



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:	Горелкина Елизавета Егоровна
Группа:	РК6-52Б
Тип задания:	лабораторная работа
Тема:	Модель биологического нейрона

Студент _____
подпись, дата

Горелкина Е.Е.
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 Реализация функции численного решения задачи Коши ОДУ методом Эйлера	5
5 Реализация функции численного решения системы ОДУ неявным методом Эйлера	8
6 Реализация функции численного решения системы ОДУ методом Рунге-Кутта 4-го порядка	9
7 Численное интегрирование модели Ижикевича	10
8 Анализ особенностей режимов динамической системы	14
9 Сравнительный анализ реализованных методов	19
10 Реализация функций для моделирования нейронной сети	23
11 Моделирование нейронной сети и определение частот колебаний нейронов	27
12 Заключение	33

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 будет исследована одна из простейших моделей подобного типа: модель Ижикевича.

Задача 20(Модель Ижикевича)

Дана система из двух ОДУ 1-го порядка:

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

и дополнительного условия, определяющего возникновение импульса в нейроне:

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

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

Данная система имеет параметры:

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

b – чувствительность переменной восстановления к флуктуациям разности потенциалов;

c – значение потенциала мембранны сразу после импульса;

d – значение переменной восстановления мембранны сразу после импульса.

Таблица 3: 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

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.

Требуется (**продвинутая часть**).

1. Объяснить, в чем состоят принципиальные отличия реализованных методов? В чем они схожи?

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: $a = 0.02$, $b = 0.2$, $c = -65 + 15\alpha$, $d = 8 - 6\beta^2$ 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 $u(0) = bv(0)$.

б) Нейронная сеть может быть смоделирована с помощью полного графа. Матрица смежности W этого графа описывает значения токов, передаваемых от нейрона к нейрону 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

Целью данной работы является изучение численных методов решения задачи Коши для системы ОДУ.

В базовой части данной работы предлагается познакомиться с тремя различными методами численного решения задачи Коши для системы ОДУ на примере модели Ижикевича биологического нейрона.

В продвинутой части предлагается проанализировать различия и сходства реализованных методов, а также проинтегрировать нейронную сеть, где каждый нейрон описывается моделью Ижикевича.

3 Выполненные задачи

1. На языке программирования Python реализована функция, возвращающая дискретную траекторию системы ОДУ, где дискретная траектория строится с помощью метода Эйлера.
2. На языке программирования Python реализована функция, возвращающая дискретную траекторию системы ОДУ, где дискретная траектория строится с помощью неявного метода Эйлера.
3. На языке программирования Python реализована функция, возвращающая дискретную траекторию системы ОДУ, где дискретная траектория строится с помощью метода Рунге-Кutta 4-го порядка.
4. Для каждого из реализованных методов численно найдена траектория заданной динамической системы (1) для каждого из четырех режимов (таблица 1). На четырех отдельных графиках выведены полученные траектории как зависимости потенциала мембрани v от времени t , где каждый график соответствует своему режиму работы.
5. Кратко описаны особенности указанных режимов динамической системы.
6. Проанализированы особенности реализованных методов: найдены принципиальные отличия и сходства.
7. Определены все необходимые функции для моделирования динамики нейронной сети.
8. Произведено интегрирование во времени нейронной сети. Построен график зависимости номера нейрона, в котором произошел импульс, от момента времени, в который он произошел. Определены частоты синхронных колебаний нейронов в сети.

4 Реализация функции численного решения задачи Коши ОДУ методом Эйлера

Для реализации функции численного решения задачи Коши для системы ОДУ методом Эйлера необходимо воспользоваться рядом следующих определений из лекций [1].

Нормальным обыкновенным дифференциальным уравнением n -го порядка является уравнение вида:

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

где a, b – вещественные числа, $y^{(k)}$ – k -ая производная функции $y(t)$.

Для случая ОДУ 1-го порядка, выражение (3) принимает вид:

$$y'(t) = f(t, y), \quad t \in [a, b]. \quad (4)$$

Тогда система ОДУ 1-го порядка из n уравнений выглядит следующим образом:

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

что эквивалентно:

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}), \quad t \in [a, b], \quad (6)$$

где \mathbf{y} – вектор функций $y_i(t)$, \mathbf{f} – вектор функций из правой части выражения (5).

Задача Коши для системы ОДУ 1-го порядка из n уравнений формулируется следующим образом:

$$\begin{cases} y_1(a) = \alpha_1 \\ y_2(a) = \alpha_2 \\ \dots \\ y_n(a) = \alpha_n \end{cases}, \quad (7)$$

где $a, \alpha_1, \alpha_2, \dots, \alpha_n$ – вещественные числа.

Все методы, рассматриваемые в данной работе, включая метод Эйлера, предполагают дискретизацию координаты t в сетку вида $t_i = a + ih$, $i = 1, \dots, m$, где $h = \frac{b-a}{m} = t_{i+1} - t_i$ называют шагом, а m – это количество узлов [1].

Без потери общности метод Эйлера для численного решения задачи Коши для ОДУ 1-го порядка (4) с начальными условиями $y(a) = \alpha$ будет сформулирован следующим образом. Пусть $y(t) \in C^2[a; b]$. Пусть функция $y(t)$ разложена в ряд Тейлора в точке t_i . Тогда значение ряда Тейлора для функции $y(t)$ в точке t_{i+1} равно:

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2 y''(\xi_i)}{2}, \quad (8)$$

где $\xi_i \in (t_i, t_{i+1})$. Теперь предположив, что h мало, можно отбросить член порядка $O(h^2)$, что дает формулировку метода Эйлера:

$$\begin{aligned} w_0 &= \alpha, \\ w_{i+1} &= w_i + hf(t_i, w_i), \quad i = 0, 1, \dots, m-1, \end{aligned} \quad (9)$$

где $w_i \approx y(t_i)$.

Обобщая (9) на случай n ОДУ, можно получить метод Эйлера для системы ОДУ 1-го порядка из n уравнений:

$$\begin{aligned} \mathbf{w}_0 &= \boldsymbol{\alpha}, \\ \mathbf{w}_{i+1} &= \mathbf{w}_i + h \cdot \mathbf{f}(t_i, \mathbf{w}_i), \quad i = 0, 1, \dots, m - 1, \end{aligned} \tag{10}$$

где $\boldsymbol{\alpha}$ – вектор начальных условий $y_1(a) = \alpha_1, y_2(a) = \alpha_2, \dots, y_n(a) = \alpha_n$ размера n , а \mathbf{w}_i – вектор решений ОДУ размера n на $i - 1$ шаге.

Реализация функции `euler(x_0, t_n, f, h)`, приведенная в листинге 1, не предполагает привязки к какой-то конкретной системе ОДУ, за исключением лишь того факта, что в ней предусмотрена проверка на ограничение значений решений задачи Коши. Для модели Ижикевича этой проверкой является дополнительное условие (2).

Входные данные функции `euler(x_0, t_n, f, h)`: x_0 – список с начальными условиями, элементы которого равны значениям вектора $\boldsymbol{\alpha}$, t_n – целочисленная переменная, равная b для $t \in [a, b]$ (здесь $a = 0$), f – список функций, который содержит определения функций из правой части (5) и еще одно определение функции для проверки дополнительного условия, h – переменная с плавающей точкой, равная шагу по времени h .

Алгоритм, реализованный в функции `euler(x_0, t_n, f, h)`, следующий. Сначала вычисляется значение n – количество уравнений в системе ОДУ. Затем создается список w , куда заносится список с начальными условиями x_0 . Далее в цикле от h до t_n с шагом h происходит численное решение задачи Коши для системы ОДУ, согласно (10). В конец w добавляется пустой список. В $w[-1]$ в цикле по уравнениям системы добавляется очередное значение $w_{i+1,j}$. Таким образом, в итоге $w[-1]$ содержит значения \mathbf{w}_{i+1} . На следующем шаге алгоритма происходит проверка дополнительного условия путем вызова функции $f[-1]$, так как текущая реализация предполагает, что данная функция – последний элемент списка функций f .

Выходные данные функции `euler(x_0, t_n, f, h)`: w – список со списками, в которых элементы равны вычисленным значениям \mathbf{w}_{i+1} для каждого t_i , что и является дискретной траекторией системы ОДУ.

Листинг 1: Определение функции `euler(x_0, t_n, f, h)` численного решения задачи Коши для системы ОДУ методом Эйлера.

```
def euler(x_0, t_n, f, h):
    n = len(f) - 1
    w = [x_0]

    for t in np.arange(h, t_n + h, h):
        w.append([])
        for j in range(n):
            w[-1].append(w[-2][j] + h * f[j](t, w[-2]))
    w[-1] = f[-1](w[-1])

    return w
```

5 Реализация функции численного решения системы ОДУ неявным методом Эйлера

Формулировка неявного метода Эйлера похожа на формулировку явного метода Эйлера, рассмотренного выше. Однако имеется существенное различие. Согласно лекциям [1], формулировка неявного метода Эйлера для системы ОДУ из n уравнений имеет вид:

$$\begin{aligned} \mathbf{w}_0 &= \boldsymbol{\alpha}, \\ \mathbf{w}_{i+1} &= \mathbf{w}_i + h \cdot \mathbf{f}(t_i, \mathbf{w}_{i+1}), \quad i = 0, 1, \dots, m - 1, \end{aligned} \tag{11}$$

где все обозначения аналогичны обозначениям в явном методе Эйлера.

Как видно из (11), второе уравнение является нелинейным в общем случае относительно вектора \mathbf{w}_{i+1} . Значит, необходимо решить нелинейную систему уравнений для нахождения численного решения задачи Коши с помощью неявного метода Эйлера. В общем случае, такие системы не всегда удается решить аналитически. Поэтому найти решение можно, например, с помощью функции `root` из модуля `scipy.optimize` языка программирования Python. Согласно документации [2], в данной функции реализованы различные решатели систем нелинейных уравнений, выбор которых можно осуществить с помощью параметра `method`, передаваемого в указанную функцию. Для данной реализации неявного метода Эйлера в качестве значения параметра `method` было выбрано значение `df-sane`, его алгоритм подробно описан в статье [3].

В листинге 2 представлена реализация функции `implicit_euler(x_0, t_n, f, h)` точно так же без привязки как какой-либо определенной системе ОДУ, аналогично тому, как это было реализовано в `euler(x_0, t_n, f, h)`.

Входные данные функции `implicit_euler(x_0, t_n, f, h)` аналогичны входным данным функции `euler(x_0, t_n, f, h)`.

Алгоритм, реализованный в функции `implicit_euler(x_0, t_n, f, h)`, в отличие от алгоритма для метода Эйлера, предполагает определение функции `sys_eq(eq)`, которая задает систему уравнений, согласно второму выражению в (11). В этой функции \mathbf{x} – список с элементами, равными значениям вектора решений данной системы относительно \mathbf{w}_{i+1} . Далее, как и в `euler(x_0, t_n, f, h)` идет цикл по времени. В нем сначала численно решается система нелинейных уравнений с помощью функции `root`. В качестве первых двух аргументов ей передается имя функции, задающей систему нелинейных уравнений, `sys_eq` и начальное значение, с которого решатель функции `root` начнет итерации. В качестве начального значения передается $\mathbf{w}[-1]$ – предыдущее значение вектора решений \mathbf{w} . Результат работы функции `root` присваивается переменной \mathbf{x}_0 . Далее аналогично алгоритму в `euler(x_0, t_n, f, h)`, за тем лишь исключением, что вместо $\mathbf{w}[-2]$ в $\mathbf{f}[i]$ передается \mathbf{x}_0 .

Выходные данные функции `implicit_euler(x_0, t_n, f, h)` аналогичны выходным данным функции `euler(x_0, t_n, f, h)`.

Листинг 2: Определение функции `implicit_euler(x_0, t_n, f, h)` численного решения задачи Коши для системы ОДУ неявным методом Эйлера.

```

def implicit_euler(x_0, t_n, f, h):
    n = len(f) - 1
    w = [x_0]

    def sys_eq(eq):
        x = eq
        w_ = [x[i] - w[-1][i] - h * f[0](t, x) for i in range(n)]
        return w_

    for t in np.arange(h, t_n + h, h):
        x_0 = root(sys_eq, w[-1], method = 'df-sane').x
        w.append([])
        for i in range(n):
            w[-1].append(w[-2][i] + h * f[i](t, x_0))

    w[-1] = f[-1](w[-1])

    return w

```

6 Реализация функции численного решения системы ОДУ методом Рунге-Кутта 4-го порядка

Согласно лекциям [1], формулировка метода Рунге-Кутта 4-го порядка для системы ОДУ 1-го порядка имеет вид:

$$\begin{aligned}
 \mathbf{w}_0 &= \boldsymbol{\alpha}, \\
 \mathbf{k}_1 &= h \cdot \mathbf{f}(t_i, \mathbf{w}_i), \\
 \mathbf{k}_2 &= h \cdot \mathbf{f}\left(t_i + \frac{h}{2}, \mathbf{w}_i + \frac{1}{2}\mathbf{k}_1\right), \\
 \mathbf{k}_3 &= h \cdot \mathbf{f}\left(t_i + \frac{h}{2}, \mathbf{w}_i + \frac{1}{2}\mathbf{k}_2\right), \\
 \mathbf{k}_4 &= h \cdot \mathbf{f}(t_i + h, \mathbf{w}_i + \mathbf{k}_3), \\
 \mathbf{w}_{i+1} &= \mathbf{w}_i + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4), \quad i = 0, 1, \dots, m-1,
 \end{aligned} \tag{12}$$

где все обозначения аналогичны обозначениям в явном методе Эйлера.

Данный метод реализован в функции `runge_kutta(x_0, t_n, f, h)`, описанной в листинге 3.

Входные данные функции `runge_kutta(x_0, t_n, f, h)` аналогичны входным данным функции `euler(x_0, t_n, f, h)`.

Алгоритм, реализованный в функции `runge_kutta(x_0, t_n, f, h)`, следующий. До цикла по времени все действия аналогичны действиям в `euler(x_0, t_n, f, h)`. Далее в цикле по времени заполняются списки $\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4$ элементы которых равны значениям векторов $\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4$ (12). Для поэлементного суммирования списков используется функция `zip`. Далее в список \mathbf{w} добавляется результат вычисления значения \mathbf{w}_{i+1} (последняя строка в (12)). Проверяется дополнительное условие.

Выходные данные функции `runge_kutta(x_0, t_n, f, h)` аналогичны выходным данным функции `euler(x_0, t_n, f, h)`.

Листинг 3: Определение функции `runge_kutta(x_0, t_n, f, h)` численного решения задачи Коши для системы ОДУ методом Рунге-Кутта 4-го порядка.

```
def runge_kutta(x_0, t_n, f, h):
    n = len(f) - 1
    w = [x_0]

    for t in np.arange(h, t_n + h, h):
        k1 = np.array([h * f[i](t, w[-1]) for i in range(n)])
        k2 = np.array([h * f[i](t + h / 2, [x + y / 2 for x, y in zip(w[-1], k1)]) for i in range(n)])
        k3 = np.array([h * f[i](t + h / 2, [x + y / 2 for x, y in zip(w[-1], k2)]) for i in range(n)])
        k4 = np.array([h * f[i](t + h, [x + y for x, y in zip(w[-1], k3)]) for i in range(n)])
        w.append(list([x + (p + 2*y + 2*z + q) / 6 for x, p, y, z, q in zip(w[-1], k1, k2, k3, k4)]))

    w[-1] = f[-1](w[-1])

return w
```

7 Численное интегрирование модели Ижикевича

Чтобы найти численно траектории заданной динамической системы для каждого из реализованных методов, определим функцию, которая будет возвращать функции, соответствующие системе ОДУ (1) и дополнительному условию (2). В листинге 4 реализована функция `function(dsm)`, которая в качестве аргумента получает список значений параметров для определенного режима работы динамической системы. В этой функции определены три необходимые функции: первая `dv_dt(t, y)` возвращает значение первого уравнения системы ОДУ, вторая `du_dt(t, y)` возвращает значение второго уравнения системы ОДУ, третья `if_pulse(y)` проверяет выполнение дополнительного условия и в соответствии с этим возвращает измененный список значений `y`, согласно (2), если условие выполнено, иначе возвращает неизмененный список. Функция `function(dsm)` возвращает список из этих трех функций.

Стоит отметить, что сигнатура функций `dv_dt(t, y)` и `du_dt(t, y)` включает параметр `t`, хотя он нигде не используется, так как в системе (1) переменная `t` в явном виде отсутствует. Однако, как оговаривалось ранее, все три метода численного решения задачи Коши реализованы как можно более универсально, поэтому элементы списка функций, получаемые в результате вызова `function(dsm)`, требуют наличие параметра `t`.

Листинг 4: Определение функции `function(dsm)`, которая возвращает функции, соответствующие системе ОДУ (1) и дополнительному условию (2).

```
def function(dsm):
    def dv_dt(t, y):
        return 0.04 * y[0] ** 2 + 5 * y[0] + 140 - y[1] + 5

    def du_dt(t, y):
        return dsm[0] * (dsm[1] * y[0] - y[1])

    def if_pulse(y):
        if y[0] >= 30:
            return [dsm[2], y[1] + dsm[3]]
        else:
            return y

    return [dv_dt, du_dt, if_pulse]
```

Теперь можно создать списки из функций для четырех режимов работы. Для этого удобно хранить параметры режимов работы в словаре DSM, где ключ – название режима, значение – список параметров. Здесь же содается функция `f`, которая является списком из списков функций, где каждый список соответствует списку, возвращаемому `function(dsm)`. Все это реализовано в листинге 5.

Листинг 5: Создание словаря DSM и списка из списков функций `f`.

```
DSM = {'TS' : [0.02, 0.2, -65, 6],
       'PS' : [0.02, 0.25, -65, 6],
       'C' : [0.02, 0.2, -50, 2],
       'FS' : [0.1, 0.2, -65, 2]}

f = [function(DSM['TS']), function(DSM['PS']), function(DSM['C']), function(DSM['FS'])]
```

Дальнейшая реализация: генерация списков с траекториями и функции вывода графиков, – подробно представлены в файле `compmath_lab3.ipynb`.

На рисунках 1 - 3, представленных ниже, отображены графики для всех трех реализованных методов, с помощью каждого из них были вычислены дискретные траектории системы ОДУ для четырех различных режимов.

Далее будет проведен анализ особенностей каждого из четырех режимов и сравнительный анализ каждого из трех реализованных методов.

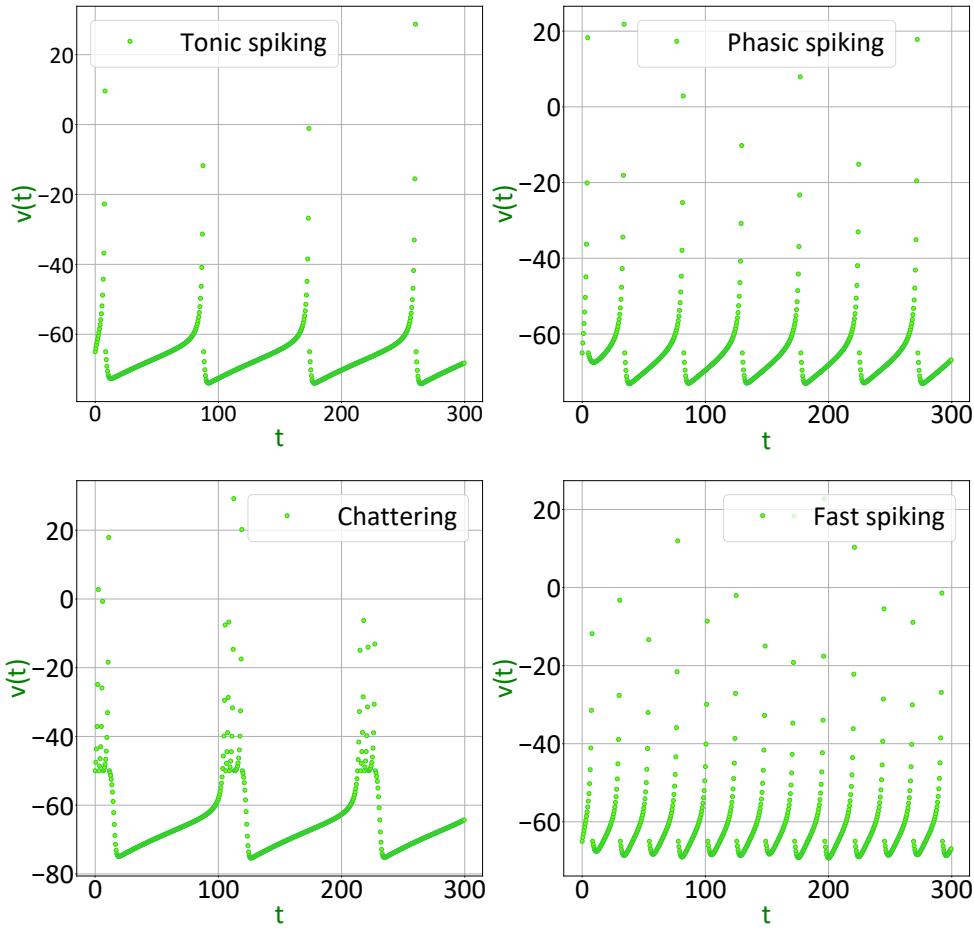


Рисунок 1. Графики зависимости дискретных траекторий $v(t)$ от времени t из системы ОДУ (1), построенных с помощью метода Эйлера, для четырех указанных режимов из таблицы 1.

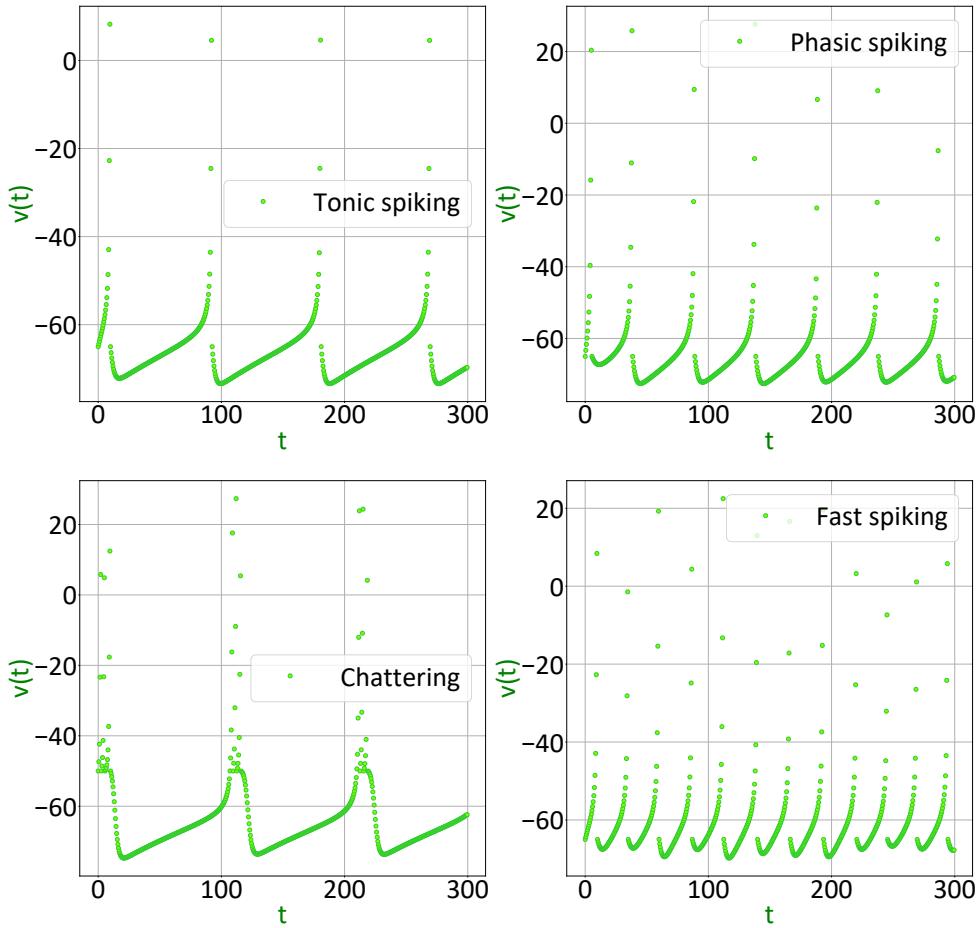


Рисунок 2. Графики зависимости дискретных траекторий $v(t)$ от времени t из системы ОДУ (1), построенных с помощью неявного метода Эйлера, для четырех указанных режимов из таблицы 1.

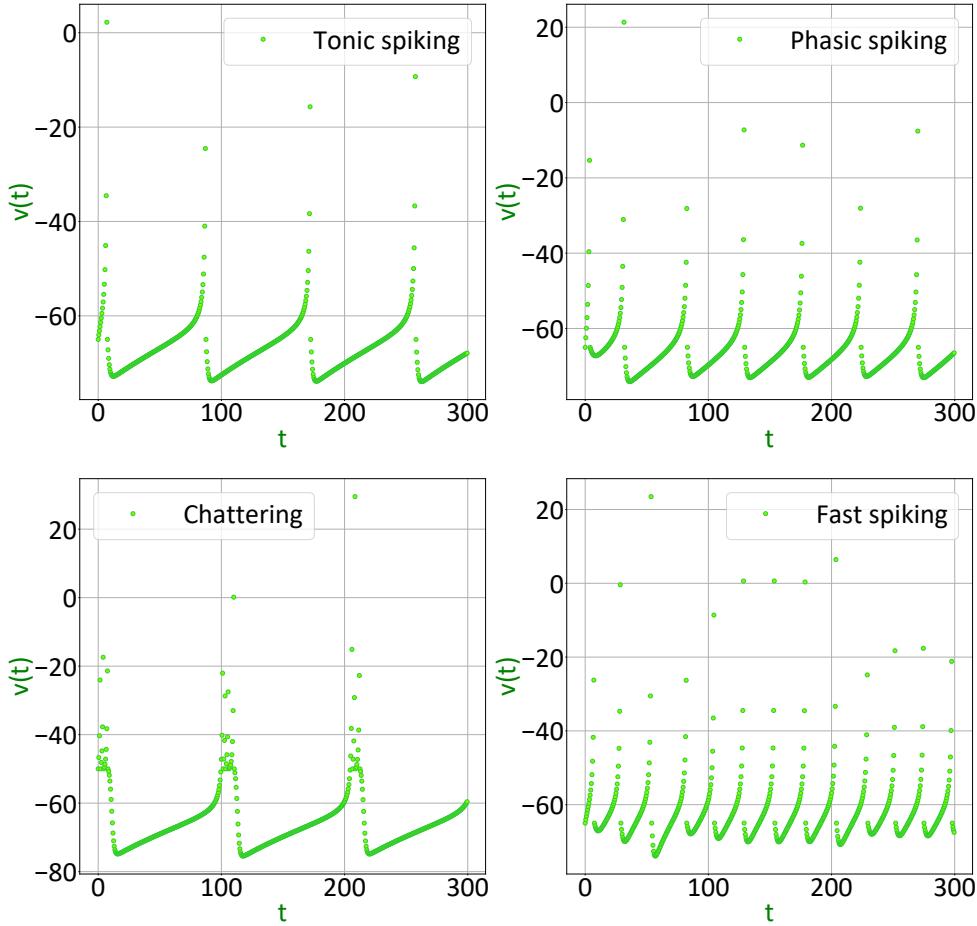


Рисунок 3. Графики зависимости дискретных траекторий $v(t)$ от времени t из системы ОДУ (1), построенных с помощью метода Рунге-Кутта 4-го порядка, для четырех указанных режимов из таблицы 1.

8 Анализ особенностей режимов динамической системы

Для анализа особенностей режимов динамической системы нужно обратиться к таблице 1, в которой указаны параметры каждого режима. Также понадобятся графики, на которых будут отображены как дискретная траектория $v(t)$, так и дискретная траектория $u(t)$ в зависимости от времени. Можно ограничиться использованием результата работы одного какого-либо метода численного решения задачи Коши, реализованного ранее. В данном случае, решения получены с помощью метода Эйлера.

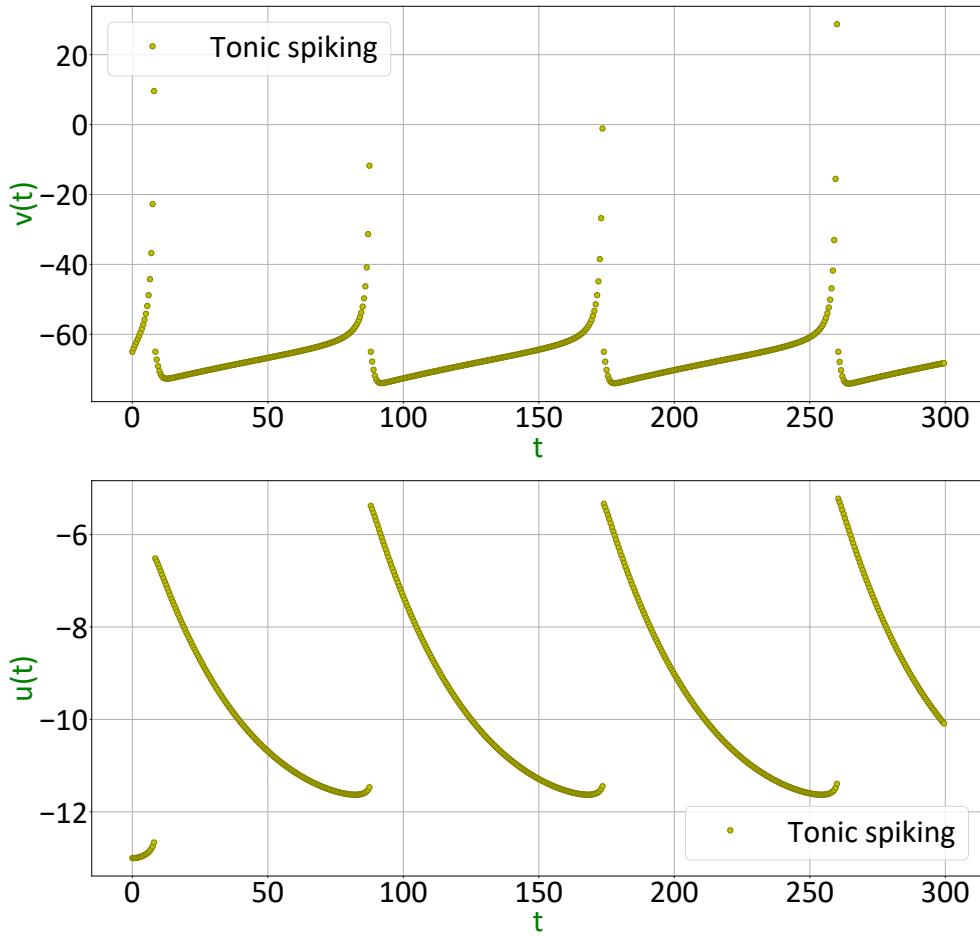


Рисунок 4. Графики зависимости дискретных траекторий $v(t)$ и $u(t)$ от времени t для режима Tonic spiking, построенных с помощью метода Эйлера.

На рисунке 4 изображены дискретные траектории для режима Tonic spiking. Почти в самом начале временного промежутка наблюдается импульс (график $v(t)$), восстановление длится около 80 мс (примерная разница между максимумами на $u(t)$), что объяснимо достаточно маленьким значением параметра $a = 0.02$, отвечающего за временной масштаб восстановления, а также большим значением параметра $d = 6$, отвечающего за значение переменной восстановления (u) сразу после импульса.

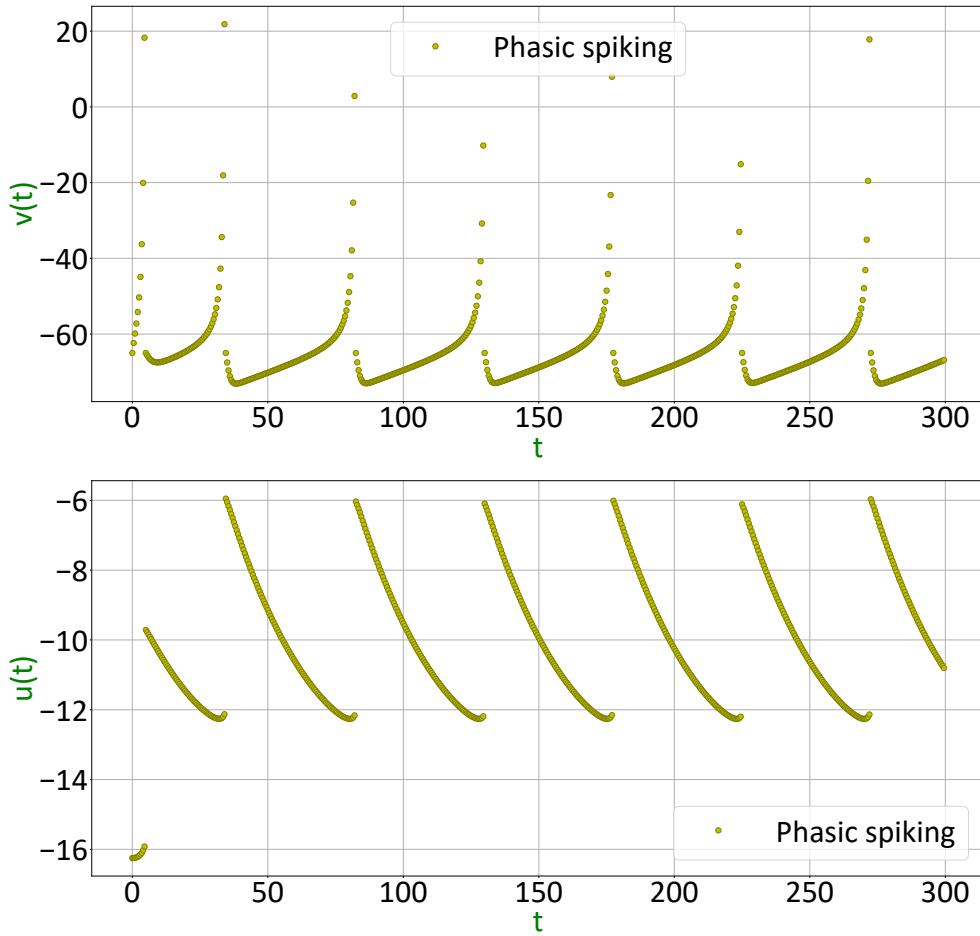


Рисунок 5. Графики зависимости дискретных траекторий $v(t)$ и $u(t)$ от времени t для режима Phasic spiking, построенных с помощью метода Эйлера.

На рисунке 5 изображены дискретные траектории для режима Phasic spiking. В самом начале также наблюдается импульс. Количество импульсов за время 300 мс в режиме Tonic spiking равно 4 – количество разрывов на графике $v(t)$ (рис. 4), в режиме Phasic spiking это количество равно 7. То есть частота импульсов в данном режиме почти в 2 раза больше, чем в предыдущем. Это можно объяснить тем, что значение переменной $b = 0.25$ для данного режима больше, чем значение переменной $b = 0.2$ для режима Tonic spiking. Как указано в условии задания, данная переменная отвечает за чувствительность переменной восстановления мембранны к флюктуациям. Для Phasic spiking чувствительность к флюктуациям выше, следовательно, импульсы происходят чаще. Время восстановления примерно равно 50 мс.

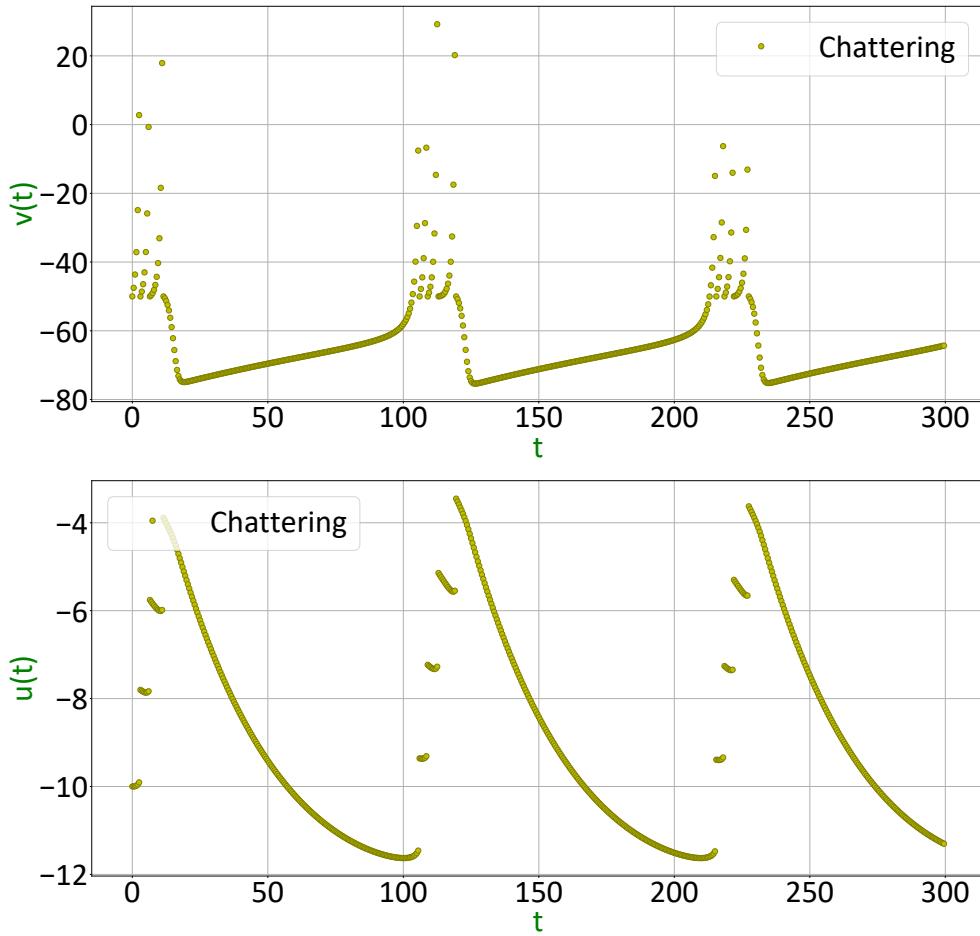


Рисунок 6. Графики зависимости дискретных траекторий $v(t)$ и $u(t)$ от времени t для режима Chattering, построенных с помощью метода Эйлера.

На рисунке 6 изображены дискретные траектории для режима Chattering. На графике $u(t)$ имеются частые разрывы, возникающие с определенной периодичностью. Наблюдаются близко расположенные подряд идущие импульсы (3 импульса в промежутке около 10 мс), после которых следует достаточно долгий период восстановления. Это можно объяснить тем, что значение $c = -50$ потенциала мембранны сразу после импульса для данного режима выше, чем у рассмотренных ранее двух режимов (для них $c = -65$). Из-за чего и возникают подряд идущие импульсы. Однако значение $d = 2$ переменной восстановления мембранны сразу после импульса ниже (у рассмотренных ранее $d = 6$), поэтому период время восстановления около 100 мс.

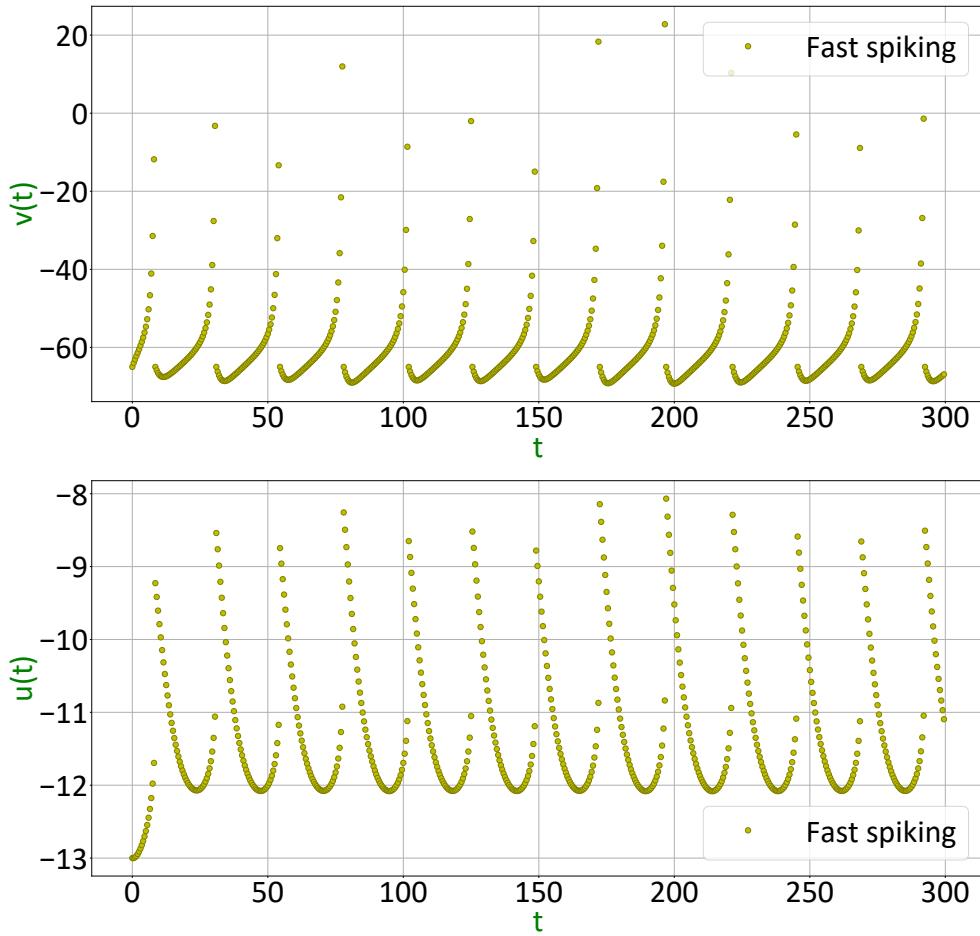


Рисунок 7. Графики зависимости дискретных траекторий $v(t)$ и $u(t)$ от времени t для режима Fast spiking, построенных с помощью метода Эйлера.

На рисунке 7 изображены дискретные траектории для режима Fast spiking. По графикам видно, что его частота более, чем в 3 раза выше, чем у режима Tonic Spiking (количество разрывов на графике $v(t)$ равно 13). При этом разница между двумя этими режимами заключается в том, что переменная $a = 0.1$ для режима Fast spiking в 5 раз больше переменной $a = 0.02$ для режима Tonic Spiking, означает, что восстановление после импульса в режиме Fast spiking происходит в 5 раз быстрее, чем в режиме Tonic Spiking. Однако из-за более низкого значения $d = 2$ для Fast spiking частота импульсов отличается менее, чем в 5 раз. Время восстановления около 20 мс.

9 Сравнительный анализ реализованных методов

На рисунках 8 - 11 приведены графики зависимости траекторий $v(t)$ для трех реализованных методов. На каждом графике свой режим работы динамической системы.

Уже из рисунков видно, что результаты численного решения задачи Коши для трех методов немного отличаются, графики совпадают не полностью. Отличия наиболее явные в местах, где происходит разрыв.

Из формулы (8) остаточный член ряда Тейлора, отброшенный при переходе к методу Эйлера, является остаточным членом второго порядка. Отсюда локальная погрешность метода Эйлера на шаге t_i равна:

$$L = y(t_i) - w_i = O(h^2). \quad (13)$$

Учитывая, что всего в результате численного решения задачи Коши для ОДУ (или системы ОДУ) требуется $\frac{T}{h}$ шагов, где T – время, в течение которого происходит интегрирование (длина отрезка интегрирования $b - a$ для $t \in [a, b]$ из (4)), глобальная, или накопленная погрешность, метода Эйлера равна:

$$G = O\left(h^2 \cdot \frac{T}{h}\right) = O(h). \quad (14)$$

Значит, метод Эйлера является методом 1-го порядка точности и при этом имеет локальную погрешность $O(h^2)$ на каждом шаге.

Так как явные методы почти всегда условно устойчивы [1], то метод Эйлера считается также условно устойчивым.

Что касается вычислений, то с помощью явного метода Эйлера они происходят быстрее всего, так как функция из правой части уравнения (10) в программной реализации вызывается один раз в цикле по времени.

Неявный метод Эйлера имеет те же локальную (13) и глобальную (14) погрешности, что и явный метод Эйлера. Однако важно учитывать, что в данной работе при реализации метода Эйлера было использовано численное решение нелинейной системы уравнений, которое имеет также свою погрешность. Поэтому что-то сказать о реальной погрешности сложно.

Неявный метод Эйлера не такой быстрый, как явный метод Эйлера, потому что для каждого шага требовалось решать нелинейную систему уравнений.

Неявные методы считаются абсолютно устойчивыми [1]. Но если используется численное решение нелинейной системы уравнений, то могут возникать погрешности уже метода решения данной системы. По этой же причине неявный метод Эйлера может требовать большого числа операций и временных затрат.

Что касается метода Рунге-Кутта 4-го порядка, как было показано в лекциях [1], он имеет локальную погрешность $O(h^5)$ на каждом шаге (остаточный член ряда Тейлора для функции, разложенной до 4 членов). Тогда, исходя из тех же соображений, как для

(14), глобальная погрешность метода Рунге-Кутта 4-го порядка равна $O(h^4)$. То есть метод Рунге-Кутта 4-го порядка является методом 4-го порядка точности. При этом на каждом шаге требуется 4 раза вычислить функцию \mathbf{f} . Также стоит отметить, что метод Рунге-Кутта 4-го порядка все еще является достаточно устойчивым, хоть и условно. С повышением порядка точности область устойчивости становится меньше. При этом также повышается сложность вычислений. Таким образом, метод Рунге-Кутта 4-го порядка является оптимальным методом, где соблюден баланс между устойчивостью, точностью и вычислительной сложностью, из всех трех рассмотренных методов.

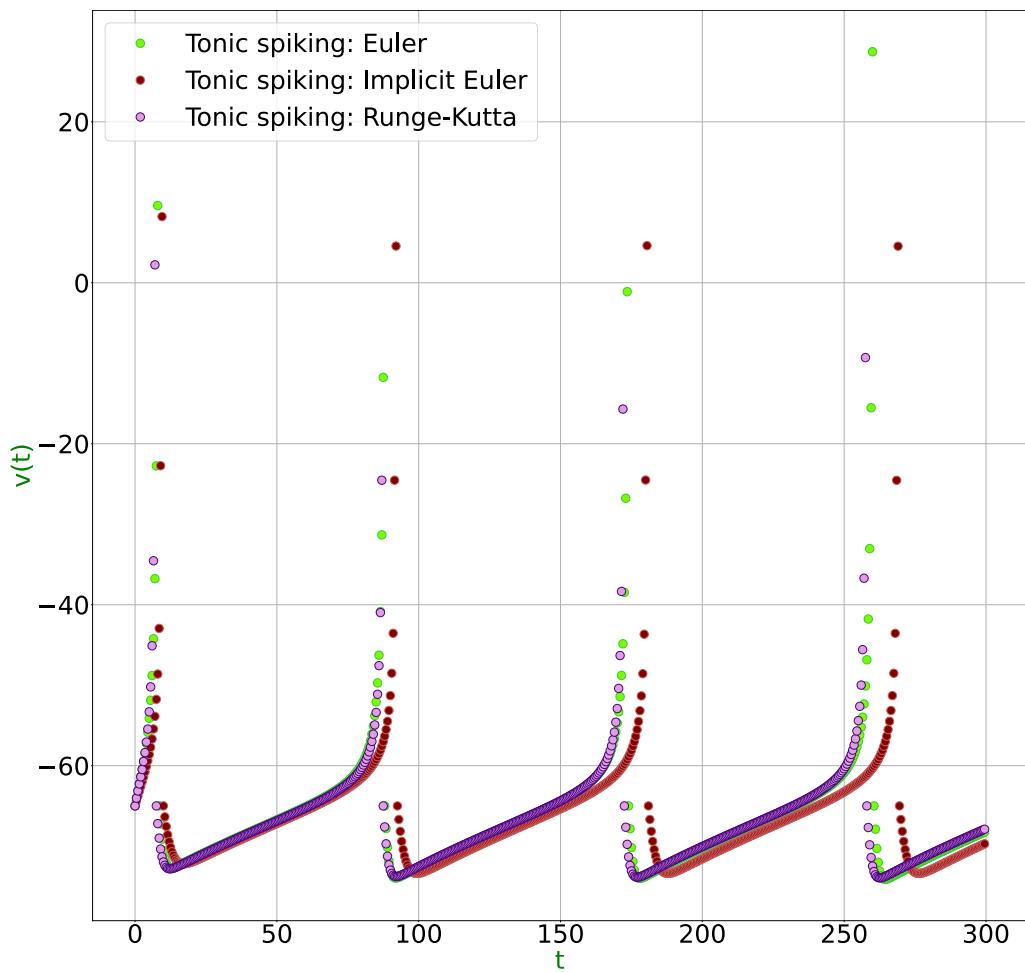


Рисунок 8. График зависимости дискретных траекторий $v(t)$ от времени t , построенных с помощью: метода Эйлера, неявного метода Эйлера и метода Рунге-Кутта 4-го порядка, – для режима Tonic spiking.

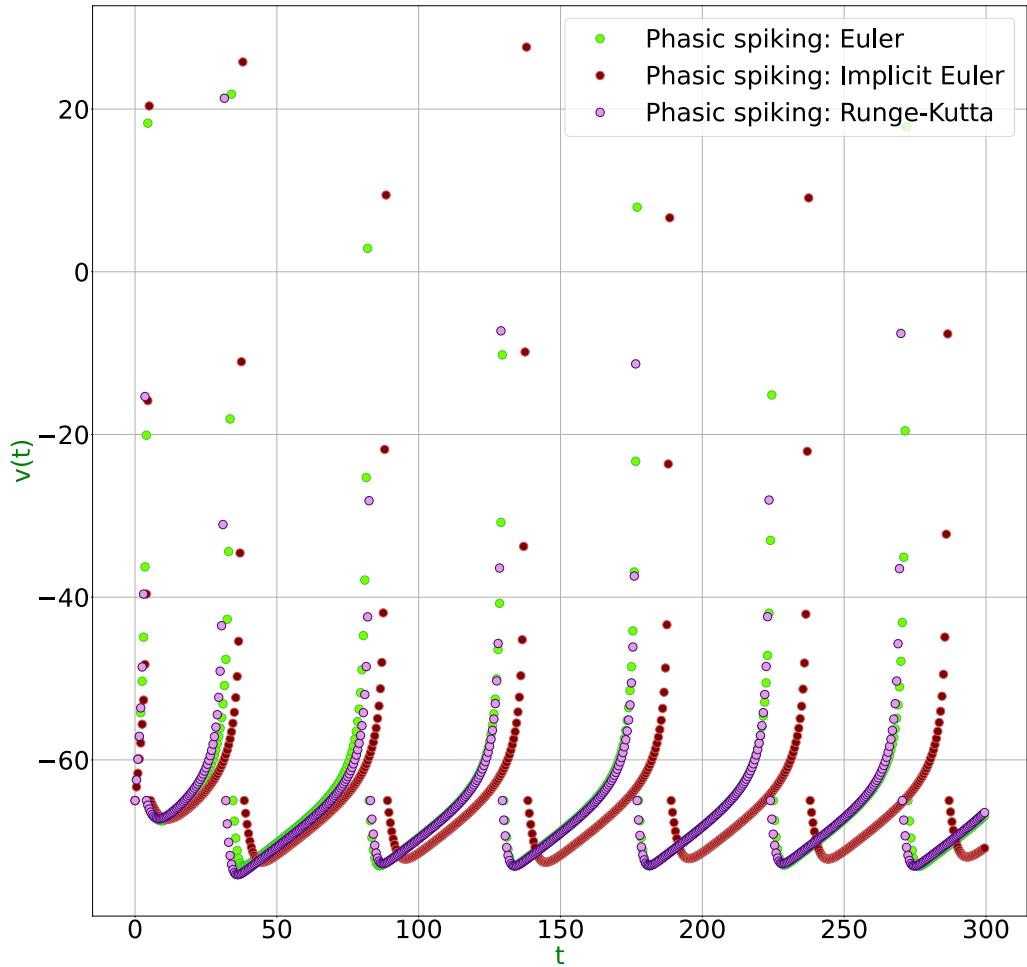


Рисунок 9. График зависимости дискретных траекторий $v(t)$ от времени t , построенных с помощью: метода Эйлера, неявного метода Эйлера и метода Рунге-Кутта 4-го порядка, – для режима Phasic spiking.

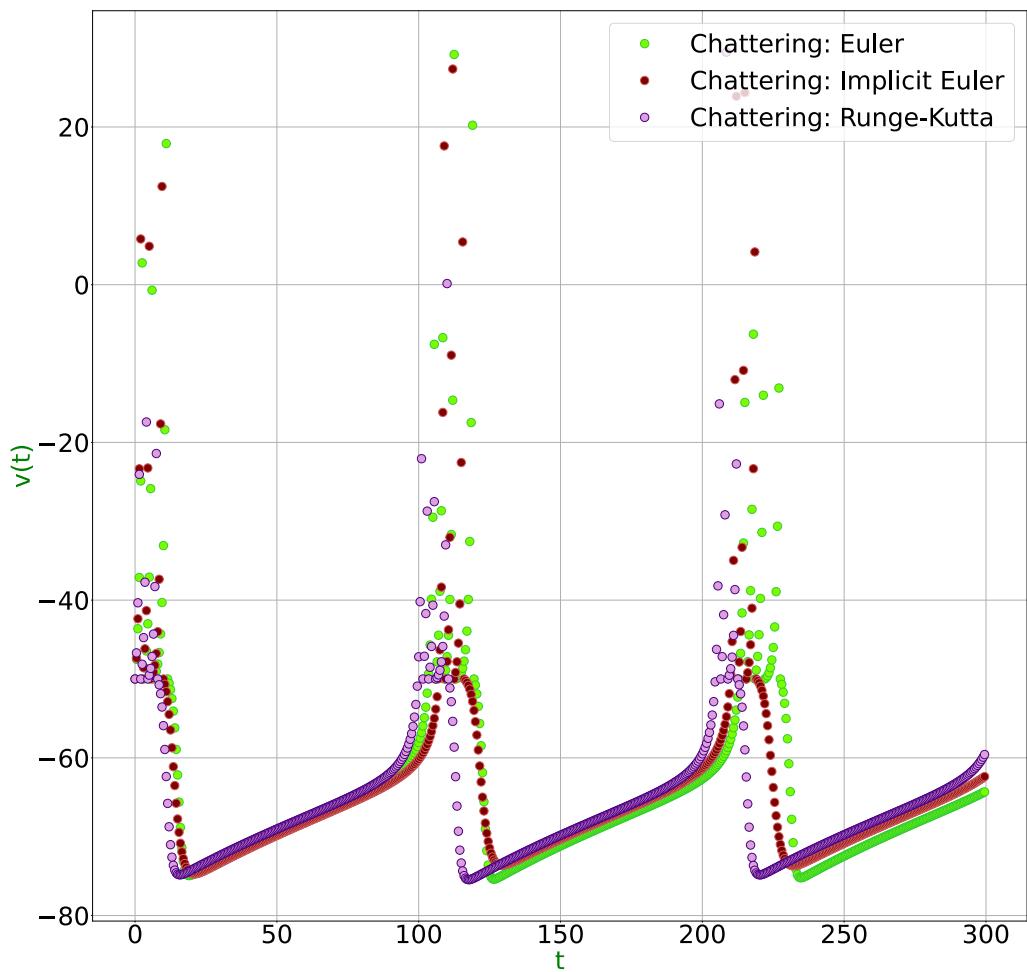


Рисунок 10. График зависимости дискретных траекторий $v(t)$ от времени t , построенных с помощью: метода Эйлера, неявного метода Эйлера и метода Рунге-Кутта 4-го порядка, – для режима Chattering.

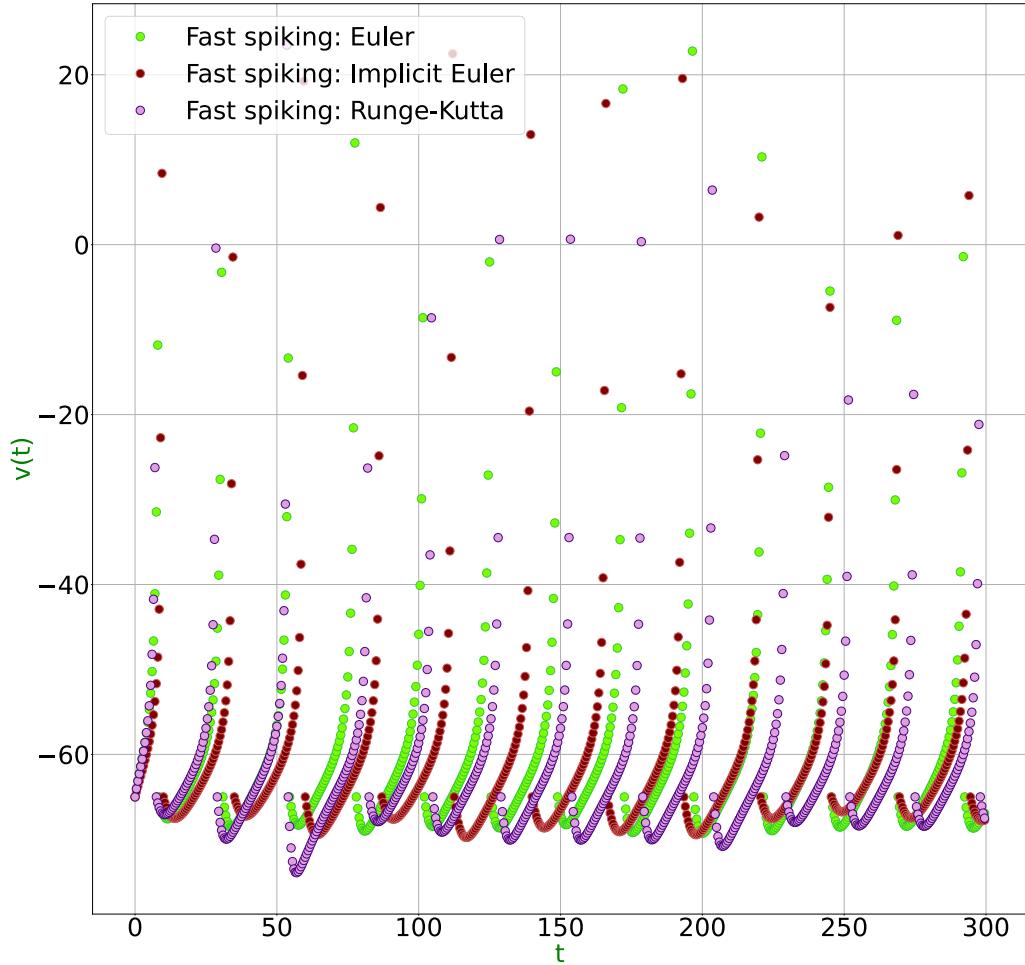


Рисунок 11. График зависимости дискретных траекторий $v(t)$ от времени t , построенных с помощью: метода Эйлера, неявного метода Эйлера и метода Рунге-Кутта 4-го порядка, – для режима Fast spiking.

10 Реализация функций для моделирования нейронной сети

Главным результатом моделирования импульсной нейронной сети является график зависимости номера нейрона, в котором произошел импульс, от момента времени, в который он произошел. Такой набор данных представляет собой многомерный временной ряд.

Для реализации нейронной сети с динамикой нейронов, описываемой моделью Ижикевича, удобно определить лямбда-выражения для каждого параметра нейрона (воз-

буждающего или тормозного), в котором присутствует случайная величина (15, 16). В листинге 6 такие параметры прокомментированы.

Параметры для каждого возбуждающего нейрона:

$$\begin{cases} a = 0.02, \\ b = 0.2, \\ c = -65 + 15\alpha, \\ d = 8 - 6\beta^2, \\ I = I_0 = 5\xi, \\ W_{ij} = 0.5\theta, \end{cases} \quad (15)$$

где $\alpha, \beta, \xi, \theta$ – случайные величины.

Параметры для каждого тормозного нейрона:

$$\begin{cases} a = 0.02 + 0.08\gamma, \\ b = 0.25 - 0.05\delta, \\ c = -65, \\ d = 2, \\ I = I_0 = 2\zeta, \\ W_{ij} = -\tau, \end{cases} \quad (16)$$

где $\gamma, \delta, \zeta, \tau$ – случайные величины.

Листинг 6: Определение лямбда-выражений для параметров в нейронах нейронной сети, согласно формулам из пункта 2 продвинутой части данной лабораторной работы.

```
a_ex, b_ex = 0.02, 0.2
c_ex = lambda x: -65 + 15 * x ** 2      # с возбуждающего нейрона
d_ex = lambda x: 8 - 6 * x ** 2            # d возбуждающего нейрона
a_in = lambda x: 0.02 + 0.08 * x           # a тормозного нейрона
b_in = lambda x: 0.25 - 0.05 * x           # b тормозного нейрона
c_in, d_in = -65, 2
I_ex = lambda x: 5 * x                      # внешний ток возбуждающего нейрона
I_in = lambda x: 2 * x                      # внешний ток тормозного нейрона
```

Далее будет смоделировано три нейронных сети со случайными величинами $\alpha, \beta, \gamma, \delta$ – из следующих распределений:

$$\alpha, \beta, \gamma, \delta \sim U[0, 1], \quad (17)$$

$$\alpha, \beta, \gamma, \delta \sim N\left[\frac{1}{2}, \frac{1}{12}\right], \quad (18)$$

$$\alpha, \beta, \gamma, \delta \sim \frac{1}{2}U[0, 1] + \frac{1}{2}N\left[\frac{1}{2}, \frac{1}{12}\right], \quad (19)$$

где последнее распределение является смесью нормального и равномерного распределений с равными весами.

Остальные случайные величины: ξ, ζ, τ, θ остаются такими же, как в условии, то есть распределенными равномерно на отрезке $[0, 1]$.

Параметры нормального распределения (18) взяты таким образом, чтобы у данных случайных величин совпадали первые начальные моменты – математические ожидания, и вторые центральные – дисперсии. Действительно, математическое ожидание для $\alpha \sim U[0, 1]$ равно $\frac{0+1}{2} = \frac{1}{2}$. А дисперсия равна $\frac{1-0}{12} = \frac{1}{12}$.

Для удобства генерации трех нейронных сетей определена функция `rv(random)`. Её параметр – строковая переменная `random`. В зависимости от ее значения '`uniform`', '`norm`', '`mixture`' функция `rv` возвращает одну из трех возможных реализаций случайной величины, согласно (17 - 19). Все это реализовано в листинге 7. Для генерации случайных величин был использован модуль языка программирования Python `scipy.stats`. Чтобы получить смесь распределений, необходимо сначала сгенерировать случайную величину из равномерного распределения на $[0, 1]$. Если она меньше или равна $\frac{1}{2}$, то генерируется случайная величина из нормального распределения, иначе – из равномерного.

Листинг 7: Определение функции `rv(random)`, возвращающей случайную величину из указанного распределения `random`.

```
def rv(random):
    if random == 'norm':
        return float(sts.norm.rvs(loc = 1/2, scale = 1/12, size = 1))
    if random == 'uniform':
        return float(sts.uniform.rvs(loc = 0, scale = 1, size = 1))
    if random == 'mixture':
        un = sts.uniform.rvs()
        if un <= 1/2:
            return float(sts.norm.rvs(loc = 1/2, scale = 1/12, size = 1))
        else:
            return float(sts.uniform.rvs(loc = 0, scale = 1, size = 1))
```

Далее необходимо задать параметры нейронов моделируемой нейронной сети. Для этого была определена функция `neural_network_params(n, ex_n, random)`.

Входные данные функции `neural_network_params(n, ex_n, random)`: `n` – общее число нейронов, `ex_n` – число возбуждающих нейронов, `random` – параметр, отвечающий за распределение случайных величин в параметрах a, b, c, d нейронов.

Алгоритм, реализованный в `neural_network_params(n, ex_n, random)`, достаточно прост. Сначала создаются списки со параметрами для модели нейронной сети. Все списки прокомментированы в листинге 8. Стоит отметить, что списки `V` и `U` состоят из `n` элементов, потому что для моделирования нейронной сети достаточно запоминать лишь последнее значение решения системы ОДУ на каждом шаге для вычисления следующего, так как цель состоит в получении многомерного временного ряда. Далее идут заполнения списков в циклах. Сначала заполняются первые `ex_n` элементов – для возбуждающих нейронов. Затем заполняются последние `n - ex_n` элементов – для тормозных нейронов.

Выходные данные функции `neural_network_params(n, ex_n, random)` – список со списками всех параметров нейронной сети.

Листинг 8: Определение функции `neural_network_params(n, ex_n, random)`, возвращающей параметры нейронов нейронной сети.

```

def neural_network_params(n, ex_n, random):
    W = [[0 for i in range(n)] for j in range(n)]           # матрица W, в которой хранятся
    # значения синаптических токов
    coef = [[] for i in range(n)]                          # список со списками коэффициентов
    # для каждого нейрона
    I_d = [0 for i in range(n)]                           # внешние токи для каждого нейрона
    I_syn = [0 for i in range(n)]                         # синаптические токи для нейрона
    V = [-65 for i in range(n)]                           # начальные значения для v(t) из ОДУ
    U = [-65 * 0.2 for i in range(n)]                     # начальные значения для u(t) из ОДУ

    for i in range(ex_n):
        W[i] = sts.uniform.rvs(loc=0, scale=1/2, size=n)
        coef[i] = [a_ex, b_ex, c_ex(rv(random)), d_ex(rv(random))]
        I_d[i] = I_ex(float(sts.uniform.rvs(loc=0, scale=1, size=1)))

    for i in range(n - ex_n):
        W[n - 1 - i] = sts.uniform.rvs(loc=-1, scale=1, size=n)
        coef[n - 1 - i] = [a_in(rv(random)), b_in(rv(random)), c_in, d_in]
        I_d[n - 1 - i] = I_in(float(sts.uniform.rvs(loc=0, scale=1, size=1)))
        U[n - 1 - i] = -65 * coef[n - 1 - i][1]

    return [W, coef, I_d, I_syn, V, U]

```

Далее необходимо реализовать функцию интегрирования импульсной нейронной сети и генерацию многомерного временного ряда, описанного ранее. Для этого определена функция `neur(n, ex_n, random, synapsis = True)`.

Входные данные функции `neur(n, ex_n, random, synapsis = True)` имеет те же самые входные данные, что и функция `neural_network_params(n, ex_n, random)`, и еще один булевый параметр `synapsis`, отвечающий за учет синапсов в нейронной сети. По умолчанию он равен `True`.

Алгоритм, реализованный в функции `neur(n, ex_n, random, synapsis = True)` состоит в следующем. Сначала генерируются параметры нейронной сети путем вызова функции `neural_network_params(n, ex_n, random)`. Далее создаются три пустых списка: для хранения времени, когда произошел импульс, для хранения номеров всех нейронов, в которых произошел импульс и для хранения номеров нейронов, в которых произошел импульс в данный момент времени t (буферный список). Данные списки прокомментированы в листинге 9. После идет цикл по времени до 1000 мс с шагом 0.5 мс. В этом цикле по всем нейронам численно решается задача Коши для системы ОДУ (1) явным методом Эйлера для каждого нейрона. При возникновении импульса (условие (2)) заполняются списки, описанные выше. Наконец, если данная модель учитывает синаптические токи, то для всех нейронов из буферного списка происходит суммирование синаптических токов для связанных с ними нейронов: переменная `I_syn[k]` аккумулирует токи, значения которых заданы в матрице W .

Выходные данные функции `neur(n, ex_n, random, synapsis = True)` – списки `t_pulse` и `id_pulse`, в которых хранятся времена возникновения импульсов и номера нейронов, в которых происходили импульсы.

Листинг 9: Определение функции `neur(n, ex_n, random, synapsis = True)`, возвращающей многомерный временной ряд.

```
def neur(n, ex_n, random, synapsis = True):
    W, coef, I_d, I_syn, V, U = neural_network_params(n, ex_n, random)

    t_pulse = []          # список, в котором хранятся значения времени,
                          # когда произошел импульс
    id_pulse = []          # список, в котором хранятся номера нейронов,
                          # в которых произошел импульс
    buf_pulse = []          # список, в котором хранятся номера нейронов,
                          # в которых произошел импульс в данный момент t

    for t in np.arange(h, 1005, h):
        buf_pulse = []
        for k in range(n):
            V[k] = V[k] + h * (0.04 * V[k] ** 2 + 5 * V[k]           # явный метод Эйлера
                                + 140 - U[k] + I_syn[k] + I_d[k])
            U[k] = U[k] + h * coef[k] * (coef[k] * V[k] - U[k])      # явный метод Эйлера
            I_syn[k] = 0          # синаптические токи обнуляются

            if V[k] >= 30:      # если произошел импульс
                t_pulse.append(t) # добавляется время, когда произошел импульс
                id_pulse.append(k) # добавляется нейрон, в котором произошел импульс
                buf_pulse.append(k)
                V[k] = coef[k][2]
                U[k] += coef[k][3]

        if synapsis:          # если данная модель учитывает синаптические токи
            for x in buf_pulse:
                for k in range(n):
                    if k not in buf_pulse:
                        I_syn[k] += W[x][k]      # синаптические токи суммируются
                                         # от всех нейронов,
                                         # в которых произошел импульс

    return [t_pulse, id_pulse]
```

11 Моделирование нейронной сети и определение частот колебаний нейронов

Можно приступать к моделированию нейронной сети. Вспомогательная функция, осуществляющая вывод графиков, представлена в файле `comprmath_lab3.ipynb`.

Как было сказано ранее, было смоделировано три нейронных сети, для каждой из которых будет свой закон распределения случайных величин в параметрах a, b, c, d нейронов.

На рисунке 12 модель нейронной сети смоделирована согласно условию пункта 2 продвинутой части данной работы. То есть все случайные величины имеют равномерное распределение на отрезке $[0, 1]$.

Глядя на данный рисунок, можно заметить различия между первыми 800 возбуждающими нейронами и последними 200 тормозными нейронами.

Возбуждающий слой нейронов имеет одинаковые параметры a и b , которые отвечают за время восстановления мембранны и чувствительность переменной восстановления к флюктуациям. А вот значения потенциала мембранны c и переменной восстановления

d сразу после импульса являются функциями от случайной величины. В то время как у x нейронов ситуация противоположная.

Отсюда можно сделать вывод, что более синхронные колебания возбуждающего слоя вызваны тем, что все они имеют одинаковый временной масштаб восстановления и одинаковую чувствительность к флуктуациям, что позволяет компенсировать разброс значений потенциала мембранны и переменной восстановления сразу после импульса.

Менее синхронные колебания тормозного слоя могут быть объяснены аналогично: временной масштаб и чувствительность к флуктуациям в данном случае – функции от случайной величины. Поэтому, несмотря на одинаковые значения потенциала мембранны и переменной восстановления сразу после импульса, нейроны в тормозном слое восстанавливаются за разное время или какие-то из них более чувствительны к флуктуациям, чем другие.

Таким образом получается более асинхронные колебания у нейронов тормозного слоя, нежели у нейронов возбуждающего слоя.

За время моделирования 1000 мс произошло 10 различимых синхронных и частично синхронных импульсов. Значит частота колебаний нейронов примерно равна 10 Гц.

На рисунке 13 параметры нейронов a, b, c, d являются функциями от случайной величины, имеющей нормальное распределение (18). Как видно, колебания обоих слоев стали более синхронными. Кроме того, нейроны тормозного слоя теперь имеют большую синхронность по сравнению с нейронами возбуждающего слоя. Несмотря на равенство матожиданий и дисперсий у случайных величин из равномерного и нормального распределения, наблюдается разница в работе двух данных нейронных сетей. Это можно объяснить особенностями законов распределения случайных величин. Для нормального распределения случайная величина наиболее вероятно принимает значения близкие к ее матожиданию, в то время как для равномерного распределения вероятность принять значения близкие к матожиданию или к краям распределения одинакова. Поэтому для второго случая, когда в качестве случайных величин взяты нормально распределенные, отклонение от среднего значения гораздо меньше, что и дает более синхронные колебания нейронов.

Частота колебаний, как и в случае первой нейронной сети, приблизительно равна 10 Гц.

На рисунке 14 можно наблюдать некий "усредненный" вариант описанных выше двух нейронных сетей. Здесь случайная величина в параметрах a, b, c, d либо из равномерного распределения, либо из нормального распределения (19). При этом вероятность быть распределенной по первому или по второму закону распределения для данной случайной величины одинакова и равна $\frac{1}{2}$. Как видно здесь колебания более синхронные, чем у первой нейронной сети, и менее синхронные, чем у второй. Такой результат был ожидаем.

Частота колебаний и в этом случае приблизительно равна 10 Гц.

Далее будет смоделирована еще одна нейронная сеть, таким образом, что у нее отсутствуют вообще какие-либо связи. Иными словами она является несвязанным графом из 1000 вершин (имеет 1000 компонент связности).

На рисунке 15 показан временной ряд такой нейронной сети. Видно, что практически невозможно выделить какие-либо синхронные импульсы. Нейронная сеть ведет себя достаточно хаотично, что объясняется отсутствием какой-либо передачи информации (в данном случае значения тока) от одного нейрона к другому в случае возникновения импульса.

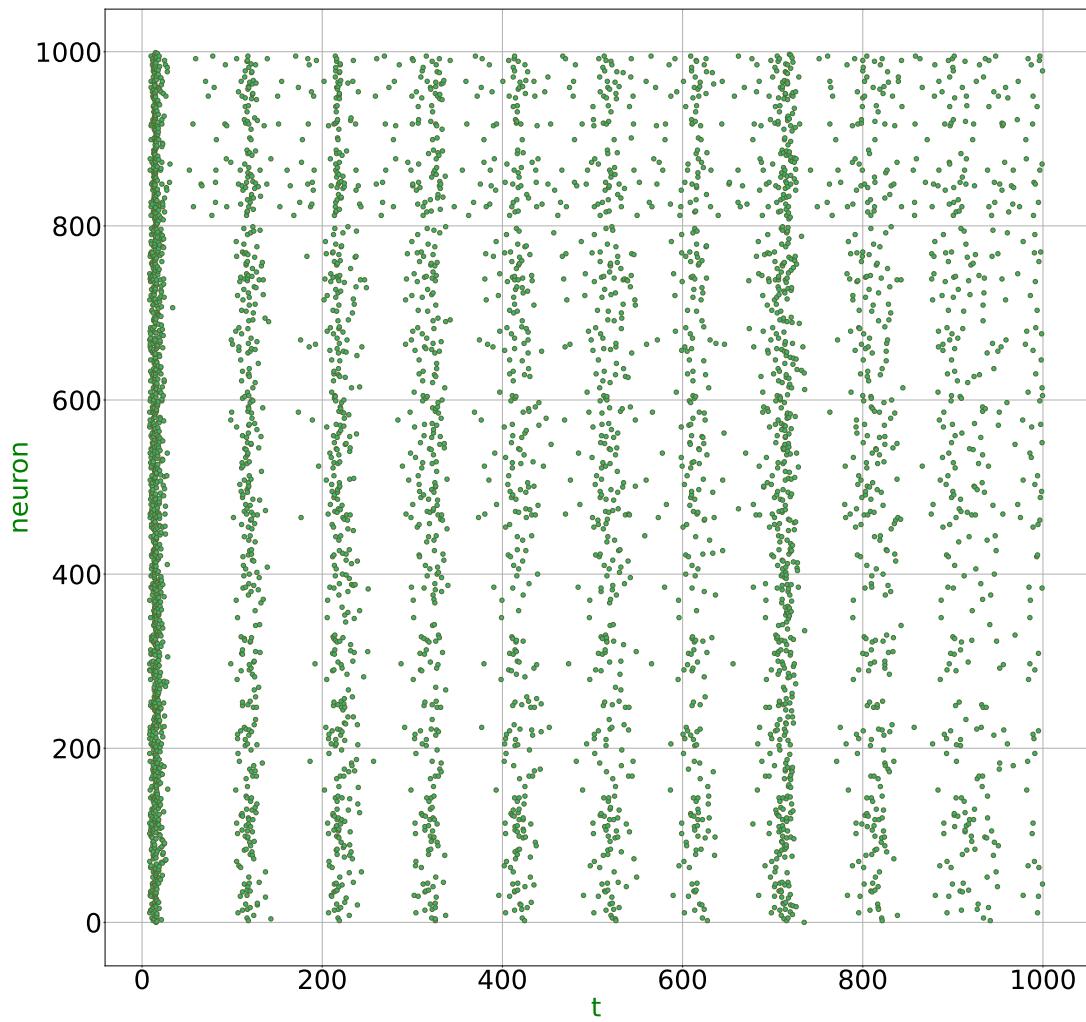


Рисунок 12. График зависимости номера нейрона $neuron$ от времени t , полученный в результате интегрирования нейронной сети, где случайные величины в параметрах нейронов имеют равномерное распределение.

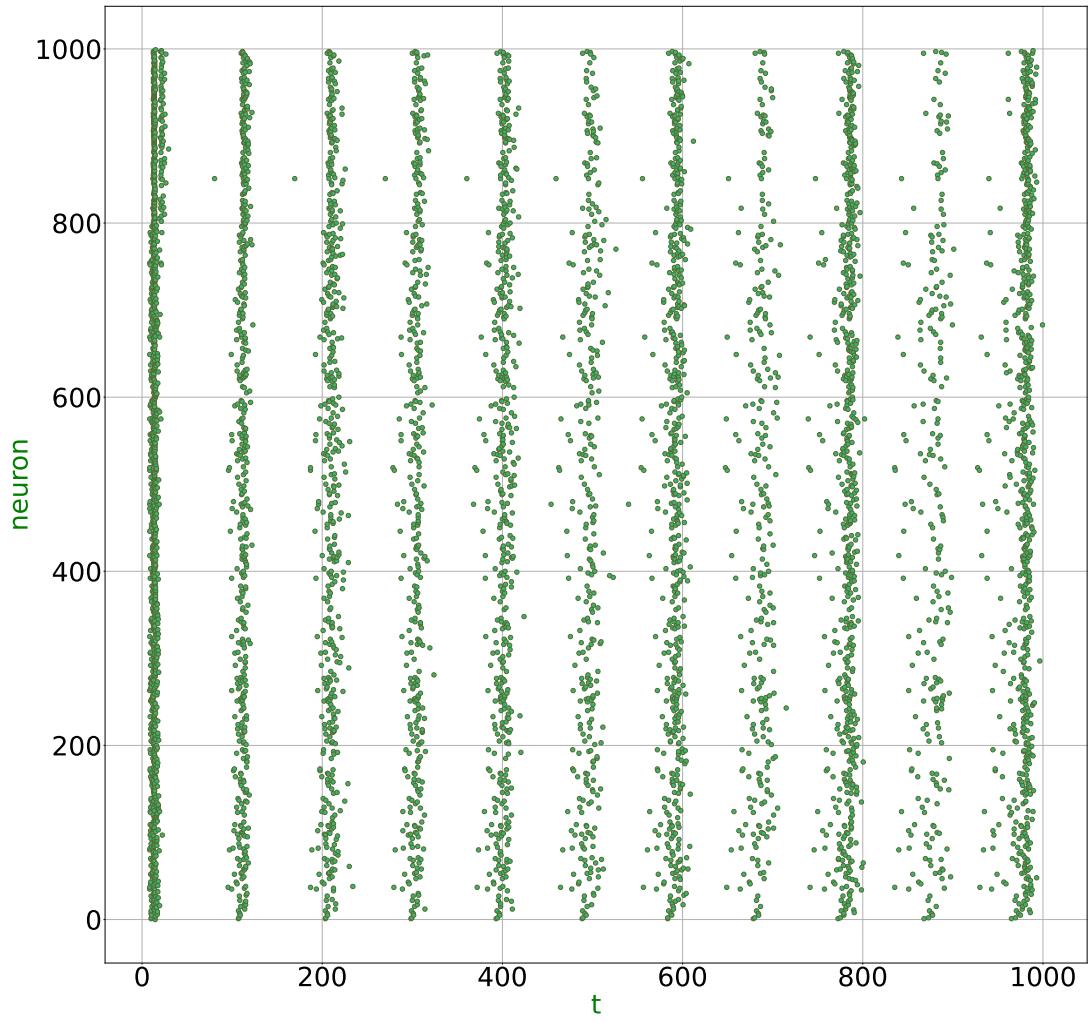


Рисунок 13. График зависимости номера нейрона $neuron$ от времени t , полученный в результате интегрирования нейронной сети, где случайные величины в параметрах нейронов имеют нормальное распределение.

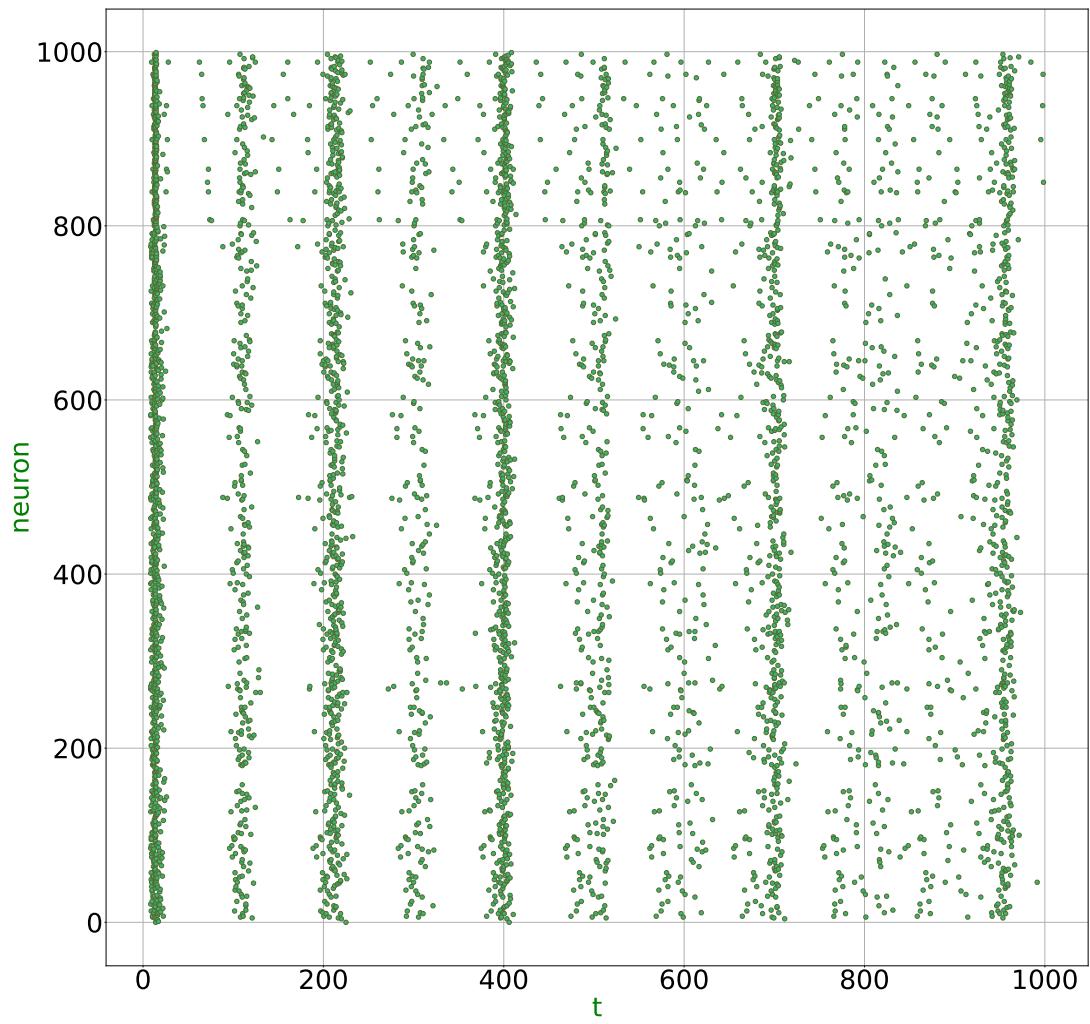


Рисунок 14. График зависимости номера нейрона $neuron$ от времени t , полученный в результате интегрирования нейронной сети, где случайные величины в параметрах нейронов имеют распределение 19.

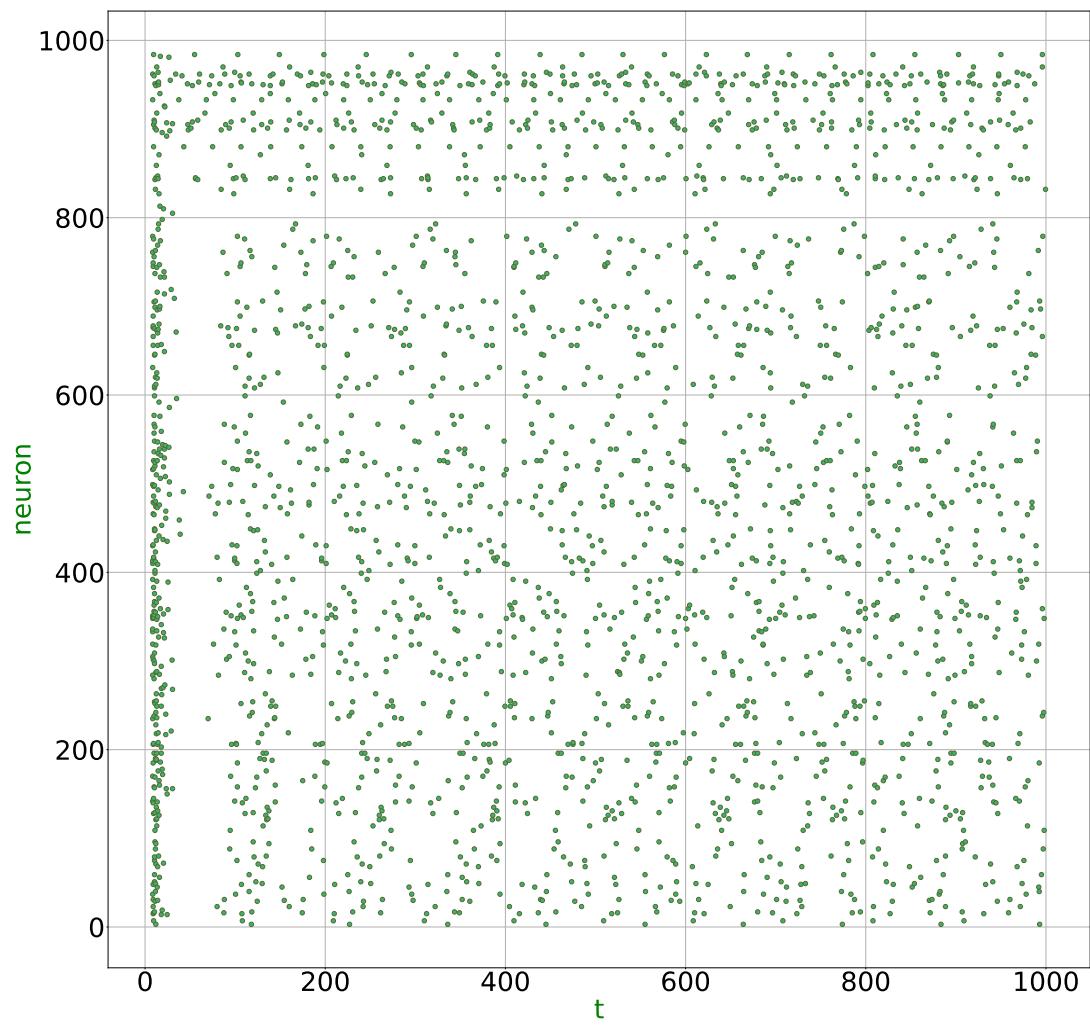


Рисунок 15. График зависимости номера нейрона $neuron$ от времени t , полученный в результате интегрирования нейронной сети, где случайные величины в параметрах нейронов имеют равномерное распределение. Синаптические токи не учтены.

12 Заключение

1. В результате проделанной работы были реализованы три метода численного решения задачи Коши для системы ОДУ 1-го порядка. Каждый из этих методов имеет как своих недостатки, так и определенные достоинства.
2. Метод Эйлера является самым быстрым, при этом уступает по устойчивости и точности двум другим. Неявный метод Эйлера является абсолютно устойчивым, однако, когда требуется численно решить нелинейную систему уравнений, то его нельзя назвать быстрым, возможно, его точность из-за этого также снижается. Метод Рунге-Кутта 4-го порядка представляет собой некий баланс между устойчивостью, точностью и алгоритмической сложностью.
3. Моделирование нейронной сети с разными параметрами для нейронов наглядно показало, как отклонения в различных параметрах динамической системы модели Ижикевича влияют на синхронность или асинхронность работы нейронов.
4. Моделирование нейронных сетей с разными случайными величинами в параметрах a, b, c, d , дало понять, что чем менее отклоняется случайная величина от какого-то определенного значения, тем более ярко выражены импульсы и более синхронно работают группы нейронов.

Список использованных источников

1. Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018-2021. С. 140.
2. <https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.root.html>
3. Cruz William, Martínez José, Raydan Marcos. Spectral Residual Method without Gradient Information for Solving Large-Scale Nonlinear Systems of Equations // Math. Comput. 2006. 07. Т. 75. С. 1429–1448.

Выходные данные

Горелкина Е.Е.. Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] – Москва: 2021. – 33 с. URL: <https://sa2systems.ru:88> (система контроля версий кафедры РК6)

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

Решение и вёрстка:  студент группы РК6-52Б, Горелкина Е.Е.

2021, осенний семестр