



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:	Степанов Никита Николаевич
Группа:	РК6-55Б
Тип задания:	лабораторная работа
Тема:	Модель биологического нейрона (вариант 5)

Студент

\_\_\_\_\_  
подпись, дата

Степанов Н. Н.  
Фамилия, И.О.

Преподаватель

\_\_\_\_\_  
подпись, дата

\_\_\_\_\_  
Фамилия, И.О.

Москва, 2021

# Содержание

<b>Модель биологического нейрона (вариант 5)</b>	<b>3</b>
1   Задание . . . . .	3
2   Цель выполнения лабораторной работы . . . . .	5
3   Дискретная траектория системы ОДУ . . . . .	5
1. Метод Эйлера . . . . .	5
2. Неявный метод Эйлера . . . . .	6
3. Метод Рунге-Кутты 4-го порядка . . . . .	7
4. Вывод полученных траекторий . . . . .	9
Особенности режимов . . . . .	10
4   Моделирование Нейронной сети . . . . .	11
Сравнение методов Эйлера (явный/неявный) и метода Рунге-Кутты . . .	11
Моделирование во времени нейронной сети с помощью метода Эйлера .	11
5   Заключение . . . . .	16

# Модель биологического нейрона (вариант 5)

## 1 Задание

### Условие

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 20 (Модель Ижикевича)

Дана система из двух ОДУ 1-го порядка:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I, \quad (1)$$

$$\frac{du}{dt} = a(bv - u), \quad (2)$$

и дополнительного условия, определяющего возникновение импульса в нейроне:

$$v \geq 30, = \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases}, \quad (3)$$

где  $v$  - потенциал мембраны (мВ),  $u$  - переменная восстановления мембраны (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 мембраны; чем больше  $a$ , тем быстрее происходит восстановление после импульса),  $b$  (чувствительность переменной восстановления к флуктуациям разности потенциалов),  $c$  (значение потенциала мембраны сразу после импульса),  $d$  (значение переменной восстановления мембраны сразу после импульса)

## Базовая часть

1. Написать 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-Кутты 4-го порядка;

2. Для каждого из реализованных методов численно найти траекторию 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 зависимость потенциала мембраны от времени, где каждый график должен соответствовать своему характерному режиму работы нейрона.

4. По полученным графикам кратко описать особенности указанных режимов.

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

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

## 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:  $a = 0.02$ ,  $b = 0.2$ ,  $c = -65 + 15\alpha^2$ ,  $d = 8 - 6\beta^2$  и внешний ток в отсутствие токов от других нейронов равен:  $I = I_0 = 5\xi$ , где  $\alpha, \beta, \xi$  - случайные числа от 0 до 1. 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  $\gamma, \delta, \zeta$  - 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. Весовая

матрица смежности  $\mathbf{W}$  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 ними нейрона  $i$  одновременно увеличивается на величину  $W_{ij}$  и затем сразу же падает до нуля, что и моделирует передачу импульса по нейронной сети. Значение  $W_{ij}$  равно  $0.5\theta$ , если нейрон  $j$  является возбуждающим, и  $-\tau$ , если тормозным, где  $\theta$  и  $\tau$  - 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 – с помощью численных методов решения задачи Коши для ОДУ 1-го порядка исследовать одну из простейших моделей спайковых нейронных сетей - модель Ижикевича.

## 3 Дискретная траектория системы ОДУ

### 1. Метод Эйлера

Необходимо реализовать функцию -  $euler(x\_0, t\_n, f, h)$ , где дискретная траектория строится с помощью метода Эйлера, все принимаемые параметры данной функции - описаны в условии;

Для ясности и лаконичности изложения, представлено пояснение и вывод данного метода согласно лекционным материалам.

Метод Эйлера - простейший численный метод решения систем обыкновенных дифференциальных уравнений.

Рассмотрим следующее ОДУ:

$$\frac{dy}{dt} = f(t, y), \quad (4)$$

где  $t \in [a; b]$  и  $y(a) = \alpha$ . Кроме того, предполагается дискретизация координаты  $t$  в сетку вида:  $t_i = a + ih, i = 1, \dots, m$ ; где  $h = \frac{b-a}{m} = t_{i+1} - t_i$  - что является шагом. Предположим, что  $y(t) \in C^2[a; b]$  и разложим функцию  $y(t)$  в ряд Тейлора в точке  $t_i$ :

$$y(t) = y(t_i) + y'(t_i)(t - t_i) + \frac{y''(\xi)}{2}(t - t_i)^2 \quad (5)$$

где  $\xi \in (t; t_{i+1})$  для  $t > t_i$ . Используя соотношение (5) вычислим значение ряда в точке  $t_{i+1}$ :

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2 y''(\xi_i)}{2} \Rightarrow y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2 y''(\xi_i)}{2}, \quad (6)$$

где  $\xi_i \in (t; t_{i+1})$ . Предположив, что  $h$  мало, отбросим член порядка  $O(h^2)$ , что и дает формулировку метода Эйлера:

$$\omega_0 = \alpha, \omega_{i+1} = \omega_i + hf(t_i, \omega_i), i = 0, 1, \dots, m - 1, \quad (7)$$

где ожидается, что  $\omega_i \approx y(t_i)$

Теперь, представим программную реализацию функции `euler(x_0, t_n, f, h)`:

#### Листинг 1. Реализация функции `euler(x_0, t_n, f, h)`

---

```
1 def euler (t_0, t_n, f, h, l=5):
2     t_nodes = int((t_n - t_0)/h)
3     u = [0 for i in range (t_nodes+1)]
4     v = [0 for i in range (t_nodes+1)]
5     v[0] = c
6     u[0] = b * v[0]
7     for i in range (t_nodes):
8         v[i+1] = v[i] + h * f[0](u[i], v[i], l)
9         u[i+1] = u[i] + h * f[1](u[i], v[i])
10
11     if v[i+1] >= 30:
12         v[i+1] = c
13         u[i+1] = u[i+1] + d
14     return u, v
```

---

## 2. Неявный метод Эйлера

Необходимо реализовать функцию - `implicit_euler(x_0, t_n, f, h)`, где дискретная траектория строится с помощью неявного метода Эйлера, все принимаемые параметры данной функции - описаны в условии.

Для ясности и лаконичности изложения, представлено пояснение данного метода, его отличие от явного метода Эйлера.

С идеологической точки зрения, *неявный метод Эйлера* является совсем небольшим усложнением явного метода. Основная идея неявного метода Эйлера заключается в том, что неизвестные значения  $\omega_{i+1}$  **могут входить** как в левую, так и в правую части уравнения (7), что модифицирует данное уравнение:

$$\omega_{i+1} = \omega_i + hf(t_i, \omega_{i+1}), i = 0, 1, \dots, m - 1, \quad (8)$$

тем самым **можно повысить точность** представления правой части. Значения функции  $f$  на каждой итерации берутся на новом временном слое и производится решение нелинейного уравнения.

Решение нелинейного уравнения будет производиться с помощью функции `root` пакета `scipy.optimize` библиотеки для научных вычислений - `scipy`. Данная функция `scipy.optimize.root(fun, x0, args=(), method='hybr', jac=None, tol=None, callback=None, options=None)` находит **корень (решение) функции**, заданной первым принимаемым аргументом `fun`, используя начальное условие `x0` и дополнительные аргументы `args`, необходимые для нахождения решения; остальные принимаемые параметры данной функции не используются, и, соответственно, не комментируются.

Таким образом, согласно соотношению (8), создаются две вспомогательные функции `mod_v` и `mod_u`, относительно которых и будет производиться решение с помощью функции `scipy.optimize.root`.

Теперь, представим программную реализацию функции `implicit_euler(x_0, t_n, f, h)`:

## Листинг 2. Реализация функции `implicit_euler(x_0, t_n, f, h)`

---

```

1 def implicit_euler(t_0, t_n, f, h, l = 5):
2     t_nodes = int((t_n - t_0)/h)
3     u = [0 for i in range (t_nodes+1)]
4     v = [0 for i in range (t_nodes+1)]
5
6     v[0] = c
7     u[0] = b * v[0]
8
9     def mod_v(v_i1, u_i, v_i):
10         return v_i1 - v_i - h * f[0](u[i], v[i], l)
11
12     def mod_u(u_i1, u_i, v_i):
13         return u_i1 - u_i - h * f[1](u[i], v[i])
14
15     for i in range (t_nodes):
16         v[i+1] = (optimize.root(mod_v, v[i], args = (u[i], v[i]))).x[0]
17         u[i+1] = (optimize.root(mod_u, u[i], args = (u[i], v[i]))).x[0]
18
19         if v[i+1] >= 30:
20             v[i+1] = c
21             u[i+1] = u[i+1] + d
22
23     return u, v

```

---

### 3. Метод Рунге-Кутты 4-го порядка

Вывод данного метода не будет приводится, т. к. он достаточно объемен и трудоемок. Однако он в полной мере присутствует в **лекционных материалах**, куда и ссылается автор.

Согласно лекционным материалам, формулировка метода Рунге-Кутты 4-го порядка для систем ОДУ имеет вид:

$$w_0 = \alpha, \quad (9)$$

$$k_1 = hf(t_i, w_i), \quad (10)$$

$$k_2 = hf(t_i + \frac{h}{2}, w_i + \frac{1}{2}k_1), \quad (11)$$

$$k_3 = hf(t_i + \frac{h}{2}, w_i + \frac{1}{2}k_2), \quad (12)$$

$$k_4 = hf(t_i + h, w_i + k_3), \quad (13)$$

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

Собственно, реализация соотношений (9), (10), (11), (12), (13), (14) составляет основу функции *runge\_kutta*

Теперь, представим программную реализацию функции `runge_kutta(x_0, t_n, f, h)`:

**Листинг 3. Реализация функции `runge_kutta(x_0, t_n, f, h)`**

---

```
1 def runge_kutta(t_0, t_n, f, h, l = 5):
2
3     t_nodes = int((t_n - t_0)/h)
4     u = [0 for i in range (t_nodes+1)]
5     v = [0 for i in range (t_nodes+1)]
6
7     v[0] = c
8     u[0] = b * v[0]
9     for i in range (t_nodes):
10         k1_v = h * f[0](u[i], v[i], l)
11         k1_u = h * f[1](u[i], v[i])
12
13         k2_v = h * f[0](u[i]+h/2, v[i] + 0.5 * k1_v, l)
14         k2_u = h * f[1](u[i]+0.5 * k1_u, v[i] + h/2)
15
16         k3_v = h * f[0](u[i] + h/2, v[i] + 0.5 * k2_v, l)
17         k3_u = h * f[1](u[i] + 0.5 * k2_u, v[i] + h/2)
18
19         k4_v = h * f[0](u[i] + h, v[i] + k3_v, l)
20         k4_u = h * f[1](u[i] + k3_u, v[i] + h)
21
22         v[i+1] = v[i] + (1/6) * (k1_v + 2* k2_v + 2 * k3_v + k4_v)
23         u[i+1] = u[i] + (1/6) * (k1_u + 2* k2_u + 2 * k3_u + k4_u)
24
25         if v[i+1] >= 30:
26             v[i+1] = c
27             u[i+1] = u[i+1] + d
28
29     return u, v
```

---



#### 4. Вывод полученных траекторий

Изобразим полученные траектории (зависимость потенциала мембраны от времени) на четырех отдельных графиках как зависимость потенциала мембраны от времени, где каждый график должен соответствовать своему характерному режиму работы нейрона.

Предварительно зададим словарь, содержащий характеристики каждого из режимов работы.

##### Листинг 4. Характерные режимы работы динамической системы

---

```
1 modes = {  
2     "TS" : [0.02, 0.2, -65, 6],  
3     "PS" : [0.02, 0.25, -65, 6],  
4     "C"  : [0.02, 0.2, -50, 2],  
5     "FS" : [0.1, 0.2, -65, 2]  
6 }
```

---

Теперь приведем программную реализацию вывода траекторий динамической системы, согласно заданным режимам.

##### Листинг 5. Вывод траекторий дин. системы ОДУ для соответствующего режима работы

---

```
1 fig, ax = plt.subplots(4, 1, figsize = (14, 14))  
2 captions = ["Tonic spiking (TS)", "Phasic spiking (PS)", "Chattering (C)", "Fast Spiking (FS)"]  
3 for i in range(0, 4):  
4     a = modes[list(modes.keys())[i]][0]  
5     b = modes[list(modes.keys())[i]][1]  
6     c = modes[list(modes.keys())[i]][2]  
7     d = modes[list(modes.keys())[i]][3]  
8     u1, v1 = euler(t_0, t_n, [f1, f2], h)  
9     u2, v2 = implicit_euler(t_0, t_n, [f1, f2], h)  
10    u3, v3 = runge_kutta(t_0, t_n, [f1, f2], h)  
11    ax[i].set_title(captions[i], fontsize = 12)  
12    ax[i].plot(t, v1, '--g', label = "Euler method", ms = 6 )  
13    ax[i].plot(t, v2, '-.r', label = "Implicit Euler method", ms = 7)  
14    ax[i].plot(t, v3, ':b', label = "Runge-Kutta method", ms = 8)  
15    ax[i].set_ylabel('potentials')  
16    ax[i].grid()  
17    ax[i].legend(loc = 'upper right')  
18 fig.suptitle("Dependence potential of membrane on time (ms)", fontsize=16)  
19 plt.show()
```

---

Теперь, используя Листинг (5), а также Листинги (1), (2), (3), (4), представим полученные траектории соответствующих методов и режимов.

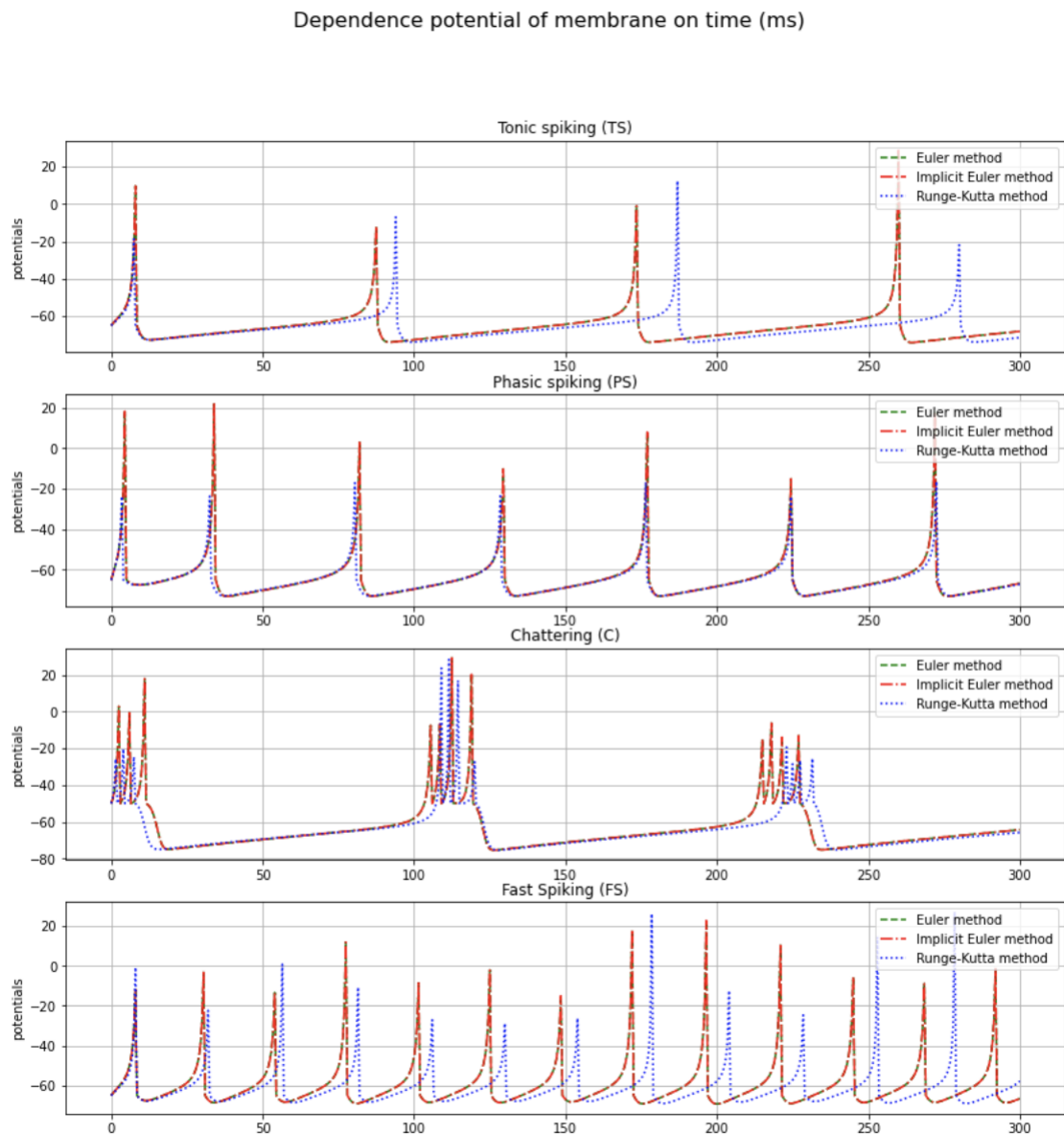


Рис. 1. Траектории динамических систем для соответствующих режимов и методов

## Особенности режимов

Как видно из рисунка (1), каждый из режимов отличен друг от друга.

Для режима *Tonic spiking* характерны 2 импульса, между которыми происходит длительное восстановление мембраны. Особенность данного режима по сравнению с другими - длительное восстановление мембраны.

Для режима *Phasic spiking* характерно, в сравнении с предыдущим, что количество импульсов увеличилось, а длительность восстановления мембраны уменьшилась.

Для режима *Chattering* видно, что происходит три импульса подряд, между которыми восстановление мембраны происходит практически мгновенно, а затем следует длительное восстановление мембраны. Особенность режима *Chattering* заключается в нескольких подряд идущих импульсах, между которыми восстановление мембраны происходит быстро, а затем, после этих импульсов - длительное восстановление мембраны.

Для режима *Fast Spiking* заметно, что количество импульсов, в сравнении с другими режимами, - максимально, а средняя длительность восстановления мембраны - минимальна.

## 4 Моделирование Нейронной сети

### Сравнение методов Эйлера (явный/неявный) и метода Рунге-Кутты

Как уже было сказано ранее - неявный метод Эйлера - небольшая модификация явного метода Эйлера. Неявные методы в общем случае всегда дают более точный результат, чем явные методы, для одного и того же количества шагов или вычислений функции. Ключевое отличие их заключается в том, что неявный метод Эйлера является абсолютно устойчивым, в то время как явный - лишь условно устойчивый. Так как точность метода определяется отброшенными членами ряда, то согласно (6), метод Эйлера имеет порядок точности  $\approx h^2$ .

Самое большое распространение из всех численных методов решения дифференциальных уравнений с помощью ЭВМ получил метод Рунге-Кутты 4-го порядка. В литературе он известен как метод Рунге-Кутты. Отметим, что метод Рунге-Кутты 2-го порядка так же называют модифицированным методом Эйлера.

$$w_0 = \alpha, w_{i+1} = w_i + hf\left(t_i + \frac{h}{2}, w_i + \frac{h}{2}f(t_i, w_i)\right) \quad (15)$$

Метод Рунге-Кутты 4-го порядка, имеет больший порядок точности чем метод Эйлера ( $\approx h^4$ ), однако он является алгебраически более трудоемким. Явный метод Рунге-Кутты так же является условно-устойчивым (неявный - абсолютно устойчивый)

Таким образом, Метод Эйлера (явный/неявный), классический Метод Рунге-Кутты - это численные методы получения решения дифференциальных уравнений. Они различаются порядком точности, степенью устойчивости и сложностью алгебраического воспроизведения (т. е. фактически - сложностью реализации и кол-во требуемых выч. ресурсов).

### Моделирование во времени нейронной сети с помощью метода Эйлера

Необходимо с помощью заданной модели Ижикевича смоделировать динамику нейронной сети, состоящей из 1000 нейронов - 800 возбуждающих, 200 - тормозных.

Прежде чем осуществлять непосредственное моделирование - необходимо произвести первичную инициализацию нейронов и их параметров. Кроме того, необходимо создать и проинициализировать весовую матрицу смежности **W** - матрицу внешних токов, при появлении в сети импульса. Так же требуется, с помощью начальных параметров задать временную сетку:

$$h1 = (t_{n1} - t_{01}) / (\text{len}(t_1) - 1)$$

Приведем программную реализацию вышеперечисленного:

#### Листинг 6. Инициализация необходимых параметров

---

```

1 W_matrix = np.zeros((1000, 1000))
2 for i in range (0, 1000):
3     for j in range (0,800):
4         if i != j:
5             W_matrix[i][j] = 0.5 * np.random.uniform()
6     for j in range (800, 1000):
7         if i != j:
8             W_matrix[i][j] = -np.random.uniform()
9
10 params_neurons = [{'a' : 0 , 'b' : 0, 'c' : 0, 'd' : 0, 'l' : 0, 'id' : 0} for i in range (1000)]
11 id = 1
12
13 for i in range (800):
14     params_neurons[i]['a'] = 0.02
15     params_neurons[i]['b'] = 0.2
16     params_neurons[i]['c'] = -65 + 15 * np.random.uniform() ** 2
17     params_neurons[i]['d'] = 8 - 6 * np.random.uniform() ** 2
18     params_neurons[i]['l'] = 5 * np.random.uniform()
19     params_neurons[i]['id'] = id
20     id += 1
21
22 for i in range (800, 1000):
23     params_neurons[i]['a'] = 0.02 + 0.08 * np.random.uniform()
24     params_neurons[i]['b'] = 0.25 - 0.05 * np.random.uniform()
25     params_neurons[i]['c'] = -65
26     params_neurons[i]['d'] = 2
27     params_neurons[i]['l'] = 2 * np.random.uniform()
28     params_neurons[i]['id'] = id
29     id += 1
30
31 t_1 = np.linspace(0, 1000, 21)
32 t_0_1 = 0
33 t_n_1 = 1
34 h1 = (t_n_1 - t_0_1) / (len(t_1) - 1)
35 increased_current = np.zeros((len(t_1), 1000))

```

---

Как можно видеть из Листинга (6), моделирование автором производится на промежутке от 0 до 1000 мс, с шагом 50 мс.

Далее автором вводится понятие - "вспомогательные параметры" - это фактически параметры нейрона, рассматриваемого в данный момент времени. Необходимость ввода таких параметров обусловлена сугубо удобством автора.

Дальнейшая реализация сводится к следующему алгоритму:

1. Производится первичная инициализация нулями двумерной матрицы *increased\_current* - содержащей значения токов нейронов в случае возникновения импульса.
2. Производится инициализация вспомогательных параметров параметрами конкретного нейрона.
3. Вспомогательный ток инициализируется либо соответствующим элементом матрицы *increased\_current* (если ранее был импульс и там не ноль), либо базовым током рассматриваемого нейрона.
4. Далее в цикле рассматривается конкретный нейрон во всех промежутках времени, если на **каком** из промежутков был импульс - выводятся номер нейрона, временной промежуток и значение потенциала; соответствующие элементы матрицы *increased\_current* увеличиваются согласно условию (ток от нейрона с импульсом передается другим нейронам)

Модель нейронной сети выводится как в трех-мерном представлении, так и в двух-мерном представлении.

Представим программную реализацию вышенаписанного алгоритма.

#### Листинг 7. Моделирование нейронной сети в 3-х мерном пространстве

```
1 fig = plt.figure(figsize=(14, 10))
2 ax_3d = Axes3D(fig)
3 ax_3d.set_xlabel('time (ms)', fontsize = 15)
4 ax_3d.set_ylabel('neuron id', fontsize = 15)
5 ax_3d.set_zlabel('V', fontsize = 17)
6 l = [0 for c in range(len(t_1))]
7 increased_current = np.zeros((len(t_1), 1000))
8 for i in range(0, 1000):
9     a = params_neurons[i]['a']
10    b = params_neurons[i]['b']
11    c = params_neurons[i]['c']
12    d = params_neurons[i]['d']
13    for k in range(len(t_1)):
14        if increased_current[k][i] == 0:
15            l[k] = params_neurons[i]['l']
16        else:
17            l[k] = increased_current[k][i]
18    id = params_neurons[i]['id']
19    v = mod_euler(t_0_1, t_n_1, [f1, f2], h1, l)
```

```

20 for p in range (len(v)):
21     if v[p] == c:
22         for j in range (1000):
23             increased_current[p][j] = increased_current[p][j] + params_neurons[j]['I'] +
                W_matrix[j][i]
24         if (i+1 <= 800):
25             _ = ax_3d.scatter(t_1[p], i+1, v[p], color = 'red')
26         else:
27             __ = ax_3d.scatter(t_1[p], i+1, v[p], color = 'blue')
28 ax_3d.set_title("Neural network modeling (3D)", loc='center', fontsize = 25)
29 _.set_label('Excitatory neuron')
30 __.set_label('Inhibitory neuron')
31 plt.legend(fontsize = 14)
32 plt.show()

```

Моделирование в 2-х мерном пространстве по своей сути аналогично 3-х мерному, поэтому для большей лаконичности изложения, **приводиться не будет.** Согласно Листингу (7) изобразим моделирование нейронной сети в 3-х мерном пространстве.

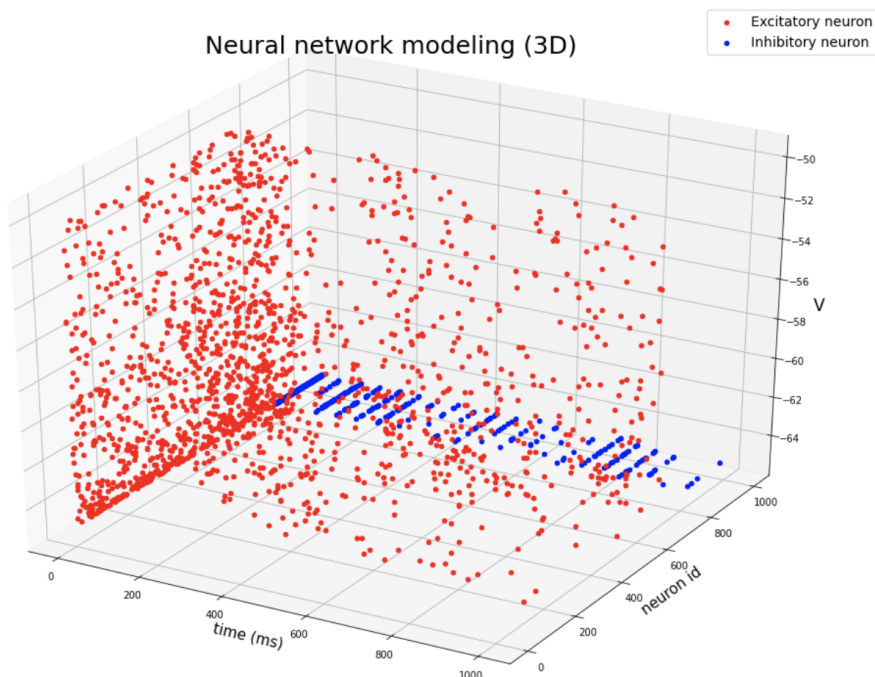


Рис. 2. Моделирование импульсов нейронов в трехмерном представлении. Красный цвет - возбуждающий нейрон, синий - тормозной нейрон.

Теперь, приведем моделирование нейронной сети в 2-х мерном пространстве. Оси ординат будет соответствовать *id* нейрона, оси абсцисс - временной момент моделирования. Точка соответствует случившемуся импульсу данного нейрона.

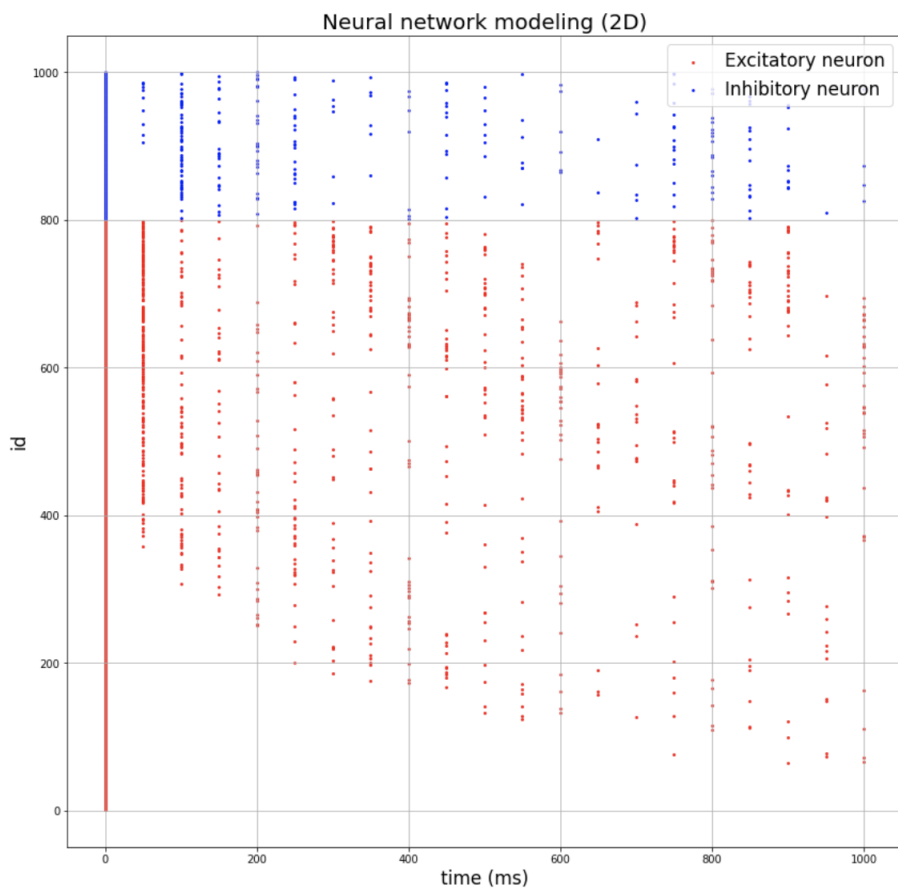


Рис. 3. Моделирование импульсов нейронов в двухмерном представлении. Красный цвет - возбуждающий нейрон, синий - тормозной нейрон. Точка - импульс.

Судя по полученным графикам (2), (3), заметим, что колебания являются частично синхронными - все зависит от сгенерированных параметров нейрона и, следовательно, - от его типа. Так, например, очевидно, что для возбуждающего нейрона возникновение импульса - гораздо характернее, чем для тормозного нейрона.

Тем не менее, примерная частота полученных колебаний составляет 10-15 Гц.

## 5 Заключение

1. Проведена реализация численных методов решения задачи Коши - явный метод Эйлера, неявный метод Эйлера, метод Рунге-Кутты.
2. С помощью вышеперечисленных методов были построены траектории динамических систем для соответствующего режима работы **Нейрона**.
3. Произведено моделирование нейронной сети, состоящей из 1000 нейронов - 800 возбуждающих, 200 тормозных.
4. Получена графическая интерпретация моделирования как в 3-х мерном пространстве, так и в 2-х мерном пространстве.
5. Определен тип колебаний импульсов нейронов - частичные колебания. Определена примерная частота полученных колебаний - 10-15 Гц.



В совокупности, данная лабораторная работа позволила лучше понять и проработать материал, связанный с понятиями: модель Ижикевича, задача Коши, явный метод Эйлера, неявный метод Эйлера, Методы Рунге-Кутты, флуктуации, возбуждающий нейрон, тормозной нейрон.

### Список использованных источников

1. Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018-2021. С. 140.
2. [Электронный ресурс] Першин А. Ю. Видео-лекции по курсу "Вычислительная математика". Москва, 2021 [https://www.youtube.com/channel/UC69GDhPVL\\_Y-7IXn3EhmcH2w](https://www.youtube.com/channel/UC69GDhPVL_Y-7IXn3EhmcH2w)

### Выходные данные

Степанов Н. Н.. Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2021. — 16 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, Степанов Н. Н.

2021, осенний семестр