N^4Sim: The first Nervous NaNoNetwork Simulator with Synaptic Molecular Communications

by

Nafi Ahmet Turgut

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Electrical and Electronics Engineering

February, 2022

N^4Sim: The first Nervous NaNoNetwork Simulator with Synaptic Molecular Communications

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Nafi Ahmet Turgut

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Özgür B. Akan

_____

Prof. Fatih Alagöz

_____

Prof. Alper T. Erdoğan

Date: _____

To my parents

ABSTRACT

The unconventional nature of molecular communication necessitates contributions from a host of scientific fields making the simulator design for such systems to be quite challenging. The nervous system is one of the largest and most important nanonetworks of the body. Several molecular and nano communication simulators exist in literature along with a few neural network simulators, however, most existing simulators are neither specific for the nervous system nor they ignore synaptic diffusion because of the computational complexity required to model it. Additionally, information and communication theoretical (ICT) analysis of the system is not directly supported by existing neural network simulators. In this work, we present and describe Neural NaNoNetwork Simulator, N^4Sim, which can resolve the issues in existing simulators. We describe key components of the simulator and methods to solve the synaptic communication in a fast and efficient manner. Our results are comparable to those achieved by Monte Carlo simulations for the synapse while using a fraction of time and processing resources. The presented simulator opens a large set of design options for applications in nervous system.

ÖZETÇE

Moleküler haberleşme (MH), nano makineler arasında kullanım için önerilmiş bir haberleşme teknolojisidir. MH'nin sıra dışı doğası bu alanda çalışanların simulator geliştirmelerini zorlaştırmıştır. İnsan vücudunun nano ölçekteki en büyük ve önemli parçası sinir sistemidir. Literatürde bir takım sinir ağı simulatorleri geliştirilmiş olmakla beraber, hiçbiri sinir sistemleri üzerine odaklanmamış ve hesaplama zorluğu sebebiyle sinaptik difüzyon adres edilmemiştir. Bununla birlikte bilgi ve iletişim teknolojileri metrikleri analizi ve değerlendirmesi desteklenmemiştir. Bu çalışmada, mevcut simulator algoritmalarındaki belirtilen sorunlara çözüm amaçlı olarak Sinir ve Nano Ağ Simulator programı N^4Sim oluşturulmuştur. Bu amaçla simulatorun kilit parçaları ve sinaptik haberleşmeyi modellemenin hızlı ve verimli methodları açıklanmıştır. Kısıtlı işlemci kaynaklarıyla sonuçlarımız, sinaps Monte Carlo simulasyonlarına kıyaslanabilir sonuçlar üretmiştir. Önerilen simulator sinir sistemleri için geniş dizayn seçenekleri sunmaktadır.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude and sincere thanks to my advisor Dr. Özgür Barış Akan for his everlasting guidance, encouragement and tolerance. Without his support, it would be impossible to complete this thesis.

I would like to thank to Dr. Fatih Alagöz and Dr. Alper Tunga Erdoğan for taking part in my thesis committee and for their valuable suggestions and comments.

I want to thank to all members of Next-generation and Wireless Communications Laboratory (NWCL) for their collaboration, constant help and friendship. I specially want to thank Meltem Civaş, Dr. Oktay Çetinkaya and Dr. Bilgesu Arif Bilgin for their valuable comments.

I would also like to acknowledge the support of Koç University, which made the completion of this thesis possible.

I wish to thank my wife Safiye and my family for motivating and encouraging me throughout my study. I would also like to thank Merve Saimler for her invaluable support and friendship.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 N^4Sim: The first Nervous NaNoNetwork Simulator with Synaptic Molecular Communications

With the advent of nanotechnology, nanonetworking remains a special focus since individual nanomachines possess scant resources in terms of memory, processing, size and networking capabilities [3]. Molecular communication (MC) is one of the most promising paradigms proposed for the realization of nanonetworks [4]. This is chiefly because MC is based on natural implements all around and within us. The study of MC in nature provides multi-fold benefits since it can be used in applications for natural systems and can also help in the design of artificial nanonetworks based on natural approaches [5].

The nervous system is composed of networks of neurons connected by means of synapses that can either be electrical or chemical. Synapses function to receive, transmit and process information in the body. Chemical synapses, which occur more frequently in nature, use the transmission of neurotransmitter molecules to transfer information between neurons [6]. Manipulation of information by the nervous system is still not clearly understood and an information and communication technology (ICT) framework is required to evaluate the relationships between various entities of this system [7]. In our previous works, we target the nervous nanonetwork from an ICT perspective to find a number of important parameters such as the channel capacity and network delays from the perspective of single and multiple neurons [8–13]. However, for large-scale nervous nanonetworks, a dedicated simulator is required

that can analyze the system in an efficient and comprehensive manner.

Specialized network simulators are required for the analysis of large networks since the evaluation of closed-form solutions for network parameters using traditional analytical tools is not possible. From a MC perspective, parameters such as particle size, shape, interaction with the surrounding environment and other particles, and timescales of different processes are important for the design of any simulator. Although general purpose molecular dynamics softwares such as Large-scale atomic/molecular massively parallel simulator (LAMMPS) [14], may be used to simulate nanonetworks, they do not to scale up for large networks because of high computational overheads.

Thus, domain-specific simulators are more preferred because they are generally more robust, accurate and efficient. Various research groups around the world are implementing different simulators tailored to their research needs. In this regard, *NanoNS* and *N3Sim* are general purpose simulation frameworks for diffusion-based nanonetworks [15, 16]. *Nano-Sim* is another nanonetwork simulator that specifically targets wireless nanosensor networks [17]. *BiNS* is a Java-based simulation package for MC systems that allows the creation of objects that model the behavior of biological entities [18]. *CalComSim* is a calcium signaling simulator for both synthetic and natural cellular communication [19] and *BNSim* is multithread Java simulator for bacteria-based networks [20]. The discussed simulators cannot be directly used for the nervous system either because they are domain specific to a particular type of communication or their general purpose nature makes it difficult to target the networking problem of nervous communication.

Apart from MC simulators, some programs target the nervous system directly. *NEURON* and *GENESIS* are two of the most prominent neural network simulators in this regard [21], [22] .

NEURON is intended to be a flexible framework for implementing problems where membrane properties are spatially inhomogeneous. *NEURON* is one of the first simulator lies in the literature that successfully implement nervous system. It gets its

respect with well-designed conduction based models(HH,IF). *NEURON* has power graphical user interface (gui), in which user does not required to translate the problem into another domain. It has functions controlling the simulations and displaying the results of neuro-physiological problems. It has a strong computational engine which is particularly efficient due to special methods that uses the advantage of nerve equations. *NEURON* uses two implicit integration methods: Backward Euler and a variant of Crank-Nicholson. The former can be used for large time steps and it has robust numerical stability. The latter is useful for small time steps. These methods improve the performance of *NEURON* by reducing the simulation time. Users have the choice of built-in clock driven and event-driven methods which mainly advances simulation time [21].

The design of *GENESIS* is based on "building-block" approach which makes it possible to add new commands or simulation components without modifying the base code. Its' gui allows users to change simulation parameters in real time so that they observe the results without any change in the simulation time. *GENESIS* libraries contain building blocks which many simulations can be constructed and "cell reader" allows to build model neurons by reading specifications from a data file. It uses Hodgkin-Huxley method to model the nerves [22].

Both of these simulators are scalable from single to multi neuron structures, however, they ignore MC in the synapse and instead use basic circuit models of neuron for simplicity. *NEST* is another important neural network simulator that does target synapses but it is computationally expensive for large networks. *NEST* is a simulation tool which optimized for large scale neural networks. While implementing large network synapse and connectivity are maintained which causes memory consumption and work load dissipated by synapses. It can take advantage of multiple processors and computers in a local area network. *NEST* has different conduction-based models for neurons (HH,IF,STDP). It uses high level scripting language SLI, for user interface. SLI has very simple syntax compared to high level languages [23].

PCSIM is a Python based simulation environment for simulating neural circuits.

It is designed for simulating neural circuits with a support of distributed simulation of large scale neural networks. *PCSIM* uses C++ for computational core code, user interface is written on Python which gives the chance to user to simply integrate neural circuit with visualization tools and data analysis. The main difference of *PCSIM* from other simulators in the literature is that it enables hybrid modeling approach. More computers needed to have linear speed-up in large scale of networks with 10000 neurons or above. PCSIM does not have a significant impact on literature, it only generates new model with Python user interface and hybrid modeling [24]. *Nengo* is a software program that provides a Python simulation framework to build large scale networks based Neural Engineering Framework [25].

NeoCortical Simulator is optimized to model characteristics mammalian neocortex neurons. ASCII based command input file is used and clock based IF neurons with compartments containing HH formulations and conductance based synaptic dynamics. No nonlinear simplifications supported. It does not provide any direct visualization software, report files straight- forward to view in any environment [26]. Brian is simulation environment which mainly targets computational efficiency where other simulators lack [27]. SpikeNET is a basic simulation tool which avoids complexity in neuro-spike communication and useful in different areas as well [28].

This is the general pattern with the neural network simulators available in literature. Most of them ignore the synapse and in cases where it is considered, the networks are not scalable to generate meaningful results. In table 1.1, detailed comparison is available for existing simulators in the literature [23].

The discussion above provides the motivation for a neural network simulator that represents the realistic behavior of MC in the nervous nanonetwork and is efficient to allow simulations of large scale neuronal networks. Therefore, in this work, we present our neural network simulator *Nervous NaNoNetwork Simulator*, N^4Sim, [29] for the first time to mitigate the issues present in the current neural network simulators, and to assist in the ICT analysis of the nervous system. Since synaptic communication is one of the most complex processes in the design of any simulator, we describe an

Table 1.1: Overall Comparison of Simulators

	NEURON	GENESIS	NEST	PCSIM	NCS	BRIAN	**N^{4}Sim**
MC in synapse	NO	NO	NO	NO	NO	NO	**YES**
Conduction-based models	YES	US	YES	US	US	US	**US**
Parallel division architect	NO	NO	YES	YES	YES	YES	**YES**
Distributive computation	US	US	YES	YES	YES	US	**YES**
Clock driven strategies	YES	YES	YES*	NO	YES	YES	**NO**
Event driven strategies	YES	NO	YES*	YES	NO	NO	**YES**
Small Scale Networks	YES	YES	US	US	YES	US	**YES**
Large Scale Networks	NO	NO	YES	YES	YES	YES	**YES**
Multi Compartmental Model	YES	YES	NO	NO	NO	US	**YES**
Diversified Integration Methods	YES	US	YES	US	NO	US	**US**
Powerful Gui Interface	YES	YES	NO	US	US	US	**YES**
Provides BOOK	YES	YES	NO	NO	NO	NO	**NO**
Provides MANUAL	YES	YES	YES	YES	YES	YES	**YES**
Interface Language	Hoc	SLI	SLI	Python	NO	Python	**C++**
SpikeTimingDependentPlasticity	YES*	YES	YES*	YES	YES	US	**YES**
Short-Term Plasticity	YES*	YES*	YES*	YES	YES	US	**YES**
Supports hybrid modelling	YES	NO	NO	YES	NO	NO	**YES**
Direct Evaluation of ICT parameters	NO	NO	NO	NO	NO	NO	**YES**

YES: Succesfull US: Unsatisfactory NO:Fails YES*:No but easy to implement

efficient way to solve this problem and make our simulator scalable. N^4Sim provides an object-oriented framework which allows fast contributions to simulator, and extend the design for various neural networks. Furthermore, with a novel solution which is explained in Chapter 2 for complex synaptic communication N^4Sim is applicable for complex neural network simulations.

The rest of this thesis is organized as follows. Section II discusses the method to efficiently simulate synapses and our plan for its incorporation in the simulator. Section III introduce ICT analysis of nervous network in N^4Sim. Section IV presents the simulator algorithm, platform details, major classes in the simulator and the network types. Section V presents a sample scenario for a small network and also provides some performance analysis. We discuss future directions in Section VI and provide the link for the current release in Section VII. Finally, Section VIII concludes the manuscript.

1.2 Main Contributions

Regarding neural-network simulators of nanoscale communication, main contributions of this thesis are listed as follows.

- First simulator which considers and gives detailed model of MC.
- Describe an efficient and scalable way to solve the design of complex processes of synaptic communication, is able to simulate large neural networks.
- Assists the ICT-based analysis of networks.

1.3 Thesis Outline

This thesis is organized as follows.

Chapter 2: In this chapter, we consider the method to efficiently simulate synapses and our plan for its incorporation in the simulator. We also introduce ICT analysis of nervous network in N^4Sim.

Chapter 3: In this chapter, presents the simulator algorithm in details. The event-driven architecture of N^4Sim is explained, platform details are also presented. Class architecture in the simulator and the network types are presented. We also share a sample scenario for a small network which provides some performance analysis.

Chapter 4: In this chapter, future directions of the simulator are provided and details of current release is shared. Open-source look-up-tables (LUT) will be provided in future, for continuous development of synaptic communication variations. Synaptic plasticity mechanisms can will be applied in the next releases. Parallel processing and GPU based processing will also be considered in the future.

Chapter 2

NEURAL NETWORK MODEL OF N^4SIM

2.1 Introduction

Biological neural networks (BNNs), in a simplified view, are composed of neurons that are connected to each other at synapses, where electrical signals, that initiate from presynaptic neuron's soma, i.e., the cell body, and propagate through its axon, a fine tubular and branching outgrowth from the soma, terminate at the synapse and are transduced into chemical signals by means of secretion of neurotransmitters into the synaptic cleft. The chemical signals are then received by receptors on the postsynpatic membrane at the synaptic cleft, which facilitates ion flow into the post-synaptic neuron, i.e., chemical-to-electrical transduction. At the postsynaptic end, the synapses are most commonly located on the dendrites, tubular outgrowths from soma, and on the soma itself. Thus, the main communication channels that comprise a neuro-spike communication network can be summarized as the axonal, synaptic and somatodendritic channels. In what follows, we explain the neural network model of N^4Sim in terms of these channels.

The rest of the chapter is organized as follows. Section 2.2 describes the equivalent network model for spinal cord motor nucleus. In Section 2.3, bound on the total rate across the network is formulated. In Section 2.4, numerical results are discussed.

2.2 Axonal channel

In accordance with the pioneering simple cable model of axons [30], in N^4Sim axons are primitively modeled by two parameters, its length and conduction velocity, the speed at which an action potential (AP) propagates through the axon. The former

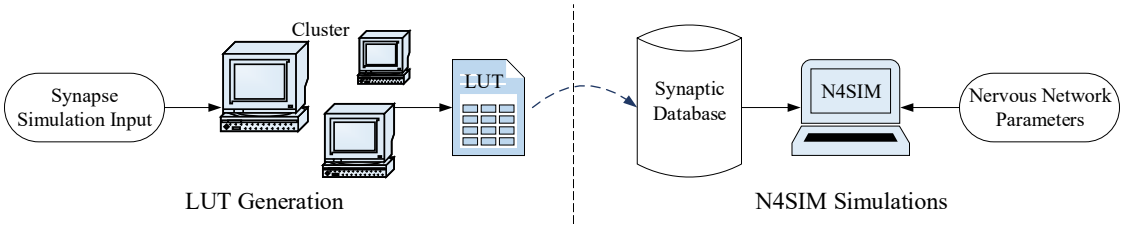


Figure 2.1: Conceptual framework of simulations in N^4Sim.

is determined by consideration of distance between the corresponding neurons, and the average curvature along the axon, which is proportional to the increase of axon length compared to interneuron distance, and is chosen according to experimental estimates in literature. The latter is also experimentally measured for a large variety of neurons, and in N^4Sim, unless otherwise specified, the conduction velocity of an axon is randomly assigned according to measured values. In its initial phase, N^4Sim is also designed very simplistically in terms of geometrical considerations of axons. In particular, between two given neurons there is at most one axon per direction, there is no branching, and each axon terminates with a single synapse.

2.3 Synaptic Channel

As discussed earlier, accurate simulation of MC dynamics in chemical synapses, which involves a diffusion-reaction process, is slow and expensive in terms of computing resources. Even though there have been attempts at developing fast algorithms emulating the expected behavior of MC dynamics at a synapse, [31], [32], these algorithms are still not efficient to be utilized as part of a large-scale network simulator. This renders the task of incorporating MC dynamics at synapses into large scale neural network simulators seemingly impracticable. In N^4Sim, we overcome this problem by employing look-up tables (LUTs), which contain, for varying synaptic variables such as number of receptors at postsynaptic density and neurotransmitters released from the presynaptic neuron, the amount of ion flow into the postsynaptic neuron during a synaptic event. Below, we clarify the incorporation of MC at synapses via

LUT approach, and elaborate on the novel capabilities this introduces to N^4Sim over existing neural network simulators.

2.3.1 LUTs for MC at Synapses in N^4Sim

Simulations that capture MC at synapses consume considerable amount of time and computing resources, which renders the direct integration of them into a large-scale neural network simulator unfeasible, especially as the number of synaptic transmissions during a given network operation are excessive in number. To overcome this obstacle, in N^4Sim we employ LUTs, which, for a given type of synapse, i.e., glutamatergic, gabaergic, etc., contain the amount of charge injected into the postsynaptic neuron for synaptic parameters, i.e., synaptic geometry, diffusion coefficient inside the cleft, number of neurotransmitters released and number of receptors situated at the postsynaptic density (PSD), spanning over the permissible physiological values. To achieve this, the parameter space is discretized into finitely many points, which determines the size of the LUT for the given type of synapse. The synaptic output for any point in the parameter space is then found by linear interpolation or extrapolation using values provided by the LUT. The conceptual framework of N^4Sim simulation system is shown in Fig. 2.1. The entire simulation process is divided in two independent processes.

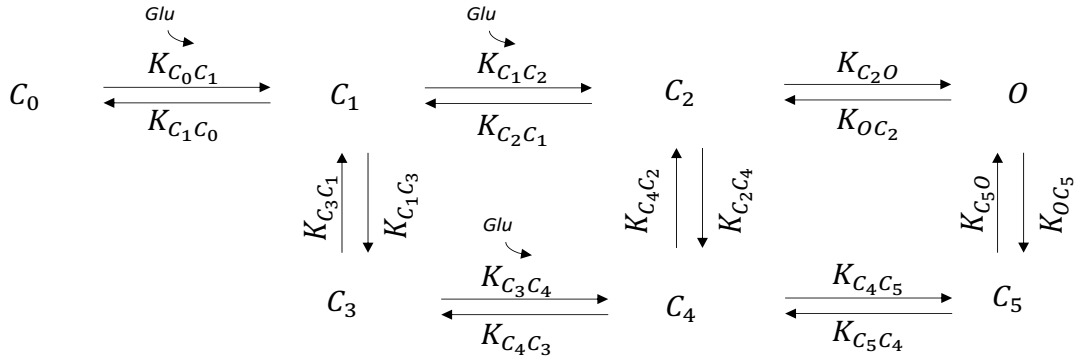
- Detailed simulations of a single synapse are done and look-up tables (LUTs) against each different type of synapse are generated by means of parallel processing. Many of these LUTs then come together to form a master synaptic database. We describe our synaptic implementation in the next subsection.

- Users of N^4Sim can describe the network through our dedicated software. N^4Sim then uses the LUTs based on synapse types and thus, instead of doing complex calculations, the problem is reduced to efficient logical indexing. The detailed design of the simulator is presented in the section III.

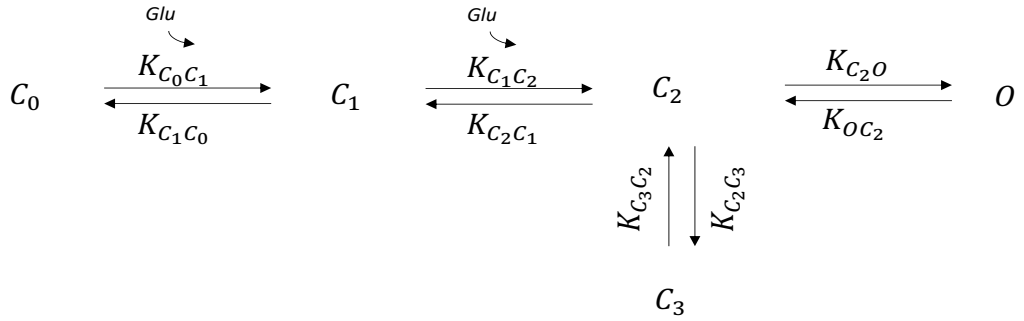
To this end, we have so far created the LUT for glutamatergic synapses only,

for which we have utilized the emulation algorithm provided in [32], since it is very efficient in comparison to standard Monte Carlo simulation approaches found in literature, e.g., see [31]. The algorithm is capable of accurately capturing expected MC dynamics in a synapse with complex neurotransmitter-receptor reaction schemes, such as the one shown in Fig. 2.2 for AMPA and NMDA receptors.

The most important feature of our approach is the fact that the time consuming LUT creation does not need to be repeated. In fact, if the users of N^4Sim contribute the LUTs they generated for particular synapse types, their LUTs can be added to the synaptic database. Thus, as the number of N^4Sim users increases, the number of synapse types stored in the synaptic database of N^4Sim will also increase, and



(a) AMPAR reaction scheme



(b) NMDAR reaction scheme

Figure 2.2: Reaction schemes for the kinetic models of (a) AMPARs [1] and (b) NMDARs [2].

eventually, N^4Sim's synaptic database will become complete.

2.3.2 Advantages of MC in synapses

There are multiple advantages of incorporating MC at synapses into N^4Sim. Apart, from improving the accuracy of N^4Sim, the use of MC at synapses enables the opportunity of exploring molecular mechanisms underlying synaptic plasticity, which is an essential feature of BNNs that governs the rules of learning. The understanding of the molecular basis of mechanisms involved in synaptic plasticity has the potential of assisting in development of novel treatments for various nervous system ailments causing learning difficulties, as well as, providing valuable information on data processing in BNNs, which may serve as a basis in the creation of a universal architecture for neuromorphic computing.

Furthermore, utilizing appropriate boundary conditions for the horizontal boundary of the synaptic cleft, the model provided in [32] is capable of emulating different scenarios such as a clogged synapse, where the dispersion of glutamate molecules out of the cleft is hindered by aggregation of large molecules around the horizontal cleft boundary. Fig. 2.3 illustrates the difference in responses of the normal and clogged glutamatergic synapses to 100 Hz impulse train stimulus of 10 pulses length. The accumulation of glutamate molecules in the cleft due to clogging severely limits the frequency response of the channel, and causes high direct currents to flow into the postsynaptic neuron, which may cause the death of the neuron due to excitotoxicity. Applicability of such scenarios may permit N^4Sim to predict some of the nervous system disorders such as Alzheimer's disease, where clogging of the synapse can be attributed to aggregation of beta-amyloid plaques [33].

2.4 ICT Analysis of Neural Networks in N^4Sim

2.4.1 The Contribution Metric

As explained in Section II, in general, a spike at a given neuron is triggered by the contribution of other neurons' activities to the membrane voltage at the axonal hillock of that neuron. One can argue that, $c(s, r)$, the contribution of incoming spikes, s, to the generated spike, r, can be taken as the ratio of the corresponding voltage contribution, at the spiking instant, to the total voltage contribution that is necessary to generate a spike from resting state of that neuron, i.e., the difference between the resting and threshold voltages of that neuron. By "at the spiking" instant

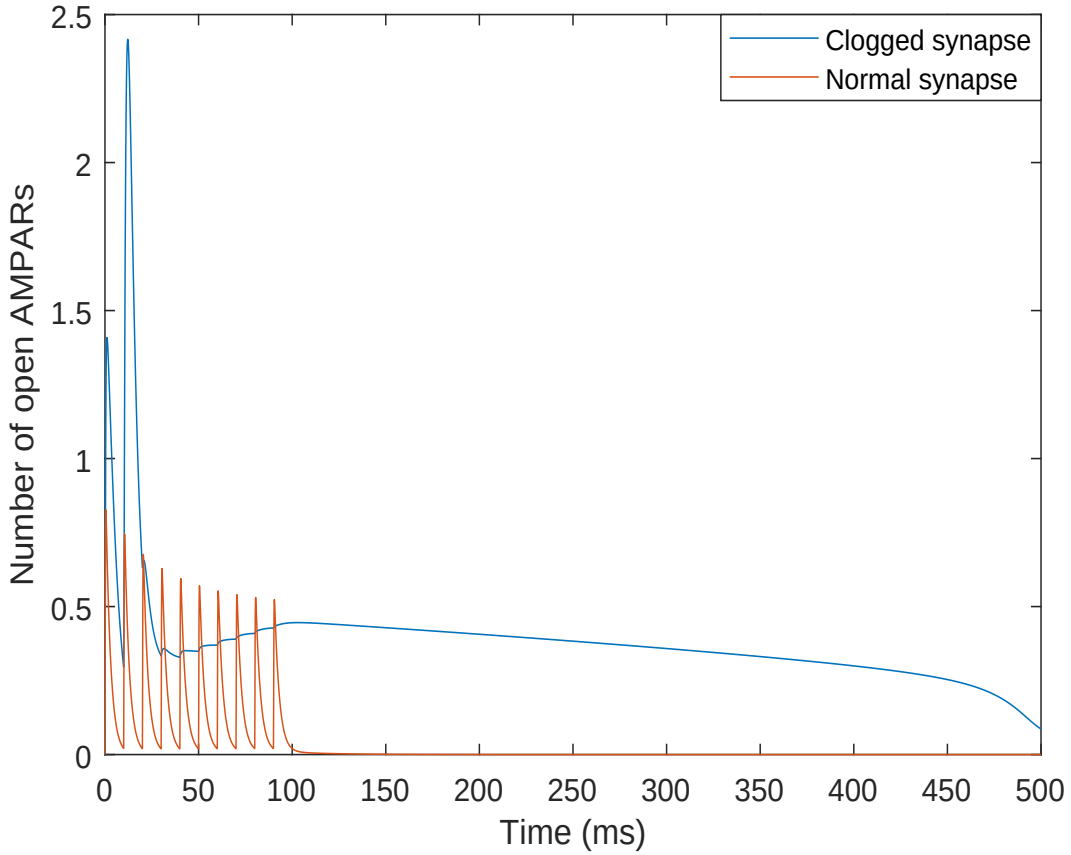


Figure 2.3: Emulated responses of normal and clogged glutamatergic synapses to 100 Hz impulse train stimulus of 10 pulses length.

we mean that the contributions of incoming spikes are calculated by also taking into consideration the decay contributions suffer until the spiking of the contributed neuron. In calculating the contribution of the final spike, we only take the partial contribution up to the threshold voltage and consider the rest as wasted. In this way, for every spike generated, except the user-defined input spikes which initiate the network, one can define the contributions of spikes that give rise to it. In this work, we only consider excitatory synapses, which generate excitatory postsynaptic potentials (EPSPs), i.e., positive voltage differences at the membrane of the postsynaptic neuron, and consequently, by the definition above we have that $c(s, r) \in [0, 1]$ and

$$\sum_{s \in G_r} c(s, r) = 1,$$

where G_r is the set of generator or parent spikes of the spike r.

The suitable mathematical object to investigate the contributions between spikes and sets of spikes, or *neural activities*, are directed acyclic graphs (DAGs). DAGs are directed graphs, i.e., a finite set of nodes connected by directed edges, which contain no cycles, that is, starting from any given node and following the directed edges one can never return back to the original node. In our context, the DAG of spikes (DAGoS) has spikes as its nodes, and directed edges between nodes are created according to the relation of contribution just described, where to each edge we attach the numerical value of the contribution. This directed graph is obviously acyclic, as the method of placement of directed edges, dictated by a causal phenomenon, inherently imply temporal ordering between the corresponding nodes, where the edge is directed in increasing temporal direction. In a DAG, a path π of length $k \geq 1$ is defined as an ordered sequence of directed edges, i.e., $\pi = \{e_1, \cdots, e_k\}$, where any two consecutive edges e_i and e_{i+1} intersect at a node s with e_i directed towards and e_{i+1} directed away from s, a short notation for which is $\overrightarrow{e}_i = \overleftarrow{e}_{i+1} = s$. For convenience, as an equivalent alternative notation for a path of length $k \geq 1$ we will also use $\pi = \{s_1, \cdots, s_{k+1}\}$, where the edges e_i of the previous notation connect s_i to s_{i+1},

$1 \leq i \leq k$. We say a node r is *reachable* by the node s, or s reaches r, if there is a path from s to r.

The given definition of contribution, $c(s, r)$, between two spikes s and r in DAGoS applies only when s and r are directly connected by a single edge directed from s to r. We now choose to extend the definition to apply between any pair of spikes in DAGoS. Given a simulation of duration T, let S and U denote the set of all spikes and user-defined spikes that occur during the simulation, respectively, P denote the set of all paths in the associated DAGoS determined by contribution relations revealed by the simulation. For any $s \in S$, let DAG of contributors (DAGoC) of s be the sub-DAG of DAGoS composed of all spikes that can reach to s and the directed edges along which they reach to s. To start with, we set $c(s, s) = 0$ for all $s \in S$ implying spikes have no self-contribution, and we set $c(s, r) = 0$ if s does not belong to the DAGoC of r, signifying the non-existence of contribution paths. Since user defined spikes are not contributed by any other spike, for any $r \in U$ DAGoC of r is only composed of r itself, and consequently $c(s, r) = 0$ for all $s \in S$. Finally, given any $r \in S \setminus U$, if s belongs to DAGoC of r, then the contribution of s to r is found as

$$c(s, r) := \sum_{\pi \in \P_{s,r}} c_\pi,$$

where $\P_{s,r} \subset \P$ is the set of all paths from s to r, and c_π is the contribution along the path π. For a path $\pi = \{e_1, \cdots, e_k\}$ in $\P$ of length $k \geq 1$, c_π is defined as

$$c_\pi := \prod_{i=1}^{k} c_{e_i},$$

where

$$c_{e_i} := c(\overleftarrow{e}_i, \overrightarrow{e}_i)$$

is the contribution over the edge e_i of its prior node to its posterior. Since contributions over edges are already defined, this concludes the bi-directional definition of contribution between any pair of spikes in DAGoS.

Next, we generalize the notion of contribution to have a meaning between two sets of spikes, i.e., neural activities. Given two neural activities $S, R \subset \S$, let us denote by $\P_{S,R}$ the set of all paths emanating from S to R, that is all paths $\pi \in \P$ with the property that (using alternative notation) $\pi(1) \in S$, $\pi(k+1) \in R$ and $\pi(i) \in \S \setminus S$ for all $i \in \{2, \cdots, k\}$, where $\pi(j)$ stands for the j^{th} entry of $(k+2)$-tuple path π, $j \in \{0, \cdots, k+1\}$. Then, the contribution of spikes from S to R is defined as

$$c(S, R) := \sum_{\pi \in \P_{S,R}} c_\pi,$$

where c_π, the contribution along π, is defined as similar to before. In other words, the set of paths emanating from S to R are those that originate in S never to turn back and finally terminate in R, where multiple visits to R is allowed. The reason for defining contribution between two neural activities over such paths stems form the fact that the contribution over a path that visits multiple elements of S is already accounted for by the contribution over the path that is obtained by trimming it from its beginning so that the new path starts from S and never visits S again. The overlap is due to the fact that, both paths, the one with multiple visits to S and the one that starts from S and never comes back, terminate at the same node in R. That is not the case for the destination set R, where a multiply visiting and its trimmed from end versions of paths do not overlap in their contributions, as they necessarily terminate at different nodes in R.

We say, the neural activity R is independent of the neural activity S if $c(S, R) = 0$, and we say that they are mutually independent if the converse also holds.

Having defined contribution between any pair of neural activities, we can easily define the notion of contribution between activities of neurons and/or groups of neurons. Let N denote the set of all neurons in our neural network, and for each $n \in N$ let $\S_n \subset \S$ denote the set of spikes that belong to n. Then, for any two groups of neurons $M, N \subset N$ we define the contribution of the activity of neurons in M to the

activity of those in N as

$$c(M,N) := c(\bigcup_{n\in M} S_n, \bigcup_{n\in N} S_n).$$

Self-contribution of a neural activity S is defined as $c(S) := c(S,S)$, which captures the amount of cross-talk inside the neural activity S. It is well known that, in BNNs, distributed neural clusters of high activity serve as various functional blocks that perform neural signal processing, which in practice are identified and differentiated from the rest of the network as clusters that maximize cross-talk density. Thus, in a similar spirit, for a given group of neurons N, we define self-contribution density $c_d(N)$ as

$$c_d(N) := \frac{c(N)}{|N|} = \frac{c(N,N)}{|N|},$$

where $|N|$ is the number of neurons in N. In a given network, we call the groups of neurons that locally maximize self-contribution density functional blocks of the network.

As spike-to-spike contributions are continually calculated during an N^4Sim simulation, calculation of this metric brings about no overhead to simulation time. The contribution metric, in a sense, encodes the strength of synaptic connections taking into account resting and threshold potentials, much resembling the synaptic weights and thresholds employed in artificial neural networks (ANNs). However, additionally, as the somatodendritic channel of N^4Sim takes into account the time decay of voltage contributions from old spikes, the contribution metric is also sensitive to fluctuations in spike timing, which makes it all the more interesting and rescues neural network characterization from the static picture of synaptic weights and threshold values of ANNs. Thus, a spike or neuron that is "temporally well-coupled" with others will have more contribution to network activity then a "temporally ill-coupled" one, even if the two share the exact same relation to the network in terms of connectivity and its strength.

2.4.2 Delay

The overall delay in transmission of an AP across a neuron is the sum of delays for several individual processes such as axonal transmission, neurotransmitter release, synaptic diffusion and binding processes, and somatodendritic ion propagation. For neurons with axons that are several orders larger than the synapse, as is the case usually, most of the above described individual delays can be neglected in the favor of axonal delay since it constitutes the largest component of the delay [11]. Hence, for a neuron with an axonal length of L and conduction velocity v, we take the transmission delay as

$$r = \frac{L}{v}. \tag{2.1}$$

For a branch involving multiple neurons connected in a series one after the other, the overall delay is a sum of the delays of various neurons that constitute the branch. For a branch with j neurons, its delay is given as

$$R = \sum_{i=1}^{j} r_i, \tag{2.2}$$

where $r_1, r_2, ..., r_j$ are the individual delays for the branch neurons. Now, in the case of a network of neurons with multiple paths between input and output terminals, the overall network delay, R_N is a function of the delay for each of the individual paths between the two sides, i.e.,

$$R_N = f(R_1, R_2, ..., R_n), \tag{2.3}$$

where $R_1, R_2, ..., R_n$ are the delays for n different branches of the network, and $f()$ is a function that depends on the application of interest. For instance, in cases where an average network delay is required, $f()$ translates as the $mean()$ function. Similarly, $min()$, $max()$ and $median()$ delays may also required by some applications.

Chapter 3

N^4SIM SOFTWARE DESIGN

3.1 Introduction

Having discussed the framework and the implementation of synapses, we now turn towards the software design of N^4Sim. Our main aim is to develop a tool that is most helpful to researchers in understanding the principles of neural communication and synaptic function. To achieve this goal, N^4Sim is envisioned as open-source. The simulator is designed such that it is modular in nature and different modules can work independently, thus, providing an opportunity for parallel processing. We want to ensure that the simulator is multi-scale in time and space such that it works not only on single synapses but it is also possible to conduct large-scale simulations of a small cerebral region or the entire brain. The first step towards these goals is the selection of neural simulation technique.

3.1.1 Simulation algorithm

There are two families of algorithms for the simulation of neural networks: synchronous or 'clock-driven' algorithms, in which all neurons are updated simultaneously at every tick of a clock, and asynchronous or 'event-driven' algorithms, in which neurons are updated only when they receive or emit a spike. The plasticity of a synaptic connection is either event dependent, i.e. the weights increase or decrease as a result of APs from connected neurons, or decay over time if no event occurs. Although, the latter seems to be a continuous process, it is usually modeled by an exponential decay, which implies that the calculation depends only on the time between two events [6]. Therefore, N^4Sim uses an event-driven design because of the reduced computational complexity it offers. We define two types of events: AP event (APE), synaptic trans-

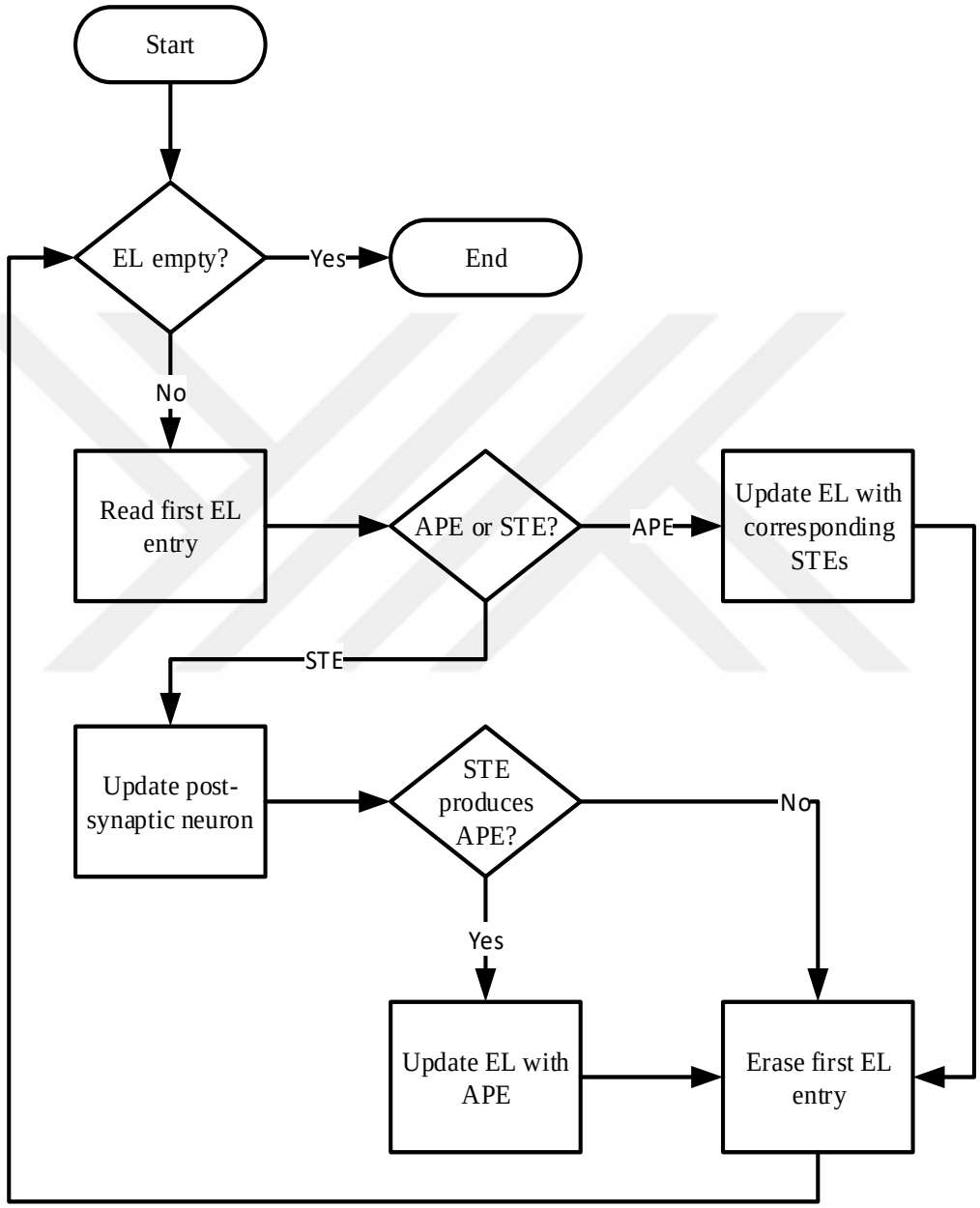


Figure 3.1: Event-driven architecture in N^4Sim.

mission event (STE). The former one happens when an AP is generated at by the soma of a neuron after stimulation from a pre-synaptic terminal. STE can be characterized as the signal transmitted to post synaptic terminals of the output neuron. Thus, whenever APE is produced, it generates some STEs that are then placed in

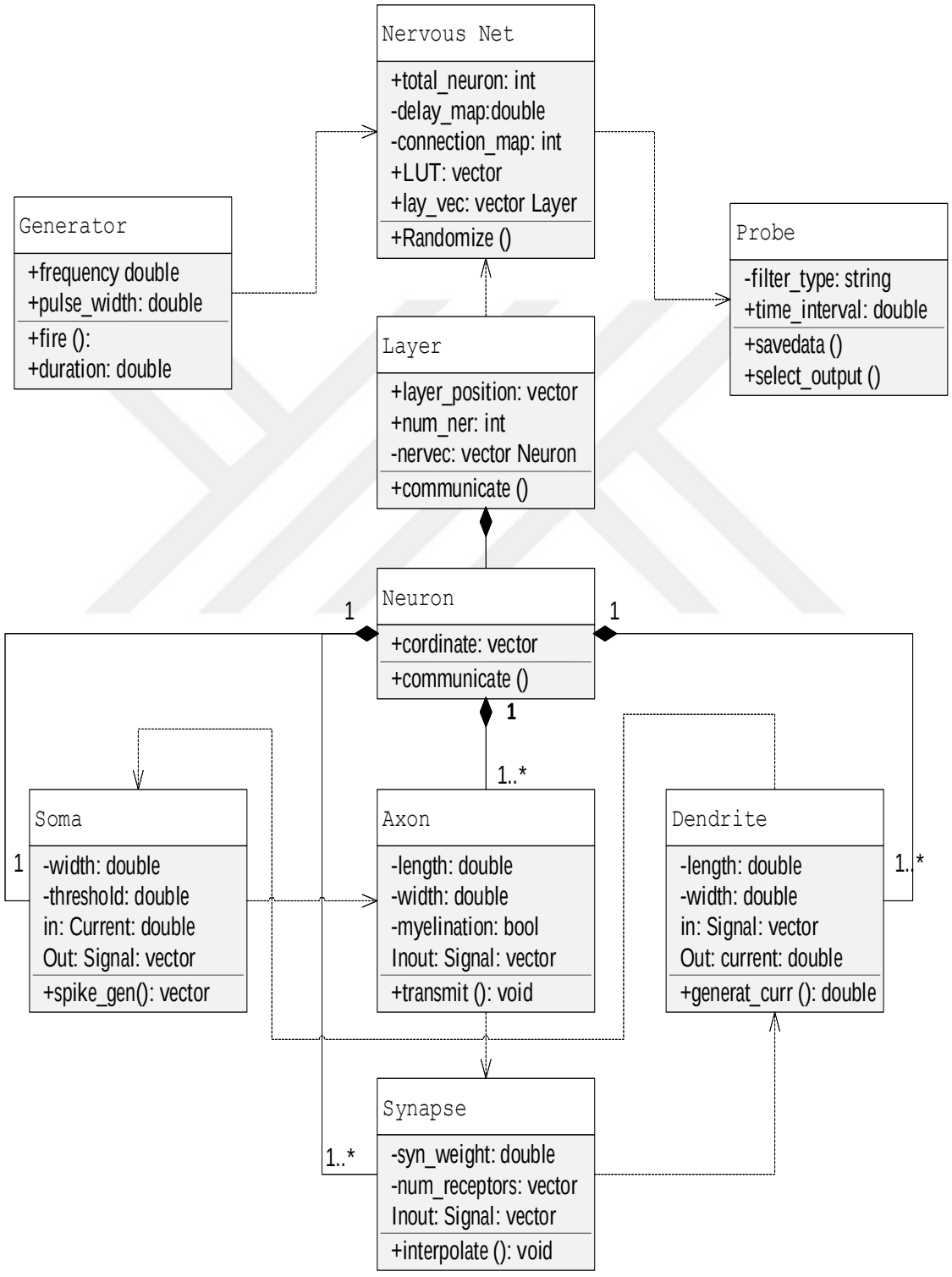


Figure 3.2: Class diagram of a neural network in N^4Sim.

the event log (EL) to record the time and the participants (transmitting and the receiving neuron) of this event. The EL is considered as the logger of the simulator that handles all APE and STE according to their order in time. Whenever the EL is updated by STE, the corresponding neurons are simulated. The receiving neuron in a neural transaction is then checked whether it produces an APE or not. This causes the EL to be updated again in order for processor to handle the next APE or STE. These processes continues, until all APEs and STEs are exhausted. A flowchart for our event-driven architecture is presented in Figure 3.1.

3.1.2 Programming platform

C++ is generally utilized in complicated protocol implementations for simulators due to its fast execution. Therefore, the design of major class definitions and libraries for N^4Sim are written in C++. To make N^4Sim user-friendly and to simplify the conduct of simulations with a multitude of options N^4Sim is provided with a graphical user interface (GUI) as depicted in Figure 3.3, which is also coded in C++. The GUI is also open source and it employs complete error handling to ensure that the software does not hang or give processing errors. Apart from configuration setting, the GUI also provides some graphical windows, where the operation of the network can be observed.

3.1.3 Major classes in N^4Sim

Three main classes of objects are planned for the network design in N^4Sim. These are the `Generator` class objects which is includes user inputs for the network, a `NervousNet` class that includes layers denoted by the `Layer` class in which neurons are defined by the `Neuron` class, and a `Probe` class to export outputs of the system to other formats such as MAT, CSV, TSV, NetCDF, XPORT or plain text. A class diagram of a neural network is shown in Figure 3.2. Using the attributes of frequency, pulse width and input duration, the `Generator` class generates input signals for the network. `NervousNet` class is the most important class in N^4Sim, that determines

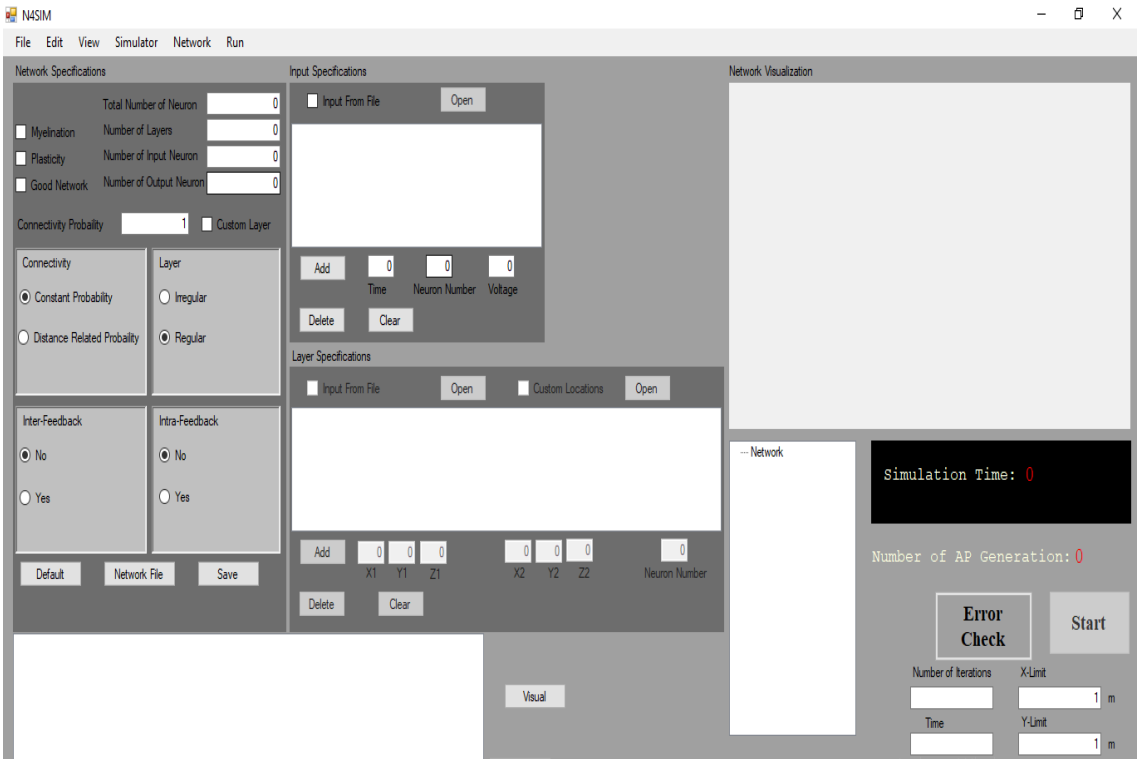


Figure 3.3: A snapshot of the graphical user interface of N^4Sim.

the network topology, which is controlled by connection maps inside `NervousNet` class . Connection maps are matrices that contain a record of all connections across the network and created either by user or by the simulator according to given network parameters. Since it is not possible for users to write connection maps for large scale networks such as the human brain, we envision a `Randomize` function in `NervousNet` class that can generate random network topologies after receiving an input on the number of neurons in the system. Moreover, `NervousNet` class involves `Layer` class denoting the confined regions which neurons are located. `Layer` class is generated in order to categorize neurons which are located in the same region. Every `Layer` object contains a vector of `Neuron` objects, which denotes the neurons inside the corresponding layer.

The `Neuron` class is designed in order to simulate individual neuron nodes. Each `Neuron` can have several associated `Dendrite`, `Axon`, `Synapse`, `Soma`. The `Axon` class is

generated to simulate Axonal transmission and communication delays are calculated in `Axon` class. `Synapse` class is responsible for simulating Molecular Communication inside Synapses and N^4Sim provides look-up tables for simulating Synapses in order to accelerate the speed of the simulator. Thus, `Dendrite` class generates Dendritic currents from the received molecules at the postsynaptic terminals. This potential is summed and sent over to the `Soma` which decides to either fire APs or not based on attributes such as the threshold. `Soma` class is the decision class inside `Neuron` which determines AP generation according to received input signals from `Dendrite` class. The Axonal transmission depends on the geometry of the axon and its myelination.

The `Synapse` class defines the molecular channel of a chemical synapse. In N^4Sim, the complxity of the simulation of molecular communication is reduced by using LUTs. Each `Synapse` class is defined through its number of AMPARs and NMDARs. `Synapse` have `Interpolate` function to get synaptic output from LUTs according to its synaptic properties. The axonal and dendridic communication is dependent on the synaptic communication.

3.1.4 Network Types

N^4Sim uses layer architecture to model complex network models in order to simulate neural networks. A layer is simply the rectangular region of network, in which neurons are placed and is denoted by down-left and up-right corner points of the rectangle. In N^4Sim, layer architecture is either randomly generated or created manually by user. Layer configuration has important role on determining networks as it mainly gives the layout of the network.

Neural networks are designed into two different network categories: custom networks, and random networks. In custom network case, the network layout is given by the user and all the network layers are defined manually. This option allows user to create simulations for specific neural networks and realistic network examples can be implemented by using this option since the user can determine positions of neurons and the types of neurons inside these networks. Random networks are generated by

determining four network options. These options are explained in detail below.

Connection option Connection between two neurons are determined either by a general connection probability or by the distance between neurons. User has to determine which option will be used in the network to determine the connections. Connection probability between neuron is basically Bernoulli process with p defined by user. Distance connectivity is calculated according to Bernoulli process but p is defined by the ratio of the distance between neurons to the maximum distance in the network. Thus, every pair of neurons have different connection probability. Smaller distance results in larger connection probability between corresponding neurons.

Regularly layered option Connection between layers is controlled with this option. Regularly layered connection allows only the connection between layer indexed i to $i + 1$.

Intra-feedback option In our simulator model, neurons are labeled with numbers and connection from larger index to smaller is considered as feedback. Intra-feedback allows the feedback connection inside the layer. Connection between neurons which are located in different layers, is not affected by this control.

Inter-feedback option Inter-feedback is the control which determines the feedback connection between different layers.

Table 3.1: Contributions of spikes of input neurons N1 and N2 to neurons N6 and N7.

	N6 S1	N6 S2	N6 S3	N6 S4	N6 S5	N6	N7 S1	N7 S2	N7 S3	N7 S4	N7 S5	N7	N6 ∪ N7
N1 S1	0,539	0,51	0,243	0,254	0,193	1,739	0,491	0,51	0,214	0,254	0,165	1,634	3,373
N1 S2	0	0	0,487	0,475	0,132	1,094	0	0	0,549	0,474	0,113	1,136	2,23
N1	0,539	0,51	0,73	0,729	0,325	2,833	0,491	0,51	0,763	0,728	0,278	2,77	5,603
N2 S1	0,461	0,49	0,27	0,271	0,214	1,706	0,509	0,49	0,237	0,272	0,183	1,691	3,397
N2 S2	0	0	0	0	0,461	0,461	0	0	0	0	0,539	0,539	1
N2	0,461	0,49	0,27	0,271	0,675	2,167	0,509	0,49	0,237	0,272	0,722	2,23	4,397
N1 ∪ N2	1	1	1	1	1	5	1	1	1	1	1	5	10

Chapter 4

RESULTS

4.1 Introduction

An infinite number of neural networks may be envisioned and analyzed each with a particular description. However, as we present N^4Sim for the first time, we simulate a small sample network that can show the capabilities of N^4Sim. Additionally, we present some results on the simulation performance for various network sizes.

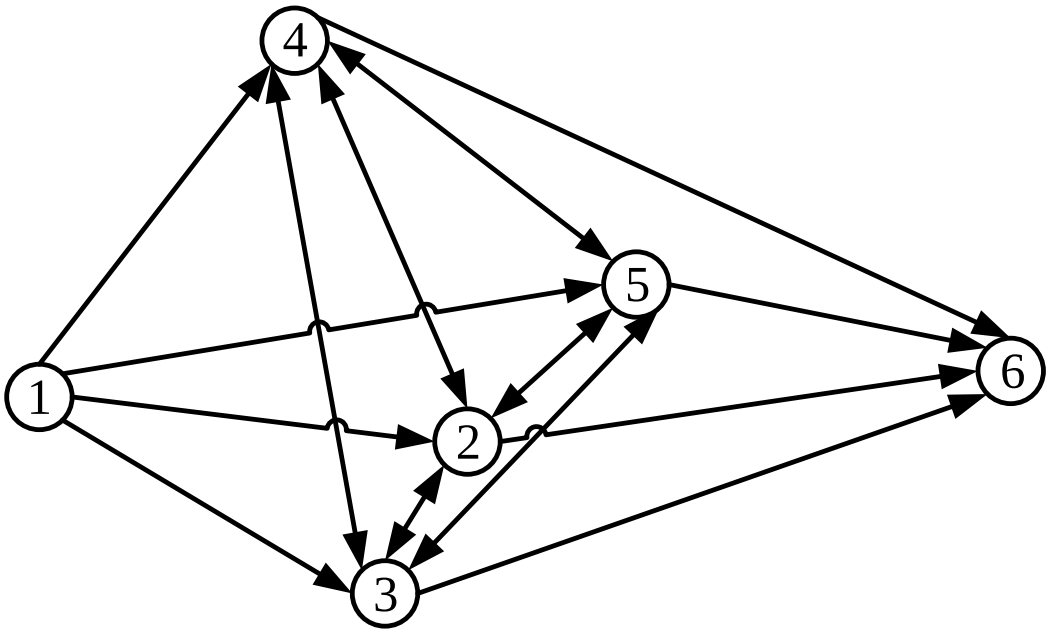


Figure 4.1: Topography of simulated neural network

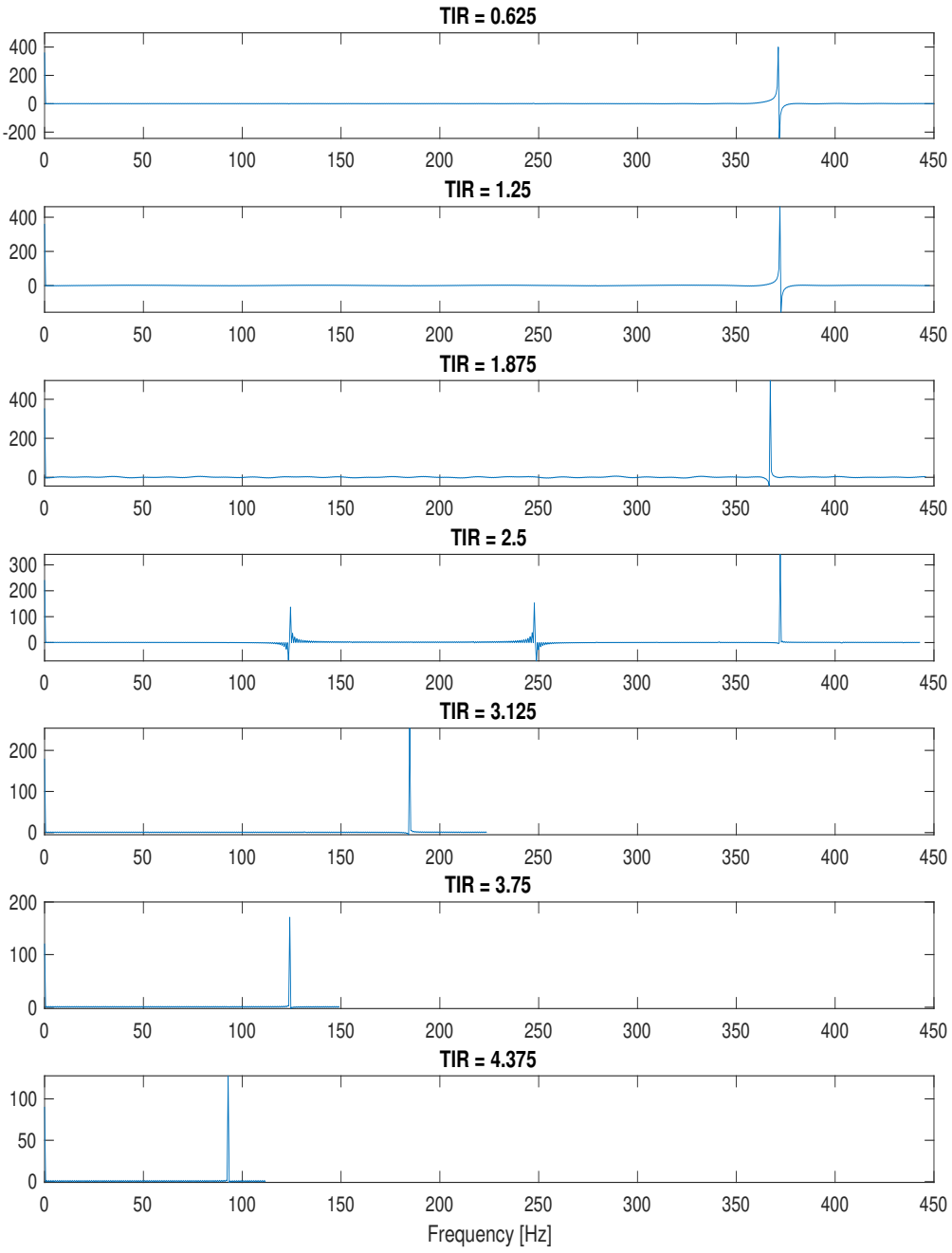


Figure 4.2: Frequency spectra of Neuron 6 activity for various TIRs

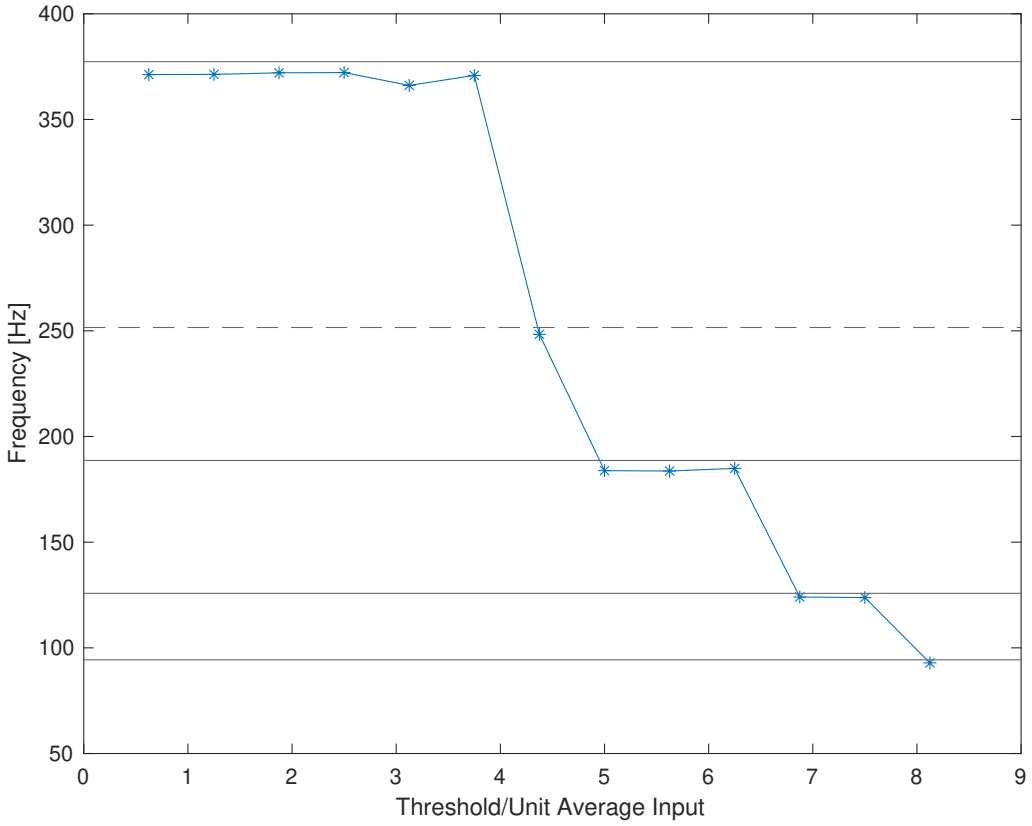


Figure 4.3: The graph output clock frequency dependence on TIR. The solid horizontal lines correspond to maximum possible frequency, $f_{max} \approx 377\ Hz$, imposed by the refractory period set 2.65 ms, its second ($f_{max}/2$), third ($f_{max}/3$), fourth ($f_{max}/4$) harmonics, and the dashed horizontal line to $f = 2f_{max}/3$.

4.1.1 A clock network in N^4Sim

To be able to illustrate N^4Sim in action we have generated an example network, as shown in Figure 4.1. The network is composed of three layers, namely the input layer (Neuron 1), the middle layer (Neuron 2-5) and the output layer (Neuron 6). This is a regularly layered network, i.e., layer I/O relationships are ordered, with no inter-layer feedback and its middle layer has intra-layer feedback. Moreover, this network is fully connected, that is, it contains all possible connections except those that are disallowed by above mentioned restrictive properties such as nonexistence of interlayer feedback.

The reason for the selection of this particular network is that, it is one of the simplest networks that generates an infinitely long output for a finite time input. Once the input neuron fires up the neurons of the middle layer, the dense feedback structure of the middle layer gives rise to an unending chatter within the layer. The output neuron simply listens to this conversation and accordingly generates the output of the network. Figure 4.2 shows the frequency spectrum of the output neuron for varying ratios of the voltage difference required to instigate the firing of the output neuron from its resting state to the average voltage difference induced by the an incoming AP, referred to as 'Threshold' and 'Unit Average Input', respectively. For all cases we see that, the output neuron exhibits an almost periodic activity, which is an indication of self-synchronisation of the middle layer neurons as a result of their endless conversation. Such networks can serve as self-organizing local clocks within the various subnetworks of the nervous system [34]. The frequency of the activity expectedly decreases with the increasing ratio, as increase in the ratio signifies the necessity of more input APs to excite the output neuron. As can be seen from Figure 4.3, the maximum frequency the output neuron can oscillate in, f_{max}, is bounded above by the inverse of the refractory period, which corresponds to $f_{max} = (2.65)^{-1} \approx 377\ Hz$ for this experiment. This value is compatible with the reported values of maximal mean neuron firing frequencies ($\sim 340\ Hz$) observed in the human brain [35]. Interestingly, we also observe that, the permissible frequencies that the output neuron may take correspond to the harmonics of f_{max}.

4.1.2 Contribution metric on a single layer network architecture

The contribution metric defined on DAGs, introduced in this thesis to capture functional relationships between two neural activities, possesses a rich mathematical structure. It is appropriate to demonstrate here one of its basic properties, namely the additivity of contributions. We have the following theorem:

Theorem 1. *Let* $\{S_i\}_{1\leq i\leq n}$, $\{R_j\}_{1\leq j\leq m}$ *be any two finite sequences of distinct spikes,*

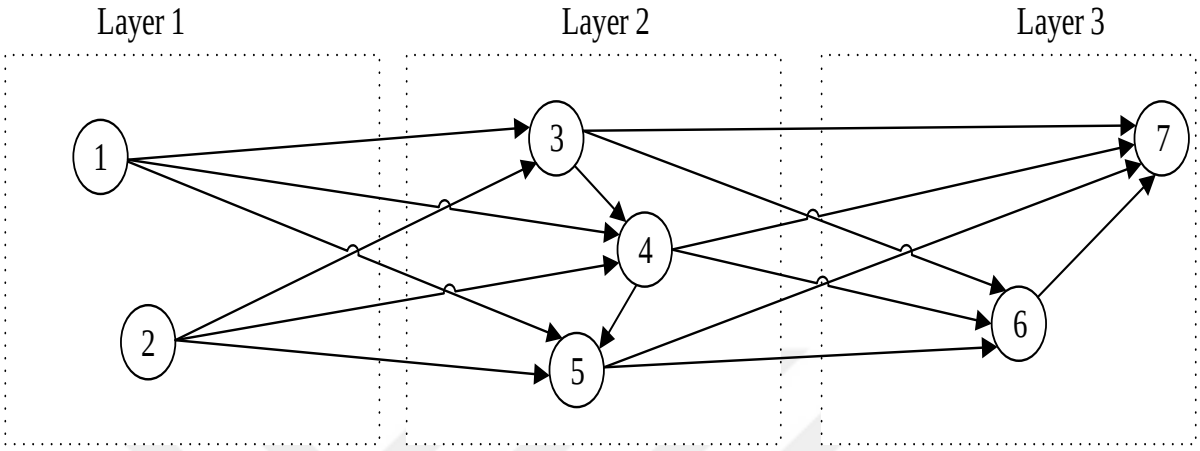


Figure 4.4: Example network connection diagram with 7 Neuron in N^4Sim.

and set $S := \bigcup_{1\leq i\leq n} S_i$, $R := \bigcup_{1\leq j\leq m} R_j$. *Then,*

$$c(S,R) = \sum_{1\leq j\leq m} c(S,R_j),$$

and

$$c(S,R) \leq \sum_{1\leq i\leq n} c(S_i,R).$$

Suppose further that $\{S_i\}_{1\leq i\leq n}$ *are mutually independent. Then,*

$$c(S,R) = \sum_{1\leq i\leq n} c(S_i,R).$$

Proof. Since $R = \bigcup_{1\leq j\leq m} R_j$, and $\{R_j\}_{1\leq j\leq m}$ are mutually disjoint, any path that emanates from S to R, i.e., $\pi \in \mathcal{P}_{S,R}$, has to terminate at some unique R_j. Thus,

$$c(S,R) = \sum_{\pi\in\mathcal{P}_{S,R}} c_\pi = \sum_{1\leq j\leq m} \sum_{\pi\in\mathcal{P}_{S,R_j}} c_\pi = \sum_{1\leq j\leq m} c(S,R_j),$$

which is the first equality in the theorem. The inequality follows from the fact that any path that emanates from S to R must emanate from S_i for some $1 \leq i \leq n$. Finally, the strict inequality occurs only when there is some $1 \leq i \leq n$ such that,

some path that emanates from S_i intersects S_j for some $1 \leq j \neq i \leq n$. Now, if $\{S_i\}_{1 \leq i \leq n}$ are mutually independent, i.e., $c(S_i, S_j) = 0$ for all $1 \leq i, j \leq n$ with $i \neq j$, then, for any given $i \neq j$ there is no path of contribution from S_i to S_j, and hence, we have equality. □

Figure 4.4 depicts a small network of two input (labeled 1 and 2) and two output (labeled 6 and 7) neurons connected by a layer of three interneurons. The input and output layers have no internal feedback, whereas the interneurons are fully connected having maximal internal feedback. In particular, all input neuron spikes are mutually independent. Thus, by Theorem 1 contributions of input neurons to output neurons can be found as the sum of contributions between corresponding spikes, which is illustrated in Table 3.1, where the network was driven by two input spikes from each input neuron, which gave rise to five spikes for both output neurons. As expected, the total contribution of input neurons to output neurons is ten, the total number of spikes produced by the output neurons.

4.1.3 Performance Analysis

Being an event based network simulator, where events are APs of neurons, N^4Sim's execution time is expected to scale linearly with the number of APs, which depends on the number of connections in the network and their strengths. To provide a measure for the speed performance and scaling behavior of N^4Sim, we have simulated regular networks with a single fully connected middle layer, a single input neuron and a single output neuron for 5 seconds of "real" time, where the number of inter-neurons in the middle layer was varied.

As can be seen from Figure 4.5 and 4.6 the execution time of N^4Sim scales linearly, with a slope of 24.4 ms per connection, with the number of connections in the network. The reason for this is that the number of calculation pathways is linearly proportional to the number of connections. Furthermore, for this network, execution time scales quadratically with the number of neurons. This is because, for the networks we consider it holds that, for a network of n inter-neurons, the network contains a total

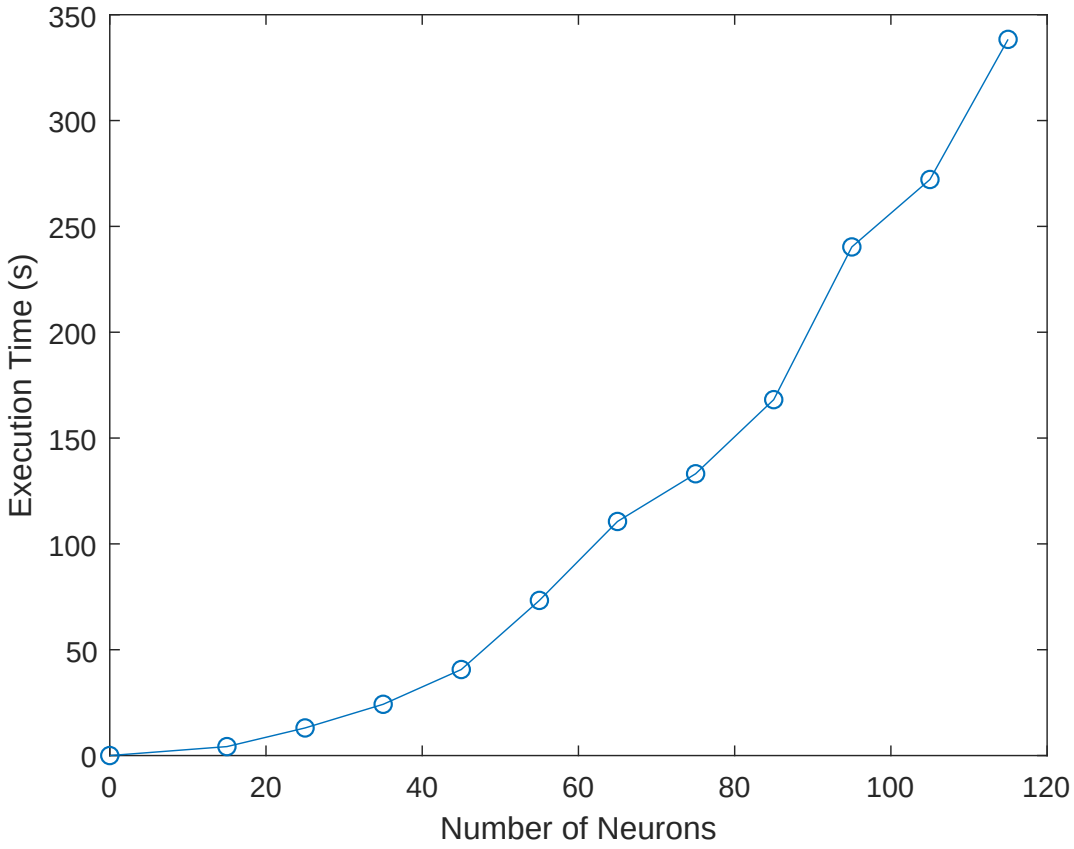


Figure 4.5: N^4Sim execution time of a regular network with 1 fully-connected middle layer, 1 input neuron and 1 output neuron for 5 seconds of "real" time as a function of time vs number of neurons.

of $n(n+1)$ connections, $n(n-1)$ inside middle layer and $2n$ in between consecutive layers. Thus, as the execution time scales linearly with number of connections, it scales quadratically with the number of neurons. For less densely connected networks, however, the execution time of N^4Sim would scale sub-quadratically.

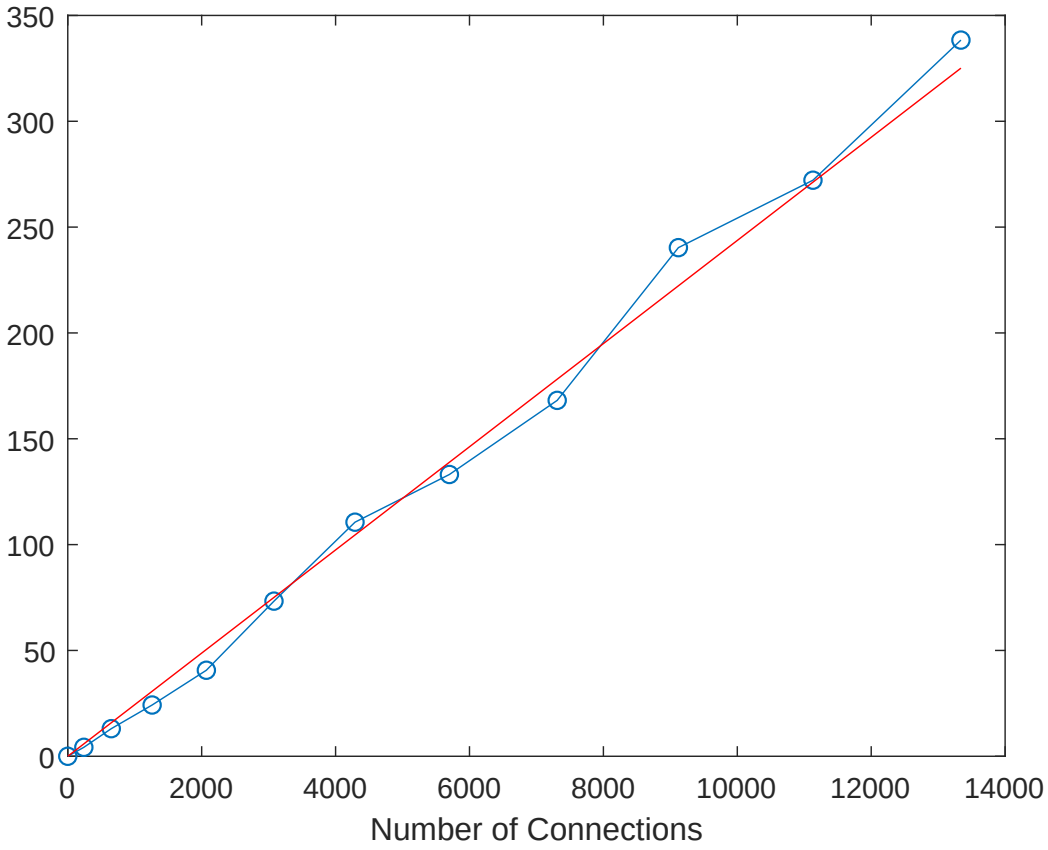


Figure 4.6: Same network in Figure 4.5 Simulation time vs number of connections.

Chapter 5

CONCLUSION AND FUTURE RESEARCH DIRECTIONS

There are numerous directions for future, however, for the sake of brevity, we limit our discussion to the most important future directions and discuss our plans of actions (PoAs) regarding each.

5.1 LUT libraries and user support

The major improvement N^4Sim has over existing neural network simulators is the inclusion of MC dynamics at synapses while still achieving fast and scalable network simulation, which is made possible by the use of LUT for synapses to reduce processing overheads. Since N^4Sim is visioned as an open-source initiative, we expect a large number of users to make contributions to our work and improve it further. In particular, so far, we have created LUTs for glutamatergic synapses utilizing the code developed in [32] for a wide range of synaptic parameters, however the same effort must be extended to other types of chemical synapses corresponding various types of neurons such as GABAergic, cholinergic and adrenergic synapses. The code in [32] is easily adaptable to such scenarios. https://github.com/nafiturgut/N4Sim . Thus, assisted by users, the library for LUTs of synaptic data can grow over time increasng the complexity of scenarios that can be enacted by N^4Sim. In addition to LUT sharing, users can discuss their problems, experiences and solutions to various scenarios in provided user forums. As the open source software grows, newer versions will be periodically released to improve the software and remove any bugs that remain.

5.2 Large time molecular dynamics and plasticity

Current version of N^4Sim is concentrated on incorporating MC in small time scales, where as biological neural networks are also subject to large time scale changes, due both spiking activity, such as depletion of molecular reservoirs and change in receptor population, and metabolic activity, such as replenishment of molecular reservoirs and decay of receptors. By addition of such large time scale dynamics into N^4Sim various synaptic plasticity mechanisms, such as long/short term potentiation or depression and spike-timing-dependent plasticity. Thus, identification of molecular mechanisms behind various observed plastic behavior of neural networks will be possible by comparing N^4Sim simulation results with existing work on plasticity such as [6]. The study of large networks with plasticity may give unique insights into mechanisms responsible for cognitive functions common to biological neural networks such as memory and learning.

5.3 Somatodendritic channel model

The current version of N^4Sim includes detailed models of the synapse and axon, however, a fast and realistic MC model that captures ionic concentration dynamics inside the dendrites and soma still needs to be developed. The existing models such as those developed in [36] and [37] are neuron morphology specific and not fast enough to be scalable to network simulations composed of thousands-millions of neurons. To achieve desired speeds, we are working on a generic model, which includes characterization of possible morphologies in terms of simpler ones, i.e., cylinders and spheres, together with I/O pipelining of these blocks to capture cell wide dynamics. Moreover, a LUT approach, as the one we employed for synapses, where LUTs will be parametrized in terms of both the molecular content, such as ion channel concentration at the membrane that effect calculated dynamics, and morphological parameters that determine the shape of the cell and relative placement of the synapses.

5.4 Axonal Model

In the current version of N^4Sim, we take the simple path of randomly attributing a conduction velocity to an axon according to experimental measurements available in literature. However, in reality, conduction velocity of an axon depends on various physiological properties such as its diameter, density of various ion channels on it, and whether it is myelinated or not. Furthermore, several studies have shown that the axonal propagation is also dependent on frequency of the input signal [38, 39]. These studies show that the input frequency is directly proportional to signal attenuation since an increase in the frequency also causes an increase in the effective internal axonal resistance. To be able to account for the variance in axonal behavior with respect to such parameters, our team is working on a physiologically realistic model that derives conduction velocity through the axon from these parameters via keeping track of molecular dynamics during AP propagation, which we hope to integrate to N^4Sim in future releases. Moreover, in N^4Sim, between two given neurons there is at most one axon per direction, there is no branching, and each axon terminates with a single synapse. However, in nature, axons commonly branch and terminate with multiple synapses. Therefore, another task we are working on is the incorporation of complex axon geometries to N^4Sim.

5.5 Parallel processing

To achieve neural network simulations on scales near to those of the human brain, parallel processing is essential. We aim to use thread-level parallelization (TLP) on the EL. The event-driven architecture will thus disintegrate into a number of tasks that are mutually exclusive, each of which can then be processed separately From a hardware perspective, we aim to support core parallelization as well as graphical processing unit (GPU) processing. CUDA-based solutions will be used for GPU processing.

5.6 Current Release

The current release of N^4Sim is available at http://nwcl.ku.edu.tr/n4sim.html Apart from the codes, the page also provides a GitHub link, where it is possible to download executable versions of N^4Sim. Future releases will be periodically posted on the same page.

5.7 Conclusion

Reviewing the existing simulators for molecular and neural communication in literature, we see that the most of these are based on empirical electrical models or do not fit the scenarios required for study of the ICT foundations of the nervous systems. Therefore, we propose N^4Sim as a novel molecular communication based simulator for the neural nanonetwork, which will be an open source, object oriented, discrete event based simulator. Furthermore, an efficient algorithm to solve the synapse using a deterministic approach is presented in this thesis which will assist in resolving the scalability issues faced by other simulators. N^4Sim will find immense application not only in the study of the nervous system, it may also assist in the development of novel ICT based methods of disease diagnosis and treatment.

BIBLIOGRAPHY

[1] P. Jonas, G. Major, and B. Sakmann, "Quantal components of unitary epscs at the mossy fibre synapse on ca3 pyramidal cells of rat hippocampus." *The Journal of Physiology*, vol. 472, no. 1, pp. 615–663, 1993.

[2] R. Lester and C. E. Jahr, "Nmda channel behavior depends on agonist affinity," *The Journal of neuroscience*, vol. 12, no. 2, pp. 635–643, 1992.

[3] I. F. Akyildiz, F. Brunetti, and C. Blázquez, "Nanonetworks: A new communication paradigm," *Computer Networks*, vol. 52, no. 12, pp. 2260–2279, 2008.

[4] O. B. Akan, H. Ramezani, T. Khan, N. A. Abbasi, and M. Kuscu, "Fundamentals of molecular information and communication science," *Proceedings of the IEEE*, 2016.

[5] D. Malak and O. B. Akan, "Molecular communication nanonetworks inside human body," *Nano Communication Networks*, vol. 3, no. 1, pp. 19–35, 2012.

[6] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[7] D. Malak and O. B. Akan, "Communication theoretical understanding of intra-body nervous nanonetworks," *IEEE Communications Magazine*, vol. 52, no. 4, pp. 129–135, 2014.

[8] E. Balevi and O. B. Akan, "A physical channel model for nanoscale neuro-spike communications," *IEEE Transactions on Communications*, vol. 61, no. 3, pp. 1178–1187, 2013.

[9] D. Malak and O. B. Akan, "A communication theoretical analysis of synaptic multiple-access channel in hippocampal-cortical neurons," *IEEE Transactions on communications*, vol. 61, no. 6, pp. 2457–2467, 2013.

[10] ——, "Synaptic interference channel," in *Com. Workshops (ICC), IEEE Int. Conf.* IEEE, 2013, pp. 771–775.

[11] N. A. Abbasi and O. B. Akan, "A queueing-theoretical delay analysis for intra-body nervous nanonetwork," *Nano Communication Networks*, vol. 6, no. 4, pp. 166–177, 2015.

[12] T. Khan, B. A. Bilgin, and O. B. Akan, "Diffusion-based model for synaptic molecular communication channel," *IEEE transactions on nanobioscience*, vol. 16, no. 4, pp. 299–308, 2017.

[13] N. A. Abbasi, D. Lafci, and O. B. Akan, "Controlled information transfer through an in vivo nervous system," *Scientific reports*, vol. 8, no. 1, p. 2298, 2018.

[14] S. Plimpton, P. Crozier, and A. Thompson, "Lammps-large-scale atomic/molecular massively parallel simulator," *Sandia National Laboratories*, vol. 18, 2007.

[15] E. Gul, B. Atakan, and O. B. Akan, "Nanons: A nanoscale network simulator framework for molecular communications," *Nano Communication Networks*, vol. 1, no. 2, pp. 138–156, 2010.

[16] I. Llatser, I. Pascual, N. Garralda, A. Cabellos-Aparicio, and E. Alarcón, "N3sim: a simulation framework for diffusion-based molecular communication," 2011.

[17] G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Nano-sim: simulating electromagnetic-based nanonetworks in the network simulator 3," in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013, pp. 203–210.

[18] L. Felicetti, M. Femminella, and G. Reali, "A simulation tool for nanoscale biological networks," *Nano Communication Networks*, vol. 3, no. 1, pp. 2–18, 2012.

[19] ——, "Simulation of molecular signaling in blood vessels: Software design and application to atherogenesis," *Nano Communication Networks*, vol. 4, no. 3, pp. 98–119, 2013.

[20] G. Wei, P. Bogdan, and R. Marculescu, "Efficient modeling and simulation of bacteria-based nanonetworks with bnsim," *Selected Areas in Communications, IEEE Journal on*, vol. 31, no. 12, pp. 868–878, 2013.

[21] M. L. Hines and N. T. Carnevale, "The neuron simulation environment," *Neural computation*, vol. 9, no. 6, pp. 1179–1209, 1997.

[22] J. M. Bower and D. Beeman, *The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System.* Springer Science & Business Media, 2012.

[23] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris *et al.*, "Simulation of networks of spiking neurons: a review of tools and strategies," *Journal of computational neuroscience*, vol. 23, no. 3, pp. 349–398, 2007.

[24] D. Pecevski, T. Natschläger, and K. Schuch, "Pcsim: A parallel simulation environment for neural circuits fully integrated with python," *Frontiers in Neuroinformatics*, vol. 3, p. 11, 2009.

[25] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: a python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, 2014.

[26] T. Devyani, "Ncs: Neuron models, user interface, and modeling," *University of Nevada, Reno*, 2014.

[27] D. Goodman and R. Brette, "The brian simulator," *Frontiers in Neuroscience*, vol. 3, 2009.

[28] A. Delorme and S. Thorpe, "Spikenet: An event-driven simulation package for modelling large networks of spiking neurons," *Network (Bristol, England)*, vol. 14, pp. 613–27, 2003.

[29] N. A. Turgut, B. A. Bilgin, and O. B. Akan, "N4sim: The first nervous nanonetwork simulator with synaptic molecular communications," *Transactions on NanoBioscience*, 2021.

[30] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The J. of physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[31] K. M. Franks, T. M. Bartol, and T. J. Sejnowski, "A monte carlo model reveals independent signaling at central glutamatergic synapses," *Biophysical journal*, vol. 83, no. 5, pp. 2333–2348, 2002.

[32] B. A. Bilgin and O. B. Akan, "A fast algorithm for analysis of molecular communication in artificial synapse," *IEEE transactions on nanobioscience*, vol. 16, no. 6, pp. 408–417, 2017.

[33] K. Hensley, J. Carney, M. Mattson, M. Aksenova, M. Harris, J. Wu, R. Floyd, and D. Butterfield, "A model for beta-amyloid aggregation and neurotoxicity based on free radical generation by the peptide: relevance to alzheimer disease." *Proceedings of the National Academy of Sciences*, vol. 91, no. 8, pp. 3270–3274, 1994.

[34] C. Dibner, U. Schibler, and U. Albrecht, "The mammalian circadian timing system: organization and coordination of central and peripheral clocks," *Annual review of physiology*, vol. 72, pp. 517–549, 2010.

[35] B. Wang, W. Ke, J. Guang, G. Chen, L. Yin, S. Deng, Q. He, Y. Liu, T. He, R. Zheng *et al.*, "Firing frequency maxima of fast-spiking neurons in human, monkey, and mouse neocortex," *Frontiers in cellular neuroscience*, vol. 10, p. 239, 2016.

[36] A. Destexhe, "Simplified models of neocortical pyramidal cells preserving somatodendritic voltage attenuation," *Neurocomputing*, vol. 38, pp. 167–173, 2001.

[37] A.-E. Tobin, S. D. Van Hooser, and R. L. Calabrese, "Creation and reduction of a morphologically detailed model of a leech heart interneuron," *Journal of neurophysiology*, vol. 96, no. 4, pp. 2107–2120, 2006.

[38] R. Scott, A. Ruiz, C. Henneberger, D. M. Kullmann, and D. A. Rusakov, "Analog modulation of mossy fiber transmission is uncoupled from changes in presynaptic ca2+," *Journal of Neuroscience*, vol. 28, no. 31, pp. 7765–7773, 2008.

[39] M. Raastad and G. M. Shepherd, "Single-axon action potentials in the rat hippocampal cortex," *The Journal of physiology*, vol. 548, no. 3, pp. 745–752, 2003.