Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ ДЛЯ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ СПАЙКІНГОВИХ НЕЙРОНІВ

Виконав: студент 4 курсу, групи 4КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Малюгов І. М.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф.КН
Паночишин Ю. М.
(прізвище та ініціали)
«4» червня 2025 р.

Рецензент: к.т.н., проф. каф. КСУ
Биков М. М.
(прізвище та ініціали)
«5» червня 2025 р.

Допущено до захисту
Зав. кафедри Яровий А.А.
«6» червня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо - професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«5» травня 2025 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Малюгову Івану Миколайовичу

1. Тема роботи: Програмний модуль для комп'ютерного моделювання спайкінгових нейронів.
керівник роботи: Паночишин Юрій Миколайович, к.т.н., доц.
затверджені наказом вищого навчального закладу "20" березня 2025 року №97

2. Термін подання студентом роботи 30 травня 2025р.

3. Вихідні дані до роботи :
тип входу – імпульсний або аналоговий, кількість входів – від 1 до 32, тривалість моделювання – до 10 000 кроків, відображення графіку мембранного потенціалу та вихідних імпульсів, відображення на мобільних пристроях.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
Вступ, аналіз предметної області спайкінгових нейронів, обґрунтування вибору моделі спайкінгового нейрона, аналіз математичної моделі SRM-нейрона, обґрунтування вибору мови та середовища програмування, розробка програми для комп'ютерного моделювання SRM -нейрона, аналіз результатів роботи комп'ютерної моделі SRM -нейрона, висновки, перелік використаних джерел, додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
алгоритм роботи комп'ютерної моделі SRM-нейрона,
результати моделювання SRM-нейрону,
порівняння вхідного сигналу з різними значеннями генерації імпульсів,
зовнішній вигляд додатку після відкриття.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Паночишин Ю. М., к.т.н., доц. каф. КН		

7. Дата видачі завдання 5 травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області спайкінгових нейронів	05.05.25	07.05.25	
2	Обґрунтування вибору моделі спайкінгового нейрона	08.05.25	11.05.25	
3	Аналіз математичної моделі SRM - нейрона	12.05.25	15.05.25	
4	Обгрунтування вибору середовища програмування	16.05.25	26.05.25	
5	Розробка програмного забезпечення для комп'ютерного моделювання SRM - нейрона	27.05.25	29.05.25	
6	Аналіз результатів роботи комп'ютерної моделі SRM -нейрона	30.05.25	01.06.25	
7	Оформлення матеріалів до захисту БКР	02.06.25	04.06.25	

Студент _____ **Малюгов І. М.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Паночишин Ю. М.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Малюгов І. М. Програмний модуль для комп'ютерного моделювання спайкінгових нейронів. Бакалаврська кваліфікаційна робота скаладається з 76 сторінок формату А4, на яких є 25 рисунків, 2 таблиці, список використаних джерел містить 21 найменувань.

Метою роботи є розширення функціональних можливостей програмного модуля комп'ютерного моделювання спайкінгових нейронів.

У цій роботі розроблено додаток для мобільних пристроїв, який дозволяє моделювати роботу спайкінгового нейрона. Модель SRM було обрано для комп’ютерного моделювання за критерієм максимум функцій – мінімум обчислювальних витрат. Програмний модуль для комп’ютерного моделювання спайкінгового SRM-нейрона створено мовою програмування JavaScript у середовищі розробки JetBrains WebStorm. Графічний інтерфейс реалізовано за допомогою HTML та CSS, а весь функціонал реалізовано мовою JavaScript. Для полегшення розробки та економії часу на написання типових функцій, використовувалась бібліотека JQuery. Було продемонстровано роботу модуля у різних режимах задання імпульсних та аналогових вхідних сигналів. Програма-аналог має 4 функціональні можливості, а розроблений програмний модуль – 7, тобто розроблений програмний модуль має на 75% більше функціональних можливостей, ніж програма-аналог.

Ключові слова: комп'ютерне моделювання, спайкінговий нейрон, функціональні можливості, SRM-модель нейрона.

ABSTRACT

Maliuhov I. M. Software module for computer modeling of spiking neurons. The bachelor thesis consists of 76 pages of A4 format, on which there are 25 figures, 2 tables, the list of used sources contains 21 names.

The aim of the work is to expand the functionality of the software module for computer simulation of spiking neurons.

In this work, an application for mobile devices has been developed that allows modeling the operation of a spiking neuron. The SRM model was selected for computer modeling according to the criterion of maximum functions - minimum computational costs. The software module for computer simulation of a spiking SRM neuron was created in the JavaScript programming language in the JetBrains WebStorm development environment. The graphical interface is implemented using HTML and CSS, and all the functionality is implemented in JavaScript. To facilitate development and save time on writing typical functions, the JQuery library was used. The module's operation was demonstrated in different modes of specifying pulse and analog input signals. The analog program has 4 functionalities, and the developed software module has 7, i.e. the developed software module has 75% more functionalities than the analog program.

Keywords: computer modeling, spiking neuron, functionality, SRM model of neuron.

ЗМІСТ

ВСТУП

Актуальність досліджень. У наш час інформаційних технологій штучний інтелект глибоко проник майже в усі сфери людського життя. З теоретичного боку кінцевою метою досліджень з штучного інтелекту є розкриття таємниць роботи мозку (мислення) та створення його моделі. На практиці штучний інтелект – це технічна (у більшості випадків практичної реалізації – комп'ютерна) система, яка має ознаки інтелекту, тобто здатна:

- розуміти та розпізнавати;
- знаходити спосіб досягнення мети та самостійно приймати рішення;
- навчатися.

Одним із напрямків досліджень штучного інтелекту є штучні нейронні мережі – це математичні моделі, а також їх програмна та апаратна реалізації, побудовані за принципом функціювання біологічних нейромереж, тобто мереж нервових клітин живих організмів.

Останнім часом дуже популярними стали спайкінгові нейронні мережі, які володіють низкою переваг порівняно із традиційними нейронними мережами. Наприклад, вони здатні швидко та ефективно обробляти багатоканальну інформацію в режимі реального часу [1]. Через це задача моделювання спайкінгових нейронів та спайкінгових нейромереж є надзвичайно актуальною, тому що може вплинути на покращення результатів роботи практичних впроваджень нейрокомп’ютерних систем на їх основі.

Успішний математичний опис потенціалів дії Ходжкіна і Хакслі призвів до цілої серії робіт, які намагаються детально описати динаміку різних іонних струмів. Тим не менш, точний опис нейронної активності включає в себе широкий ряд змінних, що часто запобігає ясному розумінню основ динаміки. Отже, був необхідний спрощений опис.

Об’єкт дослідження – процес комп’ютерного моделювання спайкінгових нейронів.

Предмет дослідження – програмні засоби комп’ютерного моделювання спайкінгових нейронів.

Мета роботи – розширення функціональних можливостей програмного модуля комп'ютерного моделювання спайкінгових нейронів (здатність працювати online з мобільних пристроїв).

Задачі дослідження:

Для досягнення поставленої мети потрібно вирішити наступні завдання:

1) провести аналіз предметної області спайкінгових нейронів;

2) проаналізувати математичну модель спайкінгового SRM-нейрона;

3) провести розробку алгоритму роботи програмного модуля моделювання спайкінгового нейрона;

4) здійснити програмну реалізацію моделі спайкінгового SRM-нейрона;

5) виконати тестування програми та проаналізувати отримані результати.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ МОДЕЛЮВАННЯ СПАЙКІНГОВИХ НЕЙРОНІВ

1.1 Постановка задачі

Ставиться задача розробки додатка для мобільних пристроїв, який можна було б використовувати для моделювання спайкінгових нейронів.

Моделювання полягає у тому, що потрібно згідно математичної моделі обробити вхідні сигнали нейрона та обчислити та відобразити у вигляді графіку вихідний сигнал нейрона, який змінюється у часі.

Кількість входів нейрона повинна бути від 1 до 32.

Нейрон може мати входи двох типів – імпульсний або аналоговий.

По імпульсному входу можуть надходити тільки імпульсні сигнали. Всі імпульси мають однакову амплітуду і тривалість, різниться тільки момент появи імпульсу у часі.

По аналоговому входу можуть надходити сигнали, які є неперервними і змінюються у часі по якомусь закону, який можна задати у вигляді функціональної залежності від часу.

Тривалість моделювання – до 10 000 кроків, Під кроком моделювання розуміється часовий інтервал, протягом якого може з’явитись лише один імпульс і протягом якого всі сигнали вважаються незмінними. Крок еквівалентний часовому інтервалу чисельного інтегрування.

Програма має відображати як графіки зміни вхідних сигналів, які задаються користувачем, так і два вихідних графіки:

1) графік мембранного потенціалу нейрона,
2) графік вихідних імпульсів нейрона.

Ці вихідні графіки визначаються кількістю, формою та амплітудами вхідних сигналів та параметрами моделі нейрона та є результатами комп’ютерного моделювання спайкінгового нейрона.

1.2 Огляд відомих методів моделювання спайкінгових нейронів

На сучасному етапі розвитку інформаційних технологій штучні нейронні мережі все частіше застосовують у різноманітних системах штучного інтелекту, зокрема, для розпізнавання динамічних образів різної природи, прогнозування багатопараметричних нестаціонарних процесів, для підтримки прийняття рішень у системах управління високого рівня за умов невизначеності та для розв'язання інших нетривіальних когнітивних задач, де традиційні комп'ютерні технології і системи, які працюють за чітко визначеними алгоритмами та програмами, виявляють свою неефективність.

Зацікавлення у вивченні принципів та секретів функціонування біологічних нейромереж і створенні їх різноманітних моделей викликане прагненням вчених поліпшити функціональність, ефективність, швидкодію, надійність і гнучкість технічних систем різного рівня. Результати дослідження штучних нейронних мереж знаходять широке застосування при створенні автоматизованих пристроїв для розпізнавання візуальних та аудіо образів, для діагностування стану технічних засобів та об'єктів, розробки інтелектуальних методів адаптивного управління, конструювання високопродуктивних, гнучких і надійних суперкомп’ютерів, навчання роботів складній поведінці у мінливому середовищі та у багатьох інших завданнях.

При класифікуванні штучних нейронних мереж з огляду на їхній процесорний блок [2], то можна виокремити три покоління.

Перше покоління має процесорний блок, що базується на нейронах МакКалока-Пітца. Їх також називають персептронами. Вони стали поштовхом до появи багатьох моделей нейромереж, зокрема, таких як багатошаровий персептрон, нейромережа Гопфілда та машина Больцмана. Характерною ознакою цього покоління нейромереж є те, що вони можуть видавати лише цифровий сигнал на виході. Фактично, ці нейромережі є універсальними для обчислень із цифровим (тобто 0 або 1) входом та виходом і довільну булеву

функцію можна обчислити на багатошаровому персептроні з одним прихованим шаром нейронів.

Базою другого покоління нейромереж є процесорні елементи, які використовують «активаційну функцію» з неперервним діапазоном можливих вихідних значень відносно зваженої (поліноміальної) суми вхідних сигналів. Найчастіше як функція активації використовуються сигмоїдальна функція $\sigma=1/(1+e^{-y})$ або лінійна функція з насиченням π, для якої $\pi(y)=y$ для $0\leq y\leq 1$, $\pi(y)=0$ при $y<0$ і $\pi(y)=1$ для $y>1$. Типовими прикладами представників другого покоління нейромереж є мережі зі зворотнім розповсюдженням помилки та рекурентні сигмоїдальні нейромережі. На теперішній час показано, що нейромережі другого покоління можуть реалізовувати булеві функції меншою кількістю нейронів, ніж нейромережі першого покоління. На додаток, нейромережі другого покоління можуть обчислювати функції з аналоговими значеннями входів та виходів. Фактично, мережі другого покоління є універсальними для виконання обробки аналогових (неперервних) функцій у тому сенсі, що довільна неперервна (аналогова) функція на певному відрізку області визначення може бути доволі точно апроксимована нейромережею цього типу лише з одним прихованим шаром. Іншою важливою рисою нейромереж другого покоління є те, що їх можна навчати з використанням методів, заснованих на градієнтному спуску, наприклад, методом зворотного поширення помилки.

З огляду на біологічну інтерпретацію нейромереж другого покоління, можна вважати вихідний сигнал сигмоїдального нейрона рівнем струму генерації імпульсу (fire) у біологічного нейрона. Оскільки встановлено, що у біологічних нейронів струм генерації імпульсу при різних середніх частотах лежить між мінімальним та максимальним, то нейромережі другого покоління вважають біологічно реалістичнішими, ніж нейромережі першого покоління.

Проте, класичні нейромережі (нейромережі першого та другого поколінь) містять і низку недоліків [3]:

1. Обчислення входу нейрона вважається моментальним, тобто таким, що не має затримки; а отже за такого припущення не є можливим моделювати динамічних систем, для яких суттєвим є затримки проходження сигналів та внутрішній стан.

2. Відсутність ефектів синхронізації, коли деякі множини нейронів обробляють стан синхронно, тобто відповідно до розповсюдження хвиль збудження і гальмування нейронів.

3. Традиційна нейромережа працює зі статичними сигналами (числовими векторами), а тому для дослідження обробки динамічних сигналів, їх необхідно додатково перетворити в статичний набір числових параметрів (у вектори чисел). Такий підхід потребує додаткових витрат часу на очікування завершення передачі сигналу (або окремої визначеної частини його) та на відповідні перетворення сигналу.

Третє покоління моделей нейромереж використовує як процесорні блоки спайкінгові (або англійською IF – integrate-and-fire) нейрони.

Такі нейромережі мають низку переваг [4]:

1. *Швидкодія*. Нейромережі спайкінгових нейронів можуть передавати та отримувати великі обсяги інформації за відносно невеликий час кількох імпульсів. Це є причиною високої продуктивності, швидкості та ефективності їх застосування.

2. *Можливість роботи в режимі реального часу*. Спайкінгові нейромережі розробляються для задач використання потокової інформації, а тому можуть реально бути інтегрованими в динамічне середовище.

3. *Продуктивність обробки.* Спайкінгові нейромережі здатні обчислювати будь-яку функцію, яку може обчислювати нейромережа другого покоління, і зачасту можуть робити це із залученням меншої кількості нейронів.

4. Біологічна правдоподібність. Оскільки принципи функціонування спайкінгових нейромереж більш пподібне до того, що відомо наразі про

біологічні нейромережі, то вони можуть більше вигравати від швидко зростаючої бази знань, що отримується із сфери психофізіології.

Спайкінгові нейрони є фізичними моделями біологічних нейронів (3-тє покоління моделей нейронів) і за своїми властивостями та характеристиками більш подібні до своїх прототипів (біологічних нейронів), ніж бінарні нейрони (формальні нейрони Мак-Каллока і Пітса – 1-е покоління моделей нейронів) та аналогові нейрони з потенційним виходом (2-ге покоління моделей нейронів) [5].

Щоб вірно вибрати модель спайкінгового нейрона, потрібно знати принципи функціонування біологічного нейрона. Найважливішими принципами функціонування біологічного нейрона є:

– частотний вихід – це коли вихідна інформація про ступінь збудження нейрона кодується в серії нервових імпульсів з пропорційною частотою;

– амплітуда і тривалість окремих нервових імпульсів, які проходять по одному і тому ж нервовому волокну, є постійними, а частота і кількість нервових імпульсів у послідовності залежать від інтенсивності збудження ннейрона (закон «все або нічого»). Такий спосіб кодування інформації є найбільш завадостійким, тобто у широких межах не залежить від стану нервових волокон, що проводять імпульси.

Біологічний нейрон має такі функціональні властивості:

1) просторове підсумовування;

2) часове підсумовування;

3) підпорогове підсумовування;

4) залежність «сила–тривалість» – це залежність між амплітудою і тривалістю струму, необхідного для досягнення порогового стану нейрона (рис. 1.1) .

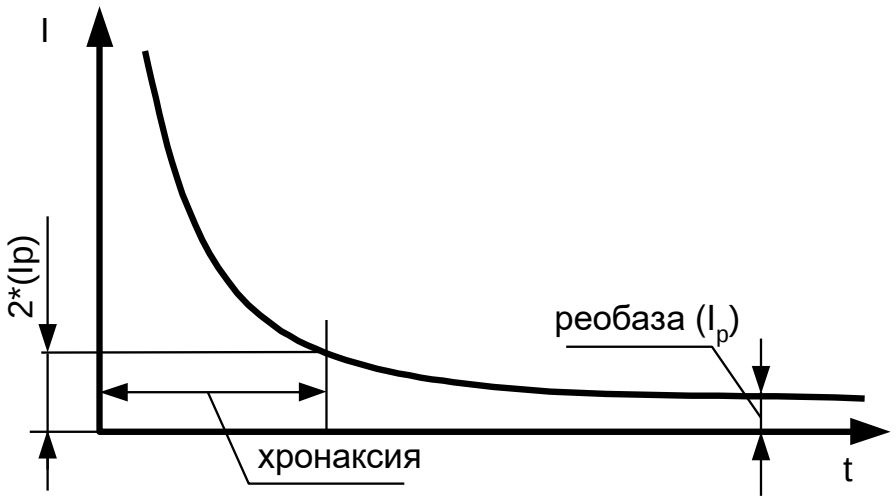


Рисунок 1.1 - Залежність «сила – тривалість»

5) залежність частоти ритмічного збудження від амплітуди сигналу збудження спочатку має лінійно зростаючу ділянку, а потім виходить на ділянку насичення;

6) трансформація ритму – нейрони мають властивість змінювати частоту імпульсів, які передаються, тобто властивість трансформації ритму;

7) залпова активність – це коли відповідь нейрону може виникнути у вигляді одиночного потенціалу дії, серії імпульсів з частотою, що знижується, а також у вигляді «пачок» імпульсів, які з'являються через деякі інтервали;

8) фонова активність – це коли нервові клітини мають постійну імпульсну активність. Їх потенціал мембрани періодично коливається, то збільшується, то зменшується;

9) акомодація – це пристосування нейрону до дії збудження, яке повільно наростає за силою;

10) рефрактерність – це відсутність чутливості нервової клітини до вхідних збуджень під час формування потенціалу дії. В момент виникнення високовольтної частини — спайка, нейрон не може відповідати на збудження новим потенціалом дії, тобто є абсолютно незбудливим (абсолютна рефрактерна фаза). Після цього збудливість нервової клітини поступово

відновлюється до початкового рівня (відносна рефрактерная фаза) і навіть протягом деякого часу може його перевищувати (екзальтаційна фаза).

1.3 Аналіз відомих програм-аналогів для моделювання спайкінгових нейронів

Існує декілька програмних засобів, що дозволяють моделювати спайкінгові SRM-нейрони (PCSIM, GENESIS, NEURONS),

Розглянемо, наприклад, програму Genesis Simulator. Ця програма є середовищем для моделювання нервових клітин, нейронних мереж та окремих нейронів. Результати моделювання відображуються на великому наборі графіків, які демонструють роботу моделі нейрона. GENESIS будує моделі нейронів або нейронних систем, створюючи імітаційні середовища. Головний екран програми зображено на рисунку 1.2.

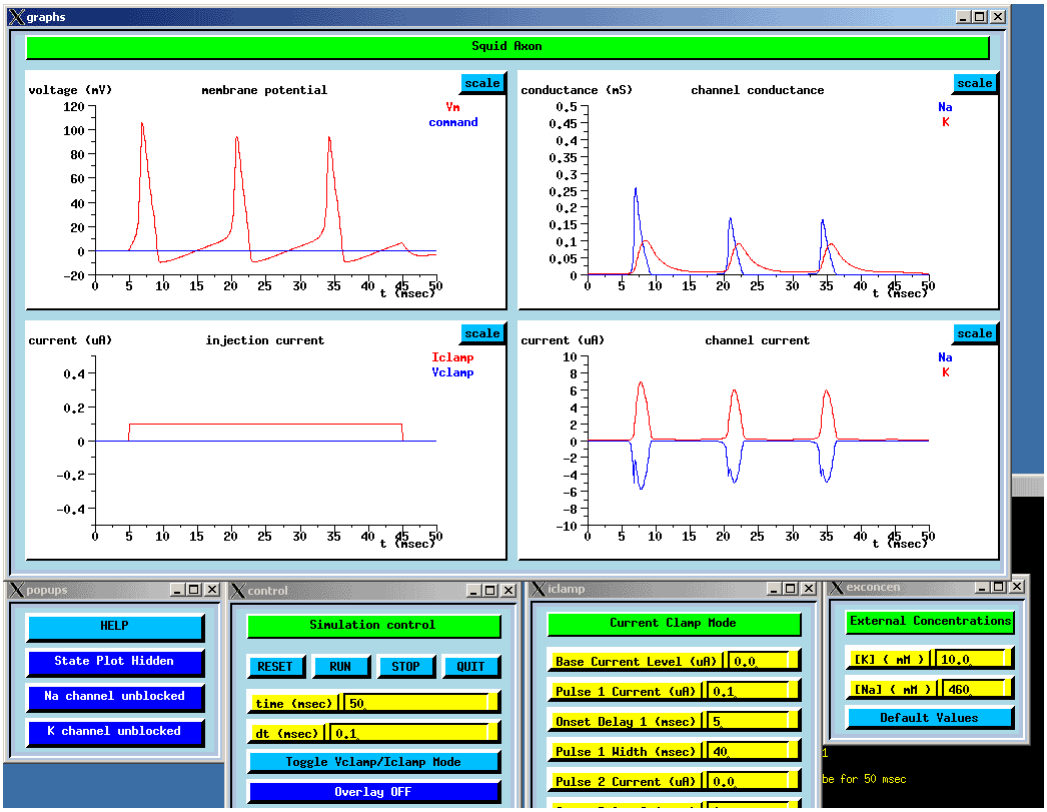


Рисунок 1.2 – Головний екран програми GENESIS

Більшість сучасних додатків для GENESIS реалізують реалістичне моделювання біологічних систем. GENESIS зазвичай використовується для комп'ютерного моделювання поведінки великих структур мозку, таких як, кора головного мозку. Ці дослідження найчастіше зустрічаються на курсах нейронного моделювання в університеті Caltech та Морській біологічній лабораторії у Вудс-Хілл, штат Массачусетс (США).

Часто GENESIS використовується разом з програмним забезпеченням NEURON Єльського університету як засіб для вчених співпрацювати для побудови моделей нервової системи. Програма GENESIS також може використовуватись з Kinetikit для моделювання шляхів передачі сигналу. GENESIS використовувався у багатьох сучасних дослідженнях, деякі з яких спрямовані на розробку програмного забезпечення, що є корисним для багатьох дисциплін нейроінформатики. В цій програмі навіть є моделі, які максимально наближені до моделювання роботи біологічного головного мозку тварин. Очевидно, що ця програма містить у собі занадто багато функціоналу [6]. Але не призначена для використання на мобільних пристроях.

Усі програми-аналоги для комп'ютерного моделювання спайкінгових нейронів PCSIM, GENESIS, NEURONS мають багато недоліків. Ось найбільш суттєві з них:

- громіздкість (більшість з відомих засобів потребують інсталяції чи встановлення пакету MatLab),
- недружній інтерфейс (графічні інтерфейси мають лише назви параметрів без пояснень, що вони значать; усі дані виводяться в одному вікні),
- не є можливим задавати власну формулу генерації імпульсів.

Усі ці недоліки роблять користування наведеними вище програмами моделювання зовсім незручним і саме тому розроблюваний в цій роботі програмний модуль має бути простим у встановленні та користуванні, щоб будь-хто з користувачів міг би легко користуватися ним.

1.4 Висновок до розділу 1

У розділі описано детальну постановку задачі комп'ютерного моделювання спайкінгових нейронів. Також розглянуто предметну область спайкінгових нейронів та спайкінгових нейронних мереж, їх місце серед моделей нейронів першого та другого покоління, переваги та недоліки спайкінгових моделей нейронів, їх більшу адекватність біологічним нейронвм і правдоподібність у моделюванні поведінки до біологічних нейронів в нейронних мереж. Було показано перспективність комп’ютерного моделювання спайкінгових нейронів. Крім цього, було проаналізовано такі програми-аналоги як PCSIM, GENESIS, NEURONS, визначено їх переваги та недоліки та поставлено задачі дослідження.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ДЛЯ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ СПАЙКІНГОВИХ НЕЙРОНІВ

2.1 Обґрунтування вибору моделі спайкінгового нейрона

Відомо багато різних видів математичних моделей спайкінгових нейронів [7]. Усі вони різняться своєю складністю та ступенем біологічної правдоподібності. Ступінь біологічної правдоподібності прийнято виражати через кількість функцій біологічного нейрона, які може виконувати дана модель нейрона. А складність прийнято виражати кількістю елементарних операцій процесора з плаваючою комою (FLOP – FLoat Operations Per second: додавання, множення та ін.), які є необхідними для чисельного комп’ютерного моделювання нейрона протягом визначеного інтервалу часу (скажімо, 1 мілісекунда).

Розглянемо деякі найпоширеніші математичні моделі спайкінгових нейронів та проведемо їх порівняння за складністю та ступенем біологічної правдоподібності з метою вибрати найкращу модель та розробити для неї програму комп’ютерного моделювання.

Деякі широко відомі математичні моделі спайкінгових нейронів виражаються у формі звичайних диференціальних рівнянь. Почнемо розгляд у першу чергу з простих моделей. Порівняльна характеристика моделей нейронів представлена у табл. 2.1 та на рис. 2.1.

Введемо такі позначення: ν позначає потенціал мембрани нейрона і ν' позначає похідну потенціалу мембрани нейрона за часом. Усі параметри в математичних моделях обрано таким чином, що ν має одиниці вимірювання мВ - мілівольт, а час вимірюється у мс - мілісекундах.

Таблиця 2.1 – Порівняльна характеристика функціонально-обчислювальних властивостей математичних моделей спайкінгових нейронів

№ п/п	Назва моделі	Біологічна правдоподібність (к-ть функцій)	Кількість FLOPs
1.	Інтегрально-імпульсна модель (SRM)	3	5
2.	Квадратична інтегрально-імпульсна модель (QI&F)	7	7
3.	Резонансно-імпульсна модель (R&F)	14	10
4.	Модель Хіндмарша-Розе	21	120
5.	Модель Ходжкіна-Хаксли	22	1200

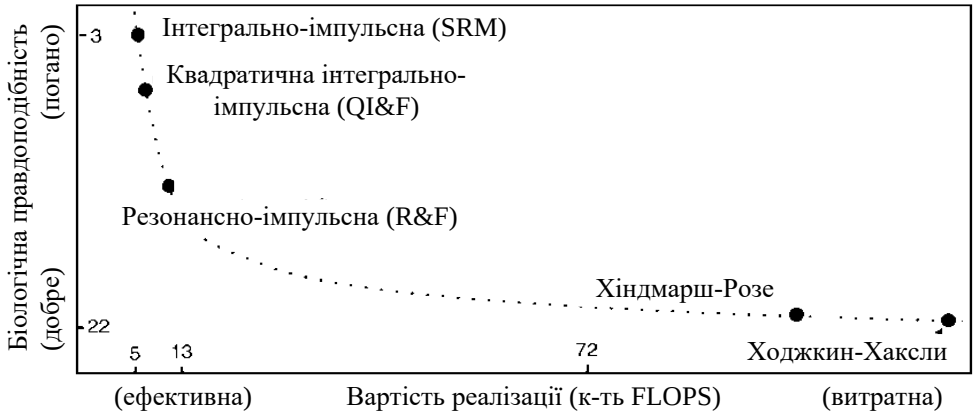


Рисунок 2.1 – Порівняльна діаграма для різних видів математичних моделей спайкінгових нейронів

Для порівняння обчислювальних витрат, будемо вважати, що кожна модель, записана як динамічна система $\overset{*}{x} = f(x)$, реалізується за допомогою однокрокового чисельного методу Ейлера першого порядку $x(t+\tau) = x(t) + \tau \cdot f(x(t))$ з кроком інтегрування τ, який обирається так, щоб досягти розумної чисельної точності.

1) Інтегрально-імпульсна модель нейрона (друга назва SRM - SpikeResponseModel)

Однією з найпоширеніших математичних моделей в нейро-обчислюваних науках є інтегрально-імпульсна модель нейрон (I&F) [4]

$$v' = I + a - b\,v, якщо\, v \geq v_{поріг}, тоді\, v \leftarrow c \tag{2.1}$$

де v - потенціал мембрани нейрона, I - вхідний струм, $a, b, c, v_{поріг}$ - інші параметри. Коли потенціал мембрани нейрона v досягає значення $v_{поріг}$, нейрон вистрілює імпульс, і v скидається до величини c.

I&F моделі нейронів відносяться до нейронів збудження 1-го типу. Вони можуть видавати послідовності імпульсів з постійною частотою, тобто працювати в режимі інтегратора. Це найпростіша матмодель для реалізації коли крок інтегрування τ становить 1 мс. Дійсно, ітерація $v(t+1) = v(t) + (I + a - b\,v(t))$ потребує всього чотири операції з плаваючою комою (додавання, множення та ін.), а також одну операцію порівняння з пороговим значенням $v_{поріг}$. Через те, що модель I&F має лише одну змінну, вона не може мати функцій фазової імпульсації, пакетної імпульсації будь-якого виду, змінного порогу, бістабільності, автономної хаотичної динаміки. Через фіксований поріг імпульси не будуть мати затримки. Таким чином, модель I&F є найпростішою з моделей, але і виконує менше функцій біологічного нейрона.

2) Квадратично-імпульсна модель нейрона (quadratic I&F)

Одним із різновидів I&F моделей нейронів є квадратичний I&F-нейрон, також відомий як тета-нейрон [7]:

$$v' = I + a(v - v_{спокою})(v - v_{порогу})$$

$$якщо\, v = v_{граничне}, тоді - v \leftarrow v_{спокою} \tag{2.2}$$

де $v_{спокою}, v_{порогу}$ - відповідно значення спокою і порогове значення потенціалу мембрани нейрона. Ця модель нейрона є канонічною в тому сенсі, що будь-який нейрон зі «збудженням 1-го типу», що описується гладкими диференційними рівняннями, може бути перетворений у цю форму через безперервну зміну змінних. Ця модель нейрона потребує лише сім FLOP операцій, щоби моделювати нейрон протягом 1 мс. Це - найкраща модель, коли потрібно моделювати великомасштабну нейромережу інтеграторів. На відміну від лінійних аналогів, квадратична I&F модель нейрона має затримку імпульсів, що залежить від порогу (який дорівнює $v_{порогу}$ тільки тоді, коли I=0), функцію бістабільності і режим послідовної імпульсації.

3) Резонансно-імпульсна модель нейрона (resonate-and-fire R&F)

Резонансно-імпульсна модель нейрона [8] - це двовимірна (2D) математична модель нейрона, яка аналогічна I&F моделі нейрону:

$$z' = I + (b + i\omega)z$$

$$\text{якщо} Imz = a_{\text{порогу}}, \text{то} \quad z \leftarrow z_0(z) \tag{2.3}$$

де дійсна частина комплексної змінної z є потенціалом мембрани. Також $b, \omega, a_{\text{порогу}}$ - інші параметри, і $z_0(z)$ - це довільна функція, яка описує скидання потенціалу після формування імпульсу, яке залежить від попередньої активності нейрона. Така модель нейрона є простою та ефективною, вона потребує 10 операцій для моделювання протягом 1 мс. Якщо частота коливань $\omega = 0$, то модель стає інтегратором. Її функціонально-обчислювальні властивості наведено у табл. 2.1.

4) Модель спайкінгового нейрона Хіндмарша-Розе

Модель нейрона Хіндмарша-Розе [9] – це модель нейрона таламуса і вона може бути описана формулами:

$$\begin{aligned} v' &= \upsilon - F(v) + I - \omega \\ \upsilon' &= G(v) - \upsilon \\ \omega' &= \frac{(H(v) - \omega)}{\tau} \end{aligned} \tag{2.4}$$

де *F, G, H* - це деякі функції. В залежності від їхнього вибору, такі моделі, в принципі, можуть мати всі функціонально-обчислювальні властивості (рис. 2.1). Проблема полягає в тому, як знайти функції для моделювання, наприклад, регулярно-спайкінгових (RS) і низько-порогових спайкінгових (LTS) нейронів. Якщо припустити, що ця проблема якось вирішена і що функції є поліномами третього ступеня (у кращому випадку), то нам потрібно для комп'ютерної імітації форми потенціалу дії аби максимальний крок за часом інтегрування складав 0,25 мілісекунд. У цьому випадку дана модель нейрона потребує 30 операцій з плаваючою комою для моделювання протягом 0.25 мс. Тоді буде потрібно 120 операцій для моделювання протягом 1 мс. Знову ж таки, це оптимістичні оцінки, які в принципі ніколи не можуть бути досягнуті.

5) Модель спайкінгового нейрона Ходжкіна-Хакслі

Модель нейрона Ходжкіна-Хакслі є однією з найважливіших моделей в обчислювальних нейронауках [10]. Вона містить чотири рівняння і десятки параметрів, які ми не розглядаємо в цій роботі. Вони описують потенціал мембрани, активацію Na і K-струмів, дезактивацію Na-струму. Хоча ця модель нейрона має досить обмежений спектр поведінки для дійсних значень параметрів, вона фактично володіє всіма властивостями біологічних нейронів, якщо ці параметри налаштовувати потрібним чином.

Загалом, в науці широко використовуються всі засновані на провідності моделі нейронів, подібні до моделі Ходжкіна-Хакслі. Такі моделі нейронів мають важливе значення не тільки тому, що їх параметри біофізично важливі і точні, але і тому, що вони дозволяють досліджувати питання, що пов'язані з синаптичною інтеграцією, ефектами дендритної морфології, дендритно-

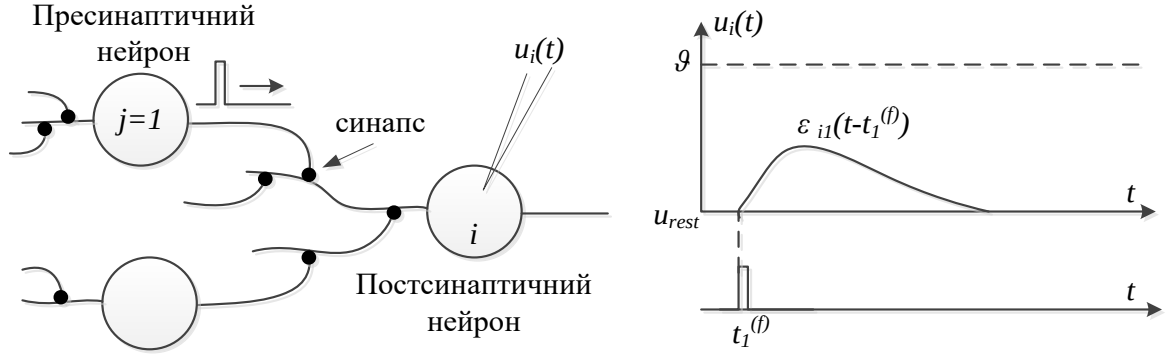
кабельною фільтрацією, взаємодією між іонними струмами, а також інші питання, що пов'язані з імпульсною динамікою окремоїнервової клітини.

Модель нейрона Ходжкіна-Хакслі є надзвичайно складною для реалізації. Вона потребує 120 операцій з плаваючою комою для 0,1 мс часу моделювання (у припущенні, що кожна експонента потребує тільки 10 операцій). Отже, треба 1200 операцій на 1 мс. Таким чином, можна використовувати модель нейрона Ходжкіна-Хакслі формально тільки для моделювання невеликої кількості нейронів або коли при моделюванні не є критичною велика тривалість часу моделювання.

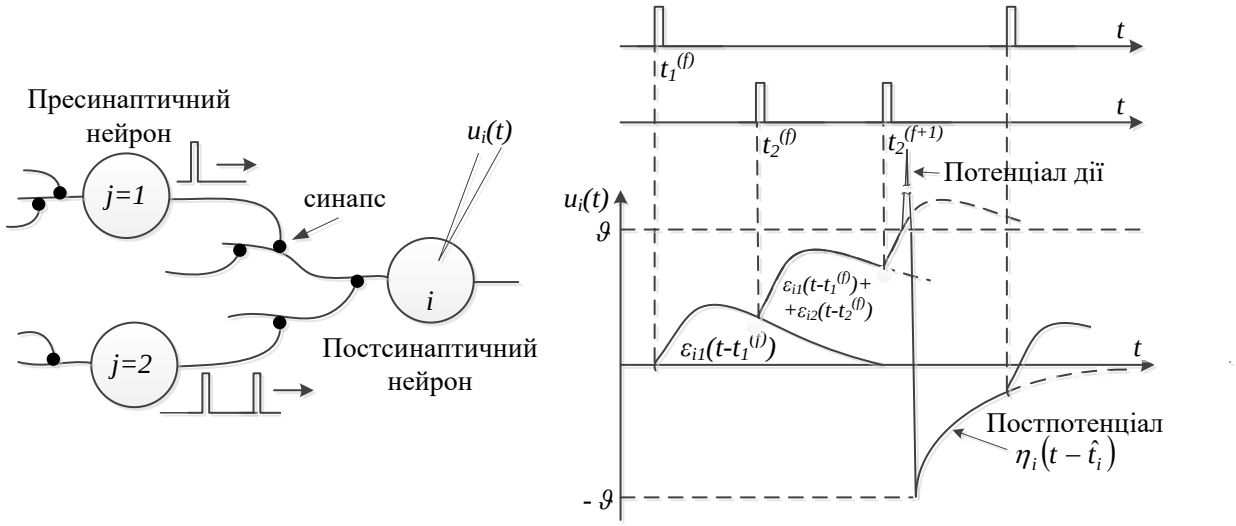
Таким чином, було зроблено аналіз відомих математичних моделей спайкінгових нейронів з точки зору виконання ними основних функцій біологічного нейрона. Зроблено висновое, що для комп’ютерної реалізації варто обирати найпростішу модель нейрона з огляду на економію ресурсів комп’ютера (пам'яті та завантаженості процесора). Тому варто обрати SRM модель нейрона, оскільки вона є найпростішою в моделюванні і будь-який мобільний пристрій зможе легко працювати з нею, не витрачаючи великого обсягу ресурсів.

2.2 Аналіз математичної моделі спайкінгового SRM-нейрона

У матмоделі SRM (SpikeResponseModel) стан нейрона i подається однією змінною u_i – мембранний потенціал нейрона. За відсутності вхідних імпульсів мембранний потенціал u_{i} нейрона має значення потенціалу спокою $u_{rest} = 0$. Кожен вхідний імпульс буде збуджувати мембранний потенціал u_i на деякий час, після чого мембранний потенціал u_i буде повертатися у стан спокою (рис.2.2а). Функція ε описує часову залежність відгуку потенціалу мембрани нейрона на вхідний імпульс, тобто ε_{ij} моделює постсинаптичний потенціал.



а)



б)

Рисунок 2.2 – Формування а) постсинаптичного потенціалу $\varepsilon_{ij}(t-t_j^{(f)})$ нейрона і б) потенціалу дії та пост-потенціалу $\eta_i(t-\hat{t}_i)$ нейрона

Якщо після додавання у часі відгуків від кількох вхідних імпульсів,мембранний потенціал u_i досягне порогу ϑ, то відбудеться генерація вихідного імпульсу нейрона (потенціалу дії). Потенціал дії (додатній імпульс) представляється δ-функцією Дірака. Форма пост-потенціалу (від'ємний імпульс) представляється функцією η, яка зображена на рис. 2.2б. Припускається, що нейрон i перед цим згенерував імпульс у момент часу $\hat{t}_i$. Після генерації вихідного імпульсу нейрона змінна u_i описується таким рівнянням:

$$u_i(t) = \eta(t - \hat{t}_i) + \sum_j w_{ij} \sum_f \varepsilon_{ij}(t - \hat{t}_i, t - t_j^{(f)}) + \int_0^\infty \kappa(t - \hat{t}_i, s) I^{ext}(t - s) ds, \quad (2.5)$$

$$\hat{t}_i = max\ \{t_i^{(f)} < t\}. \quad (2.6)$$

де w_{ij} – вага синаптичногозв'язку,

$t_j^{(f)}$– описує час генерації імпульсів пресинаптичним нейроном j,

I^{ext} – зовнішній вхідний струм,

$\kappa(t - \hat{t}_i, s)$ – лінійна відповідь мембранного потенціалу на вхідний струм, яка описує відхилення мембранного потенціалу від потенціалу стану спокою, яке спричиняється короткими імпульсами струму, $s = t - t_j^{(f)}$.

Останній доданок враховує зовнішній вхідний струм I^{ext}. Дві групові суми організовують підрахунок по всіх пресинаптичних нейронах j (просторове пілсумовування) і всіх моментах генерації спайків $t_j^{(f)} < t$ нейрона j (часове підсумовування).

При цьому всі змінні залежать від $t - \hat{t}_i$, тобто від часу останньої генерації вихідного імпульсу (спайку). Поріг ϑ може бути не фіксованим, а залежати також від $(t - \hat{t}_i)$:

$$\vartheta \rightarrow \vartheta(t - \hat{t}_i). \quad (2.7)$$

Абсолютна рефрактерність з періодом Δ^{abs} реалізується у моделі шляхом надання θ великого додатного значення, аби уникнути видачі імпульсу і дати потенціалу мембрани можливість повернутись до стану рівноваги протягом $t > \hat{t}_i + \Delta^{abs}$. Генерування імпульсу трапиться тоді, коли мембранний потенціал u_i досягне динамічного порогу $\theta(t - \hat{t}_i)$ знизу:

$$t = t_i^{(f)} <=> \; u_i(t) = \theta\,(t - \hat{t}_i\,)\;\text{і}\,\frac{du_i(t)}{dt} > 0\;. \qquad (2.8)$$

Фаза відносної рефрактерності імітується кількома способами. Після імпульсу мембранний потенціал, а отже і η, проходять через режим гіперполяризації (пост-потенціалу), де напруга мембрани нижча за потенціал спокою. Протягом цієї фази потрібна більша стимуляція, ніж зазвичай, для того, щоб перевести потенціал мембрани через поріг. Це рівноцінно короткочасному зростанню величини порогу. Також, одразу після видачі потенціалу дії відповідь на вхідні імпульси буде коротша, що спостерігається на прикладі виду функції κ - відповідь на перший вхідний імпульс у момент часу t' коротша і менш значуща, ніж на другий вхідний імпульс у момент часу t'' . Тому потрібно надходження більшой кількості імпульсів для того, щоб спричинити генерування вихідного імпульсу. Перший аргумент функцій ξ і κ дозволяє внести цей ефект.

Загальний постсинаптичний потенціал представлено у (2.9):

$$h(t|\hat{t}_i) = \sum_j w_{ij} \sum_{t_j^{(f)}} \varepsilon_{ij}(t-\hat{t}_i, t-t_j^{(f)}) + \int_0^\infty \kappa(t-\hat{t}_i, s) I^{ext}(t-s)ds\,. \qquad (2.9)$$

Тоді (2.5) може бутиа представлена у компактній формі:

$$u_i(t) = \eta(t - \hat{t}_i\,) + h(t|\hat{t}_i\,)\;. \qquad (2.10)$$

Спрощена версія моделі SRM ігнорує залежність κ і ξ від першого аргументу

$$\xi_0(s) = \xi_{ij}\,(\infty, s);\;\; \kappa_0(s) = \kappa_{ij}(\infty, s)\;, \qquad (2.11)$$

і (2.5) перетворюється у (2.12):

$$u_i(t) = \eta(t - \hat{t}_i) + \sum_j w_{ij} \sum_{t_j^{(f)}} \varepsilon_0(t - t_j^{(f)}) + \int_0^\infty \kappa_0(s) I^{ext}(t - s) ds. \qquad (2.12)$$

При цьому кожен вихідний імпульс апроксимується функцією Дірака δ:

$$\eta(t - \hat{t}) = \delta(t - \hat{t}) - \eta_0 \exp\left(-\frac{t - \hat{t}}{\tau_{recov}}\right), \qquad (2.13)$$

де $\eta_0 > 0$.

Пост-потенціал затухає до 0, якщо враховувати часову константу відновлення τ_{recov}.

Таким чином, було проаналізовано математичну модель спайкінгового SRM-нейрона та визначено її основні характеристики та залежності.

2.3 Розробка алгоритму роботи програмного модуля для комп'ютерного моделювання спайкінгового SRM-нейрона

В процесі функціонування комп'ютерної моделі спайкінгового SRM-нейрону, вона має формувати вихідний сигнал у вигляді графіків імпульсної активності на входах та виході нейрона. Згідно завдання, вхідним сигналом комп'ютерної моделі нейрона мають бути імпульсні та аналогові збудження або гальмування.

На екран завжди потрібно виводити графік, який відображує вихідні імпульси нейрону та мембранний потенціал. Кожен вхідний сигнал повинен мати свій графік, щоб їх зручно було перевіряти та порівнювати.

Спочатку програма запам'ятовує ідентифікатори елементів сторінки, з якими вона часто буде працювати, потім оголошує стандартні значення таких

змінних як: mainTime - тривалість симуляції, hMax - порогове значення мемебранного потенціалу та деякі інші (блок 1 на рис. 2.3).

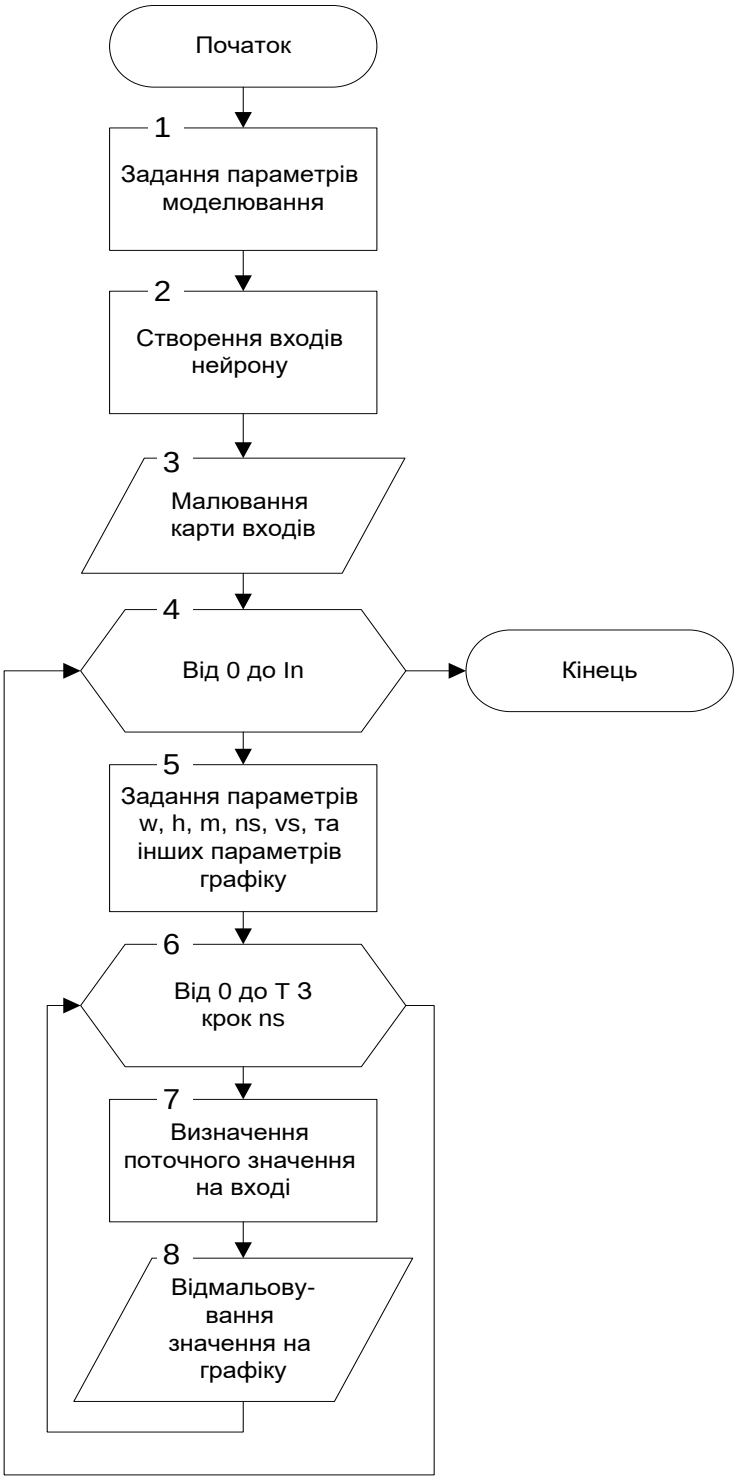


Рисунок 2.3 – Алгоритм роботи комп’ютерної моделі спайкінгового SRM-нейрона

Далі здійснюється оголошення стандартних входів нейрону (блок 2 на рис. 2.3). Для того, щоб створити новий вхід, потрібно створити новий об'єкт Input, а потім задати зв'язок цього входу з нейроном з допомогою функції createConnection.

Коли налаштування входів та самого нейрону задані, викликається функція redrawGraphs (блок 3 на рис. 2.3). Вона підраховує число графіків та відмальовує іх. Це здійснюється всереді циклу, що починається з блоку 4 на рис. 2.3). Тут, ln - кількість графіків, що наразі відображуються на сторінці. Для кожного графіку викликається функція drawGraph, у якій визначаються окремо параметри графіку: w - ширина, h - висота, m - середина графіку (тобто положення часової вісі по вертикалі), ns - крок, який визначається як відношення ширини екрану до тривалості моделювання (блок 5 на рис. 2.3).

Коли всі параметри задано, то починається цикл для підрахунку значень і іх відмальовування на графіках (блок 6 на рис. 2.3). Спочатку визначається, що саме має відмальовувати графік - вхід нейрону, чи сигнал на виході. В залежності від цього, використовуються різні алгоритми для обрахунку поточного значення. Якщо відмальовується графік вхідного сигналу, то просто обраховується його функція. Якщо ж відмальовується графік вихідного сигналу, то його поточне значення підраховується як сума всіх вхідних сигналів (блок 7 на рис. 2.3). Коли поточне значення визначено, то в залежності від того, який графік відмальовується, використовується різний алгоритм для його відображення (блок 8 на рис. 2.3). Якщо відмальовується графік вихідного сигналу, то спочатку визначається, чи перевищує мембранний потенціал порогове значення. Якщо перевищує, то відмалюється один вихідний імпульс, якщо ж не перевищує, то на виході нічого не генерується і не будується. Після відмальовування значення (імпульсу) на виході нейрону, також відмальовується значення мембранного потенціалу, щоб їх було зручніше порівнювати. При відмалюванні вхідного сигналу, визначається його тип, тобто аналоговий він чи імпульсний і, в залежності від цього, використовуються різні алгоритми для відмалювання значення. Якщо

сигнал аналоговий, то лінія відмальовується неперервною, а якщо сигнал імпульсний, то графіки відмальовуються у вигляді окремих імпульсів.

2.4 Висновок до розділу 2

У розділі було обґрунтовано вибір моделі спайкінгового нейрона. Із таких моделей як Інтегрально-імпульсна (SRM), Квадратична інтегрально-імпульсна (QI&F), Резонансно-імпульсна (R&F), модель Хіндмарша-Розе, модель Ходжкина-Хаксли було обрано для комп’ютерного моделювання за критерієм максимум функцій – мінімум обчислювальних витрат таку модель як Інтегрально-імпульсна (SRM), яка лежить в основі запропонованого модуля. Проаналізовано математичну модель спайкінгового SRM-нейрона. Розроблено алгоритм роботи програмного модуля для комп’ютерного моделювання спайкінгового SRM-нейрона.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ ДЛЯ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ СПАЙКІНГОВИХ НЕЙРОНІВ

3.1 Обґрунтування вибору мови програмування та спеціалізованих бібліотек

Однією із задач, поставлених у даній роботі, є забезпечення роботи програмного модуля online, тобто з Web-браузера. Тому розглядалися такі середовища програмування, за допомогою яких можна створити web-додаток, який можливо буде виконувати при відкритті сторінки в Internet. Найпопулярнішими мовами, які дозволяють це реалізувати є: JavaScript + HTML + CSS, ActionScript, Java.

ActionScript виконується на віртуальній машині (ActionScript Virtual Machine), яка є частиною FlashPlayer [11]. Враховуючи те, що програмний модуль повинен працювати також на будь-якому пристрої, який має підключення до мережі Internet, а FlashPlayer працює далеко не на усіх сучасних пристроях, то вибір ActionScript буде недоцільним.

Java виконується всередині браузерів у вигляд Java-аплетів [12]. Java-аплет – це прикладна програма, зачасту написана мовою програмування Java у форматі байт-коду. Java-аплети виконуються у браузері, використовуючи віртуальну Java машину (JVM), або у Sun's AppletViewer, який є автономним засобом випробування аплетів.

Переваги Java-аплетів:

- кросплатформенність,
- аплет працює на «усіх» раніше встановлених версіях Java, а не тільки на останній версії. Проте, якщо аплет потребує найсвіжішу версію JRE, то клієнт змушений буде чекати тривалішого завантаження,
- аплети підтримуються більшістю браузерів;

- аплет кешується у більшості браузерів, а тому швидко завантажується при поверненні на web-сторінку; але він може зберігатися у кеш-пам'яті та створювати проблеми після залучення нових версій;
- аплет може мати повний доступ до віртуальної машини, на якій виконується у випадку, коли користувач на це згоден;
- аплет може поліпшити використання після першого його запуску, коли JVM вже виконується і запускається швидко у постійних користувачів Java. Проте, JVM доведеться перезапустити всякий раз, коли запускають новий браузер.

Недоліки Java-аплетів:

- аплет вимагає встановлення Java-розширення (plug-in), що доступно не у всіх браузерах за замовчуванням;
- аплет не може запуститися доти, доки не запуститься віртуальна машина Java, а це може зайняти тривалий час при першому запуску;
- створення і дизайн якісного користувацького інтерфейсу з використанням аплетів вважається складнішим завданням, ніж за допомогою технології HTML;
- деякі організації дозволяють лише програмне забезпечення, що встановлене адміністраторами. Як наслідок, багато користувачів не можуть бачити за замовчуванням аплети.
- аплети можуть вимагати використання певного JRE.

Оскільки для нас важливим є користувацький інтерфейс та детальність відображення процесу моделювання у графічному виді, а в Java-аплетах це - складна задача, то мова Java також не пасує до розробки нашого модуля.

JavaScript (JS) — це динамічна, об'єктно-орієнтована мова програмування, реалізація стандарту ECMAScript [13]. Найчастіше JavaScript використовується як частина браузера, яка надає можливість коду на клієнтській стороні (такій, що виконується на пристрої кінцевого юзера), взаємодіяти з ним, керувати браузером, обмінюватися асинхронно даними з сервером, змінювати структуру та зовнішній вид web-сторінки. JavaScript

також використовується для програмування на серверній стороні (подібно до таких мов як C# і Java), для розробки ігор, стаціонарних і мобільних додатків, сценаріїв у прикладному програмному забезпеченні (напр., у програмах Adobe Creative Suite), всередині документів PDF тощо.

Не зважаючи на схожість назв, мови JavaScript та Java є двома різними мовами, які мають різну семантику, але й мають схожі риси: стандартні бібліотеки та правила іменування. Синтаксис обох мов отримано у спадок від мови C, але семантика та дизайн мови JavaScript є результатом впливу таких мов програмування як Self та Scheme.

Мова JavaScript має C-подібний синтаксис, але порівняно з мовою C має такі відмінності:

- об'єкти з можливістю інтроспективи та динамічної зміни типу через механізм прототипів,
- функції як об'єкти першого класу,
- обробка винятків,
- автоматичне приведення типів даних,
- автоматичне прибирання сміття,
- анонімні функції.

JavaScript містить кілька вбудованих об'єктів: Global, Error, Object, Function, String, Array, Boolean, Math, Number, Date, RegExp. Крім того, JavaScript містить низку вбудованих операцій, які, не обов'язково є функціями чи методами, а також низку вбудованих операторів, які керують логікою виконання програм. Синтаксис мови JavaScript, в основному, подібний до синтаксису мови Java (тобто, також успадкований від Сі), але є спрощеним порівняно з ним для того, щоб зробити мову сценаріїв легшою для вивчення. Так, наприклад, декларування змінної не містить її тип. Властивості не мають типів також, а декларування функції може стояти у тексті програми і після неї.

Мова JavaScript підтримується усіма сучасними браузерами на усіх стаціонарних комп'ютерах та мобільних пристроях, має достатню для комфортного користування швидкість. При розробці веб-додатку на мові

JavaScript інтерфейс користувача створюється мовою HTML, яка надає великі можливості з налаштування інтерфейсу та є легка у використанні. Враховуючи всі переваги мови JavaScript, вона обрана оптимальною мовою для розробки програмного модуля для комп'ютерного моделювання спайкінгових нейронів.

HTML (англ. HyperText Markup Language — мова розмітки гіпертекстових документів) — це стандартна мова розмітки web-сторінок в мережі Інтернет. Більшість web-сторінок створюються з використанням мови HTML (чи XHTML) [14]. Документ HTML обробляється браузером та відтворюється на екрані у звичному виді для людини.

Мова HTML є похідною від SGML, успадкувала від неї визначення типу документів та ідеологію структурної розмітки текстів.

Не зважаючи на те, що HTML — це штучна комп'ютерна мова, вона не є мовою програмування.

HTML поряд із каскадними таблицями стилів та вбудованими скриптами формує три основні технології побудови web-сторінок.

HTML має засоби для:

- створення структурованих документів позначенням структурного складу тексту: заголовки, абзаци, таблиці, списки, цитати та інше;
- отримання інформації із мережі Internet через гіперпосилання;
- створення інтерактивних форм;
- залучення зображень, відео, звуку та інших об'єктів до тексту.

Каскадні таблиці стилів (англ. Cascading Style Sheets чи скорочено CSS) [15] — це спеціальна мова, яка використовується для опису сторінок, які написано мовами розмітки даних.

Найчастіше CSS застосовують для візуальної презентації сторінок, які написано на HTML і XHTML, але формат CSS може використовуватися і до інших типів XML-документів.

Специфікації CSS були створені Консорціумом Всесвітньої мережі та розвиваються ним.

CSS має різні профілі та рівні. Наступні рівні CSS створюються на основі попередніх, додають нову функціональність або розширюють уже наявні функції. Рівні позначають як CSS1, CSS2 і CSS3. Профілі — це сукупність правил CSS одного або більшої кількості рівнів, створені для окремих видів інтерфейсів або пристроїв. Наприклад, є профілі CSS для мобільних пристроїв, принтерів тощо.

CSS (блочна або каскадна верстка) прийшла на зміну до табличної верстки web-сторінок. Основна перевага блочної верстки — це розділення змісту даних (сторінки) та їх візуальної презентації.

CSS використовується розробниками та відвідувачами web-сторінок для того, щоб визначити шрифти, кольори, верстку та інші параметри вигляду сторінок. Одна з основних переваг — це можливість відокремити контент сторінки (або зміст, наповнення, зазвичай XML, HTML або подібна мова розмітки) від вигляду документу (який описується в CSS).

Таке відокремлення покращує сприйняття та доступність наповнення, забезпечує більшу гнучкість та контроль за відображенням наповнення за різних умов, робить наповнення структурованішим та простішим, прибирає повтори тощо. CSS дозволяє також адаптувати наповнення до різних умов відображення (на екрані монітору, мобільного гаджету (КПК), у роздрукованому виді, на екрані ТВ, пристроях, що підтримують шрифт Брайля або на голосових браузерах та ін.).

Залежно від використаного CSS один і той самий HTML або XML документ може відображатися по-різному. Можуть бути такі стилі відображення сторінки:

- стилі автора (інформація надається автором сторінки):
  - таблиці стилів зовнішні (англ. stylesheet), часто окремий файл або файли .css,
  - таблиці стилів внутрішні, включені як частина документу або блоку,
  - стилі окремого елементу;

- стилі користувача:
  - локальний .css-файл, який вказано користувачем для використання на сторінках і вказано у налаштуваннях браузера (напр., Opera);
- стилі браузера (переглядача):
  - стандартний стиль переглядача, наприклад, стандартні стилі для елементів, визначених браузером, застосовуються коли відсутня інформація про стиль елементу або вона є неповною.

Стандарт CSS визначає діапазон та порядок застосування стилів, тобто те, у якій послідовності і для яких саме елементів використовуються стилі. Таким чином, застосовується принцип каскадності: для елементів зазначається тільки та інформація про стилі, яка змінилася або не була визначена загальними стилями.

Переваги CSS:

- інформація про стиль для всього сайту або для його частин може розташовуватися в одному .css-файлі, що дозволяє швидко вносити зміни у дизайн та презентацію сторінок;
- різна інформація про стилі для різних користувачів: наприклад, великий розмір шрифту для користувачів з поганим зором, стилі для виведення сторінки на принтер, стилі для мобільних пристроїв;
- сторінки зменшуються в об'сязі та стають структурованішими, так як інформація про стилі відокремлена від тексту та має визначені правила застосування, а також сторінка побудована з їх урахуванням;
- прискорене завантаження сторінок і зменшені обсяги інформації, яка передається, навантаження на сервер і канал передавання. Досягається тим, що сучасні браузери здатні запам'ятовувати (кешувати) інформацію про стилі і використовувати її для усіх сторінок, а не завантажувати для кожної сторінки окремо.

Для розробки мовою JavaScript було обрано середовище програмування JetBrainsWebStorm. WebStorm [16] – це середовище для розробки на мові

JavaScript, яке підійде і для front-end-розробки, і для створення додатків на Node.js. Основна перевага WebStorm - зручний та розумний редактор JavaScript, HTML та CSS, що підтримує також нові технології та мови програмування, такі як CoffeeScript, Dart, TypeScript, Sass, LESS та Stylus.

WebStorm забезпечує автозаповнення, аналіз коду «з льоту», навігацію по коду, зневадження, рефакторинг та інтеграцію із системами управління версіями. Особливою перевагою інтегрованого середовища WebStorm є робота з проектами (включно з рефакторингом коду JavaScript, який міститься у різних файлах та теках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (це коли в документ HTML вкладено скрипт на мові Javascript, в який також вкладено інший код HTML, всередині якого вкладений ще Javascript) — у таких конструкціях підтримується коректний рефакторинг.

Головні можливості:

- модифікація файлів .css, .js, html з одночасним переглядданням результатів (Live Edit, у деяких джерелах ця функція називається «редагування файлів на льоту» або «в реальному масштабі часу» або «без перезавантаження сторінки»),
- підтримка HTML5,
- підтримка JSDoc,
- підтримка Node.js,
- можливості Zen Coding і Emmet,
- зневадження коду JavaScript,
- віддалене розгортання згідно протоколів FTP, SFTP на мережевих дисках, тощо з можливістю автоматичної синхронізації,
- інтеграція з системами управління версіями Subversion, GitHub, Git, Perforce, CVS, Mercurial підтримуються «з коробки» з можливістю побудови переліку змін і відкладених змін,
- інтеграція з системами відстеження помилок.

Все перераховане робить WebStorm кращим середовищем розробки для програмного модуля комп'ютерного моделювання спайкінгових нейронів.

3.2 Програмна реалізація комп’ютерної моделі спайкінгового SRM-нейрона

Так як програмна реалізація модуля комп'ютерного моделювання спайкінгових нейронів виконана за допомогою JavaScript та HTML, то логіка роботи та опис інтерфейсу модуля чітко відокремлені один від одного.

При зверненні до серверу модуля, на клієнтський бік передається файл index.html, у якому описано структуру програмного модуля та його графічний інтерфейс (рис. 3.1). Файлову структуру програмного модуля відображено на рисунку 3.2. Коли файл index.html повністю завантажено клієнту, то спочатку відбувається формування графічного інтерфейсу за допомогою HTML та CSS, а потім під’єднується JavaScript.

Для того, щоб додаток вірно відображався на мобільних пристроях, були додані спеціальні мета-теги [17], які вказують пристроям, яким саме чином відображати додаток:

```
<meta name="mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-capable" content="yes" />
<meta content="minimal-ui" name="viewport"/>
```

Для зручності розробки весь функціонал мовою JavaScript було розділено на декілька підключених файлів:

- main.js - опис основних змінних та функцій програмного модуля,
- input.js - опис об'єкту «вхідний імпульс»,
- neuron.js - опис об'єкту «нейрон»,
- map-init.js - формування карти входів.

```
▼ <div class="main-div">
  ▼ <div class="map-display">
      <div class="title">Карта зв'язків</div>
      <div class="hide-button"></div>
    ▼ <div class="neurons-map">
      ▼ <div class="neuron-inputs">
          <div class="neuron input" data-nid="1" style="width: 40px; height: 40px;">in</div>
          <div class="neuron input" data-nid="2" style="width: 40px; height: 40px;">in</div>
        </div>
        <canvas id="neuron-map-helper" height="300" width="860">
        <div class="neuron" data-nid="3" style="top: 90px; left: 660px; width: 60px; height: 60px;">N1</div>
      </div>
    ▼ <div class="control-panel">
      ▶ <div class="option">…</div>
      ▶ <div class="option">…</div>
      ▶ <div class="option">…</div>
      ▶ <div class="map-actions option">…</div>
      </div>
    </div>
  ▼ <div class="display-panel" style="text-align: left">
    ▼ <div class="graphs-list">
      ▼ <div class="graph-wrapper input-config" data-nid="1">
          <div class="title">Вхід</div>
          <div class="hide-button"></div>
          <canvas class="output" id="result-nid-1" data-nid="1" height="200" width="990">
        ▶ <div class="formula-config">…</div>
        </div>
      ▼ <div class="graph-wrapper input-config" data-nid="2">
          <div class="title">Вхід</div>
          <div class="hide-button"></div>
          <canvas class="output" id="result-nid-2" data-nid="2" height="200" width="990">
        ▶ <div class="formula-config">…</div>
        </div>
      </div>
    ▼ <div class="last-graph graph-wrapper">
        <div class="title">Потенціал мембрани та вихідні імпульси</div>
        <canvas id="end-result" height="200" width="990" class="end-res">
      </div>
    </div>
    <div class="clear"></div>
  </div>
  <div class="popups-wrapper"></div>
  <script src="main.js"></script>
  <script src="input.js"></script>
  <script src="neuron.js"></script>
  <script src="map-init.js"></script>
```

Рисунок 3.1 – Структура HTML

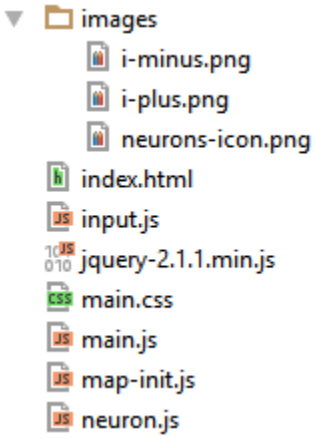


Рисунок 3.2 – Структура додатку

Також, для полегшення розробки та економії часу на написання типових функцій, використовувалась бібліотека JQuery [18].

Під час виконання файла main.js спочатку в змінні зберігаються ідентифікатори елементів сторінки, які часто використовуються. Ця дія виконується за допомогою функції getElementById об'єкту document. Далі створюються масиви для зберігання об'єктів нейрона, входів та підключень.

Після оголошення змінних здійснюється оголошення функцій і обробників подій. Створення обробників подій здійснюється за допомогою методу delegate бібліотеки JQuery.

Після оголошенення всіх змінних та функцій, описується створення входів. Наведемо приклад створення першого входу:

```
inp1 = new Input({
   title: 'in',
   dir: 1,
   fType: 1,
   formula: 'sin(x/6)*2'
});
```

Об'єкт Input може приймати для попереднього налаштування входу різні параметри. Параметр title відповідає назві входу, яка буде відображатися на його графічній репрезентації, параметр dir визначає збуджуючий цей сигнал чи гальмівний, Параметр fType визначає тип сигналу (1 - аналоговий сигнал, 2 – імпульсний сигнал), параметр formula визначає формулу для формування аналогового сигналу чи послідовність імпульсів для імпульсного сигналу відповідно. При заданні аналогового сигналу можна використовувати всі стандартні математичні функції JavaScript. Наприклад, функція Math.abs(sin(x/6)*2) буде генерувати синусоїду, на якій буде «віддзеркалено» від'ємні значення (рис. 3.3).

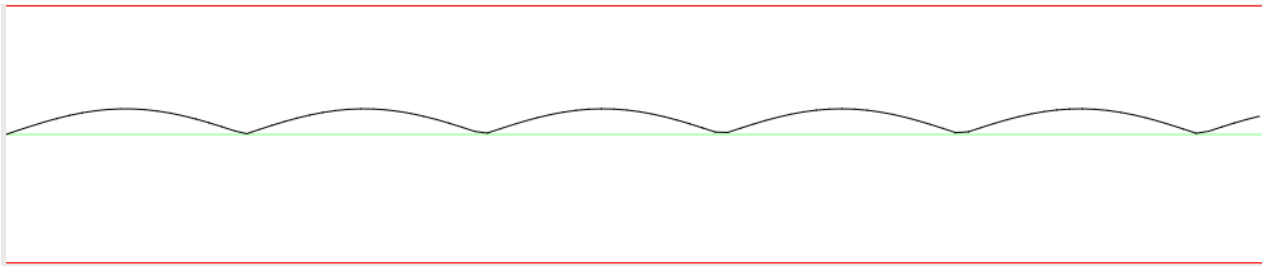


Рисунок 3.3 – Графік функції Math.abs(sin(x/6)*2)

Для генерування аналогового вхідного сигналу у вигляді пилки (рис. 3.4) потрібно у полі ввести формулу x%8. Символ % у JavaScript означає знаходження модуля числа.

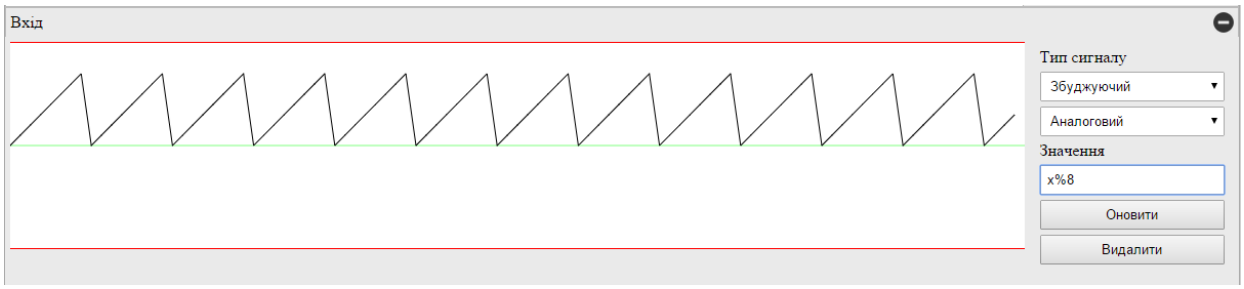


Рисунок 3.4 – Графік функції у вигляді пилки

Імпульсний сигнал, на відміну від аналогового, задається у вигляді набору числових значень, які вказують, коли саме (на яких кроках моделювання) генерувати вхідні імпульси. Для зручності користувача задавати повторювані сигнали, було додано можливість задавати проміжок часу, на якому нейрони будуть повторюватися та частоту цих повторень. Якщо вписати в набір числових значень 10-20, то отримаємо 11 повторюваних кожний крок підряд імпульсів від 10 до 20 кроків включно (рис. 3.5).

Для зручності задання повторюваних імпульсів було додано шаблон x+y+z, де x - початкове значення номеру кроку моделювання, y – період повторення імпульсів, а z - кінцеве значення номеру кроку моделювання. Тобто, якщо в набір числових значень імпульсів вписати 5+5+40, то отримаємо

8 імпульсів з періодом 5, причому перший імпульс буде на 5-му кроці, а останній на 40-му кроці. (рис. 3.6).

Рисунок 3.5 – Графік вхідних імпульсів, заданий значенням 10-20

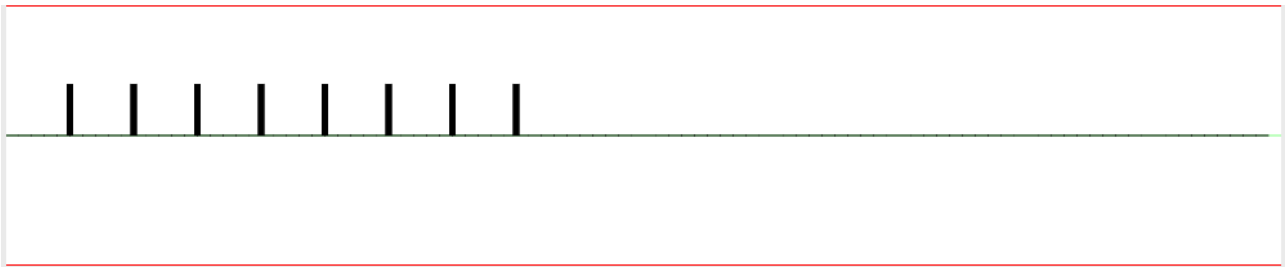


Рисунок 3.6 – Графік, заданий значенням 5+5+40

Все зазначене можна комбінувати та створювати бажану послідовність імпульсів. Наприклад, якщо ввести значення: "1,5,10, 12-18, 50+4+90", то отримаємо послідовність імпульсів на 1-му, 5-му, 10-му кроках моделювання, потім з 12 по 18 кроки буде 7 імпульсів підряд, а потім з 50-го кроку по 90-й буде 11 імпульсів через 4 кроки (тобто на 50-му, 54-му, 58-му, 62-му, 66-му, 70-му, 74-му, 78-му, 82-му, 86-му, 90-му кроках), як зображено на рисунку 3.7.

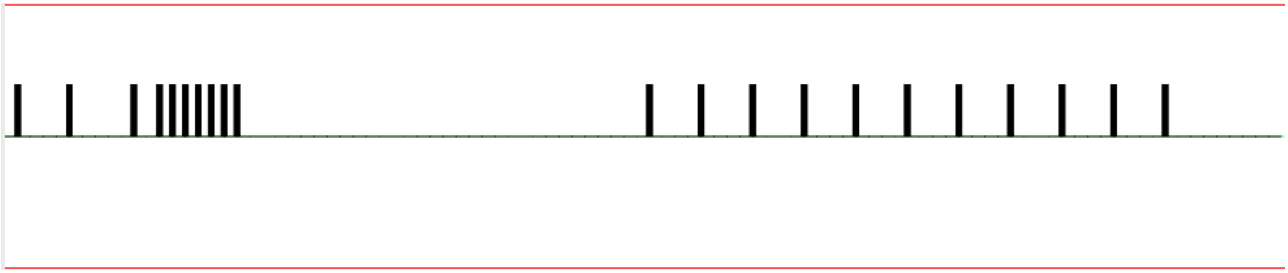


Рисунок 3.7 – Графік, заданий значеннями "1,5,10, 12-18, 50+4+90"

Гнучкість мови програмування JavaScript дозволяє задавати умови навіть всередині аналогового сигналу, заданого функцією. Наприклад, можливо використати тернарний оператор і побудувати, наприклад, функцію x%8 > 3 ? 1 : 5 (рис. 3.8). Це буде сигнал, в основі якого «пилка» y=x, але тепер, якщо значення x в поточний момент часу є менше 3, то воно замінюється на значення 1, а якщо ж значення x в поточний момент часу більше трьох, то воно замінюється на 5. Це дозволяє створювати функціонально надзвичайно складні сигнали, не використовуючи складні інструменти.

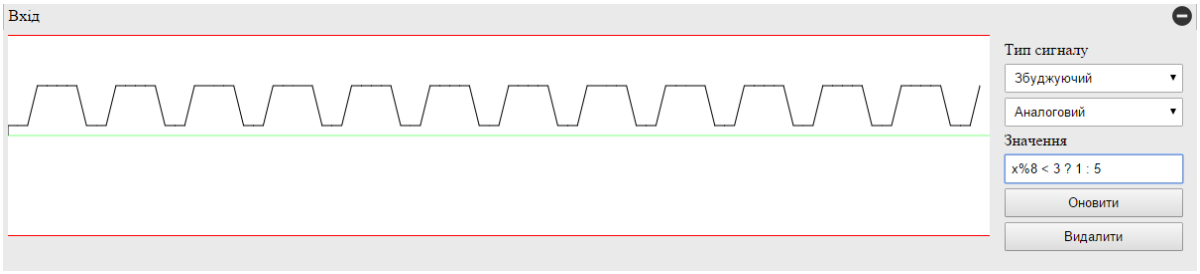


Рисунок 3.8 – Графік, заданий виразом « x%8 > 3 ? 1 : 5 »

Метод Math.random() дозволяє генерувати випадкові числа. З його допомогою можна додавати до стаціонарного сигналу довільну випадкову величину. Наприклад, задавши формулу «sin(x/10+Math.random())*2», отримаємо вхідний сигнал у вигляді синусоїди з деякими випадковими спотвореннями (рис. 3.9).

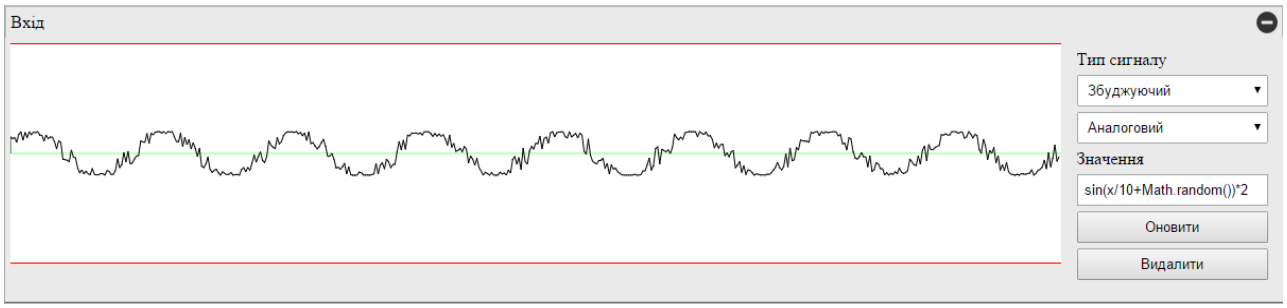


Рисунок 3.9 – Графік сигналу, заданий виразом «sin(x/10+Math.random())*2»

Нижче наведемо список математичних методів, які підтримуються JavaScript [19] і які можна використовувати для задання вхідних сигналів нейрона:

- Math.acos,
- Math.atan,
- Math.asin,
- Math.atan2,
- Math.exp,
- Math.random,
- Math.min,
- Math.sqrt,
- Math.round,
- Math.log,
- Math.floor,
- Math.sin,
- Math.ceil,
- Math.cos,
- Math.tan,
- Math.max,
- Math.pow,
- Math.abs.

Також для задання вхідних сигналів нейрона можна використовувати константи, які підтримуються JavaScript:

- Math.E,
- Math.LOG2E,
- Math.LOG10E,
- Math.LN2,
- Math.PI,
- Math.SQRT2,
- Math.SQRT12,

- Math.LN10.

Після створення потрібної кількості входів нейрону створюється власне об'єкт самого нейрону:

```
n1 = new Neuron({
title: 'N1',
x: 660,
y: 90
});
```

При створенні об'єкту самого нейрону, так само, як і при створенні входу нейрона, передається параметр title, який задає назву даному нейрону, а також координати x та y нейрону на карті вікна програми.

Перераховані об'єкти мають всередині метод _draw, який відмальовує їх одразу після створення.

Після створення об’єкту нейрон та його входів, потрібно також створити зв’язки між входами та нейроном. Для цього було використано функцію createConnection, що приймає два параметри. Перший із цих двох параметрів задає об'єкт, від якого проводиться зв'язок, а другий параметр задає об'єкт, до якого проводиться зв'язок. Отже, для здійснення з'єднання першого входу з нейроном необхібно виконати команду:

```
createConnection(inp1.nid, n1.nid).
```

Після побудови карти нейронів та зв'язків викликається метод redrawGraphs, який перемальовує графіки моделювання з урахуванням створених об'єктів.

Знаходження значень функції здійснюється всередині методу drawGraph. У цьому методі відбувається спочатку перевірка заданої формули. Якщо формула задана, то відбувається відмальовування графіку входів нейрону. Якщо ж формула не задана, тоді відмальовується значення вихідного сигналу даного нейрона.

Для входів нейрону значення підраховуються дуже просто, оскільки потрібно опрацювати лише одну змінну. Якщо ж відбувається відмалювання

графіка вихідного сигналу нейрона, то потрібно обраховувати значення одразу всіх вхідних сигналів нейрона. Саме для цього було створено цикл, який обходить усі вхідні сигнали, підраховує їхні значення, а потім, залежно від того, збуджуючий сигнал чи гальмівний на конкретному вході, цей цикл додає чи віднімає значення сигналу від значення мембранного потенціалу.

Для аналогового сигналу та імпульсного сигналу поточні значення обраховуються по-різному. У випадку аналогового сигналу, потрібно лише виконувати функцію eval [20], передавши їй формулу як параметр. У випадку ж імпульсного сигналу, здійснюється пошук конкретного значення імпульсу в послідовності імпульсів.

Отже, поєднання всіх зазначених вище функціональних можливостей дозволило створити такий додаток, який в дійсності легко налаштовується. А завдяки тому, що він має просту внутрішню структуру, його в майбутньому буде легко розширювати та вдосконалювати.

3.3 Висновок до розділу 3

У розділі було обґрунтовано вибір для створення комп’ютерної моделі спайкінгового нейрона мови програмування JavaScript та середовища розробки JetBrainsWebStorm, оскільки він є зручним та розумним редактором JavaScript, HTML та CSS, що підтримує також нові технології. Здійснено програмну реалізація комп’ютерної моделі спайкінгового SRM-нейрона. Графічний інтерфейс реалізовано за допомогою HTML та CSS. А весь функціонал реалізовано мовою JavaScript та розділено на декілька підключених файлів: main.js, input.js, neuron.js, map-init.js. Також, для полегшення розробки та економії часу на написання типових функцій, використовувалась бібліотека JQuery.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМНОГО МОДУЛЯ ДЛЯ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ СПАЙКІНГОВИХ НЕЙРОНІВ

4.1 Тестування програмного модуля для комп'ютерного моделювання спайкінгових нейронів

Одразу після відкриття Web-сторінки програми, відображається карта зв’язків з двома входами (in) та нейроном (N1), а також графіки вхідних сигналів та вихідних (рис.4.1).

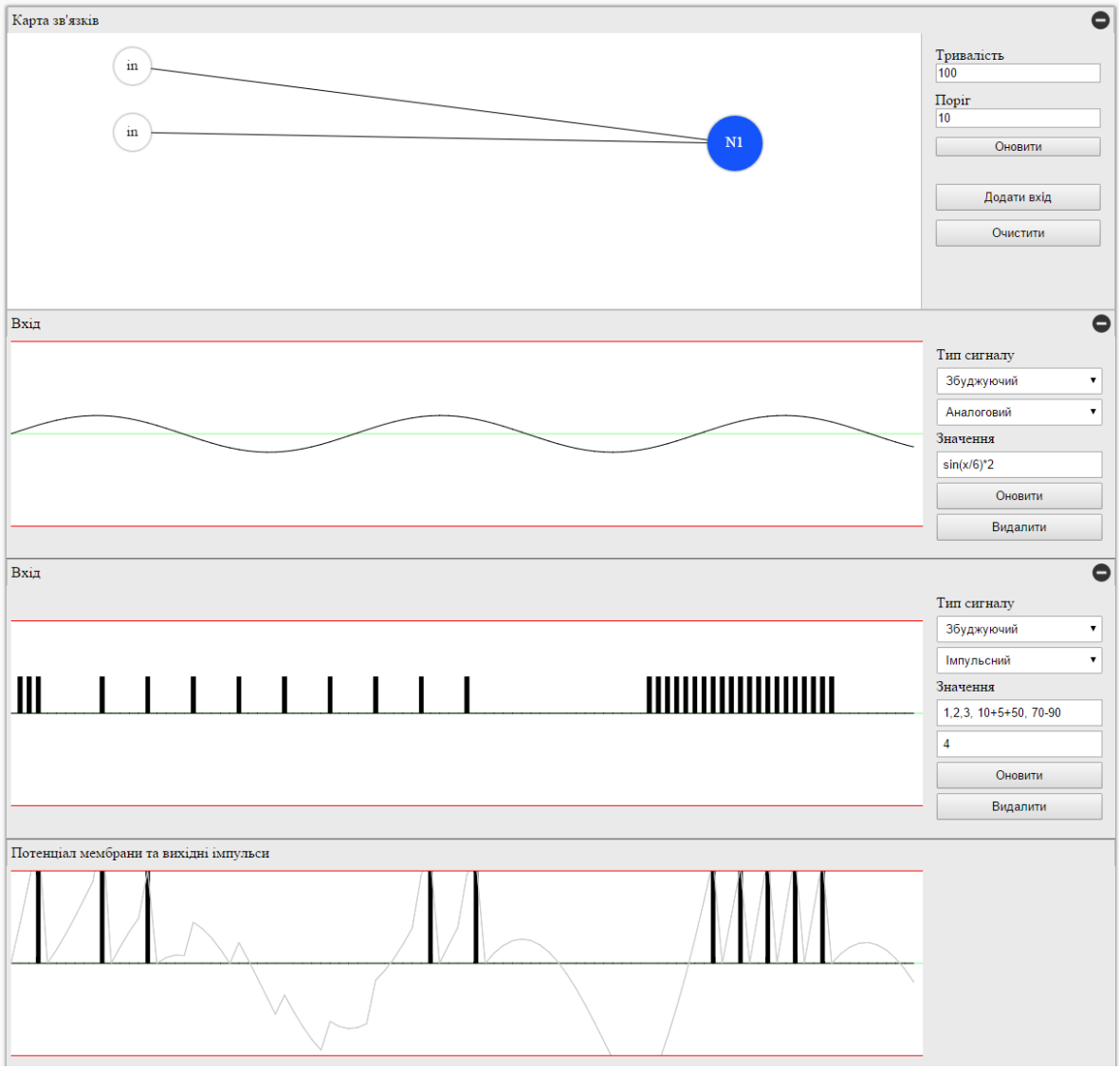


Рисунок 4.1 – Вид вікна додатку після відкриття

Робоче поле програми умовно поділено на дві частини: ліва частина – зона графіків та права частина – зона налаштувань. Напроти графіків вхідних сигналів відображаються налаштування для відповідного окремого входу (рис. 4.2).

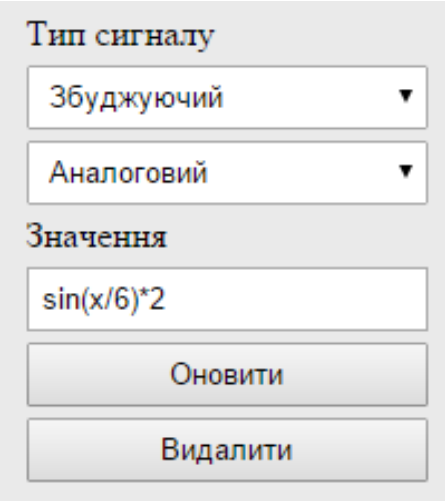


Рисунок 4.2 – Налаштування для входу нейрона

Якщо тип сигналу задано як «Збуджуючий», то його значення щоразу (на кожному кроці моделювання) буде додаватися, а якщо тип сигналу задано як «Гальмівний», то - відніматися. Наприклад, спочатку моделювали нейрон з використанням сигналу звичайної синусоїди на першому вході та сигналом лінійної функції на другому вході (рис. 4.3). А потім змінили тип входу сигналу з лінійною функцією на гальмівний (рис. 4.4), то побачимо, що графік мембранного потенціалу різко пішов донизу.

Кнопка "Оновити" призначена для оновлення графіків моделювання після зміни типу та значення вхідного сигналу, а кнопка "Видалити" призначена для видалення відповідного входу.

Також кожен блок робочого поля для графіків та карти має кнопку для приховування чи відображення відповідного блоку (рис. 4.5), що робить роботу з додатком більш комфортною у випадку коли створено багато входів.

Адже ця кнопка дозволяє приховати ті входи, які на даний момент не цікаві для моделювання, а залишити ті входи, які необхідні.

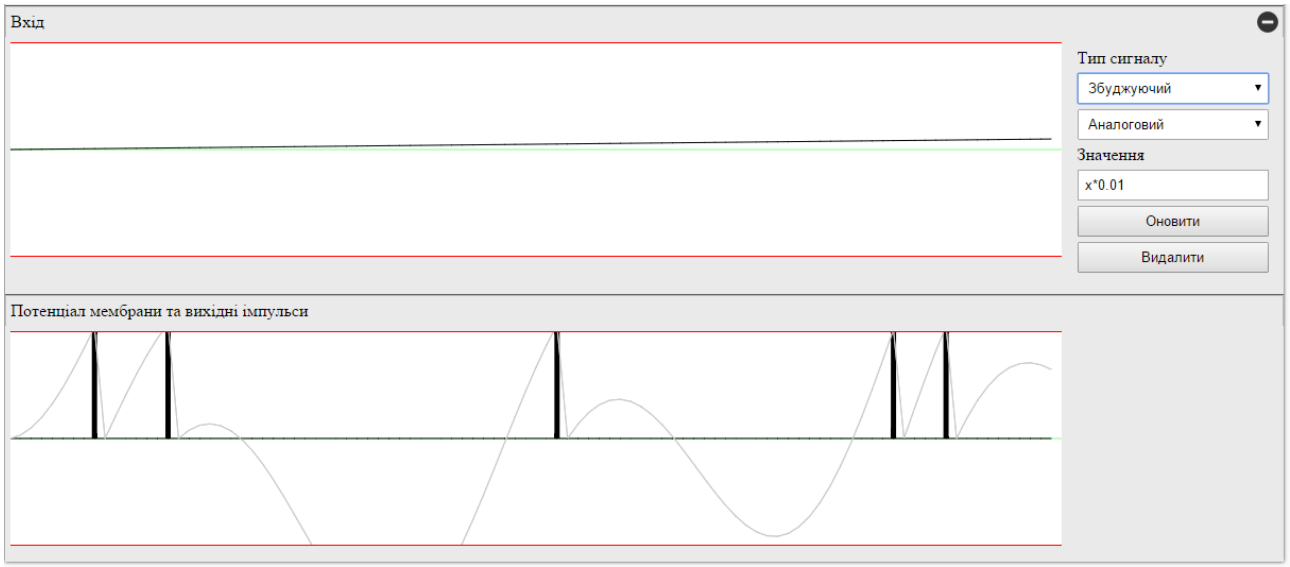


Рисунок 4.3 – Вид графіків при використанні збуджуючого сигналу

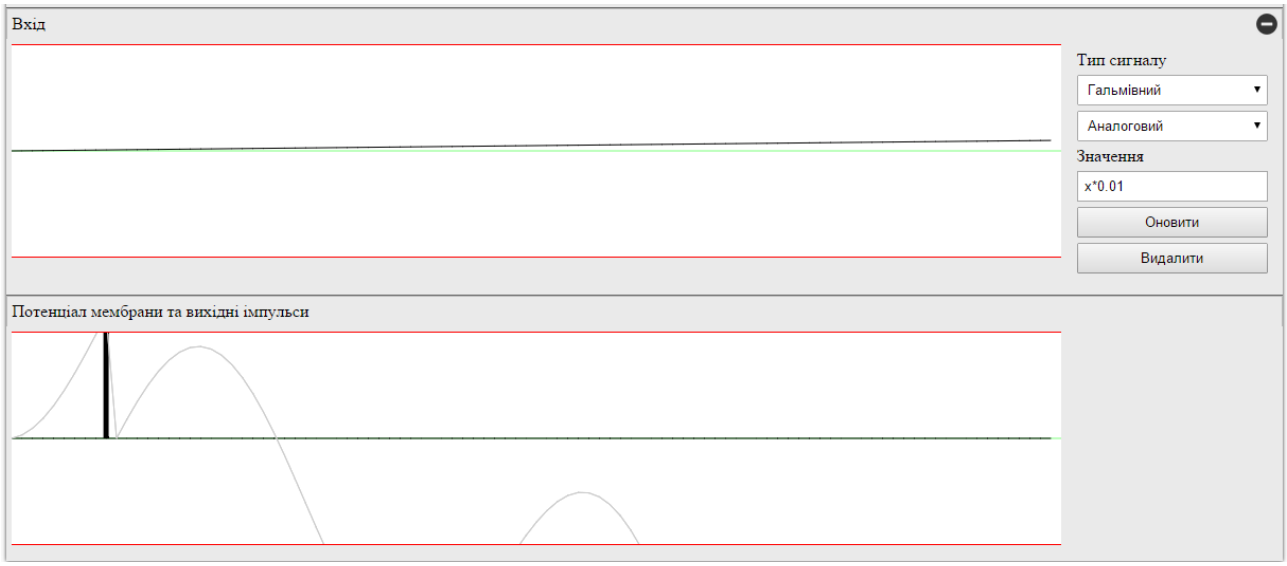


Рисунок 4.4 – Вид графіків при використанні гальмівного сигналу

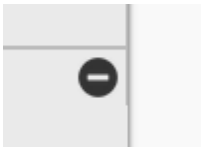


Рисунок 4.5 – Кнопка вмикання/вимикання видимості блоку

Праворуч від карти зв'язків нейрона є вікно налаштування процесу моделювання (рис. 4.6). У полі «Тривалість» задається кількість часових кроків, протягом яких буде проводитись моделювання. В залежності від обраної тривалості визначається значення кроку між ітераціями. Значення цього кроку дорівнює відношенню ширини елементу Canvas, на якому відбудовується графік, до тривалості. За замовчуванням обирається крок 100, а ширина елементу Canvas - 990 пікселів. Значить, поділивши 990 на 100, визначимо, що крок = 9.9 пікселів. Це можна перевірити, задавши тривалість рівну 10. У цьому випадку вид графіку зміниться і буде відображено лише десяту частину початкового графіку (рис. 4.7).

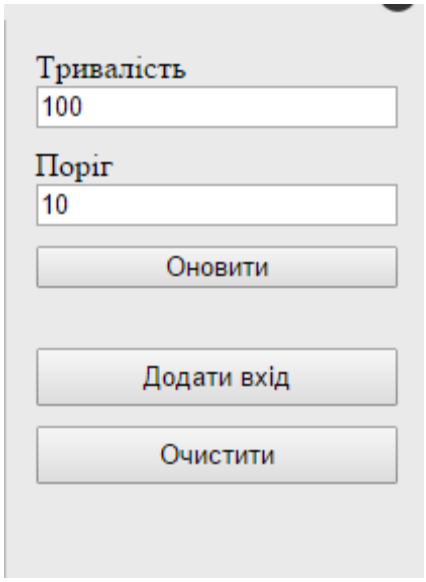


Рисунок 4.6 – Вікно налаштування моделювання

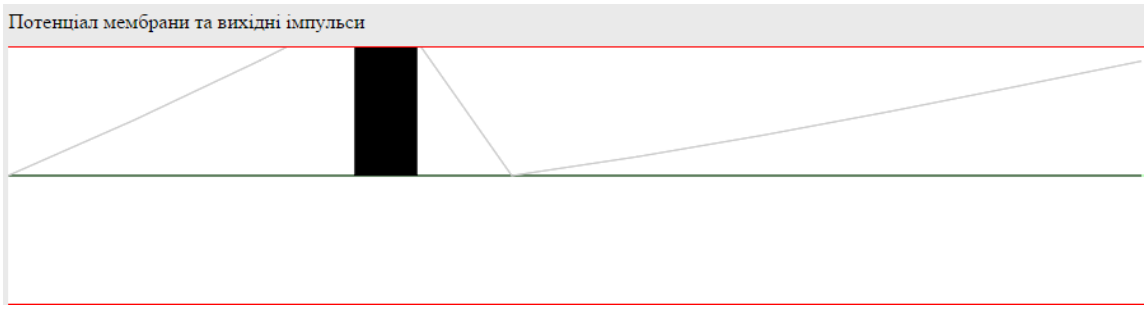


Рисунок 4.7 – Вид графіку при тривалості 10

Поле «Поріг» задає значення мембранного потенціалу, при досягненні якого буде генеруватись імпульс на виході нейрона. Збільшивши поріг до 20, можна побачити на рисунку 4.8, як змінився вид графіку відносно рис. 4.1.

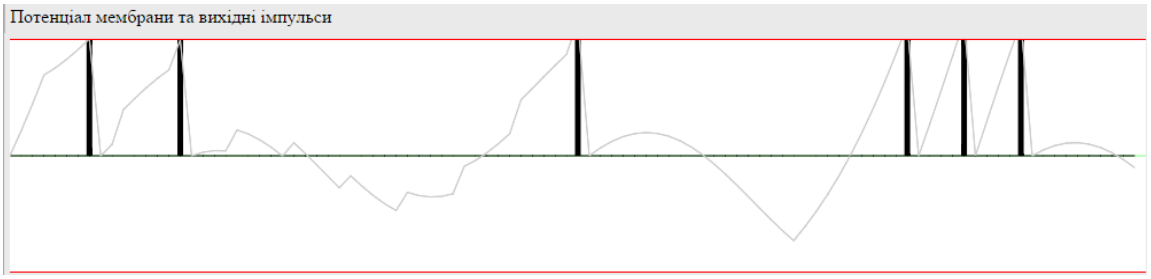


Рисунок 4.8– Вид графіку при значенні порогу 20

Кнопка «Додати вхід» дозволяє додати ще один вхід нейрона на карту та створить для нього додаткову зону графіку. За замовчуванням нові входи не генерують жодних сигналів (рис. 4.9).

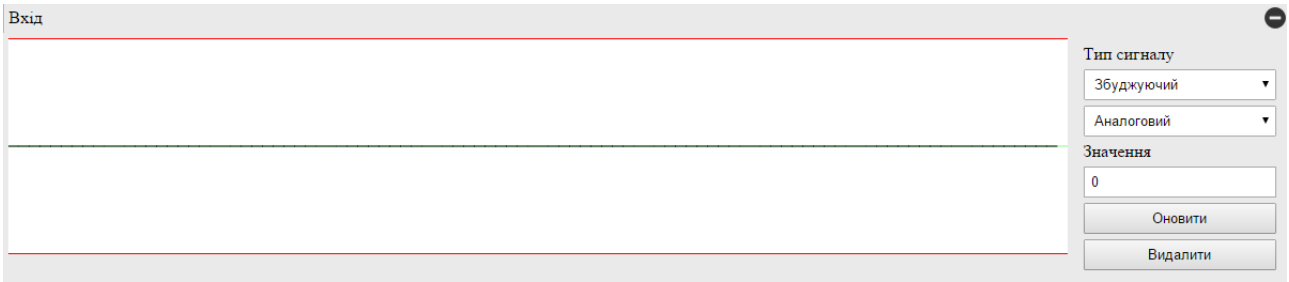


Рисунок 4.9 – Вид зони графіку та вікна налаштувань щойно створеного входу

Кнопка «Очистити» дозволяє видалити всі входи та зони їх графіків (рис. 4.10).

Розроблений додаток може працювати на будь-якому пристрої, який має доступ до Internet. Достатньо перейти за посиланням http://www.neurons.redelium.com і одразу відкриється web-сторінка додатку.

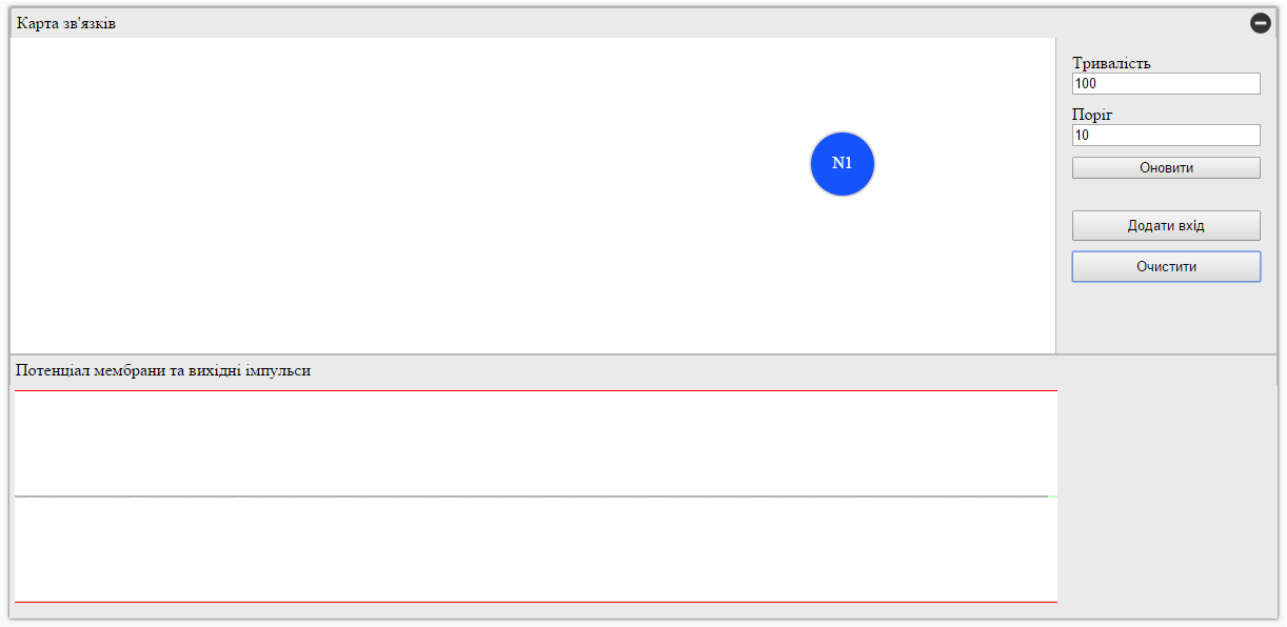


Рисунок 4.10 – Вид сторінки додатку після натискання кнопки «Очищення»

Додаток було протестовано на декількох пристроях, що мають різні характеристиками та роздільну здатність екрану і на всіх працював добре.

Роботу додатку на планшеті показано на рис. 4.11.

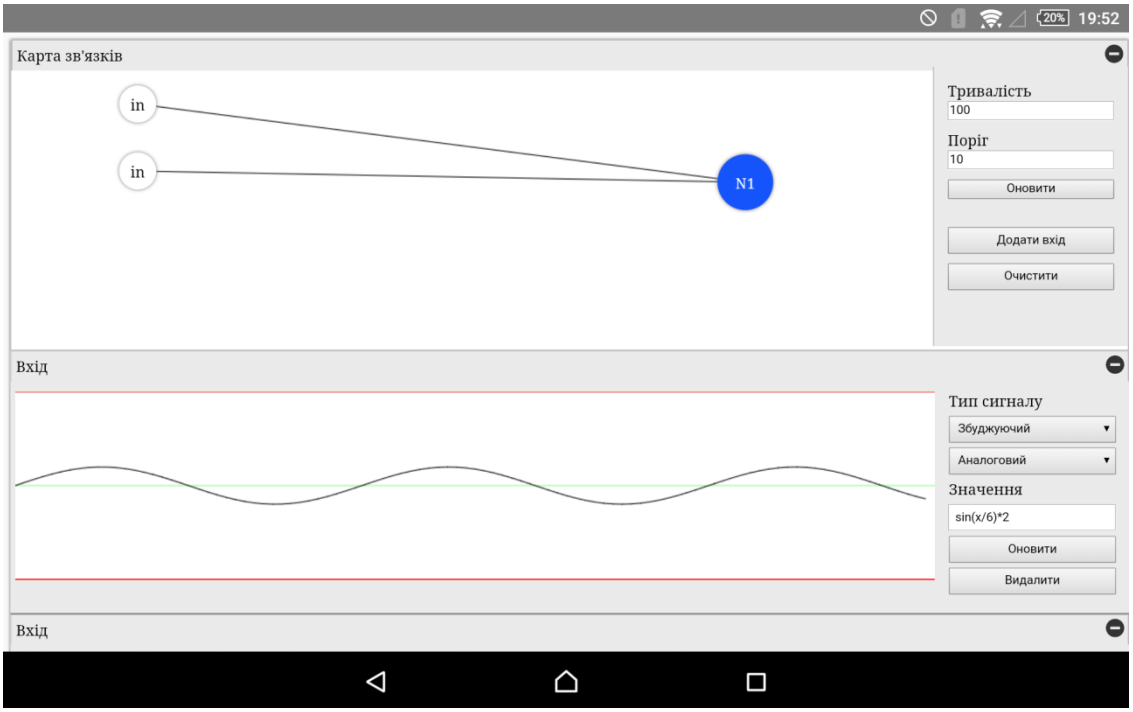


Рисунок 4.11 – Вид додатку на планшеті

4.2 Аналіз результатів роботи програмного модуля для комп'ютерного моделювання спайкінгових нейронів

Для доведення досягнення поставленої мети бакалаврської кваліфікаційної роботи, а саме – розширення функціональних можливостей програмного модуля для комп'ютерного моделювання спайкінгових нейронів, наведемо порівняння його функціональних можливостей з функціональними можливостями аналога – програми GENESIS (рис. 1.2).

Таблиця 4.1 - Порівняння функціональних можливостей розробленого програмного модуля та аналога

№ п/п	Назва функціональної можливості	Аналог (GENESIS)	Розроблений модуль
1	Можливість моделювання SRM-нейрона	+	+
2	Можливість задання кількості входів від 0 до 32	+	+
3	Можливість роботи на мобільних пристроях	_	+
4	Можливість відображення мембранного потенціалу та вихідних імпульсів на одному графіку	_	+
5	Можливість автоматично підлаштовувати поле виведення графіка під кількість кроків моделювання	+	+
6	Можливість додавати до аналогового вхідного сигналу випадкову складову	_	+
7	Можливість задавати довільну послідовність імпульсів на імпульсному вході	+	+

Із таблиці 4.1 видно, що програма-аналог має 4 функціональні можливості, а розроблений програмний модуль – 7, тобто розроблений

програмний модуль має на 75% більше функціональних можливостей, ніж програма-аналог. Тобто мета роботи досягнута - функціональні можливості розробленого програмного модуля розширені.

Розроблену комп’ютерної модель SRM-нейрона можна в подальшому використовувати в дослідженнях для проведення різноманітних експериментів з нею. Крім цього, розроблену комп’ютерну модель можна застосовувати для побудови і дослідження поведінки спайкінгових нейронних мереж як з невеликим числом нейронів, так і великомасштабних спайкінгових нейронних мереж. Завдяки своїй простоті з точки зору необхідної кількості операцій з плаваючою комою в одиницю часу моделювання, розроблена комп’ютерна модель SRM-нейрона особливо буде корисна при моделюванні поведінки саме великомасштабних спайкінгових нейронних мереж.

4.3 Висновок до розділу 4

У результаті тестування програмного модуля для комп'ютерного моделювання спайкінгових нейронів було доведено його повну працездатність та відповідність поставленому завданню. Було продемонстровано роботу модуля у різних режимах задання імпульсних та аналогових вхідних сигналів. Порівняння переваг розробленого модуля перед аналогом було зведено у таблицю. Аналіз таблиці показав, що програма-аналог має 4 функціональні можливості, а розроблений програмний модуль – 7, тобто розроблений програмний модуль має на 75% більше функціональних можливостей, ніж програма-аналог. Таким чином, мета роботи досягнута - функціональні можливості розробленого програмного модуля розширені.

ВИСНОВКИ

У ході виконання бакалаврської дипломної роботи було зроблено детальну постановку задачі комп'ютерного моделювання спайкінгових нейронів. Також розглянуто предметну область спайкінгових нейронів та спайкінгових нейронних мереж, їх місце серед моделей нейронів першого та другого покоління, переваги та недоліки спайкінгових моделей нейронів, їх більшу адекватність біологічним нейронам і правдоподібність у моделюванні поведінки до біологічних нейронів в нейронних мереж. Було показано перспективність комп’ютерного моделювання спайкінгових нейронів. Крім цього, було проаналізовано такі програми-аналоги як PCSIM, GENESIS, NEURONS, визначено їх переваги та недоліки.

Було обґрунтовано вибір моделі спайкінгового нейрона. Із таких моделей як Інтегрально-імпульсна (SRM), Квадратична інтегрально-імпульсна (QI&F), Резонансно-імпульсна (R&F), модель Хіндмарша-Розе, модель Ходжкина-Хаксли було обрано для комп’ютерного моделювання за критерієм максимум функцій – мінімум обчислювальних витрат таку модель як Інтегрально-імпульсна (SRM), яка лежить в основі запропонованого модуля. Проаналізовано математичну модель спайкінгового SRM-нейрона. Розроблено алгоритм роботи програмного модуля для комп’ютерного моделювання спайкінгового SRM-нейрона.

Було обґрунтовано вибір для створення комп’ютерної моделі спайкінгового нейрона мови програмування JavaScript та середовища розробки JetBrains WebStorm, оскільки він є зручним та розумним редактором JavaScript, HTML та CSS, що підтримує також нові технології. Здійснено програмну реалізацію комп’ютерної моделі спайкінгового SRM-нейрона. Графічний інтерфейс реалізовано за допомогою HTML та CSS. А весь функціонал реалізовано мовою JavaScript та розділено на декілька підключених файлів: main.js, input.js, neuron.js, map-init.js. Також, для

полегшення розробки та економії часу на написання типових функцій, використовувалась бібліотека JQuery.

У результаті тестування програмного модуля для комп'ютерного моделювання спайкінгових нейронів було доведено його повну працездатність та відповідність поставленому завданню. Було продемонстровано роботу модуля у різних режимах задання імпульсних та аналогових вхідних сигналів. Порівняння переваг розробленого модуля перед аналогом було зведено у таблицю. Аналіз таблиці показав, що програма-аналог має 4 функціональні можливості, а розроблений програмний модуль – 7, тобто розроблений програмний модуль має на 75% більше функціональних можливостей, ніж програма-аналог. Таким чином, мета роботи досягнута - функціональні можливості розробленого програмного модуля розширені.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Нейронні мережі, сутність мереж [Електронний ресурс]. – Режим доступу: http://goo.gl/VHRdOM

2 Руденко О.В. Штучні нейронні мережі: Навчальний посібник / О.В.Руденко, Є.В.Бодянський. - Харків : ТОВ «Компанія СМІТ», 2006. — 404 с. - ISBN 966-8630-73-X.

3 Connors B. W. Intrinsic firing patterns of diverse neocortical neurons / B. W. Connors and M. J. Gutnick // Trends Neurosci., vol. 13, pp. 99–104, 1990.

4 Maass W. Networks of Spiking Neurons: The third generation of Neural Network Models. / W.Maass // Neural Networks – New York : Springer, 1997.

5 Gray C. M. Chattering cells: Superficial pyramidal neurons contributing to the generation of synchronous oscillations in the visual cortex / C. M. Gray and D. A. McCormick // Science, vol. 274, no. 5284, pp. 109–113, 1996.

6 Gerstner W. Spiking Neuron Models. Single Neurons, Populations, Plasticity / Wulfram Gerstner, Werner M. Kistler. – Cambridge University Press, 2002. – 494p. – ISBN 0 521 89079 9 (paperback). – ISBN 0 521 81384 0 (hardcover).

7 GENESIS (the GEneral NEural SImulation System) [Електронний ресурс]. – Режим доступу: http://www.genesis-sim.org/

8 Ermentrout G. B. Type I membranes, phase resetting curves, and synchrony / G. B. Ermentrout // Neural Comput., vol. 8, pp. 979–1001, 1996.

9 Rose R. M. The assembly of ionic currents in a thalamic neuron. I The three-dimensional model / R. M. Rose and J. L. Hindmarsh // Proc. R. Soc. Lond. B, vol. 237, pp. 267–288, 1989.

10 Hodgkin A. L. A quantitative description of membrane current and application to conduction and excitation in nerve / A. L. Hodgkin and A. F. Huxley // J.Physiol., vol. 117, pp. 500–544, 1954.

11 Мова ActionScript та його синтаксис [Електронний ресурс]. – Режим доступу: http://goo.gl/SmhzmW

12 Java [Електронний ресурс]. – Режим доступу: https://www.oracle.com/ua/java/technologies/javase/jdk11-archive-downloads.html

13 Сучасний підручник з JavaScript [Електронний ресурс]. – Режим https://uk.javascript.info/

14 HTMLBook [Електронний ресурс]. – Режим доступу: http://htmlbook.ua/content

15 Каскадні таблиці стилів [Електронний ресурс]. – Режим доступу: https://goo.gl/1wcfsf

16 JetBrains WebStorm [Електронний ресурс]. – Режим доступу: https://goo.gl/mMKN6g

17 Using the viewport meta tag to control layout on mobile browsers [Електронний ресурс]. – Режим доступу: https://goo.gl/WUHQlM

18 JQueryTutorial [Електронний ресурс]. – Режим доступу: http://www.w3schools.com/jquery/

19 Math [Електронний ресурс]. – Режим доступу: http://javascript.ua/Math

20 JavaScript eval() Function [Електронний ресурс]. – Режим доступу: http://www.w3schools.com/jsref/jsref_eval.asp

21 Методичні вказівки до виконання бакалаврської кваліфікаційної роботи для студентів денної та заочної форм навчання спеціальності 122 - «Комп’ютерні науки» / Укладачі А.А.Яровий, О.В.Сілагін, І.В.Богач. – Вінниця: ВНТУ, 2022. – 47 с.

Додаток А (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний модуль для комп'ютерного моделювання спайкінгових нейронів

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 11,80 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Паночишин Ю.М.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Малюгов І.М.
(підпис) (прізвище, ініціали)

Додаток Б (обов'язковий)

Лістинг програми

```
varnMap = document.getElementsByClassName('neurons-map')[0],
mapHelper = document.getElementById('neuron-map-helper'),
resultsCanvas = document.getElementById('results'),
isDrawing = false,
nDrawFrom = null,
nDrawTo = null,
neuronConnections = [],
neurons = {},
inputs = {},
inputsArray = [],
lastMouseX = 0,
lastMouseY = 0,
mainTime = 100,
hMax = 10,
iteration = 0,
popupId = 0;
startYPos = 130;
potential = 0;
isReset = 0;

nid = 1;

$(mapHelper).on('mousemove', function(e){
if(isDrawing) {
redrawConnections();
var ctx = mapHelper.getContext("2d");
        ctx.lineWidth = "1";
ctx.beginPath();
ctx.moveTo(lastMouseX, lastMouseY);
ctx.lineTo(e.offsetX, e.offsetY);
ctx.stroke();
    }
})

function sin(x) {
return Math.sin(x);
}

function cos(x) {
return Math.cos(x);
}

$(document).delegate('.neurons-map .connection', 'click', function(){
if($(this).hasClass('info')) {
        $(this).removeClass('info');
        $(this).html('');
    } else {
showConnectionInfo($(this).data('cid'));
    }
})

$(document).delegate('.popups-wrapper .button.close', 'click', function(e){
    $(this).parents('.popup-window').remove();
e.preventDefault();
return false;
})

function showConnectionInfo(cid){
point = $('.neurons-map .connection[data-cid='+cid+']');
point.html('Info');
point.addClass('info');
}

function redrawConnections(){
var ctx = mapHelper.getContext("2d");
ctx.clearRect(0, 0, 960, 300);
    $('.neurons-map .connection').remove();
if(neuronConnections.length > 0) {
        ctx.lineWidth = "1";
ctx.beginPath();
for(i = 0; i < neuronConnections.length; i++){
```

```
if(neuronConnections[i]) {
ctx.moveTo(neurons[neuronConnections[i].from].x, neurons[neuronConnections[i].from].y);
ctx.lineTo(neurons[neuronConnections[i].to].x, neurons[neuronConnections[i].to].y);
                /*
                 $(nMap).append('<div class="connection" data-cid="' + i + '"></div>');
var connection = $('.neurons-map .connection[data-cid='+i+']');
connection.css({
                 'left':                  ((neurons[neuronConnections[i].from].x                  +
neurons[neuronConnections[i].to].x)/2 - 10) + 'px',
                 'top':                   ((neurons[neuronConnections[i].from].y                  +
neurons[neuronConnections[i].to].y)/2 - 10) + 'px'
                 });
                 */
            }
        }
ctx.stroke();
    }
}

function createConnection(from, to){
if(!(neurons[to].from && (neurons[to].from.indexOf(from)+1))) {
neurons[to].from.push(from);
neuronConnections.push({from: from, to: to});
    }
redrawConnections();
}

$('#button-play').click(function(){
drawGraph('results');
    /*
fulltime = time = mainTime;
lastResPosX = 0;
lastResPosY = 0;
rctx = resultsCanvas.getContext("2d");
rctx.clearRect(0, 0, 300, 300);
    rctx.lineWidth = "1";
rctx.beginPath();
rctx.moveTo(0, 150);
processingCycle();
    */
});

function createMap() {
var map = {};
for (var neuron in neurons) {
if (neurons.hasOwnProperty(neuron)) {
var obj = neurons[neuron];
            //obj.
        }
    }
}

function showNeuronInfo(n) {
setTimeout(function(){
popupId++;
        $('body .popups-wrapper').append(
            '<div class="popup-window neuron-config" data-pid="'+popupId+'">' +
            '<div class="head">' +
            '<div class="title">Нейрон '+n.title+'</div>' +
            '<a href="#" class="button close">X</a>' +
            '</div>' +
            '<div class="content"><br>Сума вхідних сигналів:<br>' +
            '<canvas  id="result-nid-'+n.nid+'"  class="output"  height="100"  width="300"  data-
nid="'+n.nid+'"></canvas>' +
            '</div>' +
            '</div>'
        );

drawGraph('result-nid-'+n.nid);
    },10);
}

function showInputInfo(inp) {
setTimeout(function(){
popupId++;
        $('body .popups-wrapper').append(
            '<div     class="popup-window     input-config"     data-pid="'+popupId+'"     data-
nid="'+inp.nid+'">' +
            '<div class="head">' +
```

```
                '<div class="title">Налаштування входу</div>' +
                '<a href="#" class="button close">X</a>' +
            '</div>' +
            '<div class="content">' +
                '<div></div>' +
                '<div class="formula-config">' +
                    '<div class="type-selector">' +
                        '<label>Тип сигналу</label><br>' +
                        '<select class="formula-dir">' +
                            '<option value="1">Збуджуючий</option>' +
                            '<option value="2">Гальмівний</option>' +
                        '</select>' +
                        '<select class="formula-type">' +
                            '<option value="1">Аналоговий</option>' +
                            '<option value="2">Імпульсний</option>' +
                        '</select>' +
                    '</div>' +
                    '<label>Значення</label><br>' +
                    '<input type="text" class="input-formula" value="'+inp.formula+'">' +
                    (inp.fType    ==    2    ?    '<input    type="text"    class="input-impulse"
value="'+inp.impulse+'">' : '') +
                    '<button class="button-update">Оновити</button>' +
                '</div>' +
                '<canvas id="result-nid-'+inp.nid+'" class="output" height="100" width="300" data-
nid="'+inp.nid+'"></canvas>' +
                '<div class="additional-actions"><button class="delete">Видалити</button></div>' +
            '</div>' +
            '</div>'
        );

        $('.input-config[data-nid='+inp.nid+'] .formula-type').val(inp.fType);
        $('.input-config[data-nid='+inp.nid+'] .formula-dir').val(inp.dir);

drawGraph('result-nid-'+inp.nid);
    },10);
}

function drawGraph(canvasId, isEnd) {
if($('#'+canvasId).data('nid') > 0) {
var el = neurons[$('#'+canvasId).data('nid')];
    } else {
var el = n1;
    }
var ncElement = document.getElementById(canvasId);
var nc = ncElement.getContext("2d");
var w = ncElement.width;
var h = ncElement.height;
var m = h / 2;
var ns = w / mainTime;
var vs = h / (hMax * 2);
var lastX = 0;
var lastY = 0;
var iteration = 0;
var value = 0;
var potential = 0;
var mLast = 0;

nc.clearRect(0, 0, w, h);
    nc.lineWidth = 1;
nc.beginPath();
nc.rect(0,0,w,h);
    nc.strokeStyle = "#fff";
    nc.fillStyle="#fff";
nc.fill();
nc.stroke();
nc.beginPath();
    nc.strokeStyle = "rgba(0,255,0,0.5)";
nc.moveTo(0, m);
nc.lineTo(w, m);
nc.stroke();
nc.beginPath();
nc.moveTo(0, m);
    nc.strokeStyle = '#000';

for(var x = 0; x < mainTime; x++) {
px = x * ns;
nc.beginPath();
if(el.formula) {
value = 0;
```

```
var v = 0;
if(el.fType == 1) {
                v = eval(el.formula);
            } else if(el.fType == 2) {
var fArray = el.impulses;
for(var k=0; k<fArray.length; k++) { fArray[k] = parseInt(fArray[k], 10); }
if(fArray.indexOf(iteration)+1 > 0) {
                    v = el.impulse;
                } else {
                    v = 0;
                }
            }

value -= v;
        } else if(el.from.length > 0) {
var value = 0;
for(var i = 0; i < el.from.length; i++) {
if(neurons[el.from[i]]) {
if(neurons[el.from[i]].formula && neurons[el.from[i]].formula.length > 0) {
var v = 0;
if(neurons[el.from[i]].fType == 1) {
                            v = eval(neurons[el.from[i]].formula);
                        } else if(neurons[el.from[i]].fType == 2) {
var fArray = neurons[el.from[i]].impulses;
for(var k=0; k<fArray.length; k++) { fArray[k] = parseInt(fArray[k], 10); }
if(fArray.indexOf(iteration)+1 > 0) {
                                v = neurons[el.from[i]].impulse;
                            } else {
                                v = 0;
                            }
                        }

if(neurons[el.from[i]].dir == 1) {
value -= v;
                        } else {
value += v;
                        }
                    }
                }
            }
        }
value *= vs;
if(isEnd) {
var lX = lastX;
var lY = lastY;
            nc.strokeStyle = '#000';
if(isReset) {
                nc.lineWidth = 1;
nc.moveTo(lastX, m);
lastX = px;
nc.lineTo(px, m);
isReset = false;
potential = 0;
nc.stroke();
            } else {
potential += value;
if(potential <= -m) {
isReset = true;
nc.moveTo(lastX, m);
nc.lineTo(px, m);
nc.stroke();
nc.beginPath();
nc.moveTo(px, m);
lastX = px;
                    nc.lineWidth = ns/2;
nc.lineTo(px, 0);
nc.stroke();
                } else {
                    nc.lineWidth = 1;
nc.moveTo(lastX, m);
lastX = px;
nc.lineTo(px, m);
nc.stroke();
                }
            }
nc.beginPath();
            nc.lineWidth = 1;
            nc.strokeStyle = '#ccc';
nc.moveTo(px-ns, m+mLast);
```

```
if(value < -m) {
value = -m;
            }
if(value > m) {
value = m;
            }
nc.lineTo(px, m+potential);
mLast = potential;
nc.stroke();
        } else {
if(el.fType && el.fType == 2) {
nc.beginPath();
                nc.lineWidth = 1;
nc.moveTo(lastX, m);
if(value > 0) {
value = m;
                }
lastX = px;
nc.lineTo(lastX, m);
nc.stroke();
nc.beginPath();
                nc.lineWidth = ns/2;
nc.moveTo(lastX, m);
lastY = value;
nc.lineTo(lastX, m+lastY);
nc.stroke()
            } else {
nc.moveTo(lastX, m+lastY);
if(value < -m) {
value = -m;
                }
if(value > m) {
value = m;
                }
lastX = px;
lastY = value;
nc.lineTo(lastX, m+lastY);
            }
        }
nc.stroke();
iteration++;
    }
}

function redrawGraphs(){
    $('canvas.output').each(function(){
drawGraph($(this).attr('id'));
    })
drawGraph('end-result', true);
}

function clear() {
neurons = [];
neuronConnections = [];

    $('.neurons-map .neuron').remove();
    $('.graphs-list .graph-wrapper').remove();

    n1 = new Neuron({
title: 'N1',
x: 660,
y: 90,
width: 60,
height: 60
    });

startYPos = 10;

redrawConnections();
redrawGraphs();
}

$(document).delegate('.formula-config .button-update', 'click', function(){
var nid = $(this).parents('.graph-wrapper').data('nid');
neurons[nid].formula = $(this).siblings('.input-formula').val();
neurons[nid].impulse = $(this).siblings('.input-impulse').val();
if(neurons[nid].fType == 2) {
var fA = neurons[nid].formula.split(',');
neurons[nid].impulses = processImpulses(fA);
```

```
    }
redrawGraphs();
})

$(document).delegate('.formula-config input', 'keypress', function(e){
if(e.keyCode == 13) {
var nid = $(this).parents('.graph-wrapper').data('nid');
neurons[nid].formula = $(this).parents('.input-config').find('.input-formula').val();
neurons[nid].impulse = $(this).parents('.input-config').find('.input-impulse').val();
redrawGraphs();
    }
})

$(document).delegate('.map-actions .button-add', 'click', function(e){
if($('.neuron.input').size() < 32) {
var ix = new Input({
title: 'in',
x: 0,
y: startYPos,
type: 1,
fType: 1,
formula: '0'
        });
startYPos += 50;
createConnection(ix.nid, n1.nid);
redrawGraphs();
    }
})

$(document).delegate('.map-actions .button-clear', 'click', function(e){
clear();
})

function deleteInput(nid) {
delete neurons[nid];
for(var i = 0; i < neuronConnections.length; i++) {
if(neuronConnections[i].from == nid) {
neuronConnections.splice(i, 1);
        }
    }
for(var i = 0; i < inputsArray.length; i++) {
if(inputsArray[i].nid == nid) {
inputsArray.splice(i, 1);
        }
    }
if(n1.from[nid]) {
n1.from.splice(nid+1, 1);
    }
    $('.neuron[data-nid='+nid+']').remove();
    $('.graph-wrapper[data-nid='+nid+']').remove();
recalcPositions();
redrawConnections();
redrawGraphs();
}

$(document).delegate('.additional-actions .delete', 'click', function(e){
    $(this).parents('.popup-window').remove();
e.preventDefault();
var nid = $(this).parents('.input-config').data('nid');
deleteInput(nid);
return false;
});

$(document).delegate('.formula-config .formula-type', 'change', function(e){
var fType = $(this).val();
var nid = $(this).parents('.input-config').data('nid');
if(neurons[nid].fType != fType) {
neurons[nid].fType = fType;
if(fType == 1) {
neurons[nid].formula = 'sin(x/4)*1';
            $(this).parents('.input-config').find('.input-formula').val(neurons[nid].formula);
            $(this).parents('.input-config').find('.input-impulse').remove();
        } else if(fType == 2) {
neurons[nid].formula = '10,20,30';
            $(this).parents('.input-config').find('.input-formula').val(neurons[nid].formula);
            $(this).parents('.input-config').find('.input-formula').after('<input     type="text"
class="input-impulse" value="2">');
        }
    }
```

```
return false;
});

$(document).delegate('.formula-config .formula-dir', 'change', function(e){
var dir = $(this).val();
var nid = $(this).parents('.input-config').data('nid');
if(neurons[nid].dir != dir) {
neurons[nid].dir = dir;
    }
redrawGraphs();
return false;
});

$(document).delegate('#button-save', 'click', function(e){
mainTime = $('.option .main-time').val();
hMax = $('.option .top-limit').val();
redrawGraphs();
redrawConnections();
});

$(document).delegate('.hide-button', 'click', function(e){
    $(this).parents('.graph-wrapper, .map-display').toggleClass('hidden');
});

function processImpulses(fA) {
var newArray = [];
for(var i = 0; i < fA.length; i++) {
if(fA[i].indexOf('-') > -1) {
var pA = fA[i].split('-');
if(pA.length == 2) {
var lPart = parseInt(pA[0], 10);
var rPart = parseInt(pA[1], 10);
if(lPart && rPart) {
for(var k = lPart; k <= rPart; k++) {
newArray.push(k);
                    }
                }
            }
        } else if(fA[i].indexOf('+') > -1) {
var pA = fA[i].split('+');
if(pA.length == 2) {
var lPart = parseInt(pA[0], 10);
var rPart = parseInt(pA[1], 10);
if(lPart && rPart) {
for(var k = lPart; k <= mainTime; k+=rPart) {
newArray.push(k);
                    }
                }
            } else if(pA.length == 3) {
var lPart = parseInt(pA[0], 10);
var mPart = parseInt(pA[1], 10);
var rPart = parseInt(pA[2], 10);
if(lPart && rPart) {
for(var k = lPart; k <= rPart; k+=mPart) {
newArray.push(k);
                    }
                }
            }
        } else {
newArray.push(parseInt(fA[i], 10));
        }
    }
return newArray;
}

function recalcPositions() {
for(var i = 0; i < inputsArray.length; i++) {
var inp = inputsArray[i];
        inp.y = $(inp.element).position().top + 36;
        inp.x = $(inp.element).position().left + 36;
inputsArray[i] = inp;
    }
}

Input = function(params){
var inp = {};
    inp.title = params.title ?params.title : 'ix';
    inp.left = params.x;
    inp.top = params.y;
```

```
    inp.width = 40;
    inp.height = 40;
    inp.x = params.x + (inp.width / 2);
    inp.y = params.y + (inp.height / 2);
    inp.nid = params.nid ?params.nid : nid++;
    inp.formula = params.formula ?params.formula : false;
    inp.fType = params.fType ?params.fType : 1;
    inp.impulse = params.impulse ?params.impulse : 2;
    inp.impulses = params.impulses ?params.impulses : [];
    inp.dir = params.dir ?params.dir : 1;

if(params.nid) {
nid = params.nid;
    }

if(inp.fType == 2) {
var fA = inp.formula.split(',');
        inp.impulses = processImpulses(fA);
    }

    inp.element = null;

    this.nid = inp.nid;
    this.formula = inp.formula;
    this.fType = inp.fType;
    this.dir = inp.dir;
    this.impulse = inp.impulse;

    this._draw = function(){
        $('.neuron-inputs').append('<div class="neuron input" data-nid="' + inp.nid + '">' +
inp.title + '</div>');
        inp.element = $('.neuron[data-nid='+inp.nid+']');
inp.element.css({
            'width': inp.width + 'px',
            'height': inp.height + 'px'
        });
        inp.y = $(inp.element).position().top + 36;
        inp.x = $(inp.element).position().left + 36;
inp.element.on('mousedown', function(e){
if(e.button === 2) {
if(isDrawing) {
if(nDrawFrom == inp.nid) {

                    } else {
                        //createConnection(nDrawFrom, o.nid);
redrawConnections();
                    }
neurons[nDrawFrom].element.removeClass('line-from');
isDrawing =  false;
                } else {
nDrawFrom = inp.nid;
lastMouseX = inp.x;
lastMouseY = inp.y;
inp.element.addClass('line-from');
isDrawing =  true;
                }
            } else if(e.button === 0) {
                //showInputInfo(inp);
            }
        });
inp.element.on('mousedown', function(){

        });
inp.element.hover(function(){
if(isDrawing) {
redrawConnections();
var ctx = mapHelper.getContext("2d");
                ctx.lineWidth = "1";
ctx.beginPath();
ctx.moveTo(lastMouseX, lastMouseY);
ctx.lineTo(inp.x, inp.y);
ctx.stroke();
            }
        });
        $('.graphs-list').append(
            '<div class="graph-wrapper input-config" data-nid="'+inp.nid+'">' +
                '<div class="title">Вхід</div>' +
                '<div class="hide-button"></div>' +
```

```
                '<canvas    class="output"    id="result-nid-'+inp.nid+'"    data-nid="'+inp.nid+'"
height="200" width="990"></canvas>' +
                '<div class="formula-config">' +
                    '<div class="type-selector">' +
                        '<label>Тип сигналу</label><br>' +
                        '<select class="formula-dir">' +
                            '<option value="1">Збуджуючий</option>' +
                            '<option value="2">Гальмівний</option>' +
                        '</select>' +
                        '<select class="formula-type">' +
                            '<option value="1">Аналоговий</option>' +
                            '<option value="2">Імпульсний</option>' +
                        '</select>' +
                    '</div>' +
                    '<label>Значення</label><br>' +
                    '<input type="text" class="input-formula" value="'+inp.formula+'">' +
                    (inp.fType    ==    2    ?    '<input    type="text"    class="input-impulse"
value="'+inp.impulse+'">' : '') +
                    '<button class="button-update">Оновити</button>' +
                    '<div                                        class="additional-actions"><button
class="delete">Видалити</button></div>' +
                '</div>' +
            '</div>');

        $('.input-config[data-nid='+inp.nid+'] .formula-type').val(inp.fType);
        $('.input-config[data-nid='+inp.nid+'] .formula-dir').val(inp.dir);
    };

    this._draw();

neurons[inp.nid] = inp;
inputs[inp.nid] = inp;
inputsArray.push(inp);

return this;
};

Neuron = function(params){
    var n = {};
    n.title = params.title ? params.title : 'Nx';
    n.left = params.x;
    n.top = params.y;
    n.width = 60;
    n.height = 60;
    n.x = params.x + (n.width / 2);
    n.y = params.y + (n.height / 2);
    n.nid = params.id ? params.id : nid++;
    n.from = params.from ? params.from : [];

    if(params.nid) {
        nid = params.nid;
    }

    n.element = null;

    this.nid = n.nid;
    this.from = n.from;

    this._draw = function(){
        $(nMap).append('<div class="neuron" data-nid="' + n.nid + '">' + n.title + '</div>');
        n.element = $('.neuron[data-nid='+n.nid+']');
        n.element.css({
            'top': n.top + 'px',
            'left': n.left + 'px',
            'width': n.width + 'px',
            'height': n.height + 'px'
        })
        n.element.on('mousedown', function(e){
            if(e.button === 2) {
                if(isDrawing) {
                    if(nDrawFrom == n.nid) {

                    } else {
                        createConnection(nDrawFrom, n.nid);
                    }
                    neurons[nDrawFrom].element.removeClass('line-from');
                    isDrawing =  false;
                } else {
                    nDrawFrom = n.nid;
```

```
                    lastMouseX = n.x;
                    lastMouseY = n.y;
                    n.element.addClass('line-from');
                    isDrawing =  true;
                }
            } else if(e.button === 0) {
                showNeuronInfo(n);
            }
        });
        n.element.hover(function(){
            if(isDrawing) {
                redrawConnections();
                var ctx = mapHelper.getContext("2d");
                ctx.lineWidth = "1";
                ctx.beginPath();
                ctx.moveTo(lastMouseX, lastMouseY);
                ctx.lineTo(n.x, n.y);
                ctx.stroke();
            }
        })
    };

    this.process = function(){
        if(this.from.length > 0) {
            for(var i = 0; i < this.from.length; i++) {
                if(neurons[this.from[i]].formula) {
                    value += eval(neurons[this.from[i]].formula);
                }
            }
        }
    };

    this._draw();

    neurons[n.nid] = n;

    return this;
};

inp1 = new Input({
title: 'in',
dir: 1,
fType: 1,
formula: 'sin(x/6)*2'
});

inp2 = new Input({
title: 'in',
dir: 1,
fType: 2,
impulse: 4,
formula: '1,2,3, 10+5+50, 70-90'
});

n1 = new Neuron({
title: 'N1',
x: 660,
y: 90
});

createConnection(inp1.nid, n1.nid);
createConnection(inp2.nid, n1.nid);

redrawGraphs();
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«ПРОГРАМНИЙ МОДУЛЬ ДЛЯ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ СПАЙКІНГОВИХ НЕЙРОНІВ»

Виконав: студент 4 курсу,групи 4КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Малюгов І. М.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф.КН
Паночишин Ю. М.
(прізвище та ініціали)
« 3 » червня 2025 р.

Вінниця ВНТУ - 2025 рік

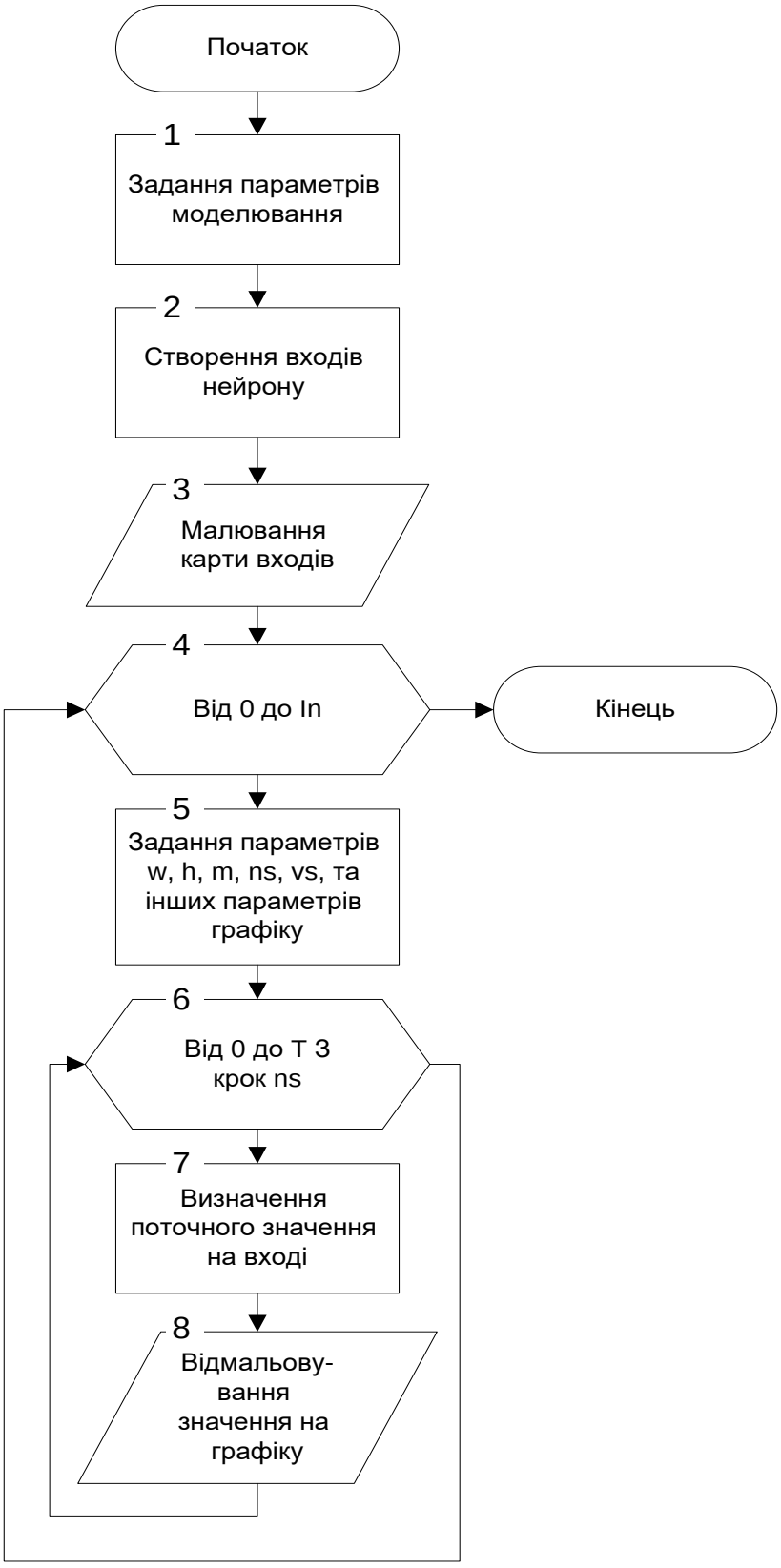


Рисунок В.1– Алгоритм роботи комп’ютерної моделі спайкінгового SRM-нейрона

Таблиця В.1 – Порівняльна характеристика функціонально-обчислювальних властивостей математичних моделей спайкінгових нейронів

№ п/п	Назва моделі	Біологічна правдоподібність (к-ть функцій)	Кількість FLOPs
6.	Інтегрально-імпульсна модель (SRM)	3	5
7.	Квадратична інтегрально-імпульсна модель (QI&F)	7	7
8.	Резонансно-імпульсна модель (R&F)	14	10
9.	Модель Хіндмарша-Розе	21	120
10.	Модель Ходжкіна-Хаксли	22	1200

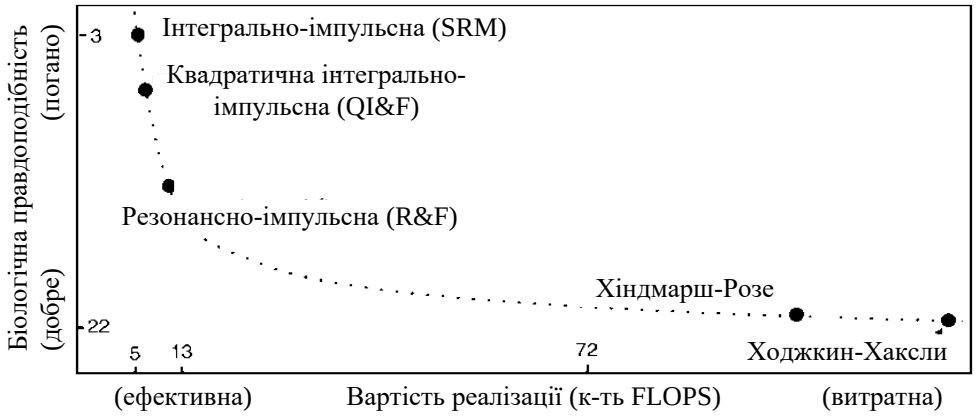


Рисунок В.2 – Порівняльна діаграма для різних видів математичних моделей спайкінгових нейронів

```
▼<div class="main-div">
  ▼<div class="map-display">
     <div class="title">Карта зв'язків</div>
     <div class="hide-button"></div>
    ▼<div class="neurons-map">
      ▼<div class="neuron-inputs">
         <div class="neuron input" data-nid="1" style="width: 40px; height: 40px;">in</div>
         <div class="neuron input" data-nid="2" style="width: 40px; height: 40px;">in</div>
       </div>
       <canvas id="neuron-map-helper" height="300" width="860">
       <div class="neuron" data-nid="3" style="top: 90px; left: 660px; width: 60px; height: 60px;">N1</div>
     </div>
    ▼<div class="control-panel">
     ▶<div class="option">…</div>
     ▶<div class="option">…</div>
     ▶<div class="option">…</div>
     ▶<div class="map-actions option">…</div>
     </div>
   </div>
  ▼<div class="display-panel" style="text-align: left">
    ▼<div class="graphs-list">
      ▼<div class="graph-wrapper input-config" data-nid="1">
         <div class="title">Вхід</div>
         <div class="hide-button"></div>
         <canvas class="output" id="result-nid-1" data-nid="1" height="200" width="990">
       ▶<div class="formula-config">…</div>
       </div>
      ▼<div class="graph-wrapper input-config" data-nid="2">
         <div class="title">Вхід</div>
         <div class="hide-button"></div>
         <canvas class="output" id="result-nid-2" data-nid="2" height="200" width="990">
       ▶<div class="formula-config">…</div>
       </div>
     </div>
    ▼<div class="last-graph graph-wrapper">
       <div class="title">Потенціал мембрани та вихідні імпульси</div>
       <canvas id="end-result" height="200" width="990" class="end-res">
     </div>
   </div>
   <div class="clear"></div>
</div>
<div class="popups-wrapper"></div>
<script src="main.js"></script>
<script src="input.js"></script>
<script src="neuron.js"></script>
<script src="map-init.js"></script>
```

Рисунок В.3 – Структура HTML

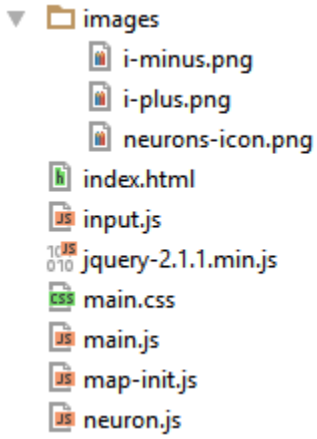


Рисунок В.4 – Структура додатку

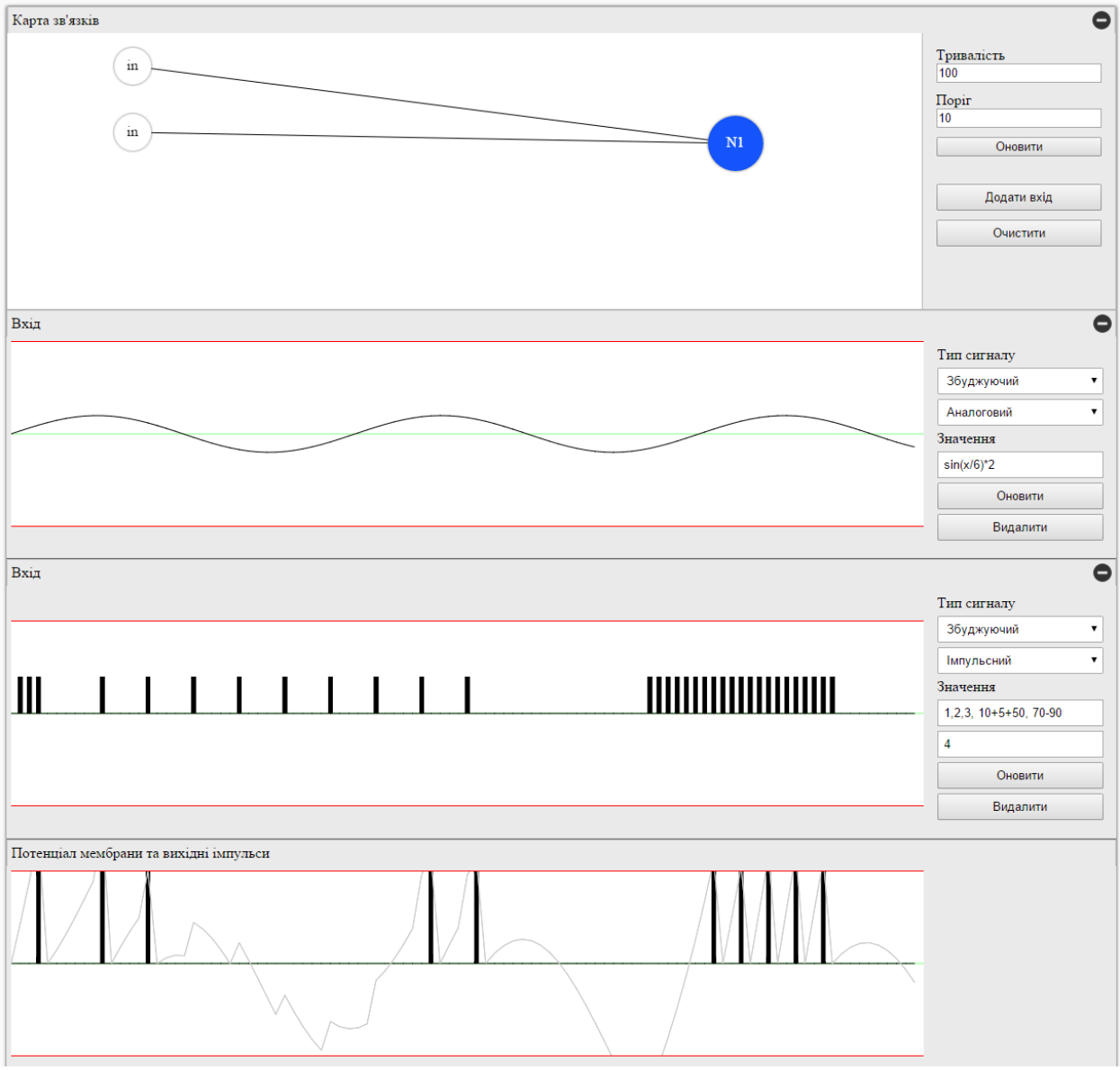


Рисунок В.5 – Результат роботи програмного модуля

Таблиця В.2 - Порівняння функціональних можливостей розробленого програмного модуля та аналога

№ п/п	Назва функціональної можливості	Аналог (GENESIS)	Розроблений модуль
1	Можливість моделювання SRM-нейрона	+	+
2	Можливість задання кількості входів від 0 до 32	+	+
3	Можливість роботи на мобільних пристроях	_	+
4	Можливість відображення мембранного потенціалу та вихідних імпульсів на одному графіку	_	+
5	Можливість автоматично підлаштовувати поле виведення графіка під кількість кроків моделювання	+	+
6	Можливість додавати до аналогового вхідного сигналу випадкову складову	_	+
7	Можливість задавати довільну послідовність імпульсів на імпульсному вході	+	+

Додаток Г (довідниковий)
Інструкція користувача

Для початку роботи з додатком необхідно відкрити його в браузері з локального файлу чи перейшовши за посиланням: http://neurons.redelium.com. Після цього відкриється сторінка з додатком і з ним вже можна буде працювати.

У верхній частині сторінки графічно відображені з'єднання входів та нейрону, а справа від них – налаштування параметрів моделювання (рис. Г.1). Тривалість задає кількість ітерацій для моделювання, а поріг - порогове значення потенціалу мембрани нейрону. Після зміни цих параметрів потрібно натиснути "Оновити", щоб вони збереглися в змінні та для оновлення графіків. Кнопка "Додати вхід" додає один вхід, що за замовчуванням має нульовий сигнал. Кнопка "Очистити" видаляє всі входи та їх графіки, залишаючи на карті лише один нейрон.

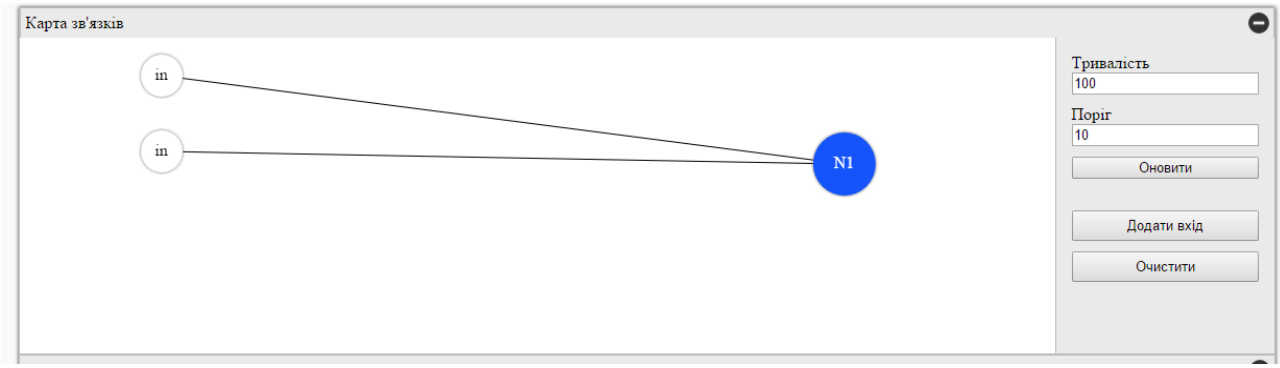


Рисунок Г.1 – Карта нейрону та налаштування моделювання

Нижче знаходяться графіки входів та їх налаштування (рис. Г.2). Сигнал може бути збуджуючий чи гальмівний та аналоговий чи імпульсний. Збуджуючий сигнал додається до потенціалу мембрани, а гальмівний відмінається. Аналоговий сигнал генерується за допомогою формули, що вписана в полі "значення". Для її задання можна використовувати будь-які математичні операції, що доступні в JavaScript. Імпульсний сигнал

генерується з масиву імпульсів, що вводиться в полі "Значення", а під цим полем задається величина імпульсу.

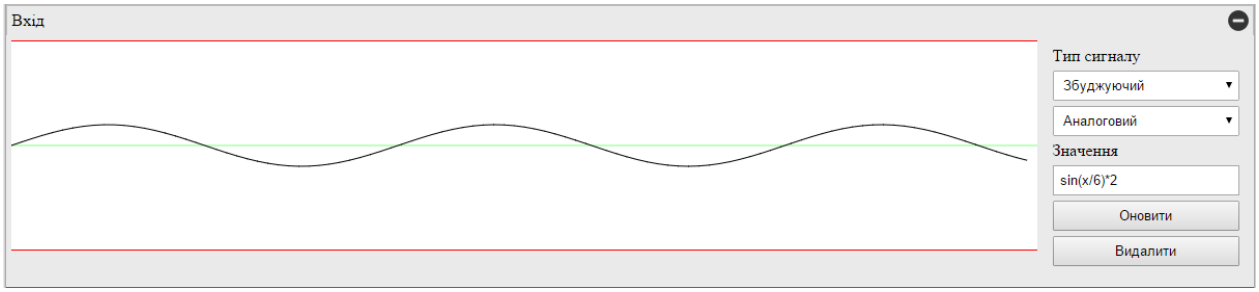


Рисунок Г.2 – Графік та налаштування входу

В самому низу сторінки знаходиться графік, що відображає імпульси на виході нейрону та зміну потенціалу мембрани (рис. Г.3). Потенціал мембрани відображається у вигляді сірої лінії. Коли потенціал досягає порогового значення, то на виході нейрону генерується імпульс, що зображений чорним кольором на графіку.

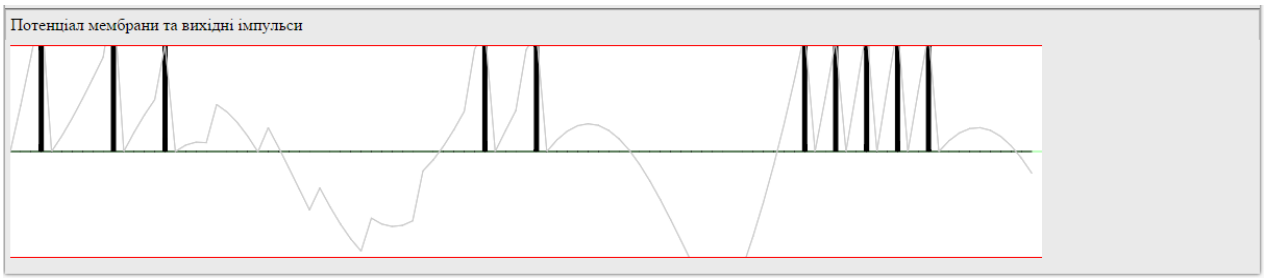


Рисунок Г.3 – Графік потенціалу мембрани та вихідних імпульсів

Входи легко налаштовуються та комбінуються, що дозволяє надзвичайно просто порівнювати сигнал на виході нейрону при різних значеннях вхідних сигналів.