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:	Кутепов Никита Сергеевич
Группа:	РК6-55Б
Тип задания:	лабораторная работа
Тема:	Модель биологического нейрона

Студент ____________ Кутепов Н.С.
подпись, дата Фамилия, И.О.

Преподаватель ____________ ____________
подпись, дата Фамилия, И.О.

Москва, 2021

Содержание

Модель биологического нейрона **3**

1 Реализация численных методов решения задачи Коши 5

2 Нахождение и построение траекторий для каждого из реализованных методов, используя разные режимы 8

3 Описание особенностей указанных режимов 10

4 Объяснение отличий и сходств у реализованных методов 11

5 Интегрирование во времени до 1000 мс нейронной сети с помощью метода Эйлера, вывод на экран импульсов всех нейронов 12

6 Заключение 14

Модель биологического нейрона

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 привелак 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 20 (Модель Ижикевича) Дана система из двух ОДУ 1-го порядка:

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

и дополнительного условия, определяющего возникновение импульса в нейроне:

если $v \geqslant 30$,

$$\begin{cases} v \longleftarrow c, \\ u \longleftarrow u + d; \end{cases} \tag{2}$$

где v – потенциал мембраны (мВ), u – переменная восстановления мембраны (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 мембраны (чем больше a, тем быстрее происходит восстановление после импульса);

b – чувствительность переменной восстановления к флуктуациям 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 мембраны сразу после импульса; d – значение переменной восстановления мембраны 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 $u(0) = du(0)$. Внешний ток принимается равным I = 5.

3. Вывести полученные траектории на четырех отдельных графиках как зависимости потенциала мембраны 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).

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

6. 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 = 0.02, $c = -65 + 15\alpha^2$, $d = 8 - 6\beta^2$ и внешний ток в отсутствие токов от других нейронов равен $I = I_0 = 5\epsilon$, где α, β и ϵ – 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 $I = I_0 = 2\varsigma$, где γ, δ и ς – случайные числа от 0 до 1. В качестве начальных условий используются значения $v(0) = -65$ и $u(0) = bv(0)$.

б) Нейронная сети 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 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 τ, если тормозным, где θ и τ – случайные числа от 0 до 1.

7. Вывести на экран импульсы всех нейронов как функцию времени и определить частоты характерных синхронных (или частично синхронных) колебаний нейронов в сети.

Цель выполнения лабораторной работы

Цель выполнения лабораторной работы – изучение методов численного решения задачи Коши: метод Эйлера, неявный метод Эйлера, метод Рунге-Кутты. Знакомство со спайковой нейронной сетью, а именно с простейшей моделью подобного типа: модель Ижикевича.

Поставленные задачи

1. Реализовать численные методы решения задачи Коши: метод Эйлера, неявный метод Эйлера, метод Рунге-Кутта 4-го порядка.

2. Найти траектории заданной динамической системы для каждого из реализованных методов и вывести полученные траектории как зависимость потенциала мембраны от времени, используя шаг $h = 0.5$, режимы из таблицы 1 с заданными начальными условиями и током $I = 5$.

3. Описать особенности указанных режимов из таблицы 1.

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

5. Произвести интегрирование во времени до 1000 мс нейронной сети с помощью метода Эйлера, вывести на экран импульсы всех нейронов как функцию времени.

1 Реализация численных методов решения задачи Коши

В данной лабораторной работе рассматривает систему ОДУ 1-го порядка 1, разрешенных относительно производной:

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

Для решения ОДУ используются начальные условия задачи Коши, данное ОДУ можно представить в виде системы ОДУ:

$$\frac{d}{dt}\begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_n(t) \end{bmatrix} = \begin{bmatrix} f_1(t, y_1, \cdots, y_n) \\ f_2(t, y_2, \cdots, y_n) \\ \vdots \\ f_n(t, y_1, \cdots, y_n) \end{bmatrix}. \tag{4}$$

или в векторном виде:

$$\frac{dy}{dt} = f(t, y) \tag{5}$$

Рассмотрев ОДУ 5 после разложения в ряд Тейлора можно записать формулировку метода Эйлера [1]:

$$w_0 = \alpha, \tag{6}$$

$$w_{i+1} = w_i + hf(t_i, w_i), i = 0, 1..., m-1, \tag{7}$$

при этом ожидаем, что $w_i \approx y(t_i)$, а $t_i = a + ih$, $i = 0, 1, ..., m$, $h = \frac{b-a}{m} = t_{i+1} - t_i$. В данном случае, учитывая, что происходит работа с системой ОДУ, представим w как вектор $w_i = [y_1(t_i), y_2(t_i)]$.

Листинг 1. Реализация метода Эйлера.

```
def euler(x_0, t_n, f, h, consts):
    # x_0 - initial conditions v(0) = c, u(0) = bv(0)
    # f - function of the ODE system
    # consts - a,b,c,d,I for a specific mode
    t_0 = 0
    t_nums = np.arange(t_0, t_n + h, h)
    # t_nums -  generation of t according to the condition of the Euler method
    y = np.zeros(shape=(len(t_nums), len(x_0)))
    y[0] = x_0

    for i in range(len(t_nums) - 1):
        w = y[i]
        w = w + h * f(w, consts)
        y[i + 1] = normal(consts, w)

    return t_nums, y
```

Функция normal(consts, w) осуществляет определение возникновения импульса в нейроне.

Листинг 2. Реализация функции normal

```
def normal(consts, values):
   # consts - a,b,c,d for a specific mode
   # values - values v, u
    v, u = values
    if v >= 30:
        v = consts['c']
        u += consts['d']

    return [v, u]
```

При построении неявного метода Эйлера значение функции f берется на новом временном слое, таким образом для нахождения приближенного значения искомой функции на новом временном слое нужно решить нелинейное уравнение относительно w_{i+1}[2].

$$w_{i+1} = w_i + hf(t_i, w_{i+1}), \tag{8}$$

Для решения нелинейных уравнений используем метод optimize.root из библиотеки scipy, который находит корень векторной функции. Параметрами данного метода являются: fun - вектор функция, корень которой необходимо найти, w - первоначальное предположение.

Листинг 3. Реализация неявного метода Эйлера.

```
def implicit_euler(x_0, t_n, f, h, consts):
    # x_0 - initial conditions v(0) = c, u(0) = bv(0)
    # f - function of the ODE system
    # consts - a,b,c,d,I for a specific mode
    t_0 = 0
    t_nums = np.arange(t_0, t_n + h, h)
    y = np.zeros(shape=(len(t_nums), len(x_0)))
    y[0] = x_0

    for i in range(len(t_nums) - 1):
        w = y[i]
        fun = lambda foo: w + h * f(foo, consts) - foo
        sol = optimize.root(fun, w)
        w = sol.x
        y[i + 1] = normal(consts, w)

    return t_nums, y
```

Запишем метод Рунге-Кутты 4-го порядка[1]:

$$w_0 = \alpha, \tag{9}$$

$$k_1 = hf(t_i, w_i), \tag{10}$$

$$k_2 = hf(t_i + \frac{h}{2}, w_i + \frac{k_1}{2}), \tag{11}$$

$$k_3 = hf(t_i + \frac{h}{2}, w_i + \frac{k_2}{2}), \tag{12}$$

$$k_4 = hf(t_i + h, w_i + k_3), \tag{13}$$

$$w_{i+1} = w_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), i = 0, 1, ..., m - 1 \tag{14}$$

Листинг 4. Реализация метода Рунге-Кутты 4-го порядка.

```
def runge_kutta(x_0, t_n, f, h, consts):
    # x_0 - initial conditions v(0) = c, u(0) = bv(0)
    # f - function of the ODE system
    # consts - a,b,c,d,I for a specific mode
    t_0 = 0
```

```
    t_nums = np.arange(t_0, t_n + h, h)
    y = np.zeros(shape=(len(t_nums), len(x_0)))
    y[0] = x_0

    for i in range(len(t_nums) - 1):
        w = y[i]
        k_1 = h * f(w, consts)
        k_2 = h * f(w + k_1 / 2, consts)
        k_3 = h * f(w + k_2 / 2, consts)
        k_4 = h * f(w + k_3, consts)

        w = w + (k_1 + 2 * k_2 + 2 * k_3 + k_4) / 6

        y[i + 1] = normal(consts, w)

    return t_nums, y
```

2 Нахождение и построение траекторий для каждого из реализованных методов, используя разные режимы

Рассмотрим промежуток времени $t \in [0; 300]$, используем шаг $h = 0.1$, так как при $h > 1e - 1$ optimize.root обращает график функции в константу, и параметры a, b, c, d, I, которые подгружаются из json файла, учитывая текущий режим из таблицы 1. Используем 4 графика, каждый соответствует одному из 4 режимов из таблицы 1, одна итерации осуществляет построение траекторий всех методов для одного режима. Отрисовку осуществим с помощью пакета matplotlib.pyplot.

Листинг 5. Нахождение и отрисовка траекторий для каждого реализованного метода.

```
if __name__ == '__main__':

    h = 0.5
    t_n = 300

    t = np.linspace(0, t_n, 201)
    fig, ax = plt.subplots(2, 2, figsize=(14, 6))
    axes = [ax[0][0], ax[0][1], ax[1][0], ax[1][1]]

    for name, current_ax in zip(BASE_CONST, axes):
        x_0 = [BASE_CONST[name]['c'], BASE_CONST[name]['c'] *
            BASE_CONST[name]['b']]
        x_1, y_1 = euler(x_0, t_n, f, h, BASE_CONST[name])
        x_2, y_2 = implicit_euler(x_0, t_n, f, h, BASE_CONST[name])
        x_3, y_3 = runge_kutta(x_0, t_n, f, h, BASE_CONST[name])

        current_ax.set_title(name, loc='left')
```

```
        current_ax.set_ylim([-80, 40])
        current_ax.set_xlabel(r'$t$', fontsize=16)
        current_ax.set_ylabel(r'$v$', fontsize=16)
        current_ax.plot(x_1, y_1[:,0], ':', label=r"Метод Эйлера", marker='o',
            markersize=2)
        current_ax.plot(x_2, y_2[:,0], ':', label=r"Неявный методЭйлера", marker='o',
            markersize=2)
        current_ax.plot(x_3, y_3[:,0], ':', label=r"Метод РунгеКута-", marker='o',
            markersize=2)

        current_ax.grid()
        current_ax.legend(loc=1)

    plt.show()
```

Функция f в данном случае соответствует системе ОДУ 1-го порядка 1.

Листинг 6. Реализация системы ОДУ 1-го порядка 1.

```
def f(nums, consts):
    # consts - a,b,c,d,I for a specific mode
    # nums - v,u values
    v, u = nums
    I = consts['I']
    current_v = 0.04 * v ** 2 + 5 * v + 140 - u + I
    current_u = consts['a'] * (consts['b'] * v - u)

    return np.asarray([current_v, current_u])
```

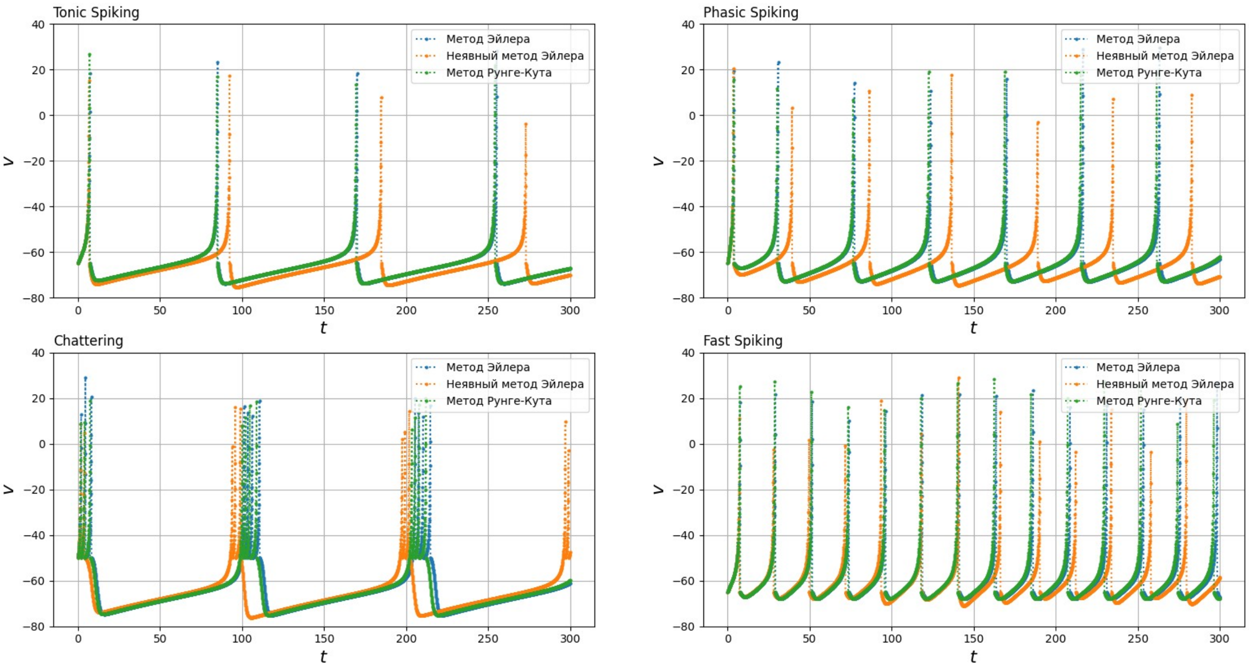

Рис. 1. Траектории для каждого из реализованных методов

3 Описание особенностей указанных режимов

Рассмотрим полученные графики 1:

Режим "Tonic Spiking"имеет низкую частоту пульсаций, это объясняется большим временем восстановления мембраны после импульса(параметр b), невысоким временным масштабом восстановления мембраны(параметр a) и недостаточно большим значением переменной восстановления к флуктуациям разности потенциалов(параметр d).

Режим "Phasic Spiking"отличается от предыдущего режима более частым проявлением пульсаций за счет большего значения переменной восстановления к флуктуациям разности потенциалов.

Режим "Chattering"имеет схожую с режимом "Tonic Spiking"частоту пульсаций, однако поведение графика в момент импульса явно отличается от других режимов, предположительно из-за значения потенциала мембраны сразу после импульса.

Режим "Fast Spiking"заметно чаще вызывает импульс нейронов, что является довольно очевидным, учитывая высокий временной масштаб для восстановления мембраны и время восстановления мембраны сразу после импульса.

4 Объяснение отличий и сходств у реализованных методов

Рассмотрим отличия реализованных методов. Явным различием является порядок точности глобальной и локальной погрешностей. Порядок точности локальной погрешности метода Эйлера - $O(n^2)$, глобальной - O(n). Для неявного метода Эйлера характерен порядок точности глобальной погрешности $O(n^2)$, локальной - $O(n^3)$. Порядок точности глобальной погрешности метода Рунге-Кутты 4-го порядка - $O(n^4)$, локальной - $O(n^5)$.

Учитывая полученные порядки точности, можно говорить о том, что метод Рунге-Кутты 4-го порядка является оптимальным вариантом в плане точности для решения данной заачи, чтобы понять, будет ли данный метод эффективен по времени, осуществим замер времени выпонения функции. Для этого был написан декоратор benchmark, который принимает функцию, запускает метод time.time(), результат которого записывается в переменную start, после чего исполняется переданная функция, далее повторно вызывается time.time(), результат которого записывается в переменную end, в итоге время выполнения функции считается как разность $end - start$

Листинг 7. Реализация декоратора для замера времени работы функции.

```
def benchmark(func):
   # func - method for measuring
    import time

    def wrapper(*args, **kwargs):
        start = time.time()
        res = func(*args, **kwargs)
        end = time.time()
        print(f'{func.__name__}: Времявыполнения: {end - start} секунд.')

        return res

    return wrapper
```

Тогда рассмотрим время выполнения функция для одного из режимов (таблица 1). Для режима "Tonic Spiking"получаем следующие результаты:

Euler: Время выполнения: 0.02200174331665039 секунд.

Implicit_euler: Время выполнения: 0.2710258960723877 секунд.

Runge_Kutta: Время выполнения: 0.18801283836364746 секунд.

Из чего можно сделать вывод, что несмотря на точность метода Рунге-Кутты 4-го порядка, метод Эйлера для данного набора данных работает быстрее.

5 Интегрирование во времени до 1000 мс нейронной сети с помощью метода Эйлера, вывод на экран импульсов всех нейронов

Нейронная сеть включает в себя 1000 нейронов, 800 из которых - возбуждающие, 200 - тормозные. Моделирование нейронной сети происходит с помощью полного графа. Используем матрицу смежности для описания значения токов, передаваемых от нейрона к нейрону, т.е. при одиночном импульсе внешнего тока происходит спайк, соответственно при подаче на нейрон постоянного внешнего тока происходит генерация последовательности спайков с определенной частотой.

Для реализации поставленной задачи необходимо сгенерировать матрицу смежности W, векторы, которые хранят параметры нейронов соответствующего типа. Первые 800 элементов вектора/матрицы соответветствуют возбуждающим нейронам, оставшиеся 200 - тормозным. Необходимо вывести на экран импульсы нейронов как функцию времени. Осуществим цикл по времени, каждую итерацию происходит решение 2000 ОДУ 1-го порядка, при этом сохраняются позиции нейронов, в которых возник импульс в данный момент времени, данные по этим нейронам(момент времени, в который возник импульс и номер нейрона) записываются в соответствующий список. В конце итерации необходимо осуществить метод Эйлера, учитывая значение шага h, в данном случае h=0.5, происходит $\frac{1}{h}$ = 2 итерации метода Эйлера в единицу времени. Построение графика выполним с помощью пакета matplotlib.

Листинг 8. Интегрирование во времени до 1000мс с помощью метода Эйлера.

```
def neural_network():

    t_n = 1000
    n = 1000
    h = 0.5
    n_b = int(0.2 * n)
    n_e = int(0.8 * n)

    W, a, b, c, d = get_consts(n, n_e, n_b)

    v = -65.0 * np.ones(n)
    u = v * b

    ex_t_plot = []
    br_t_plot = []

    ex_neuron_id = []
    br_neuron_id = []

    steps_in_t = int(1 / h)

    I = np.hstack((5 * np.random.default_rng().random(n_e), 2 *
```

```
        np.random.default_rng().random(n_b)))

    for t in range(t_n):
        impulse = v >= 30

        for i, is_impulse in enumerate(impulse):
            if is_impulse:
                if i > 799:
                    br_t_plot.append(t)
                    br_neuron_id.append(i)
                else:
                    ex_t_plot.append(t)
                    ex_neuron_id.append(i)

        v[impulse] = c[impulse]
        u[impulse] = u[impulse] + d[impulse]

        I_new = I.copy()
        I_new += np.sum(W[:, impulse], axis=1)

        for i in range(steps_in_t):
            new_v = v + h * (0.04 * v ** 2 + 5 * v + 140 - u + I)
            new_u = u + h * a * (b * v - u)
            v = new_v
            u = new_u
```

Листинг 9. Генерация матрицы смежности и векторов соответствующих констант.

```
def get_consts(n, n_e, n_b):
    # n - number of neurons
    # n_e - number of excitatory neurons
    # n_b - number of braking neurons
    W = np.hstack((np.random.default_rng().random((n, n_e)) / 2, -
        np.random.default_rng().random((n, n_b))))
    a = np.hstack((0.02 * np.ones(n_e), 0.02 + 0.08 *
        np.random.default_rng().random(n_b)))
    b = np.hstack((0.2 * np.ones(n_e), 0.25 - 0.05 *
        np.random.default_rng().random(n_b)))
    c = np.hstack((-65 + 15 * np.random.default_rng().random(n_e) ** 2, -65 *
        np.ones(n_b)))
    d = np.hstack((8 - 6 * np.random.default_rng().random(n_e) ** 2, 2 * np.ones(n_b)))

    return W, a, b, c, d
```

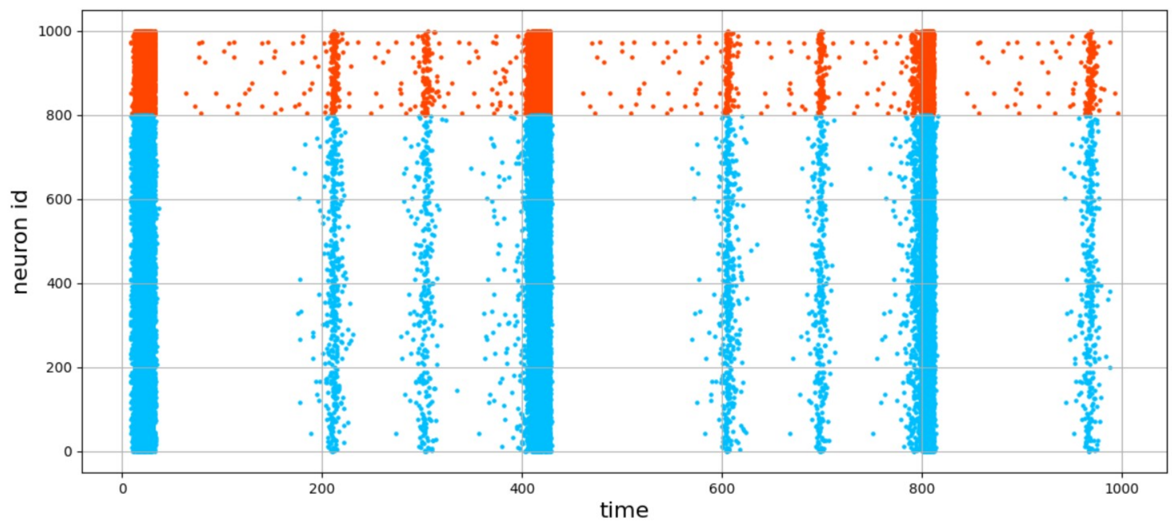

Рис. 2. Импульсы всех нейронов.

Заметим, что колебания не являются полностью синхронными, все зависит от сгенерированных параметров, примерная частота полученных колебаний 7-10 Гц.

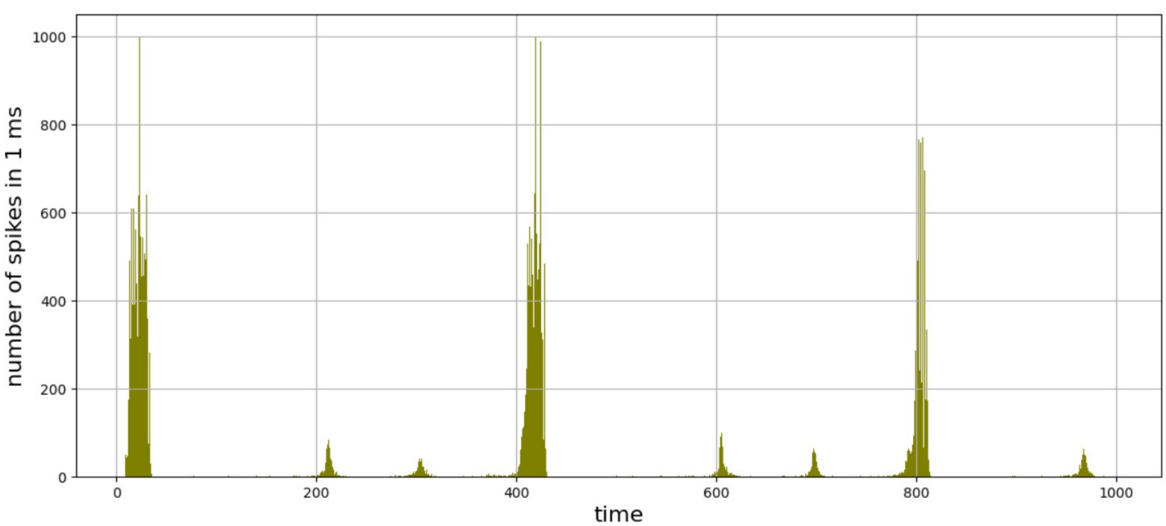

Рис. 3. Активность нейронной сети.

6 Заключение

1. Реализованы численные методы решения задачи Коши. Выявлено, что оптимальным решением является метод Рунге-Кутты 4-го порядка за счет порядка по-

грешности, однако метод Эйлера явно является интуитивно более простым и для задач данного типа проявляет хорошее соотношение степени точности и времени вычисления.

2. Проведено моделирование нейронной сети, выявлены колебания частотой $\approx 7-10$ Гц.

Список использованных источников

1. Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018-2021. С. 140.

2. Веб-ресурс(https://slemeshevsky.github.io/num-mmf/ode/html/._ode-FlatUI001.html)

Выходные данные

Кутепов Н.С.. Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2021. — 15 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 РК6-55Б, Кутепов Н.С.*

2021, осенний семестр