



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего 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

Тетерин Никита Евгеньевич

Группа

РК6-546

Тип задания

лабораторная работа №3

Тема лабораторной работы

Модель биологического нейрона
(вариант 5)

Студент

Тетерин Н.Е.

подпись, дата

фамилия, и.о.

Преподаватель

Соколов А.П.

подпись, дата

фамилия, и.о.

Оценка _____

Москва, 2021 г.

Оглавление

Задание на лабораторную работу	3
Цель выполнения лабораторной работы	3
Выполненные задачи.....	3
1. ОДУ и постановка задачи Коши	4
2. Алгоритмы для численных методов решения задачи Коши для систем ОДУ	5
3. Моделирование активности нейрона.....	9
4. Моделирование нейронной сети.....	11
Заключение.....	15
Список использованных источников	16

Задание на лабораторную работу

Реализовать различные численные схемы для решения задачи Коши для систем обыкновенных дифференциальных уравнений (ОДУ) в общем случае и проанализировать их особенности. Смоделировать траекторию динамической системы, описывающей поведение биологического нейрона в модели Ижикевича. Произвести моделирование системы из таких нейронов с учетом обратных связей по току, симулирующих их объединение в сеть.

Цель выполнения лабораторной работы

Цель выполнения лабораторной работы – имплементировать следующие алгоритмы численных методов для решения задачи Коши для систем ОДУ: прямой метод Эйлера, обратный (неявный) метод Эйлера и метод Рунге-Кутты 4-го порядка. Проанализировать их сильные и слабые стороны и применить для моделирования активности биологических нейронов.

Выполненные задачи

1. Разработать алгоритм для прямого метода Эйлера.
2. Разработать алгоритм для обратного метода Эйлера с использованием численного решения нелинейного уравнения.
3. Разработать алгоритм для метода Рунге-Кутты 4-го порядка.
4. Создать единый интерфейс для реализованных методов для задачи произвольной размерности.
5. Определить траектории динамической системы для шага $h = 0.1$ для всех характерных режимов работы нейрона с помощью каждого из численных методов. Построить графики, содержащие полученные импульсы.
6. Смоделировать активность нейронов в сети, содержащей возбуждающие и тормозящие нейроны с учетом связи по току и без.

1. ОДУ и постановка задачи Коши

Пусть имеется некоторое нормализованное (разрешенное относительно производных) ДУ первого порядка (1.1). Тогда задача Коши для него может быть сформулирована путем задания начального условия (НУ) (1.2). Системы ОДУ, а также ОДУ более высоких порядков могут быть переформулированы и сведены к форме (1.3), где функция f зависит от набора переменных y , суть есть вектора некоторой размерности.

$$(1.1) \quad y'(t) = f(t, y), \quad t \in [a; b].$$

$$(1.2) \quad y(a) = y_0$$

$$(1.3) \quad y'(t) = \frac{dy}{dt} = f(t, y), \quad y_1(a) = y_{10}, \dots, y_n(a) = y_{n0}, \quad t \in [a; b].$$

В рамках данной работы система ОДУ, описывающая активность нейрона (по Ижикевичу), формулируется следующим образом:

$$(1.4) \quad \begin{cases} \frac{dv}{dt} = f_1(v, u) = 0.04v^2 + 5v + 140 - u + I; \\ \frac{du}{dt} = f_2(v, u) = a(bv - u); \end{cases}$$

И включает также дополнительное условие, определяющее возникновение импульса в нейроне:

$$\text{если } v \geq 30, \quad \begin{cases} u \leftarrow u + d \\ v \leftarrow c \end{cases}$$

здесь используются следующие обозначения: v – потенциал мембранны (мВ), u – переменная восстановления мембранны, учитывающая активацию тока ионов K^+ и инактивацию токов ионов Na^+ , предоставляющая негативную обратную связь v . a определяет временной масштаб для восстановления потенциала мембранны, b описывает чувствительность переменной восстановления к подпороговым флюктуациям v , c является величиной потенциала сразу после импульса. Аналогично, d определяет сдвиг u в момент сразу после импульса, I – внешний ток, проходящий через синапс нейрона от всех нейронов, с которыми он связан. В табл. 1.1 приведены значения параметров модели для четырех характерных режимов работы нейрона, определяющие траекторию динамической системы. Внешний ток здесь принимается равным $I = 5$. Начальными условиями выступают $v(0) = c$, $u(0) = bv(0)$.

Режим	a	b	c	d
Tonic spiking (TS)	0.02	0.2	-65	6
Phasing 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 1.1. Характерные режимы динамической системы и соответствующие значения её параметров

2. Алгоритмы для численных методов решения задачи Коши для систем ОДУ

Сформулируем численные методы, использованные в данной работе для решения задачи Коши в форме (1.1–2). Произведем дискретизацию независимой координаты (времени) с фиксированным шагом h . В результате получаем некоторый набор абсцисс t_i , $i = 1, \dots, n$. Формулировка прямого метод Эйлера (также называемого явным) получается путем разложения функции $f(t, y)$ в ряд Тейлора в каждом из узлов $t = t_i$ и отбрасывания членов ряда порядка малости $O(h^2)$ и ниже. Получаем:

$$(2.1) \quad y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2y''(\xi_i)}{2} \Rightarrow y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2y''(\xi_i)}{2}, \quad \xi \in (t_i; t_{i+1})$$

при малом h можно записать: $w_0 = \alpha$, $w_{i+1} = w_i + hf(t_i, w_i)$, $i = 0, \dots, n - 1$

Неявный (также называемый обратным) метод Эйлера получается аналогичным образом, но путем разложения исходной функции в ряд Тейлора не в точке t_i , а в точке t_{i+1} . Запись остается той же, но в качестве аргумента функции $f(t, y)$ выступает значение w_{i+1} , таким образом получается набор в общем случае нелинейных уравнений для каждого из w_{i+1} :

$$(2.2) \quad w_0 = \alpha, w_{i+1} = w_i + hf(t_i, w_{i+1}), \quad i = 0, \dots, n - 1$$

Вывод метода Рунге-Кутты 4 порядка несколько длиннее, поэтому запишем его формулировку сразу, без доказательства:

$$\begin{aligned} (2.3) \quad & w_0 = \alpha \\ & k_1 = hf(t_i, w_i) \\ & k_2 = hf(t_i + h/2, w_i + k_1/2) \\ & k_3 = hf(t_i + h/2, w_i + k_2/2) \\ & k_4 = hf(t_i + h, w_i + k_3) \\ & w_{i+1} = w_i + (k_1 + 2k_2 + 2k_3 + k_4)/6, \quad i = 0, \dots, n - 1 \end{aligned}$$

Исходный код для данной ЛР традиционно был написан на ЯП *Python*. На листинге 2.1 приведен исходный код функции *solve_ode* с документацией, расположенной в модуле *utils.odeint.py*. Этот метод является интерфейсом для всех численных методов. Чтобы применить требуемый, во время вызова функции необходимо специфицировать аргумент *method*, принимающий одну из трех возможных опций: *euler*, *imp-euler*, *runge-kutta*, каждая из которых, очевидно, отвечает за использование соответствующего метода. Помимо спецификатора метода, сигнатура функции содержит следующие аргументы: *t_0* и *t_n* (соответственно левая и правая границы интервала интегрирования по времени), *f* (функция, содержащая правую часть ОДУ), величина шага *h*, optionalный аргумент *constraint*, в качестве которого нужно передавать функцию, которая будет ограничивать полученное решение (например, снизу или сверху, либо, как в случае рассматриваемой в данной работе системы, являясь дополнительным условием). Алгоритмически численные методы являются довольно прямолинейными в том отношении, что на каждой текущей итерации лишь рассчитывают траекторию динамической системы, основываясь на значениях, полученных на предыдущем шаге. Результатом работы функции является словарь с ключами *t* и *y*, содержащими соответственно коллекцию значений независимой переменной на заданном интервале и коллекцию из значений всех компонент функций *y* на каждом шаге.

```

odeint.py M ×
1 import numpy as np
2 from scipy.optimize import root
3
4 def solve_ode(x_0: list or np.ndarray, t_n: np.number, f, constraint=None, t_0=.0, h=5e-1, method='euler'):
5     ...
6         the function implements solvers for systems of ordinal differential equations, represented by a single
7         vector-function of single independent variable
8
9     + `x_0`: `list` or `np.ndarray` - the initial condition of dynamic system
10
11    + `t_n`: `np.number` - the right bound independent variable value
12
13    + `f`: function, the right part of the ODE
14
15    + `constraint`: callable, is called to validate output of the method at each step
16
17    + `t_0`: `np.number` - the left bound independent variable value
18
19    + `h`: float, should be less then `t_n - t_0`, the computation step
20
21    + `method`: one of ('euler', 'imp-euler', 'runge-kutta'). the method to use to obtain numeric solution of the ODE.
22        Note that 'euler' has pretty bad accuracy (the worst, actually), 'imp-euler', representing backward (implicit) Euler method
23        uses `root` to find numeric solution of the non-linear equation at each step thus is vulnerable to function shape
24        'runge-kutta' utilizes 4-th order Runge-Kutta method which has greater accuracy but is significantly slower as it makes 4 calls
25        to `f` per step
26
27
28 methods = ('euler', 'imp-euler', 'runge-kutta')
29 if method not in methods: raise ValueError(f'Invalid method name, expected one of {methods}')
30
31 if t_n < t_0: raise ValueError(f'Invalid t bounds')
32 if t_n < t_0 + h : raise ValueError(f'The step value is too big')
33 if not callable(f): raise TypeError('f provided is not a callable')
34 if constraint is not None and not callable(constraint): raise TypeError('User-provided constraint should be a callable')
35 if constraint is None: constraint = lambda t, x: x
36
37 x_0 = np.asarray(x_0)
38 t_space = np.arange(t_0, t_n + h, step=h)
39 f_space = np.zeros(shape=(len(t_space), len(x_0)))
40 f_space[0] = x_0
41
42 def euler():
43     for i, t in enumerate(t_space[:-1]):
44         w = f_space[i]
45         w = w + h*f(t, w)
46         f_space[i+1] = constraint(t, w)
47     return dict(t=t_space, y=f_space)
48
49 def imp_euler():
50     for i, t in enumerate(t_space[:-1]):
51         w = f_space[i]
52         sol = root(lambda x: w - x + h*f(t, x), w, method='hybr')
53         w = sol.x[0] if len(sol.x) == 1 else sol.x
54         f_space[i+1] = constraint(t, w)
55     return dict(t=t_space, y=f_space)
56
57 def runge_kutta():
58     for i, t in enumerate(t_space[:-1]):
59         w = f_space[i]
60
61         k1 = h * f(t, w)
62         k2 = h * f(t + .5*h, w + .5*k1(t, w))
63         k3 = h * f(t + .5*h, w + .5*k2(t, w))
64         k4 = h * f(t + h, w + k3(t, w))
65
66         w = w + (k1 + 2*k2 + 2*k3 + k4) / 6
67         f_space[i+1] = constraint(t, w)
68     return dict(t=t_space, y=f_space)
69
70
71 methods = dict(zip(methods, (euler, imp_euler, runge_kutta)))
72 return methods[method]()

```

Листинг 2.1. Исходный код функции-решателя задачи Коши для всех методов

Перед непосредственным применением для решения системы (1.4), реализованные методы были протестиированы путем аппроксимации решения задачи Коши (2.4), которое описывает параметризованную сигмоиду общего вида (рис. 2.1). Для сравнения приведено точное решение и графики абсолютной погрешности схем (шаг принимался равным 2e-1). Отметим, что при увеличении шага (например, до 5e-1) и метод Рунге начинает отклоняться от точного решения, но порядок этих отклонений ожидаемо ниже порядков обеих вариаций метода Эйлера. Можно утверждать, что неявный метод Эйлера имеет склонность к появлению ошибок округления и ошибок метода, возникающих в ходе численного решения нелинейного уравнения (2.2) с помощью функции `scipy.optimize.root`. Приведем особенности каждого из вышеупомянутых методов:

1. Явный метод Эйлера, очевидно, является наиболее быстро вычисляемым, поскольку требует единственного вызова функции из правой части ОДУ. При этом он обладает наихудшей точностью, поскольку локальная погрешность имеет порядок малости $O(h^2)$, в то время как глобальная кумулятивная погрешность имеет порядок $\sim O(h)$. Условно устойчив.
2. Неявный метод Эйлера, в свою очередь, не может считаться быстрым, поскольку на каждой своей итерации требует решения в общем случае нелинейного уравнения относительно w_{i+1} , в этом и заключается его «неявность»: значение текущего вычисляемого узла оказывается в обеих частях уравнения. Порядок локальной и глобальной погрешностей у него такой же, как и у явного, то есть он является методом первого порядка, но при этом у него появляется важное свойство, заключающееся в абсолютной устойчивости (устойчивость вне зависимости от шага). Однако, если при решении нелинейного уравнения используется некоторая численная схема, судить об устойчивости становится невозможно. Очередным минусом данной схемы является необходимость большого числа вычислений переданной функции для численного решения нелинейного уравнения/затраты на его символьное решение.
3. Особенностью метода Рунге-Кутты является вложенность, ключевой идеей вывода является замена выражения в ряде Тейлора для $f(t, y)$ вплоть до остаточного члена требуемой малости, исключающая наличие производных от $f(t, y)$ (которые, как правило, затруднительно определить, ведь численное дифференцирование резко понижает потенциально возможный порядок точности, а аналитических выражений для них обычно нет). Это производится с помощью формулы для ряда Тейлора функции двух переменных и некоторой обобщенной формы с неопределенными коэффициентами, с дальнейшим их подбором методом неопределенных коэффициентов. Метод Рунге-Кутты является оптимальной схемой для данной задачи (как и для многих других). Он совмещает удовлетворительный порядок локальной и полной погрешностей ($O(h^5)$ и $O(h^4)$ соответственно) с небольшим числом вычислений значения функции (4), хоть и является только условно устойчивым. Отметим, что при дальнейших попытках повысить порядок погрешности для его увеличения на 1 потребуется уже более 1 вычисления функции, что является не столь оптимальным (но, очевидно, может быть оправданным в некоторых приложениях). На рис.2.2-2.3 можно наблюдать разительное отличие в погрешностях первого порядка методов Эйлера и погрешности 4-го порядка метода Рунге-Кутты.

$$(2.4) \quad \frac{dy}{dt} = 3y - 2y^2, \quad C = 0, t \in [-4; 4]$$

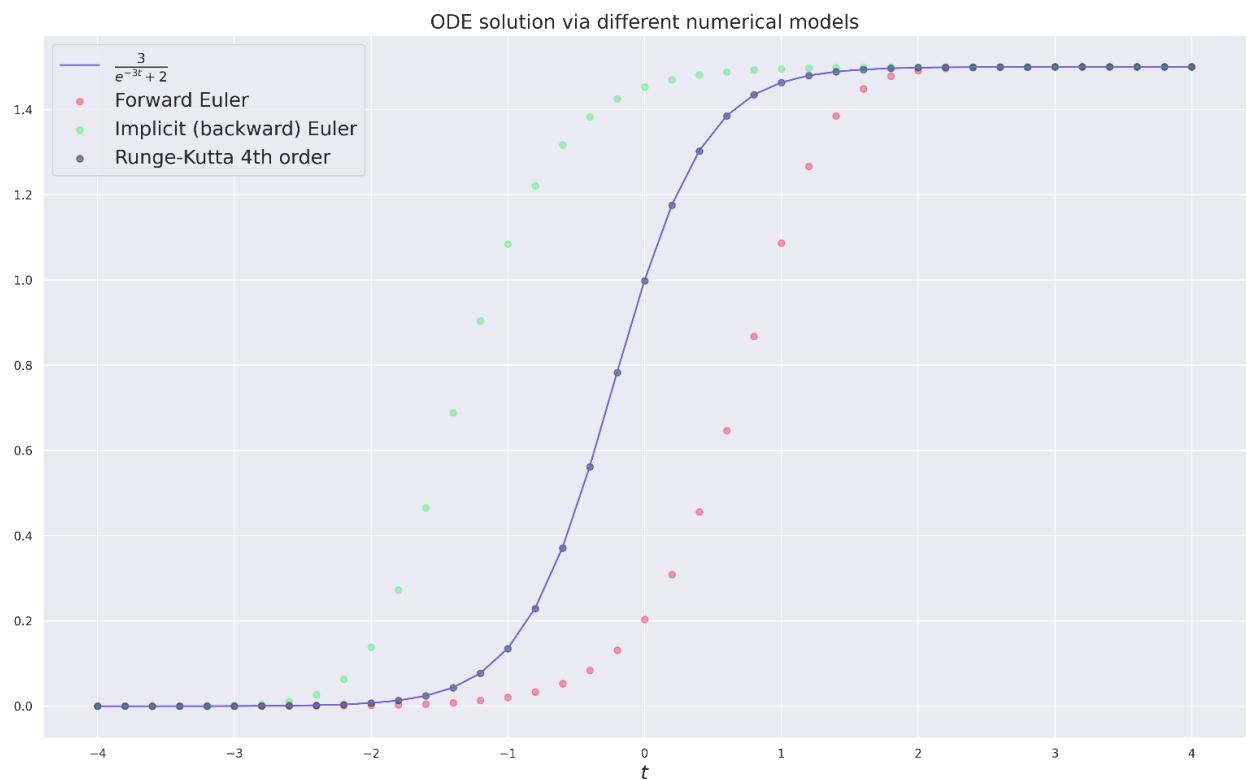


Рис. 2.1. Точное решение задачи Коши (2.4) и приближения, полученные путем применения реализованных численных методов (шаг $h=2e-1$)

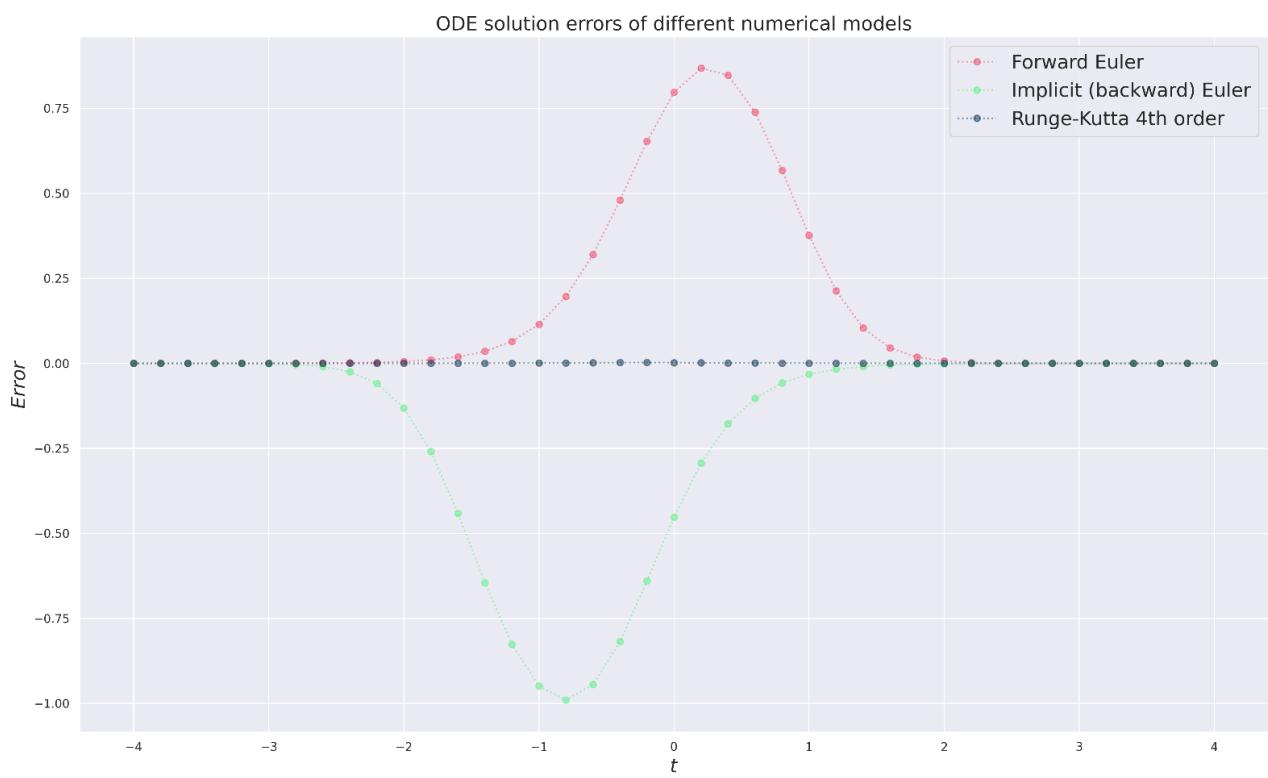


Рис. 2.2. Абсолютная погрешность аппроксимации решения задачи Коши (2.4) (со знаком)

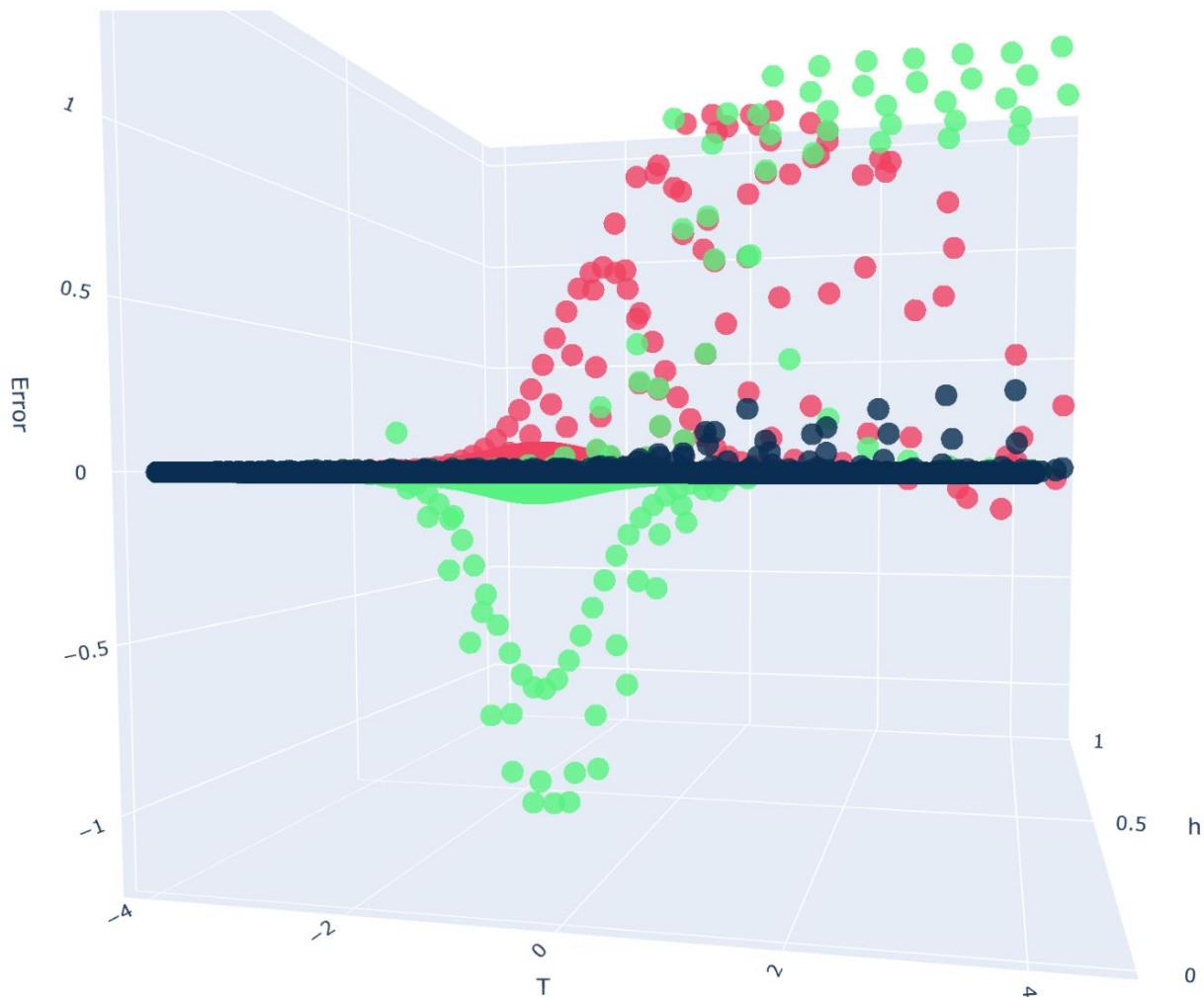


Рис. 2.3. Поверхности ошибки для различных значений шага h (от $1e-2$ до $1e0$): прямой метод Эйлера (красные круги), неявный метод Эйлера (зеленые круги), метод Рунге-Кутты 4 порядка (темно-синие круги)

3. Моделирование активности нейрона

На рис.3.1 приведены графики, соответствующие характерным режимам работы нейрона с параметрами, соответствующими табл. 1.1. Моделирование производилось на протяжении 300мс (ось абсцисс), можно обнаружить характерные особенности каждого из режимов, имеющие нейробиологическую интерпретацию. Так, режим Phasic spiking (PS) соответствует большинству возбуждающих нейронов в коре головного мозга. Имея продолжительный стимул в виде постоянного тока, такие нейроны производят несколько более коротких импульсов, после чего частота импульсов уменьшается и остается примерно постоянной (на уровне ~ 21 Гц). Такое начальное поведение обуславливается т. н. импульсной адаптацией и прослеживается у остальных режимов. Набор параметров, определяющий Tonic Spiking, описывает поведение

таламокортикальных нейронов, которые являются основным источником импульсов коры. Частота их пиков ниже, чем у PS и составляет $\sim 11\text{-}12\text{Гц}$. Следующий режим, Chattering (C), моделирует активность возбуждающих нейронов с характерными всплесками активности высокой частоты (более 200Гц). Подобное поведение связано с высоким значением c (-50мВ) и небольшому значениюю d (2). Fast spiking соответствует одному из режимов тормозящих нейронов с высокой частотой пиков. В модели подобная высокая частота ($\sim 45\text{Гц}$) соответствует $a = 0.1$ (быстрое восстановление u).

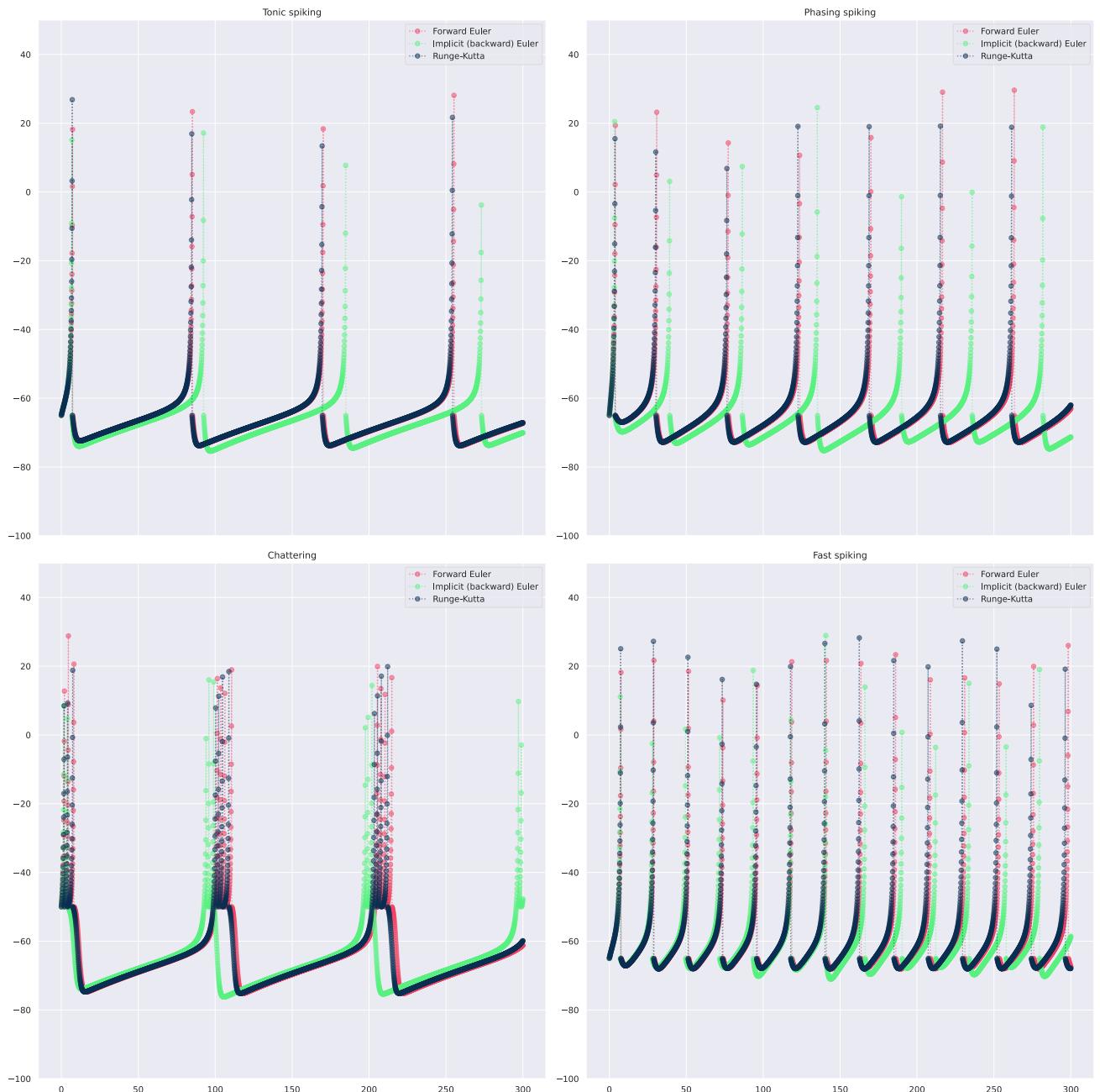


Рис 3.1. Характерные режимы нейронов, по оси ординат $v(t)$ ось абсцисс t . Приведены численные решения системы (1.4) методами Рунге-Кутты 4-го порядка, прямым и обратным методами Эйлера для шага $h=1e-1$.

4. Моделирование нейронной сети

В продвинутой части данной работы требуется произвести симулирование активности нейронной сети, состоящей из 1000 нейрокортикальных нейронов: 800 возбуждающих и 200 тормозящих (ингибиторных). Нейронная сеть может описываться полным графом, вершинами которого являются, собственно, нейроны, а ребрами связи между ними. Такой граф можно задать матрицей смежности W , элемент $W_{i,j}$ которой задает величину входного тока, получаемого нейроном i от нейрона j в случае возникновения пика последнего в следующий момент времени. Как и в случае единственного нейрона, при возникновении пика значения v и u сбрасываются. В задачу был добавлен элемент неопределенности, выраженный в задании параметров из таблицы 1.1, токов покоя и элементов матрицы W следующим образом:

$$(возбуждающие нейроны) \left\{ \begin{array}{l} a = 0.02 \\ b = 0.2 \\ c = -65 + 15\alpha^2 \\ d = 8 - 6\beta^2 \\ W_{ij} = 0.5\theta \\ I_0 = 5\xi \end{array} \right. , \text{ где } \theta, \alpha, \beta, \xi - \text{ случайные величины от 0 до 1};$$

$$(ингибиторные нейроны) \left\{ \begin{array}{l} a = 0.02 + 0.08\gamma \\ b = 0.25 - 0.05\delta \\ c = -65 \\ d = 2 \\ W_{ij} = -\tau \\ I_0 = 2\zeta \end{array} \right. , \text{ где } \gamma, \delta, \tau, \zeta - \text{ случайные величины от 0 до 1}.$$

Для решения задачи Коши использовался прямой метод Эйлера в силу своей быстроты (его нужно применить сразу к 1000 нейронам). На рис. 4.4 приведено изображение поверхности, полученной без воздействия слагаемых из матрицы смежности W , соответствующий отсутствию связей между клетками. На рис. 4.1 приведена поверхность, полученная с учетом слагаемых матрицы W , то есть при возникновении импульса в клетке j входной ток всех нейронов на следующем шаге моделирования увеличивался на соответствующие элементы W_{ij} . Можно наблюдать отличия в режимах работы ингибиторных и возбуждающих нейронов (поведение первых напоминает FS, в то время как поведение вторых скорее PS). Для эффективного решения данной задачи очень кстати оказываются векторизованные методы и операции над массивами (*np.ndarray*) пакета *numpy*. По своей сути они позволяют в параллельном режиме осуществлять как классические арифметические операции, так и, например, матричное умножение. Векторизация также позволяет эффективно находить такие статистики коллекции, как суммы её элементов вдоль произвольной оси, среднее, медиану, кумулятивные суммы и многое другое. В данном случае полезными оказались сумма элементов и арифметические операции. Так, удается применять метод Эйлера одновременно ко всем системам. Ещё одной полезной опцией является маскирование, которое позволяет фильтровать коллекцию по какому-либо признаку (поэлементно). Такую маску, являющуюся массивом той же формы, но состоящим из элементов *True/False*, затем можно применить к массиву той же формы в качестве индекса. Это очень удобно для фильтрации «загоревшихся» в определенный момент времени нейронов. Для визуализации использовался пакет *plotly.express*,

предоставляющий высокогорневый интерфейс для создания интерактивных графиков различных видов.

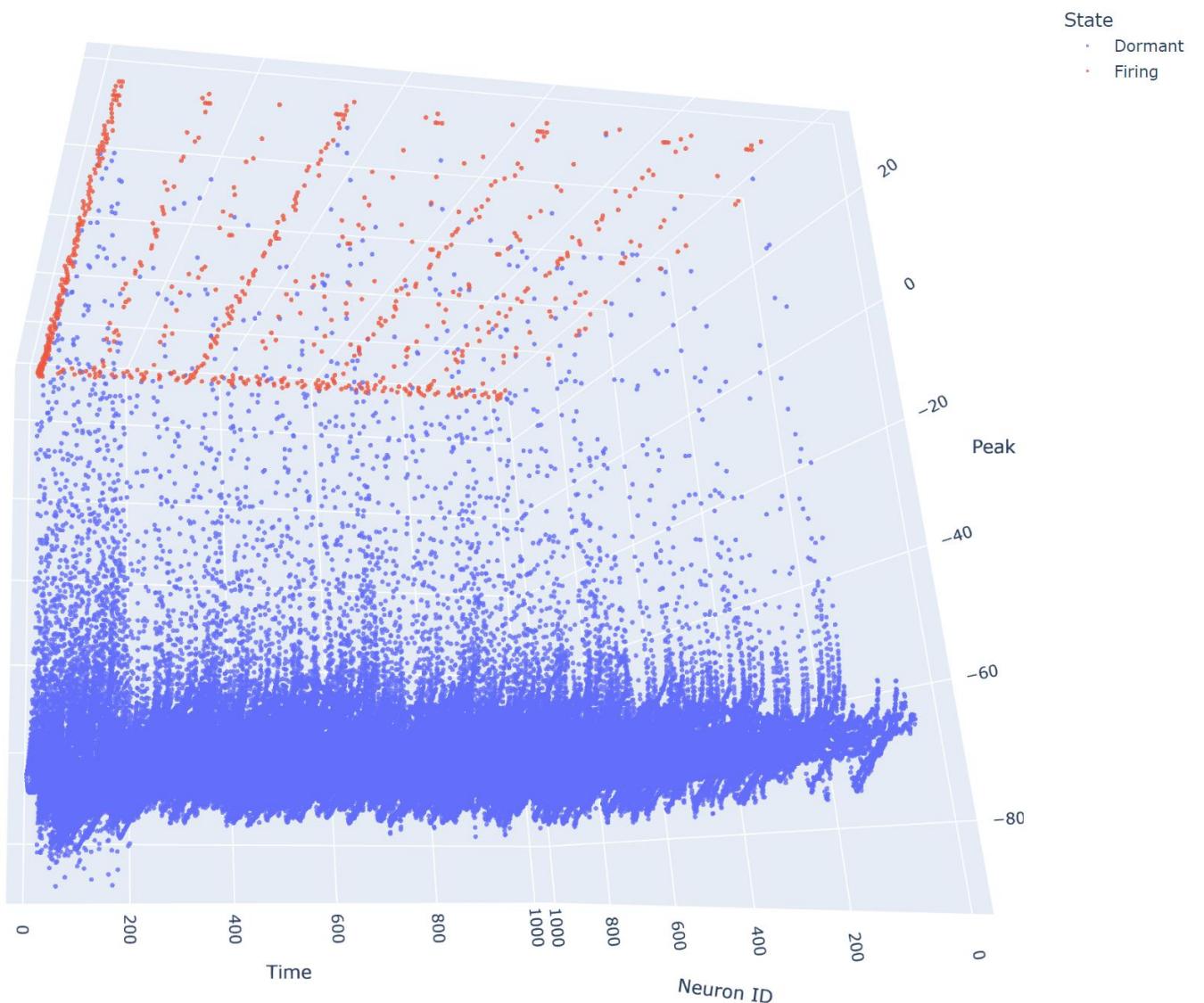


Рис. 4.1. Траектории динамической системы для 100 нейронов из числа 1000 с учетом обратных связей по току (обозначены синим). Клетки с ID до 800 являются возбуждающими, остальные – ингибиторными. Можно наблюдать выраженные синхронные импульсы ~10Гц всей сети.

Красным обозначены нейроны, которые «загорелись» в данный момент времени

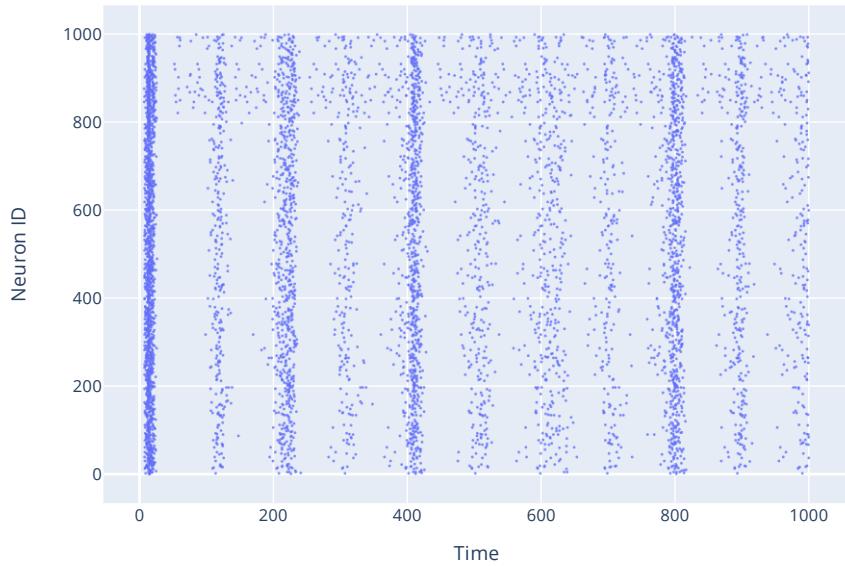


Рис. 4.2. Положения пиков всех элементов сети с учетом обратных связей по току. Точка обозначает факт возникновения пика в данном нейроне в данный момент времени. Можно наблюдать ярко выраженные синхронные колебания ~10-12Гц и различие в режимах работы клеток обоих типов. Этот график является расширением к рис. 4.1, на котором изображены лишь 10% пиков.

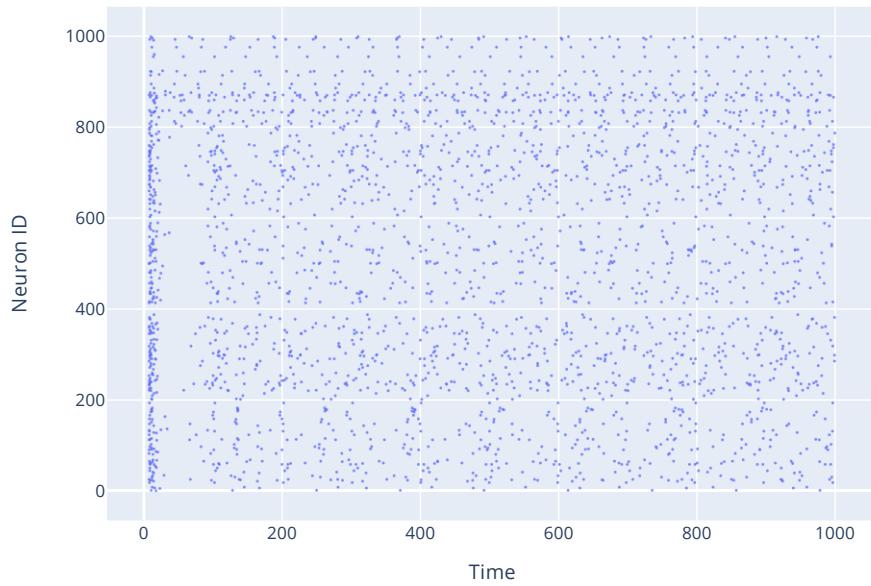


Рис. 4.3. Положения пиков всех элементов сети с учетом обратных связей по току. Синхронные колебания всей сети отсутствуют, шум вызван случайностью параметров отдельных нейронов. Различимы разные режимы активности нейронов обоих типов. Этот график является расширением к рис. 4.4, на котором изображены лишь 10% пиков.

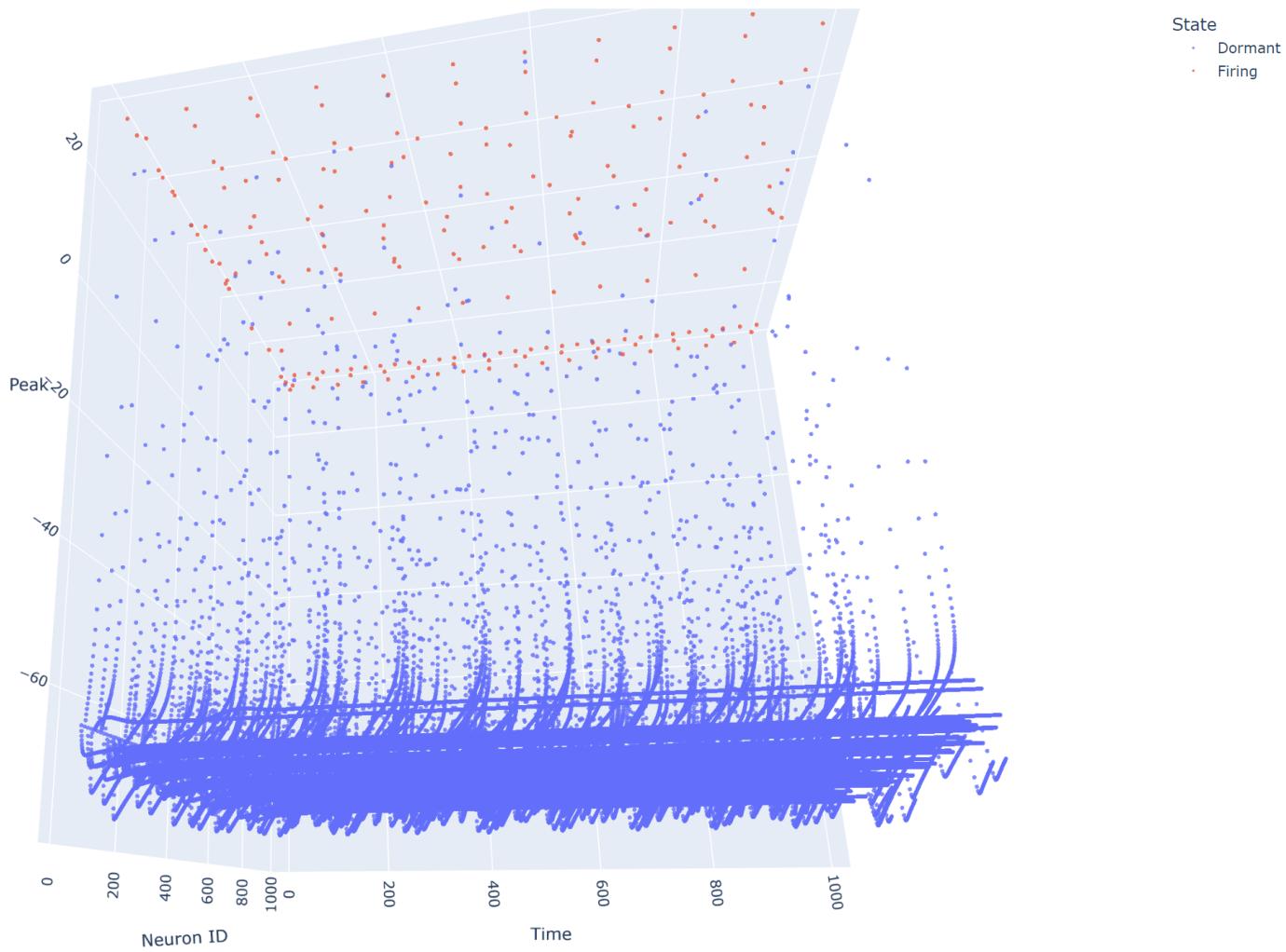


Рис. 4.4. Траектории динамической системы для 100 нейронов из числа 1000 без учета обратных связей по току. Клетки с ID до 800 являются возбуждающими, остальные – ингибиторными. Можно наблюдать, что синхронные колебания всей сети исчезли, совпадение частот может проявляться локально в силу схожести случайных параметров некоторых нейронов

Заключение

Методы решения задач Коши обыкновенных дифференциальных уравнений являются очень востребованными, поскольку очень многие процессы в мире могут описываться с помощью ДУ, при этом далеко не все уравнения имеют аналитическое решение. Как мы убедились, не только инженерные задачи могут требовать численного решения, но и, например, задачи нейробиологии. Как было упомянуто в разделе 2, самым оптимальным алгоритмом для большинства задач является метод Рунге-Кутты 4-го порядка в силу сбалансированности между числом вычислений функции и порядком точности. Метод Эйлера является самым интуитивно понятным, но не может считаться удовлетворительно точным в большинстве приложений, однако является самым быстрым в отношении вычислений. Обратный метод Эйлера обладает таким важным свойством, как абсолютная стабильность, т. е. его решение вне зависимости от шага будет сходиться. Проблема его применения состоит в побочных вычислениях и потенциальной потере точности при решении нелинейного уравнения на каждом шаге. Стоит упомянуть такой вариант реализации, как предиктор-корректор, но его применимость очень чувствительна к конкретной задаче, при том, что требует, как правило, столько же (или больше) вычислений функции на каждом шаге, сколько требуется для метода Рунге-Кутты при худшей точности. Стоит упомянуть о вопросе устойчивости численных схем для данной задачи: особенность модели заключена именно в её дополнительном условии, без него исходное ДУ было бы неустойчивым само по себе, его решение неограниченно растет. Поэтому судить об устойчивости численных схем в данной задаче несколько неуместно. Однако отметим, что высота пиков при увеличении шага заметно разнится и в общем случае падает, что, в целом, свидетельствует о больших накопленных ошибках метода.

Список использованных источников

1. **Соколов А. П., Першин А. Ю.** Инструкция по выполнению лабораторных работ (общая). // кафедра «Системы автоматизированного проектирования» МГТУ им. Н. Э. Баумана, Москва, 2021.
2. **Першин А. Ю.** Лекции по вычислительной математике. // Кафедра РК6 (Системы автоматизированного проектирования) МГТУ им. Н. Э. Баумана, 2020.
3. **Higham, Nicholas J.** Accuracy, and stability of numerical algorithms // University of Manchester, Manchester, England, 2002.
4. Официальная документация к библиотеке визуализации Plotly. Веб-ресурс. // URL: <https://plotly.com/graphing-libraries/> [Дата обращения: 22.10.2021]
5. **Eugene M. Izhikevich.** Simple model of spiking neurons. IEEE Transactions on Neural Networks (2003) 14:1569- 1572.
Веб-ресурс. // URL: <http://www.izhikevich.org/publications/spikes.pdf> [Дата обращения: 8.11.2021]