СОФТУЕР ЗА МОДЕЛИРАНЕ НА БИОЛОГИЧНИ НЕВРОННИ МРЕЖИ

ас. д-р Александър Иванов
Бургаски свободен университет

SOFTWARE FOR MODELING BIOLOGICAL NEURAL NETWORKS

Aleksandar Ivanov
Burgas Free University

Abstract: *Biological neuron modeling software tools are essential for simulating neural dynamics in computational neuroscience. This paper reviews their functionalities, applications, and comparative advantages, demonstrating how these tools advance the understanding of neuronal behavior and support the development of neurobiological hypotheses. Additionally, we discuss recent advancements and integration with other computational tools, enhancing the accuracy of neuronal simulations.*

Keywords*: biological neural networks, neural modeling, python.*

I. Въведение

Моделирането на биологични неврони е важно приложение на биоинформатиката. Тази дейност е важна за подобряването на разбирането как протичат биологичните процеси в нервната система, както от гледна точка на медицината, така и от гледна точка на когнитивната наука. Това проучване сравнява и анализира популярни софтуерни инструменти за симулиране на биологични невронни мрежи. Представени са основните характеристики на разгледаните софтуерни решения. Такъв анализ може да улесни изследователите да направят оптимален избор на инструменти въз основа на специфични изследователски нужди.

II. Модели на биологичните неврони

Невроните са основните работни единици на мозъка, съставени от клетъчно тяло (сома) с набор органели, дендрити и аксон. Невронната функция се основава основно на потока от йони през клетъчната мембрана, което води до промени в мембранния потенциал и генериране на потенциал за действие. Тези електрически импулси са в основата на невронната комуникация. В невроните се изследват различни електрически потенциали, дефиниращи работата им:

1. Мембранен потенциал: Разликата в електрическия потенциал между вътрешната и външната страна на клетката, основно задвижвана от концентрационните градиенти на йони като натрий (Na+), калий (K+) и хлор (Cl-).

2. Потенциал за действие (активационен): Бързи покачвания и спадове на потенциала на мембраната, които възникват, когато невронът изпраща сигнал по аксона.

Синаптичен пренос е процесът, чрез който невротрансмитерите се освобождават от един неврон и се свързват с рецепторите на друг, генерирайки невронната комуникация. Съществуват няколко модела за описание на електрическите процеси в неврона:

1. Моделът Leaky Integrate-and-Fire (LIF) е прост и често използван модел за симулиране на невронна активност. Той описва как мембранният потенциал се развива с течение на времето и генерира пикове (активации), когато се достигне праг.

2. Моделът на Ходжкин-Хъксли (Hodgkin-Huxley) е по-подробен и биологично точен модел на активността на невроните. Включва решаване на система от диференциални уравнения, които описват математически биофизичните свойства на потенциалите за действие в невроните. Той е разработен от Алън Ходжкин и Андрю Хъксли, които описват как потенциалите за действие в невроните се инициират и разпространяват, като се фокусират основно върху динамиката на калиеви и натриеви йони през невронната мембрана. Този модел формира основата за много невронни симулации.

3. Процесът на Поасон може да се използва за генериране на серия от активации, което е често срещан метод за симулиране на модели на задействане на неврони. Промените в мембранния потенциал могат да бъдат моделирани във времето [1].

Пробивните невронни мрежи (Spiking neural networks – SNNs) са вид изкуствени невронни мрежи, които имитират начина, по който биологичните неврони комуникират. За разлика от традиционните невронни мрежи, които използват непрекъснати стойности на активиране, пробивните невронни мрежи разчитат на дискретни събития (пробиви, пикове, активации) за предаване на информация. Този подход позволява на SNN да улавят по-реалистична мозъчна динамика и времеви модели, което ги прави изключително подходящи за невроморфни изчисления и алгоритми, вдъхновени от мозъка [2]. SNN използват различни невронни модели за представяне на биологични неврони. Тези модели определят как невронът интегрира входове с течение на времето и се активира, когато се достигне праг. SNN често включват механизми за пластичност като пластичност, зависима от времето на активации (Spike-Timing-Dependent Plasticity – STDP), където синаптичната сила се променя въз основа на относителното време на активации от свързани неврони. Този механизъм позволява на SNN да научават времеви модели и да адаптират своите връзки с течение на времето.

• В SNNs невроните комуникират чрез излъчване на електрически импулси (пробиви, активации) в определени моменти от времето. Обучението на SNN е по-сложно от традиционните невронни мрежи, тъй като алгоритъмът за обратно разпространение на грешката не работи добре с пикове и дискретни събития, базирани на времето [2]. Необходими са специализирани методи за обучение като STDP или приближения на обратното разпространение на грешката, но те все още са по-слабо развити от методите за традиционните невронни мрежи. Въпреки че някои библиотеки (като BindsNET, NEST и Brian2) поддържат SNN, инструментите за SNN не са толкова напреднали или широко разпространени, както за традиционните невронни мрежи.

II. Софтуер за моделиране на биологични невронни мрежи

Невронната симулация включва използването на изчислителни инструменти за моделиране и изследване на функциите на неврони, невронни вериги или цели невронни системи. Основната цел на тези симулации е да моделират сложните механизми, които управляват поведението на невроните, включително обработване на входове, интегриране на сигнали, генериране на изходи и формиране на функционални мрежи.

Компютърната неврология използва математически модели и теоретичен анализ, за да разкрие принципите, които управляват развитието, структурата, физиологията и когнитивните способности на нервната система. Инструменти като NEURON помагат за преодоляване на дистанцията между експерименталната невронаука и теоретичната невронаука, като позволяват симулиране на невронни процеси, които са трудни или невъзможни за физическо измерване. Симулациите могат да предскажат резултатите от експерименти, развитието на заболявания или ефектите на лекарствата върху невронната активност. Чрез моделиране на мрежи от неврони, изследователите могат да изучават възникващите свойства на невронните системи, като учене и памет, които възникват от сложните взаимодействия между много отделни неврони.

Следва списък с библиотеки за моделиране на биологични неврони:

NEURON – междуплатформен софтуер за невронна симулация, написан на C, C++ и FORTRAN, със стабилна версия (версия 8.2.0), пусната през 2022 г. и се предлага под New BSD лиценз в GitHub. Използва се за детайлно невронно моделиране на отделения; подходящ за единични неврони и мрежи. Разработен от Майкъл Хайнс, и други. [3]

NEST – междуплатформен изчислителен невронаучен инструмент, разработен от The NEST Initiative. Първоначално пуснат на 1 август 2004 г. Най-новата стабилна версия (3.7) е от 04.2024 г. Написан е на C++, Python и Cython. NEST се предлага под лиценз GPLv2+. NEST е симулатор за активации на модели на невронни мрежи, който се фокусира върху динамиката, размера и структурата на невронните системи, а не върху точната морфология на отделните неврони. Може да се използва или с интерпретирания език за програмиране Python (PyNEST), или като самостоятелно приложение (nest).[4]

Brian2 – междуплатформен софтуер за невронни мрежи, написан на Python. Удобен за потребителя, подходящ за създаване на прототипи и симулации в по-малък мащаб. Разработен е от Romain Brette, Dan Goodman и Marcel Stimberg. Последната стабилна версия, версия 2.7.1, е от 1.07.2024 г. Предлага се под лиценз CeCILL, като изходният код е достъпен в GitHub. [5]

Genesis (GEneral NEural SImulation System) – платформа на Python за симулация с общо предназначение, която е разработена да поддържа симулацията на невронни системи, вариращи от субклетъчни компоненти и биохимични реакции до сложни модели на единични неврони, симулации на големи мрежи и модели на системно ниво. [6]

PyNN – библиотека на Python. Предоставя абстрактен слой за множество симулатори, като NEST, NEURON и Brian. Преодставя независим от симулатора език за изграждане на модели на невронни мрежи. Поддържа SpiNNaker и невроморфни хардуерни системи BrainScaleS. [7][8]

The Virtual Brain (TVB) – „Виртуалният мозък“ опростява сложността на микро ниво, за да моделира организацията на мозъка на макро ниво. Чрез комбиниране на индивидуална мозъчна анатомия от образни данни с усъвършенствано математическо моделиране, TVB генерира точни EEG, MEG, BOLD и SEEG сигнали, намалявайки сложността чрез методи на статистическата физика. [9]

CARLsim – библиотека, използваща графични процесори за моделиране на пробивни невронни мрежи. [10]

Следва списък на библиотеки за моделиране на изкуствени невронни мрежи:

TensorFlow – библиотека за изкуствени невронни мрежи, разработена от Google и оптимизирана за използване на тензорни и графични процесори. [11]

PyTorch – библиотека, подобна на TensorFlow, поддържа и SNN през модула PyTorch-Spiking, който може да преобразува изкуствени невронни мрежи в SNN. [12]

Nengo – библиотека на Python, насочена специално към SNN. Позволява дефиниранее на специфични типове неврони, правила за обучение, методи за оптимизация, многократно използвани подмрежи. Може да се изграждат и управляват дълбоки невронни мрежи и да се внедряват модели на различен невронни симулатор или невроморфен хардуер. [13]

BindsNET – Python библиотека, изградена върху PyTorch, за моделиране на пробивни невронни мрежи. Подходяща за обучение с поощрения. Поддържа паралелизация върху GPU, което значително ускорява обучението на модели. [14]

Elephant – Библиотека на Python за анализ на електрофизиологични данни, съвместима с Neo и често използвана с NEST и Brian2. Elephant е с отворен код, използва се за анализиране на електрофизиологични данни, съсредоточена върху записи на активации и времеви редове, като например локални полеви потенциали (LFP). Като наследник на Neurotools, предоставя модулна рамка за междулабораторен анализ и е съвместима с инструменти като OpenElectrophy и spykeviewer. Част от разработката е финансирана от проекта на ЕС за човешкия мозък Horizon 2020. [15] Пакетът Viziphant е разработен от екипа на Elephant и я допълва с API от високо ниво за генериране на визуализации.

Помощни библиотеки и инструменти за визуализация

NeuroML – XML-базиран формат за стандартизирано описание на невронни модели и мрежи. Поддържа описание на синапси, канали, клетки.[16]

LFPy – Пакет на Python с отворен код за изчисляване на потенциали на локално поле (LFP) и извънклетъчни потенциали в нервната тъкан. Той разчита на симулатора NEURON и използва интерфейс на Python. [17]

OpenSim – OpenSimulator е мултиплатформен сървър за 3D приложения с отворен код за създаване на адаптивни виртуални светове за много потребители. Със своята функция Hypergrid потребителите могат да пътуват между различни инсталации на OpenSimulator, поддържайки концепция за разпределена метавселена. Предлага гъвкавост за разработчиците да изграждат и персонализират виртуални пространства, използвайки предпочитаните от тях технологии. Макар и не само за невронно моделиране, той се използва в неврологията за симулиране на двигателни системи. [18]

Vizier – Инструмент за визуализация на данни от невронни мрежи; интегрира се с популярни инструменти за невронна симулация. [19]

Neo – библиотека за обработка на биофизиологични данни в стандартизиран формат.

SciPy/NumPy – библиотеки на Python за научни изчисления.

В Таблица 1 са представени данни за цитирания от търсачката sciencedirect по ключови термини имената на библиотеките (първа колона). Във втората колона е обобщен броят статии за периода 2020-2023, а в третата – броят статии за 2024. (към октомври). От разгледаните данни значителен брой цитирания имат PyNN и TVB.

инструмент	2020-2023	2024
brian2	25	18
PyNN	98	20
CARLsim	0	1
TVB	139	34
bindsnet	13	8
NeuroML	16	3

Таблица 1. Данни за научни цитирания на софтуерни инструменти

III. Практическа реализация на код за симулация на биологични неврони

В тази секция са представени две сходни софтуерни симулации на неврони, активирани от външен стимул. Кодовете илюстрират възможностите на две популярни библиотеки – NEURON и абстрактния интерфейс PyNN.

NEURON е специализирана симулационна среда за моделиране на отделни и мрежи от неврони. Поддържа подробно морфологично и биофизично моделиране, което позволява на изследователите да конструират реалистични симулации на невронно поведение. NEURON е особено ефективен поради способността си да се справя със сложни дървовидни невронни структури и неговите ефективни числени методи за решаване на диференциалните уравнения, получени от динамиката на Ходжкин-Хъксли. NEURON позволява създаването на подробни невронни модели, които могат да включват различни свойства като различна плътност на каналите, сложна синаптична динамика и 3D анатомични структури. HOC (High-Order Calculator) е основният скриптов език, използван в симулационната среда NEURON, която е предназначена за моделиране и симулиране на неврони и невронни мрежи. HOC позволява на потребителите да създават и контролират симулации, да настройват сложни невронни модели и да управляват експерименти програмно. Потребителите могат да разширят възможностите на NEURON, използвайки и неговия собствен език за програмиране, NMODL, който позволява дефинирането на нови типове канали, синапси и други клетъчни свойства, които не са част от стандартната библиотека. Следва примерен код с обяснения.

Инсталация

NEURON може да се инсталира през пакетния мендижър pip:

```
pip install neuron
```

Настройка на модел

импортиране на NEURON и настройка:

```
from neuron import h, gui  # импорт на NEURON и графичен интерфейс

# създаване на клетъчно тяло (сома)
soma = h.Section(name='soma')
soma.L = 20  # дължина в микрони
soma.diam = 20  # диаметър в микрони
soma.insert('hh')  # Ходжкин-Хъксли канали
```

В тази установка сомата моделира реален неврон кто цилиндър с динамика на Ходжкин-Хъксли с калиеви (K+) и натриеви (Na+) йонни канали.

Синапс

```
# създаване на синапс
syn = h.ExpSyn(soma(0.5))  # поставяне по средата на сомата
syn.tau = 2  # времева константа на намалянето на синаптичния активационен потенциал (синаптичен разпад)
```

Стимулация

Използване на NetStim обект за генериране на изкуствени активации на синапса:

```
stim = h.NetStim()  # създаване на мрежов стимулатор
stim.number = 5  # брой активации
stim.start = 5  # начално време в милисекунди
stim.interval = 10  #  време между активациите в милисекунди

# Свързване на стимулатора със синапса
nc = h.NetCon(stim, syn)
nc.weight[0] = 0.04  # синаптично тегло
```

Тази настройка подава към синапса 5 активации, започвайки на 5-тата милисекунда с 10 милисекунди интервал между всяка

Записване

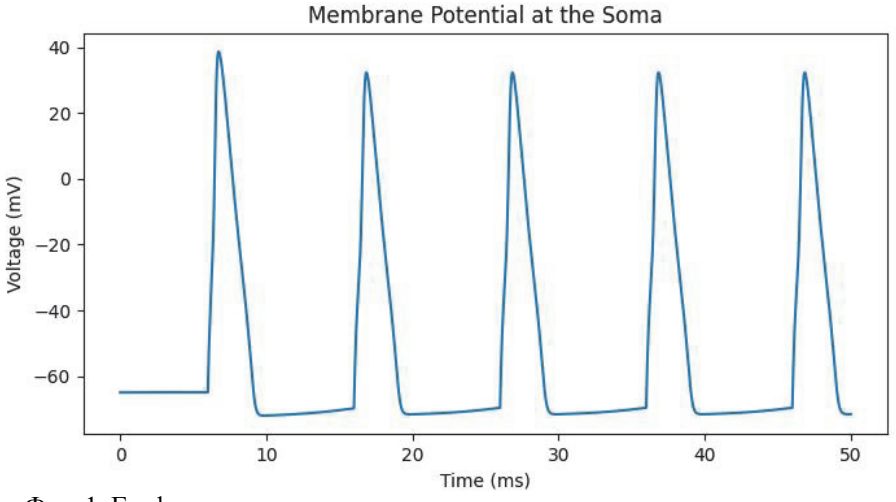
Запис на волтажа на сомата по време на симулацията:

```
v = h.Vector().record(soma(0.5)._ref_v)  # запис на волтажа в центъра на сомата
t = h.Vector().record(h._ref_t)  # време за запис

# Задаване на продължителността и началните условия на симулацията

h.tstop = 50  # милисекунди
h.run()
```

Визуализация на резултатите

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 4))
plt.plot(t, v)
plt.title('Membrane Potential at the Soma')
plt.xlabel('Time (ms)')
plt.ylabel('Voltage (mV)')
plt.show()
```

Представеният код предоставя основа за симулиране на неврон с помощта на NEURON и Python. Илюстрира възможностите на NEURON. Този модел може да се разшири с включване на по-сложни невронни вериги и разнообразна синаптична динамика.



Фиг. 1. Графика на промените във волтажа на сомата, генерирана от кода

На графиката по абсцисната ос е зададено времето в милисекунди, показвайки как мембранният потенциал в центъра на сомата (по ординатната ос, измерено в миливолтове) се променя в хода на симулацията.

PyNN е интерфейс от високо ниво за симулации на невронни мрежи. Той осигурява абстракционен слой върху различни бекендове (напр. NEURON, NEST) за създаване на невронни модели с подобен синтаксис в симулаторите. Това прави кода по-гъвкав, ако се налага превключване към различен бекенд, без да се налага променя в основната структура на кода. PyNN обикновено е по-лесен за внедряване и тестване на невронни мрежи по стандартизиран начин, но е ограничен по отношение на специфични функции и опции за фина настройка, които предлага самият NEURON.

Инсталация

```
# инсталиране и импортиране на PyNN и NEURON
pip install pyNN neuron
import neuron
import pyNN.neuron as sim
import matplotlib.pyplot as plt

# инициализиране на симулатора
sim.setup()
```

Настройка на модел

```
# създаване на неврон с динамика на Ходжкин-Хъксли
cell = sim.Population(1, sim.HH_cond_exp, {
    'cm': 1.0,  # капацитет на мембраната (mF)
    'e_rev_Na': 50.0,  # натриев реверсионен потенциал (mV)
    'e_rev_K': -77.0,  # калиев реверсионен потенциал  (mV)
    'e_rev_leak': -65.0  # потенциал в покой (mV)
'gbar_Na': 0.12,  # натриева проводимост (mV)
'gbar_K': 0.036, # калиева проводимост (mV)
'g_leak': 1 }) # теч (mV)
```

Синапс

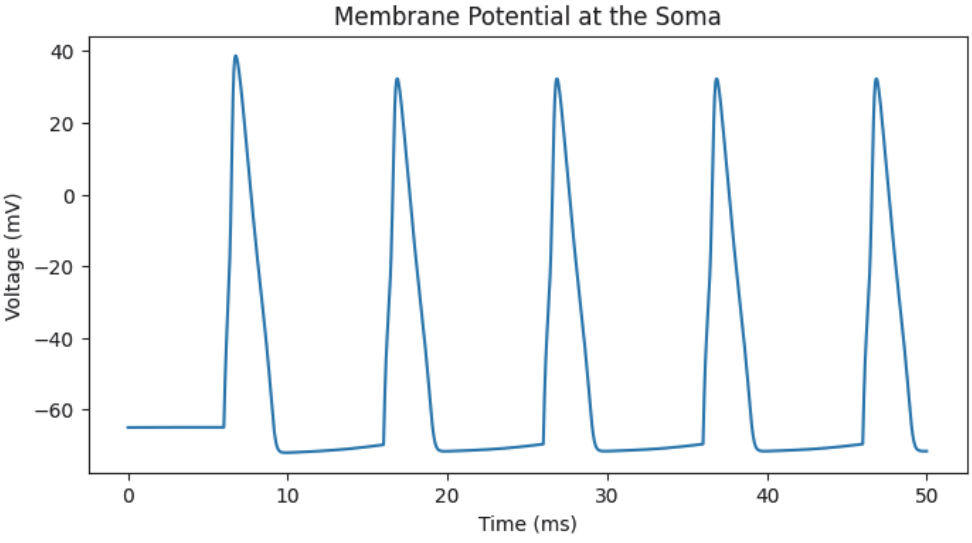
```
# създаване на синапс
synapse = sim.StaticSynapse(weight=0.04)  # синаптично тегло
```

Стимулация

```
# създаване на източник на активации за стимулиране на неврона
# в списък са зададени точните времена за активации
stimulus = sim.Population(1, sim.SpikeSourceArray, {'spike_times': [5, 15, 25, 35, 45]})

# свързване на неврона с източника на активации
synapse_connection = sim.Projection(
   stimulus, cell, connector=sim.OneToOneConnector(), receptor_type='excitatory',
synapse_type=synapse )

# запис на мембранния потенциал
cell.record('v')
# задаване на времето на симулацията и старт
sim.run(50)

# прочит на данните
data = cell.get_data().segments[0]
voltage = data.filter(name="v")[0]
time = voltage.times
```

Визуализация на резултатите

```
plt.figure(figsize=(8, 4))
plt.plot(time, voltage)
plt.title('Membrane Potential at the Soma')
plt.xlabel('Time (ms)')
plt.ylabel('Voltage (mV)')
plt.savefig(„pynn.jpeg“)
plt.show()
# край на симулацията
sim.end()
```



Фиг. 2. *Графика на промените в мембранния потенциал на симулирания неврон, генерирана от кода*

Този PyNN код ще работи на NEURON, ако е посочен като PyNN.neuron. Може да се превключи към други бекендове като NEST, като се промени pyNN.neuron на pyNN.nest, което прави PyNN отличен избор за съвместимост с няколко симулатора.

В представения код с конструктора Population се създава Ходжкин-Хъксли неврон и се задават следните характеристики:

'cm': 1,0: cm обозначава мембранния капацитет на неврона, измерен в микрофаради (μF). 1,0 μF/cm² е типична стойност за капацитет на мембраната и представлява способността на мембраната да съхранява заряд. Това влияе на времето на промените на напрежението през клетъчната мембрана, влияейки върху това колко бързо мембраната може да реагира на входящите сигнали.

'e_rev_Na': 50,0: e_rev_Na е обратният потенциал за натриеви (Na+) йони, в миливолта (mV). [20] Стойност от 50,0 mV е типична за натриевите канали и представлява равновесния потенциал, при който нетният поток от натриеви йони през мембраната би бил нула. Когато мембранният потенциал достигне тази стойност, натриевият ток не благоприятства нито притока, нито изтичането на Na+ йони.

'e_rev_K': -77,0: e_rev_K е обратният потенциал за калиеви (K+) йони, също в миливолта. -77,0 mV е обща стойност за калиев равновесен потенциал, представляваща точката, в която нетният поток от калиеви йони би бил нула. Този параметър контролира посоката на калиеви токове в зависимост от потенциала на мембраната.

'e_rev_leak': -65.0 e_rev_leak е потенциалът за обръщане на теча, който представлява потенциала на покой на клетката в отсъствието на какъвто и да е синаптичен или външен вход. -65,0 mV е типичен потенциал за покой за много неврони. Той показва базовата електрическа потенциална разлика през мембраната, дължаща се на каналите за изтичане, които са винаги отворени и позволяват на малки количества йони да текат пасивно.

'gbar_Na': 0,12: gbar_Na представлява максималната натриева проводимост на единица площ (в Siemens/cm²) на мембраната. Стойност от 0,12 S/cm² контролира колко натрий може да навлезе в клетката, когато натриевите канали са отворени, като по този начин влияе върху фазата на деполяризация на потенциала за действие. По-високата проводимост позволява навлизането на повече натрий, което води до по-бързо повишаване на мембранния потенциал.

'gbar_K': 0,036: gbar_K е максималната калиева проводимост на единица площ, също измерена в Siemens/cm². 0,036 S/cm² е типична стойност за калиева проводимост. Тази проводимост засяга фазата на реполяризация на потенциала на действие, тъй като калиевите йони напускат клетката, връщайки мембранния потенциал обратно към състоянието на покой след пик.

'g_leak': 1: g_leak е проводимостта на теч на единица площ, която позволява постоянен поток от йони през мембраната. Задаването на g_leak на 1 S/cm² осигурява малка, непрекъсната проводимост, симулираща пасивните свойства на мембраната. По-ниска проводимост на теч би означавала по-малко ток, протичащ през мембраната при отсъствие на активни канали, потенциално позволявайки по-продължителни възбуждащи сигнали.

Анализ

Първоначално невронът се намира на мембранния си потенциал в покой (напр. около -65 mV в примера с NEURON). Това е базовото напрежение, когато невронът не получава вход. За модел на неврон с динамика на Ходжкин-Хъксли, потенциалът на покой се определя от потенциала за обръщане на канала на изтичане (напр. e_rev_leak в PyNN). Когато външният стимул бъде доставен (или от NetStim в NEURON, или от SpikeSourceArray в PyNN), невронът може да се деполяризира, ако входът е достатъчно силен, за да достигне прага за задействане на потенциал за действие. Когато напрежението надхвърли прага, невронът генерира потенциал за действие (пробив, активация), причинявайки бързо повишаване на напрежението (деполяризация), последвано от рязко намаляване (реполяризация). Тази деполяризация и реполяризация образуват класическата форма на „шип“, където мембранният потенциал бързо се повишава над 0 mV и след това се връща до подпрагово ниво. След всяка активация невронът обикновено навлиза в рефракторен период, през който реагира по-слабо на входящите стимули. Това се дължи на динамиката на йонния канал, която временно инхибира по-нататъшната деполяризация. На графиката това може да изглежда като леко понижаване или период, през който невронът няма пикове, дори ако е стимулиран. Синаптичната времева константа (напр. tau в ExpSyn) определя колко време е необходимо за отслабване на ефекта от всеки стимул (синаптичен разпад). Ако синаптичната времева константа е кратка (като 2 ms в примера с NEURON), синаптичният ефект е кратък, което води до бързи пикове. По-дълга времева константа би причинила по-бавни, постепенни промени в мембранния потенциал, давайки по-плавна крива на реакция.

Аспект	PyNN код	Директен NEURON код
Ниво на абстракция	Високо, стандартизиран код	Ниско, фин контрол
Невронен модел	Предефинирано с HH_cond_exp	Ръчно с h.Section и hh
Синапс	Прост модел със StaticSynapse	Детайлен модел с ExpSyn
Стимул	SpikeSourceArray за времена на активации	Пълен контрол с NetStim
Запис	record('v') запис от клетка	Пространствен контрол
Гъвкавост	Може да използва множество бекенди	Ограничена
Персонализация	Ограничена от PyNN API	Екстензивна с NEURON API

Таблица 2. Сравнение на двата кода

IV. Заключение

В настоящата публикация са представени популярни софтуерни инструменти за моделиране на биологични неврони. Представени са работещи симулации, които илюстрират част от възможностите и ограниченията на разглеждания софтуер. Направено е кратко сравнение между разглежданите инструменти.

ЛИТЕРАТУРА

[1] M. -H. Wu et al., „Compact Probabilistic Poisson Neuron Based on Back-Hopping Oscillation in STT-MRAM for All-Spin Deep Spiking Neural Network“, 2020 IEEE Symposium on VLSI Technology, Honolulu, HI, USA, 2020, pp. 1-2,
[2] Stanojevic, A., Woźniak, S., Bellec, G. et al. High-performance deep spiking neural networks with 0.3 spikes per neuron. Nat Commun 15, 6793 (2024).
[3] https://nrn.readthedocs.io/en/latest/python/index.html
[4] https://www.nest-simulator.org/
[5] https://brian2.readthedocs.io/en/stable/
[6] http://www.genesis-sim.org/
[7] https://neuralensemble.org/PyNN/
[8] Davison A.P., Brüderle D., Eppler J.M., Kremkow J,, Muller E., et al. (2009) PyNN: a common interface for neuronal network simulators. Front. Neuroinform.
[9] https://www.thevirtualbrain.org/tvb/zwei
[10] https://sites.socsci.uci.edu/~jkrichma/CARLsim/
[11] https://www.tensorflow.org/
[12] https://www.nengo.ai/pytorch-spiking/
[13] https://www.nengo.ai/
[14] https://bindsnet-docs.readthedocs.io/
[15] https://neuralensemble.org/elephant/
[16] https://neuroml.org/
[17] https://lfpy.readthedocs.io/en/latest/
[18] http://opensimulator.org/wiki/Main_Page
[19] https://github.com/google/vizier
[20] Golowasch J,, Thomas G., Taylor AL, Patel A., Pineda A., Khalil C., Nadim F., Membrane capacitance measurements revisited: dependence of capacitance value on measurement method in nonisopotential neurons. J Neurophysiol. 2009 Oct; 102(4):2161-75.