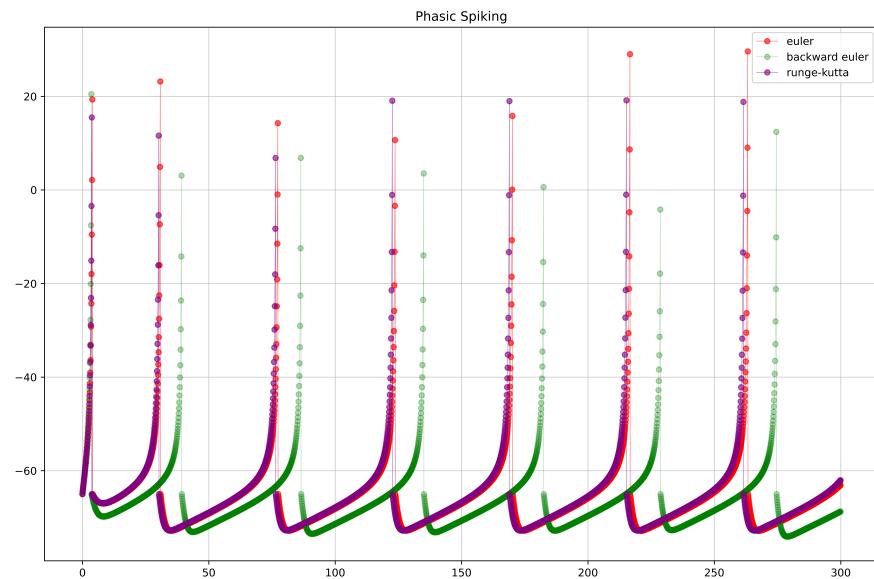


Рис. 2. Численно полученная траектория для режима Phasic Spiking

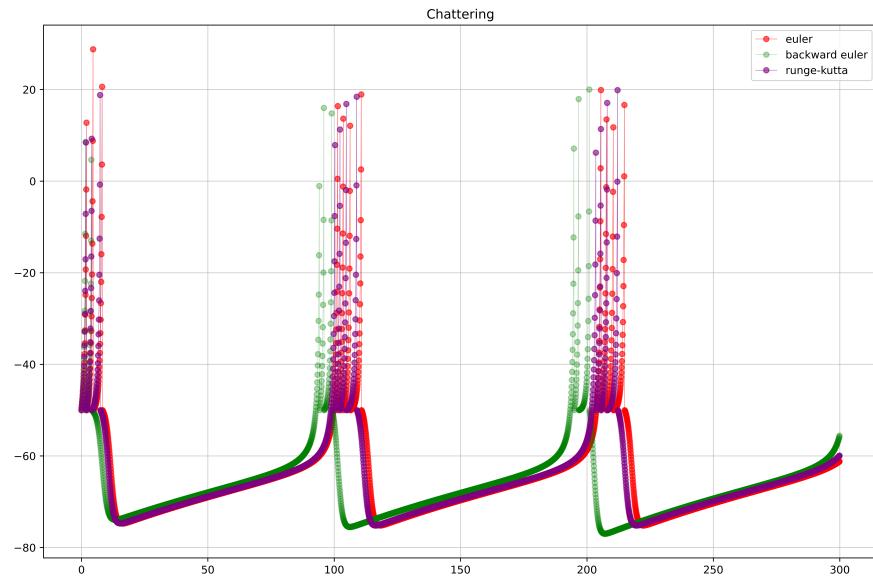


Рис. 3. Численно полученная траектория для режима Chattering

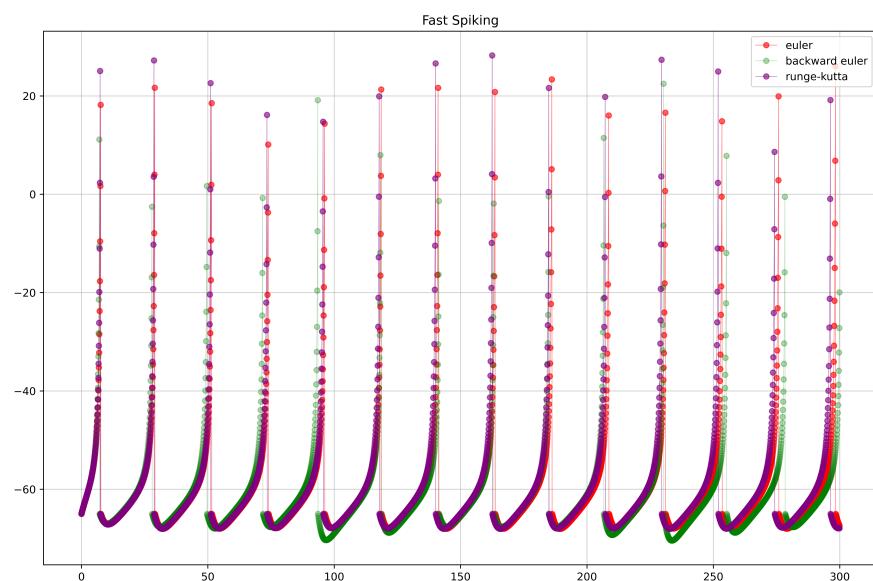


Рис. 4. Численно полученная траектория для режима Fast Spiking

3. Для режима «Chattering» частота пульсаций схожа с режимом «Tonic Spiking». Однако, из-за более низкого значения параметра d по отношению к первому режиму, и другого значения параметра c , сразу после импульса можно наблюдать несколько "скачков" подряд.
4. Для режима «Fast Spiking» характерна самая высокая частота пульсаций. Это связано с довольно большим (по отношению к другим режимам) значением параметра a , вследствие чего восстановление после импульса происходит быстрее, а также уменьшения параметра d .

6 Анализ методов получения численного решения ОДУ

Представленные методы получения численного решения систем ОДУ схожи в своем алгоритме работы: на каждой итерации вычисляется последующее значение на основе значения с предыдущего шага.

Для оценки времени работы каждого из методов, в начало функций была добавлена переменная, вычисляемая с помощью функции `time.time()` и равная текущему времени, установленному на вычислительной машине. При передаче управления из функции в основную часть программы, переменная вычисляется еще раз и, разность переменной из функции и переменной из основной части программы является примерным временем работы метода.

Таблица 2. Время выполнения методов получения численного решения систем ОДУ для системы (1).

Режим	Явный м. Эйлера	Неявный м. Эйлера	Рунге-Кутты
Tonic spiking (TS)	0.01999	0.17	0.15
Phasic spiking (PS)	0.03	0.19	0.15
Chattering (C)	0.01999	0.18	0.14
Fast spiking (FS)	0.0099	0.2	0.15
Среднее	0.01997	0.185	0.1475

Первым принципиальным различием между методами стоит отметить их время работы. В таблице 2 приведены значения времени выполнения методов для нахождения численного решения системы ОДУ (1) в секундах. Можно заметить, что время выполнения явного метода Эйлера сильно отличается от времени выполнения неявного метода Эйлера и метода Рунге-Кутты. Это объясняется тем, что функция $f(u, v)$ из правой части уравнений вызывается единожды на каждой итерации и не требует

сложных вычислений. Однако, учитывая то, что локальная погрешность явного метода Эйлера равна $O(h^2)$, а его глобальная погрешность $\frac{b-a}{h}O(h^2) \sim O(h)$ [1], можно утверждать, что данный метод обладает меньшей точностью по сравнению, например, с методом Рунге-Кутты 4-го порядка.

Неявный метод Эйлера обладает теми же характеристиками погрешности, что и явный метод Эйлера. Однако, его главным недостатком является решение нелинейного уравнения, что предполагает большое количество обращений к функции, стоящей в правой части уравнения. Несмотря на то, что методы используют, в целом, похожие уравнения для вычисления решения, из-за нахождения решения нелинейного уравнения количество времени, затраченное на получение численного решения, заметно возрастает по сравнению с явным методом Эйлера.

Метод Рунге-Кутты 4-го порядка, как можно заметить из таблицы 2, выполняется для каждого из режимов в среднем одно и то же количество времени. Локальная погрешность метода Рунге-Кутты 4-го порядка равна $O(h^4)$, а глобальная погрешность - $O(h^3)$, за счет чего метод можно считать более точным по отношению к методам Эйлера. Метод Рунге-Кутты использует относительно небольшое количество вычислений на каждой из итераций, учитывая, что время выполнения функции занимает меньшее количество времени, чем решение нелинейного уравнения и при этом имеет большую точность.

Исходя из представленных выше суждений, можно утверждать, что в данной задаче метод Рунге-Кутты 4-го порядка является самым оптимальным методом получения численного решения систем ОДУ по совокупности критериев времени выполнения и точности метода.

7 Моделирование импульсной нейронной сети по модели Ижикевича

Произведем моделирование нейронной сети, используя модель Ижикевича. Пусть в нейронной сети имеется 800 возбуждающих нейронов и 200 тормозных. Параметры нейронов представлены в таблице 3.

Таблица 3. Параметры нейронов.

Вид нейрона	a	b	c	d	I
Возбуждающий	0.02	0.2	$-65 + 15\alpha^2$	$8 - 6\beta^2$	5ξ
Тормозящий	$0.02 + 0.08\gamma$	$0.25 - 0.05\omega$	-65	2	2ζ

где $\alpha, \beta, \xi, \gamma, \omega, \zeta$ - случайные числа от 0 до 1.

Программное описание вектора параметров для 800 возбуждающих и 200 тормозных нейронов представлено на листинге 6. В качестве начальных условий используются значения $v(0) = -65$, $u(0) = bv(0)$. Сама нейронная сеть моделируется с помощью пол-

Листинг 6. Программное описание параметров для нейронов.

```

1  a_v = np.ones(800)
2  a_v = a_v * 0.02
3  b_v = np.ones(800)
4  b_v = b_v * 0.2
5  c_v = np.zeros(800)
6  c_v = -65 + 15 * (np.random.default_rng().random(nv) ** 2)
7  d_v = np.zeros(800)
8  d_v = 8 - 6 * (np.random.default_rng().random(nv) ** 2)
9  a_t = np.zeros(200)
10 a_t = 0.02 + 0.08 * np.random.default_rng().random(nt)
11 b_t = np.zeros(200)
12 b_t = 0.25 - 0.05 * np.random.default_rng().random(nt)
13 c_t = np.ones(200)
14 c_t = c_t * (-65)
15 d_t = np.ones(200)
16 d_t = d_t * 2
17 a = np.append(a_v, a_t)
18 b = np.append(b_v, b_t)
19 c = np.append(c_v, c_t)
20 d = np.append(d_v, d_t)

```

ного графа. Значения токов, которые передаются от нейрона к нейрону будем описывать матрицей смежности \mathbf{W} . Программно генерируем матрицу смежности как 1000 элементов, где первые 800 - возбуждающие нейроны, оставшиеся 200 - тормозные. Ток от нейрона равен 0.5θ , если нейрон является возбуждающим и $-\tau$, если тормозным. Программное описание матрицы смежности представлено на листинге 7.

Листинг 7. Описание начальных параметров и матрицы смежности.

```

1  v = np.ones((1000)) * (-65.)
2  u = b * v
3
4  W1 = np.random.default_rng().random((1000, 800)) * 0.5
5
6  W2 = np.random.default_rng().random((1000, 200)) * (-1)
7
8  W = np.hstack((W1, W2))

```

Для того, чтобы построить график зависимости номера нейрона от времени, когда возник импульс, необходимо двигаясь в цикле по дискретизированной сетке времени запоминать соответственно значения номера и момента времени импульса. На каждой итерации вектор v будет проверяться на возникновение импульса ($v \geq 30$) и результатом такой проверки будет вектор, состоящий из булевых значений *True/False*. Для

нейронов, в которых условие истинно, запоминается момент времени t_i и номер нейрона.

В конце каждой итерации находится численное решение ОДУ с помощью явного метода Эйлера. Так как шаг по времени в данном случае принимается равным за единицу, необходимо учесть, что за единицу времени выполняется $\frac{1}{h}$ итераций для метода Эйлера. Программная реализация описанного цикла представлена на листинге 8.

Листинг 8. Реализация дискретизации по времени.

```
1  for ti in t:
2      f_matrix = v >= 30
3      for i, is_impulse in enumerate(f_matrix):
4          if is_impulse:
5              impulse.append({'time': ti, 'id': i})
6      v[f_matrix] = c[f_matrix]
7      u[f_matrix] = u[f_matrix] + d[f_matrix]
8
9      I = np.hstack((5 * np.random.default_rng().random(800), 2 * np.random.
10                 default_rng().random(200)))
11      I = I + np.sum(W[:, f_matrix], axis=1)
12      f1 = lambda v1, u1, I1: 0.04 * (v1 ** 2) + 5 * v1 + 140 - u1 + I1
13      f2 = lambda v2, u2, a2, b2: a2 * (b2 * v2 - u2)
14
15      i = 0
16      while (i < int(1/h)):
17          v_old = v
18          u_old = u
19          v = v_old + h * f1(v_old, u_old, I)
20          u = u_old + h * f2(v_old, u_old, a, b)
          i += 1
```

Вывод зависимости времени от номера нейрона, в котором возник импульс реализован с помощью преобразования списка из словарей к виду *pandas.DataFrame*. Таким образом, легко отличить возбуждающие нейроны от тормозящих: возбуждающие нейроны имеют номер от 0 до 799, а тормозящие - от 800 до 999, т. е., расположены с сохранением нумерации, по которой они были заданы в матрице смежности \mathbf{W} . График зависимости времени от номера нейрона, в котором возник импульс, представлен на рис. 5.

Частота полученных колебаний непостоянна и в зависимости от сгенерированных параметров равна 4-8 Гц.

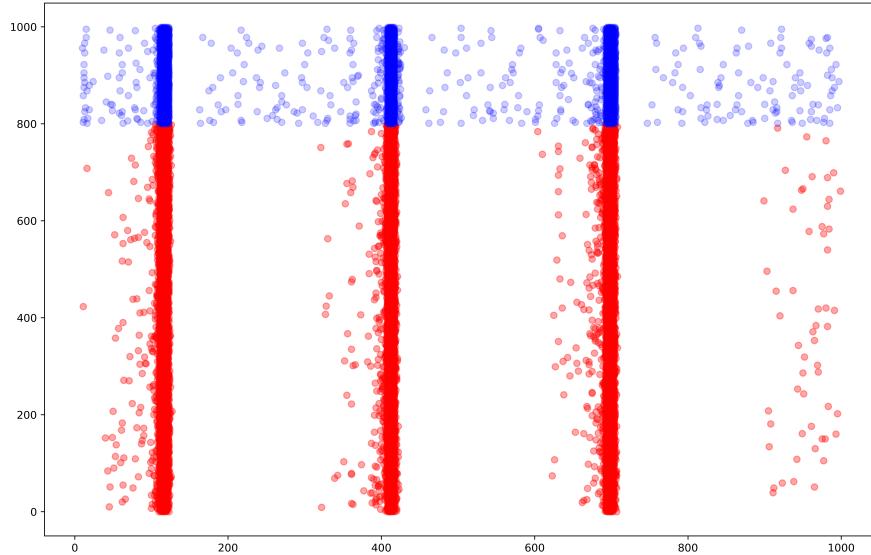


Рис. 5. График возникновения импульса в нейронах

8 Заключение

1. В ходе выполнения лабораторной работы были реализованы методы получения численного решения систем ОДУ такие, как явный и неявный методы Эйлера и метод Рунге-Кутты 4-го порядка. В результате сранения данных методов, оптимальным методом для данной задачи выбран метод Рунге-Кутты за счет более высокой точности по сравнению с другими методами. Учитывая хороший порядок точности погрешности метода, его время выполнения относительно небольшое и приемлемо.
2. Определено время выполнения методов для данной задачи. По данному критерию самым быстрым оказался явный метод Эйлера. Скорость выполнения данного метода объясняется тем, что функция обращается к правой части уравнения всего один раз за итерацию.
3. После построения графиков численно полученных траекторий для режимов из таблицы 1 были сделаны выводы об особенностях режимов и скорости восстановления мембранны.
4. В результате моделирования импульсной нейронной сети по модели Ижикевича

были выявлены колебания частотой 4-8 Гц. Из-за случайной генерации параметров нельзя точно определить количество колебаний в ИНС.

Список использованных источников

1. Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018-2021. С. 140.
2. Задача Коши для обыкновенных дифференциальных уравнений [Электронный ресурс]. Режим доступа: <https://slemeshevsky.github.io/num-mmf/ode/html/ode-FlatUI.html>

Выходные данные

Зотов Д. А.. *Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика».* [Электронный ресурс] — Москва: 2021. — 21 с. URL: <https://sa2systems.ru:88> (система контроля версий кафедры PK6)

Постановка: © ассистент кафедры PK-6, PhD А.Ю. Першин

Решение и вёрстка: © студент группы PK6-53Б, Зотов Д. А.

2021, осенний семестр