Norwegian University
of Life Sciences

Master's Thesis 2022 30 ECTS
Faculty of Science and Technology

Extending the functionality of the power law-based STDP synaptic model by implementing the support for delay value dominated by axonal propagation

Sanjayan Rengarajan
Master of Science in Data Science

Acknowledgements

I express my profound gratitude to my Supervisor, Susanne Kunkel, for her time and efforts that made this work possible. She has been responsible for conceptualizing this idea, regularly checking upon the work required for its development, till proper completion. I am glad that in spite of her busy schedule, she was always able to find the required time to mentor me. I also acknowledge the use of her drawings in this thesis, which are used with her permission. I am also grateful for her help in reviewing and revising the content of this thesis. I enjoyed all the meetings, discussions, and directions that she has provided throughout the course of this thesis. All the credit for the positive impact this work would entail belongs to her. And if there are any shortcomings, the responsibilities are solely mine.

I would also like to thank my other Supervisor, Prof. Hans Ekkehard Plesser for the valuable insights and suggestions he has provided for this scientific work. His numerous years of experience working in this domain and his sharp observational skills were crucial for the proper completion of this work. I am also grateful to him, for encouraging and providing support for the usage of his LaTeX template, making this writing work easier and well organized.

I extend my gratitude to Jochen Martin Eppler, Håkon Mørk and Stine Brekke Vennemo for their generous help in programming and explaining the procedures involved in software development with NEST Simulator. They were so patient and kind in answering my queries, for which I am truly grateful. Jochen Martin Eppler has various times directly helped in making this project a success, his long years of experience with NEST development were clearly visible with how quickly he could spot and fix the bugs.

I acknowledge the use of Fenix Infrastructure resources, which are partially funded from the European Union's Horizon 2020 research and innovation programme through the ICEI project under the grant agreement No. 800858. I also acknowledge the use of Orion Compute Cluster provided and managed by the IT

department of NMBU. Both these hardware resources have been super helpful in reducing the time taken to develop this project.

I thank my girlfriend, Wietske Stel, who has provided me constant motivation and gave her generous support throughout this arduous journey.

Finally, I thank my close friends and family for supporting me throughout the various maneuvers in my life. They are responsible for me to make it this far. I owe all the success in my life forever to them.

Sanjayan Rengarajan
Ås, December 2022

Abstract

Learning is conceptualized to be possible due to the reconfiguration of neural networks in the brain. This ability is referred to as neuroplasticity. One of the ways by which plasticity is observed can be explained by plasticity exhibited by the synaptic connections between two neurons. Spike-timing-dependent plasticity (STDP) is a biological process, which has been experimentally observed to produce the effect of synaptic plasticity. The STDP process is based on the precise arrival times of presynaptic and postsynaptic spikes at the synapse. This arrival time can be calculated using the timing of the spike offset by propagation delay to the synapse from the neuron. The STDP process is known to affect the connection strength between the neurons, affecting the amplitude of the spike generated in the postsynaptic neuron.

In this thesis, we worked on the simulation model of synapses exhibiting STDP properties. We used the NEST simulator framework for the implementation. The current state-of-the-art model of the STDP synapse supports only delay values as purely dendritic. We implemented support for axonal propagation delay in the synaptic model, along with support for axonal delay value greater than dendritic delay. We verified the implementation with an independent program for testing. The implementation was then analyzed for its impact in a network with different values for the axonal delay. This was done by taking the histogram of the weights of the STDP synapse model in a many-to-one network configuration which was simulated for hundred biological seconds. The results of the analysis matched theoretical predictions with the weights shifting to a higher value as the axonal delay value was increased. The analysis was also computed after optimizing the alpha parameter of the model, to make sure the network average firing rate is at a biologically comparable level at 10 spikes/s or below. In this result, we can not see any significant shifts in weight as predicted by theory. We also benchmarked the newer algorithm to check if it has any additional computation cost, which showed our implementation did not affect the performance.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

Abbreviation	Expansion
AI	Artificial Intelligence
GUI	Graphical User Interface
LIF	Leaky Integrate-and-Fire
ms	milliseconds
mV	milliVolts
MPI	Message Passing Interface
pA	picoAmpere
SLI	Simulation Language Interpreter
STDP	Spike-Timing-Dependent Plasticity
VM	Virtual Machine
WSL	Windows Subsystem for Linux

Chapter 1

Introduction

All the achievements that distinguish the human species from the rest of at least over 5.3 million species (Costello et al. 2013), can be attributed to the single most complex organ in the body, which is the human brain (Freeland 2014). The human brain is not just responsible for our intelligence, but as the saying goes with great power, the brain also has great responsibility and control of our voluntary and involuntary actions (Garrett and Hough 2017). Its powers are so omnipresent and ubiquitous, that the very moment you are reading this, assuming you are not a sentient AI, your vision, muscular movement, the voice in the head doing the reading, breathing, balance, posture, heart rate, and much more are administered by physical and biological processes taking place in different parts of your brain (Arslan et al. 2018). This unique specialty of the organ has piqued a great interest among researchers who study its functionalities among a wide range of organisms.

Philosophically, it is an age-old unanswered question that enquires if the brain is the organ responsible for all intelligence, then is it possible for it to completely understand itself? There are also other neurophilosophical questions on how much understanding we need to reach to attain the state of "complete understanding" (Craver 2007). But this and many other deeply thought-provoking philosophical questions have not deterred neuroscientists from steadily uncovering bit by bit more about the working of the brain (Gross 2012; Jerison 2012).

Owing to the complexity of this domain, the research required specialized focus with collaboration among multidisciplinary researchers. In 2013, European Union, initiated its flagship research project, the **Human Brain Project** (Markram 2012), with the main goal to gain far extensive knowledge about the brain and the domains surrounding it. This scientific work comes under the purview of the

Human Brain Project.

The brain is not only fascinating for the scientists trying to uncover its working logic, but also for the engineers and scientists from other fields who can learn by getting inspired by its working process. This has been now most sought out in the field of Artificial Intelligence (AI) where brain-inspired technologies are used directly to make computers perform tasks that were traditionally thought that only a human could do it satisfactorily (Y. Zhang et al. 2020). Recent trends due to an increase in computational power have increased the potential of AI dramatically.

For many brain-inspired technologies, the ability of the brain to learn and adapt to new information is the most important property. There are many different ways the brain achieves learning, but the one we sought here is its neuroplasticity. Neuroplasticity refers to the ability of neural connections present in the brain to grow and restructure their connections patterns based on new information presented to them in the form of electrical signals. For the brain to exhibit neuroplasticity, there exists some process in the brain which can modify their behavior based on different external conditions [1]. One such phenomenon that has been experimentally verified is spike-timing-dependent plasticity or STDP, which as the name suggests imparts plasticity in the system depending on the timing of the spikes. A neuronal spike is an electrical signal produced by a neuron when going through a complex chemical process, as the membrane voltage of the neuron changes with respect to time. The STDP phenomenon explains the change in the strength of the connection between two neurons connected through the synapse. A synapse is a specialized anatomical structure present between two neurons that facilitates the transmission of the electrical signal from one neuron to the other.

The weight or strength of the synapse refers to how strongly the signals are transmitted across the synapse. A positive weight indicates a stronger connection, a negative weight indicates the synapse acts as more inhibitory making the postsynaptic neuron less likely to produce a spike when the presynaptic neuron produces a spike. Since the weight of the synapse is sensitive to when the synapse receives the signal from both the neurons connected to it, a delay in either direction of the signal reaching the synapse can significantly change the weight of the synapse (Markram et al. 1997).

The current model of STDP synapse based on power law, available in the NEST Simulator lacks a functional property of supporting axonal propagation delay

[1] External here refers to the external to the structure, but not external to the organism itself. Although those conditions can be greatly affected due to changes in outside conditions perceived by the sensory systems of the organism.

greater than dendritic propagation delay. Axonal delay is a property defined by the delay in propagation of a spike from the presynaptic neuron to that of the synapse. This property of axonal delay makes the model biologically relevant in cases where different types of neurons are connected with each other. This type can be observed in the human brain (Madadi Asl et al. 2017). So, the goal of this thesis is to implement the support for axonal propagation delay higher than the dendritic propagation delay for the STDP-based synapse model. Along with the implementation, the model is also tested, and analyzed for its effect on a network and benchmarked for its performance

Following this section, an overview of the biological signal transmission process and the details of modeling neurons and synapses can be found in the next chapter. Along with this, the current state-of-the-art synaptic model's algorithm and the theoretical effects of longer axonal delay in a network model are provided in the theory section. The methodology that we proposed and implemented in this thesis, and the details of the testing and benchmarking of the implementation are provided in the methodology chapter. The results of the analysis and the benchmarks of the implementation can be found in the results chapter. Finally, the limitations of our and possible future enhancements are discussed in the discussion session along with the conclusion. The appendix contains supplementary material for the thesis covering details of the NEST-Simulator and the parameters used in this thesis, along with some code snippets.

Chapter 2

Theoretical Background

2.1 A brief explanation of biological signal transmission

The neuron is responsible for carrying information from different parts of the organism. This information carried by neurons is done in the form of electrochemical signals. Neurons can be in broader terms classified into three types based on the function they perform. *Sensory Neurons* are neurons that are responsible for carrying the information from sensory inputs of the body. The neurons which carry tactile senses present in the skin; audio signals from the hair cells present inside the ear; optical signals from the retina in the eye and olfactory sensors carrying signals about the smell from the nose; are some such examples belonging to this category. *Motor Neurons* are the neurons responsible for the motor action of the body. That is these neurons carry information to the skeletomuscular system and mostly act on muscles to aid in the movement and/or locomotion of the organism. *Interneurons* are neurons that are present in the Central Nervous System. Interneurons are mostly inhibitory in nature and function as a connection between sensory and motor neurons (Purves et al. 2008).

There are three major parts of a neuron as depicted in Figure 2.1. These are the cell body where the cell organelles of the neuron are present. The axon through which the signal transmission takes place to the consecutive neuronal cells and finally the dendrites which receive the signal from the previous neuron through a specialized gap called the synapse. The mechanism by which the electrical signal is generated is in the form of complex electrochemical action. Typically a neuron is present in

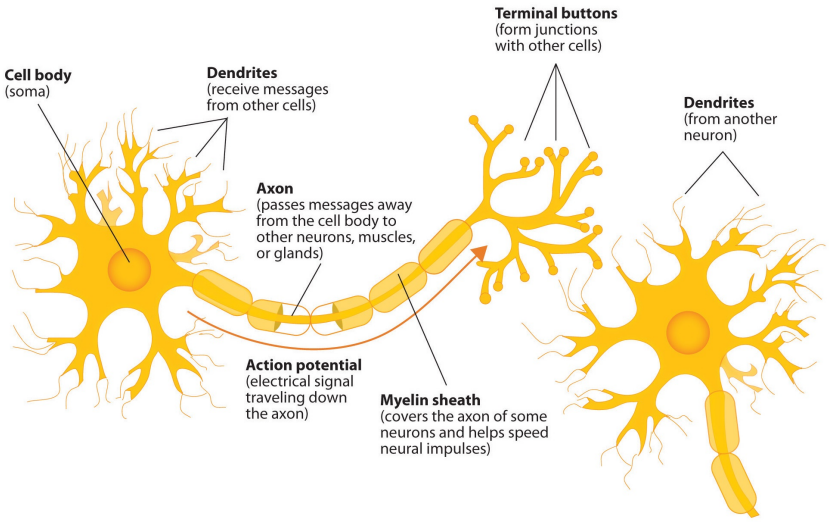


Figure 2.1: An illustration depicting the anatomical structure of the neuron. The electric signals travel from the tip of the dendrite of the left neuron to the terminal buds through the cell body and axon. Here a myelinated sheath is covering the axons, myelin is a protein which makes electrical signals travelling through the axon faster. Adapted from "Diagram of basic neuron and components", by Jennifer Walinga, Oct 2014. CCA 4.0

the polarized state with a voltage of -70 mV. This potential is referred to as the resting membrane potential of the neuron. This is maintained by actively pumping ions across the membrane of the neuron through voltage-regulated channels present in the membrane. These are commonly referred to as voltage-gated ionic channels since they regulate the pumping of ions in and out of the cell via voltage. When a neuron is sufficiently electrically stimulated, it undergoes a rapid transition and it gets depolarized to have a voltage of +40 mV. This change in polarization travels as an electrical signal throughout the length of the axon and is transmitted to the next cell. The way the neuron gets stimulated is referred to as the all-or-none principle (Lucas 1909) in neurophysiology which means that the neuron either gets polarized to the same level when excited or does not get polarized at all. The intensity of polarization is not related to the strength of excitatory signals nor its time period (Purves et al. 2008).

Typically the threshold voltage at which the neuron gets excited is around -55 mV. Once the neuron is depolarized to the peak of +40 mV it immediately goes back to maintain its polarized equilibrium state. The peak at which the neuron gets depolarized is called the action potential. When a neuron's action potential

is generated, it is known by other names such as a spike is generated or the neuron is said to have fired. The generation of spikes at proper time intervals is the key factor in the transmission of information by neurons.

The transmission of signals doesn't completely end with the neuron alone. Once the action potential reaches the terminal buds of the axon, specialized chemicals called neurotransmitters get released from the tip of the neuron. These chemicals act as biological transducers by carrying the information represented in the electrical signals in a chemical form. This all happens in a specialized structure called the synapse, which in simple words is a chemical-based[1] connection present between the neurons. The chemicals then diffuse across the synaptic cleft to bind to the neurotransmitter receptors present in the dendrite of the neuron present after the synapse (post-synaptic neuron). The binding of the neurotransmitter to the postsynaptic neuron opens up various ion channels present in the membrane of the postsynaptic neuron. As more ions enter the cell, the membrane potential changes to produce a spike in the postsynaptic neuron. This is the mechanism through which the spikes are transmitted.

2.2 Modelling of neuronal signals

To further analyze, understand, and even predict the function of a physical system, creating a mathematical model of the system is required. This mathematical model is backed by experimental research and is fitted to the data gathered from experiments to check its validity. Neurons and their signal transmission process are also similarly modeled in many different ways. Some of the models focus on the intricate details of the neuron, while others focus on the overall response properties, providing a higher level of abstraction. Since in this work, our major focus is on the synapse we chose a neuron model with a higher level of abstraction. The leaky integrate-and-fire model is one such model with linear differential equations making them easier to be solved in linear time. The equation describing the model is as given in Equation (2.1) and **??**.

$$C_m \frac{dV_m(t)}{dt} = I(t) - \frac{V_m(t) - E_l}{R_m} \tag{2.1}$$

where,

[1]There are also electrical-based synapses, but they are not within the scope of this study

R_m = Membrane resistance
E_l = Threshold Potential
C_m = Membrane Capacitance
V_m = Membrane Potrential

In this thesis, we use a specifically modified version of the LIF model. The used model has an alpha-function-shaped synaptic current. The synaptic current as the name suggests refers to the current passing in synapse after the generation of a spike in the presynaptic neuron. This model was chosen because of its simplicity to be solved computationally and exhibits complex signal properties (Rotter and Diesmann 1999; Diesmann et al. 2001). Since we only use the neuron as a post-synaptic neuron in this thesis, the postsynaptic current form is of little relevance here.

2.3 Modelling the synaptic connections

Synapse by itself doesn't generate any form of spike. Modeling of the synapse is often dependent on the model of the neuron the synapse gets connected to. The simplest form of modeling a synapse is to have a static synapse. A static synapse doesn't exhibit any type of plasticity. The simplest parameter a static synapse can have is weight. The synaptic weight in a static synapse as the name suggest stays static throughout the simulation. So, the static synaptic models will simply transmit the signal from a presynaptic neuron to the post-synaptic neuron with the only modification being the change in amplitude of the signal based on the weight of the synapse.

There are other types of synaptic models which does not exhibit plasticity but are a bit more complex than a regular static synaptic model. One such example is a stochastic synapse based on Bernoulli distribution. Synaptic models of such kind have a probabilistic transmission factor. The probability of transmission of the signal is determined by the Bernoulli statistic. This kind of signal although a bit more dynamic than the simplest models, they are quite easy to model with the only factor involving the calculation of the probability distribution (Teramae et al. 2012).

For the network of neurons to properly adapt and learn new things, there needs to be some kind of adaptation in the system that makes it robust for learning. So, synaptic models which impart plasticity to the system are of great interest. The

plasticity changes the strength of individual connections, which accumulated in a large network of neurons, enabling the network to strengthen or weaken branches of the connection. Since plasticity is observed experimentally in nature, incorporating the phenomenon into the model follows logically. Our understanding of plasticity is hugely based on the early neuroscientific theory called the Hebbian theory. The simplified version of Hebbian theory can be summarized as "Neurons that fire together, wire together" (Hebb 1949).

2.3.1 Spike-Time Dependant Plastic Synapse

A STDP synapse adjusts the relative strength of the connection between two neurons depending on the spike timings. The increase in connection strength in a STDP synapse only occurs when the spike timing of a presynaptic neuron occurs slightly before the spike timing of the postsynaptic neuron. Otherwise, the presynaptic neuron could not have caused the spike in the postsynaptic neuron as the law of causality would be violated. The time window in which this happens phenomenon holds validity is experimentally found to be in the order of a few milliseconds (Bi and Poo 1998).

STDP synapses have two ways in which the synaptic weights are adjusted. Facilitation is the process in which the synaptic weight or the relative connection strength between the neurons is increased. Depression is the phenomenon when there is a decrease in relative connection strength between the connected neurons. Facilitation and depression are a form of short-term plasticity, and all implementations of STDP synapse undergoes both processes. This interaction between both processes is called the dual-process theory of plasticity. Long-term potentiation (LTP) is an observed process in which the increases in synaptic strength stay persistently long after the cause of increment. Whereas, long-term depression (LTD) is an observed process in which the synaptic strength is persistently low. Together the activity of LTP and LTD are correlated with the learning process. (Sterratt et al. 2011)

The various implementations of STDP often vary in their usage of different mathematical functions used for facilitation and depression.

2.3.2 An overview of NEST-Simulator

The main work of this thesis is done within the NEST-Simulator framework. A simulator framework makes it easier for computational neuroscientists to just work on the modeling or simulation part alone instead of building all the necessary functionalities and optimizations around it. This is achieved by having a higher level abstraction or automatic code generation based on the model description in higher level languages (Blundell et al. 2018).

NEST Simulator is a neural simulation tool capable of phenomenological modeling of neural signals. Phenomenological modeling is a modeling paradigm in which the underlying phenomenon behind the scientific process alone is mathematically modeled. Therefore NEST, at least in its current form, ignores the spatial characteristics of the neuron and only models the phenomenon of activation potential and its propagation. Although, NEST supports the modeling of the spatial arrangement of synapses in a network of neurons.

NEST can be run on a single thread on a single node to massively parallelized supercomputers. NEST framework uses Message Parsing Interface (MPI) and openMP to support its concurrent execution. NEST interface is programmed in C++, with the additional interface accessible through Python, via PyNEST implementation which now comes bundled with the NEST installation. The Python and C++ communication is handled through Cython. NEST works on time-based simulation processing. This means all the simulation in NEST takes place in specified time steps and the communication, transmission, and reception of spikes in NEST are forced to occur in that time step. This process is contrasting with event-driven simulation technology where the process of communication happens when a specific event [2] occurs in the simulation. The advantage of time driven simulator is it forces all the neural nodes in the system to be easily monitored coherently when utilizing parallel computation (Spreizer et al. 2022). Although NEST also offers precise spike timing neural models, currently it doesn't support synaptic plasticity for such models. For a bit more depth detailed discussion about NEST-Simulator functionalities refer to Appendix A: Detailed overview of NEST objects or its official documentation site [3]

[2] For example, creation of action potential

[3] https://nest-simulator.readthedocs.io/en/v3.3/

2.4 State-of-the-Art synaptic model

The current State-of-the-Art model of the STDP synapse is based on work done by Morrison et al. 2007. The model was developed based on experimental evidence and is designed to work best with the balanced random network. A Balanced random network is a type of neural network is a network of neurons in which the total excitatory and inhibitory signals in the network are balanced with each other. And the connection of neurons with each other is set up in a random order without any preconceived shape for the connection. A balanced random network is considered for the network model, as it closely resembles the network activity dynamics present in the cortical region of the brain (Brunel and Hakim 1999; Brunel and X.-J. Wang 2003).

The working of algorithm is split into four different functions and are as given below (2.1 - 2.4), along with their explanations. The algorithms are taken from the appendix section of (Morrison et al. 2007). There are certain assumptions made by the model, for the algorithm to properly work. They are:

- The list of all neurons is available to access from the memory
- The list of spike events is recorded and also available to be accessed from the memory
- The neuron j is the postsynaptic neuron connected by the synapse to neuron i which is the presynaptic neuron as shown in Figure 2.2
- The weight of the synapse w_{ji} connecting the neurons, is stored in the post-synaptic neuron j
- The delays d_j^{Ax} representing axonal propagation delay and d_j^{Dnd} representing dendritic propagation delay are stored in the neuron j, refer Figure 2.2
- The sum of delay values is the total synaptic delay ($d_j = d_j^{Ax} + d_j^{Dnd}$) and satisfies the condition that the dendritic propagation delay is always greater or equal to the axonal propagation delay $d_j^{Dnd} \geq d_j^{Ax}$
- The target neurons hold the details of the timestamp of the last presynaptic spike $time_{old}$ and the K_+ value which controls the facilitation calculation for weight update
- Each Synapse has a value τ, which represents the cut-off time difference. If the spike times between two neurons, adjusted for the delay are higher than

the difference then weight gets depressed, otherwise, the synaptic weight gets facilitated.

In Algorithm 2.1, the function update_weight is invoked whenever a new spike is created in the presynaptic neuron i along with its timestamp. The function gets the history of time stamps as defined by Algorithm 2.2 within the time period of the previous pre-synaptic spike received by the neuron to the new presynaptic neuron shifted by the delay values. The shifting of delay values is done by adding $d_j^{Ax} - d_j^{Dnd}$ as an offset. This is done to compensate for the spike reaching the synapse from the presynaptic neuron. Then for each spike in the history that was recorded after the previous spike at $time_{old}$ adjusted for delays, the facilitation process occurs. The function calculation for facilitation function $F_+()$, is given in Equation (2.2). Finally, after doing this for every spike in the history, the depression process occurs after getting the relevant k_- value from Algorithm 2.3 and with the depression function $F_+()$ as given in Equation (2.3). Then the values of K_+ and the old timestamp $time_{old}$ are updated.

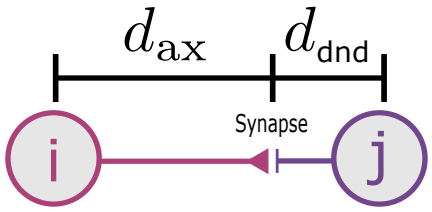


Figure 2.2: The illustration depicts the connection of presynaptic neuron i to that of postsynaptic neuron j, through a synapse in between them, d_{ax} and d_{dnd} represents the propagation delay. Note. Modified from the original illustration provided by Susanne Kunkel, using with permission.

Algorithm 2.1 Synaptic weight update

1: **procedure** `update_weight(time)`
2: **for each** post-synaptic neuron j **do**
3: $history \leftarrow j.get\_history(time_{old} + d_j^{Ax} - d_j^{Dnd}, time + d_j^{Ax} - d_j^{Dnd})$
4: **for each** spike $time_j$ in history **do**
5: $dt \leftarrow (time_j + d_j^{Dnd}) - (time_{old} + d_j^{Ax})$
6: **if** $dt \neq 0$ **then**
7: $w_j \leftarrow w_j + F_+(w_j) \cdot K_+ \cdot e^{\frac{-dt}{\tau}}$
8: $K_- \leftarrow j.get\_K\_value(time + d_i^{Ax} - d_i^{Dnd})$
9: $w_j \leftarrow w_j - F_(w_j) \cdot K_-$
10: transmit spike (w_j, d_j) to neuron j
11: $K_+ \leftarrow K_+ \cdot exponential(-\frac{time - time_{old}}{\tau}) + 1$
12: $time_{old} \leftarrow time$

$$F_+(w) = \lambda \times w^{\mu} \tag{2.2}$$

$$F_{-}(w) = \lambda \times \alpha \times w \tag{2.3}$$

where,

λ = The learning rate of the neuron
α = The asymmetric parameter, representing the depression ratio
μ = The exponentiation parameter for facilitation
F_{+} = Weight Update for facilitation
F_{-} = Weight update for depression
w = Synaptic weight

The values for the parameters were found by matching with the experimental data. The algorithm gets its power law name from the fact that in Equation (2.2) the synaptic weight is raised to the power of the parameter μ. This calculation made the model approximately coincide with the experimental data obtained from Bi and Poo 1998. The change in weights using these functions, with respect to the Δt is plotted in Figure 2.3. The explained algorithm is robust in that the parameterized equations alone simply can be modified for obtaining newer models.

To retrieve the history of spikes, the function defined in Algorithm 2.2 is used. The function is dependent on the maintenance of a special data structure called the spike_register. This spike register is a dynamic data structure containing the values of spike time t_{sp} of the postsynaptic neuron j. This function retrieves the spike between the time period $[t_1, t_2]$ with the inclusion of both extreme times. This function also increases the variable ($counter_{sp}$) which denotes the access counter for the spikes. The returned value of the function is a list of spikes within the parameterized time period.

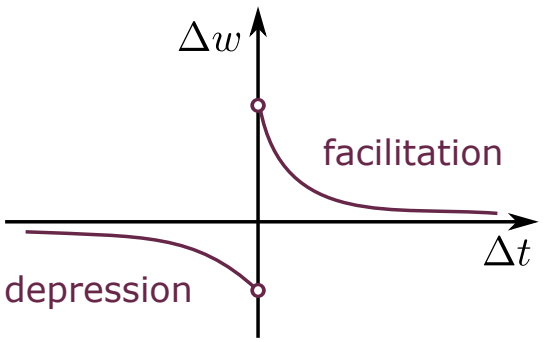


Figure 2.3: The plot depicts the change in weight with respect to the difference of spike timing in power law-based synaptic model. When Δt is 0, there is no change in weight. Note. Modified from the original illustration provided by Susanne Kunkel, using with permission.

The K_{-} value used for depression is calculated by using the previously stored k_{sp} value in the spike_register data structure.

Algorithm 2.2 spike history retrieval

```
1: procedure get_history(t_1, t_2)
2:     iteration from beginning of spike_register
3:     while t_sp ≤ t_1 do
4:         move iterator to next element
5:         Continue
6:     history ← t_sp
7:     counter_sp ← counter_sp + 1
8:     while t_sp ≤ t_2 do
9:         push t_sp to history
10:        counter_sp ← counter_sp + 1
11:        move iterator to next element
12:    return history
```

Since the K_- is used for depression of the weights, in Algorithm 2.3 the condition that the spike generated by postsynaptic neuron j at time t_{sp} should be greater than the current spike obtained from the presynaptic neuron at time t.

Algorithm 2.3 Calculation of K_- value calculation for depression

```
1: procedure get_K_value(t)
2:     reverse iteration from end of spike_register
3:     while  t_sp ≥ t do
4:         move iterator to previous element
5:         Continue
6:     return K_sp · exponential(−(t−t_sp)/tau)
```

In Algorithm 2.4, the spike register data structure is updated with the current spike time t_{sp}, k_{sp} and $counter_{sp}$. The data structure is also maintained to have a separate variable, N_{syn} which holds the value of number of incoming STDP synapse to the neuron. The function is called whenever a spike occurs in the postsynaptic neuron.

The algorithm emphasizes the strict condition of dendritic delay greater than axonal delay. Otherwise, when the get_history function is invoked in 2.1 the law of causality can become violated. This is because the two parameters it is invoked with are $time_{old} + d_j^{Ax} - d_j^{Dnd}$, $time + d_j^{Ax} - d_j^{Dnd}$. As $d_j^{Ax} > d_j^{Dnd}$ the whole delay term becomes positive. this results in the parameters being $time_{old} + \Delta$, $time + \Delta$ where Δ is a positive value denoting the difference between delays. As the upper limit of the function reaches beyond the current time, it requires the postsynaptic

Algorithm 2.4 Updation of the spike register

1: **procedure** `update_register(t)`
2: $K\_ \leftarrow K\_ \cdot exponential(-\frac{t-t_{old}}{tau}) + 1$
3: **while** length of spike_register ≥ 1 **do**
4: **if** counter$_{sp} \geq N_{syn}$ **then**
5: dequeue first element of spike_register
6: push (t, k_, 0) as last element of spike_register
7: $t_{old} \leftarrow t$

neuron to have knowledge of the spike from the future. As this would violate the law of casualty, it should be made sure that the dendritic delay in this algorithm should always be kept below axonal delay.

The algorithm is implemented in the NEST Simulator[4] [5], with some modifications. In the NEST simulator, the current implementation ignores the axonal delay part and considers the delay to be purely dendritic. This makes the only parameter that can be modified in the model the dendritic delay. The algorithm is the same as above by simply removing the dendritic part or equating the axonal delay to zero. Even if the algorithm is modified to accept the axonal delay, simply plugging the value would make the model in the NEST simulator miss a spike in its calculation. The explanation for this is given in the Figure 2.4.

[4]Documentation: https://nest-simulator.readthedocs.io/en/v3.3/models/stdp_pl_synapse_hom.html

[5]Source Code: https://github.com/nest/nest-simulator/blob/master/models/stdp_pl_synapse_hom.h

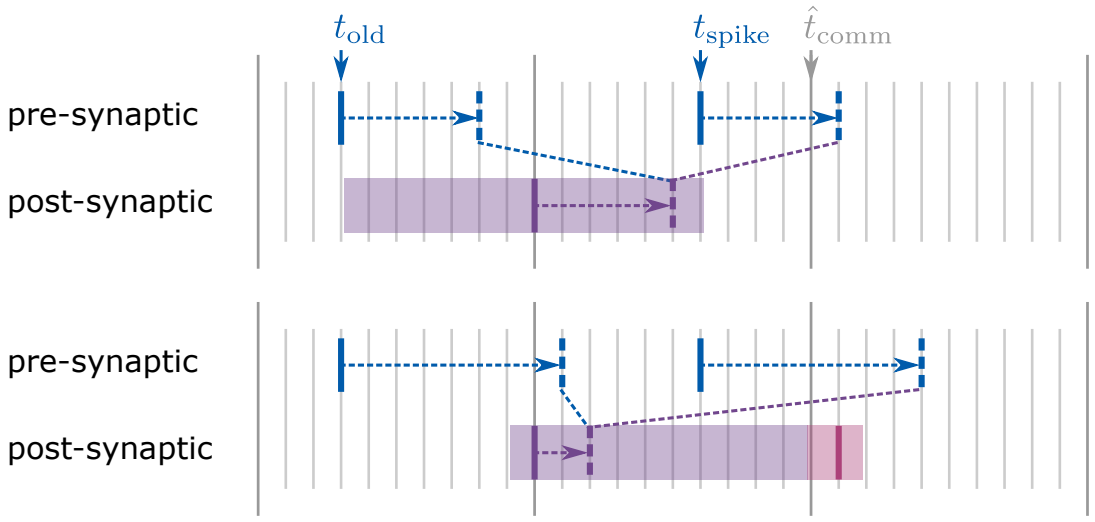


Figure 2.4: The illustration has two parts, the top part of the illustration is when the axonal delay is equal to dendritic delay, $d_{ax} = d_{dnd}$. The bottom part represents the scenario when the axonal delay is greater, $d_{ax} > d_{dnd}$. The t_{comm} bar represents the time at which NEST communicates events and the smaller gray bar represents the simulation resolution. The purple-shaded region in the postsynaptic neuron time grid represents the time window over which the get_history function looks for spikes. The dotted arrows represent the propagation delay of the spikes, and the dashed bar represents the arrival time of spikes at the synapse. The purple bar at the bottom-most part in the third time interval denotes the missed spike. Now, consider the first purple spike event in the post-synaptic neuron. When the spike occurs in the case with equal delays, the get_history function has a clear window, with which it is able to locate the previous spike as the function only gets called when the presynaptic neuron fires. In the axonal predominant delay case, when the same occurs, you can see that the function's window exceeds the current time, and thereby requires information from the future to perform the current change. And if the future spike is not available as should be the case, then it is not revisited to update the weight but rather this whole calculation would be ignored. Note. Original illustration by Susanne Kunkel, using with permission.

2.5 Motivation for including propagation delays in STDP model

The motivation for implementing axonal propagation delay as a parameter to the model comes from the support of its reference in the early experiments related to STDP (Bi and Poo 1998; Markram et al. 1997). The use of propagation delays in calculations is also logically coherent with reality. The explanation for the same is, as a spike is generated at the soma of the presynaptic neuron, it has to travel down along the length of the axon to reach the synapse. Similarly, any spike generated in the soma of the postsynaptic neuron has to propagate through the dendrite of the same neuron back to the synapse. As adaptations made in the synapse are dependent on the spike times, the synapse itself can only know about the spikes after it has arrived at its location. Since to our knowledge, the spike itself does not encode any information about its generation time, the synapse can be affected only by the arrival of the spikes. This propagation delay is a non-zero value in reality. The axons of different neurons vary in their structure and physical properties, some of the variations such as myelination or gigantism increase the speed of spike transmission across this axon (Hartline and Colman 2007). In the cortical regions of the brain, where different types of neurons are connected with each other (Somogyi and Klausberger 2005), then the models based on this network (Potjans and Diesmann 2014; Brunel 2000) will benefit with the support of having propagation delays in STDP calculations.

2.5.1 Effect of propagation delays in a network model

Since the network model, we are choosing for this work is the balanced random network. We look into how the effect of delay value dominated by axonal propagation affects the network dynamics. There are certainly different aspects in measuring this, in this work, we focus on how the weights of synapses particularly get affected over longer axonal delays. In the course of the simulation, the average value of the synaptic weight would tend to slightly shift to one way or the other. This shift in the mean of the weights is given the term drift of the mean. This drift is mathematically analogous to the drift of a random walker. The drift of synaptic weights is represented by $\dot{w}$

The calculation of the drift is done by using Fokker-Planck equation(Fokker 1914; Planck 1917). The solution for the drift of synaptic weight is given by Equation (2.4) (Morrison et al. 2008).

$$\dot{w}_{drift} = -F_{-}(w) \int_{-\infty}^{0} d\Delta t^{s}\, k_{-}(\Delta t^{s})\ \Gamma_{ij}(\Delta t^{s} + (d_{Ax} - d_{dnd})) + F_{+}(w) \int_{0}^{\infty} d\Delta t^{s}\, k_{+}(\Delta t^{s})\ \Gamma_{ij}(\Delta t^{s} + (d_{ax} - d_{dnd})) \quad (2.4)$$

where,

$F_{-}(w)$ = Depression function for updating synaptic weight
$F_{+}(w)$ = Facilitation function for updating synaptic weight
Δt^{s} = The time difference between the pre and postsynaptic spike
K_{-} = The k-value for depression
K_{+} = The k-value for facilitation
d_{Ax} = Axonal propagational delay
d_{dnd} = Dendritic propagational delay
Γ_{ij} = Cross-correlation function between presynaptic spike over postsynaptic spike

The cross-correlation function defined in the equation, drafts a correlation between the two spike trains produced in the neurons. The result of a cross-correlation function gives a rough idea of how the shape of two functions are correlated. As our assumption for the synaptic plasticity is that the spike from the presynaptic neuron should be responsible for the spike in the postsynaptic neuron, the cross-correlation function, adjusted for the propagation time, can give a factor of correlation between the spikes produced from two neurons.

The equation 2.4 has two parts, with the top part promoting negative drift and the bottom part being responsible for positive drift. In an ideal case, the drift is closer to zero. This could happen in the case when the condition, axonal delay, and dendritic delay match i.e., $d_{Ax} = d_{dnd} = d$, is satisfied. This is because in that specific scenario both the parts of the equation 2.4 will be closer to each other and the cross-correlation function will become:

$$\Gamma_{ij}(\Delta t^{s} + (d_{Ax} - d_{dnd})) \implies \Gamma_{ij}(\Delta t^{s} + (d - d)) \implies \Gamma_{ij}(\Delta t^{s}) \quad (2.5)$$

This condition of equal delay, gives us a certain frame of reference, for comparison with the other cases. Also, a note to consider is that this only works for the case where the spike train produced in the neurons is not produced by a random

process. Fortunately, the neuron exhibit some type of oscillations larger than the synaptic delay, which have been independently verified both in the experiment and simulated models (Kriener et al. 2008). Taking the case of equal propagation delays as a model, we can explore the cases where the delays are unequal.

If the axonal delay is larger than the dendritic delay i.e., $d_{Ax} - d_{dnd} > 0$, then the cross-correlation function has a slightly higher parameter. This in turn shifts the value of the correlation function more to the left-hand side. Since the top integral has limits ranging from $-\infty$ to zero, the top integral's value is increased and the bottom integral value is conversely decreased. This results in the overall drift moving towards the left side i.e., $\dot{w}_{drift} < 0$, thus making the synaptic weights on average lower.

If in the case of dendritic delay being larger than the axonal delay, then since $d_{Ax} - d_{dnd} < 0$ the result of cross-correlation is right shifted. So, the first integral decreases, and the second integral increases, following the same logic described before. This makes the whole synaptic weight drift towards the positive side $\dot{w}_{drift} > 0$ (Tchumatchenko et al. 2011).

Chapter 3

Methodology

The existing implementation of the algorithm lacks the support for having axonal propagation delays as a parameter included in its computation. So, we modified the algorithm to make the model have axonal delay as a parameter in its calculations. The addition of axonal delay as a parameter to the model is quite trivial, as just adding the parameter in locations where there was already dendritic delay, will be sufficient. The problem arises to support calculations when the axonal delay is greater than the dendritic delay. As this would result in the model missing a postsynaptic spike for its weight calculations (refer Section 2.4).

3.1 Proposed algorithm

To implement the modified algorithm, a new function is added to the existing algorithm. This new function called adjust_weight (Algorithm 3.1) is responsible for carefully recognizing the spikes, which would not have been considered in the older algorithm. The added function performs the required calculations again in the communication time of the NEST simulator, after the firing of the postsynaptic neuron. The calculation of the synaptic weight is then updated using the missed spike. This makes sure that the correct value of synaptic weight is maintained in any case.

The algorithm is best explained as a flow of the process. The flowchart for the new algorithm is given in Figure 3.1. The process starts when a presynaptic neuron generates a spike, the spike is stored as a SpikeEvent data structure in NEST,

which is communicated to all the subsequent targets of the neuron. In our case, the target is the postsynaptic neuron, but for it to reach that target the data gets to the STDP model connected between them. This is done by event-handling functions defined in the NEST kernel.

When the SpikeEvent is received by the synapse, the update_weight procedure described in Algorithm 2.1 is invoked[1]. But before the last depression step, if the axonal delay is greater than the dendritic delay, the weight before the depression step is performed, along with the previous presynaptic spike time and the current presynaptic spike time is packed into a data structure of adjustentry[2] type (refer Table 3.1 for detailed description) and stored in a queue. Then the process continues with performing the depression step and subsequently sending the SpikeEvent to the postsynaptic neuron.

Now, when the postsynaptic neuron fires, at NEST communication time, the spike time is pushed to the history of spikes, which itself is implemented as a stack-based data structure[3]. After this process, the queue storing adjustentry data is iterated over to find the postsynaptic spike which satisfies the condition of being generated after the latest presynaptic spike whose time is offset by the addition of axonal delay and subtraction of dendritic delay. This offset is done because we need to find the spike which comes to the synapse after the presynaptic spike[4]. Since the postsynaptic spike also takes over the dendritic delay to arrive at the synapse, we look for a spike that has been generated in the postsynaptic neuron at a time advanced by the value of the dendritic delay. Once the condition is satisfied, then the adjust weight procedure is invoked with the current postsynaptic spike as the missed_spike parameter along with the adjustentry data that satisfied the condition.

The adjust_weight procedure[5] (Algorithm 3.1) takes the weight before the last depression step and also computes the actual spike time of the last presynaptic spike. Then the access_counter value of the synapse is increased by one. This counter variable is used to clean off spikes from history, by removing spikes with access_counter greater than the outgoing targets from the synapse. Here, the facilitation step is actually different from the existing method, the K_+ value is actually returned to the previous value (value before the depression step) by taking

[1] In NEST Simulator it is named as the send function, Source Code line 251 - 318

[2] Adjust Entry: Source Code

[3] ArchivingNode is a class used in NEST inherited by the neuron model, the set_spiketime function which is a member of this class achieves this functionality: Source Code, line 191 -230

[4] the adjust_weights function in Archiving Node, not to be confused with the one in the model source code achieves this functionality: Source Code, line 284 -295

[5] adjust_weight function: Source Code, line 354 - 413

Table 3.1: Details of adjustentry data structure

Member variable	Description
$t_{lastspike}$	Spike time of one spike before the latest spike generated by the presynaptic neuron
$weight$	The synaptic weight before the last depression step is performed
$t_{received}$	Spike time of the latest spike generated by the presynaptic neuron, offset by the difference between axonal delay and dendritic delay
$identifier\_data$	An identifier data to identify the synapse, and the neurons connected to it, useful for NEST Kernel to know which synapse to perform the operations on

the mathematical inverse of the step previous k_+ update (Line 11 of Algorithm 2.1). Then the facilitation step is again performed with this new value. Since the value of dt is negative, the facilitation step results in a very small increase in weight.

Before the depression step is done, the weights are updated in the adjustentry data structure, so as to have the correct data if there are any subsequent postsynaptic spikes. Finally, the k_- value is computed based on the recent data and then the depression step is performed. In the NEST simulator, there is one more practical step in which the synapse's weight is updated by calling its setter function.

Algorithm 3.1 Adjustment of the synaptic weight based on missed spike

1: **procedure** `adjust_weight`(`adjustentry, missed_spike`)
2: $original\_weight \leftarrow current\_weight$
3: $current\_weight \leftarrow adjustentry.weight$
4: $t_{spike} = adjustentry.t_{received} - d_{Ax} + d_{Dnd}$
5: Increase the synapse access counter, counter$_{sp}$
6: $dt \leftarrow missed\_spike + d_{dnd} - (adjustentry.t_{lastspike} + d_{ax})$
7: assert $dt > 0$
8: $k_+^{inv} \leftarrow (k_+ - 1) \cdot (\exp \frac{adjustentry.time_{lastspike} - t_{spike}}{\tau})^{-1}$
9: $current\_weight \leftarrow current\_weight + F_+(current\_weight) \cdot K_+^{inv} \cdot e^{\frac{-dt}{\tau}}$
10: $adjustentry.weight \leftarrow current\_weight$
11: $K_- \leftarrow target\_neuron.get\_K\_value(t_{spike} + d_{Ax} - d_{Dnd})$
12: $current\_weight \leftarrow current\_weight - F_-(current\_weight) \cdot K_-$

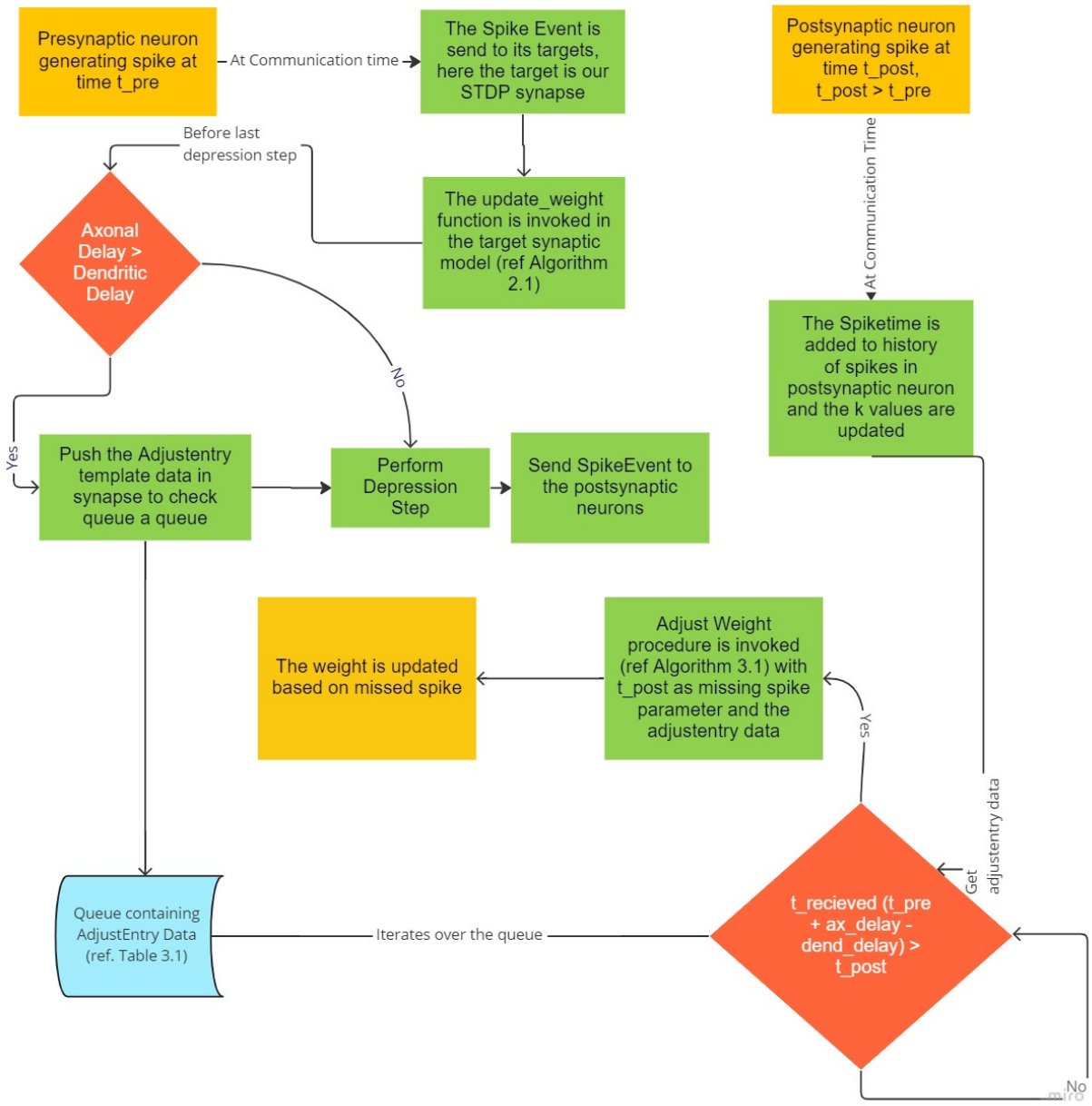


Figure 3.1: The flowchart depicts the process through which the proposed algorithm works. The yellow blocks are the initiating/terminating event blocks and the green boxes are processes that can be traced to specific procedures in the NEST Simulator source code. The Orange boxes represent branching conditions

3.2 Testing the new model

To test the model, the NEST native language SLI script was used. The script was made to run simulations for 100 biological seconds. The script used a simple network model of one presynaptic neuron connected to a postsynaptic neuron through the new STDP model. Additionally, the postsynaptic neuron has connections representing the population of inhibitory neurons and the external neuron population. This is done to maintain the same condition as a balanced random network exhibiting asynchronous irregular regime (Brunel 2000) as that was the target network model for this synaptic model (Morrison et al. 2007). The model is similar to the Figure 3.2, but here we only use one presynaptic neuron. The parameters chosen for the script were taken from the (Morrison et al. 2007) and the particular values were chosen to maintain the network exhibiting the said regime. The values are given in Appendix B along with their description. The (Brunel 2000) network has an excitatory population four times in size as the inhibitory population, the balanced state is achieved by having a greater firing rate for the inhibitory population than the excitatory population. In testing the models are replaced by a single neuron having its firing rate equal to that of population size times the firing rate for the population. The spikes are generated by a Poisson generator connected to the neurons. We use a Leaky Integrate-and-Fire neuron model for the postsynaptic neuron, and the rest of the neurons representing the population are Poisson generators connected to the postsynaptic neuron through a parrot neuron.

The test script[6] contains two parts. In the first part, the script runs the simulation using our model implemented in the NEST simulator along with all the other models from the NEST. The spikes generated by both the presynaptic neuron and the postsynaptic neuron are collected and stored. These spikes are later then used in the second part of the test script. The second part of the test script takes the spikes alone and independently calculates the weight of the synapse with each consecutive presynaptic and postsynaptic spike pair. Finally, after calculations, the final weight of the synapse obtained from the first part is compared with the final weight obtained by independent calculation in the second part. If the weight matches then the test has passed otherwise the test has failed, implying there is an error at least in one of the parts of the script.

This testing was looped for different values of axonal delays. The test was first performed for ten axonal delay values, ranging from zero to one second, for every

[6]The SLI script used for testing: Source Code

one-tenth of a second, while keeping the total propagational delay as one second. Then the test was modified to verify for every one-tenth axonal delay over the range of zero to ten seconds, with the total propagation delay set as ten seconds. The test passed in all the cases. A wide variety of axonal delays were needed to make sure that the test passes on all the border cases.

3.3 Analysis of weight distribution

To perform the analysis of weights we constructed a network model, which is very similar to the model used for testing. The chosen architecture is called the many-to-one architecture and its overview is given by Figure 3.2. Since we only want to observe the effects of the implementation only in balanced random networks in the asynchronous irregular regimen, we use the same model. The only difference being there are many presynaptic neurons and synapses connected to a single postsynaptic neuron.

After building the network, the model was simulated for 100.0 seconds in biological time[7]. The parameters used for this network model can be found in Appendix C.

[7]The python code used for analysis: Source Code

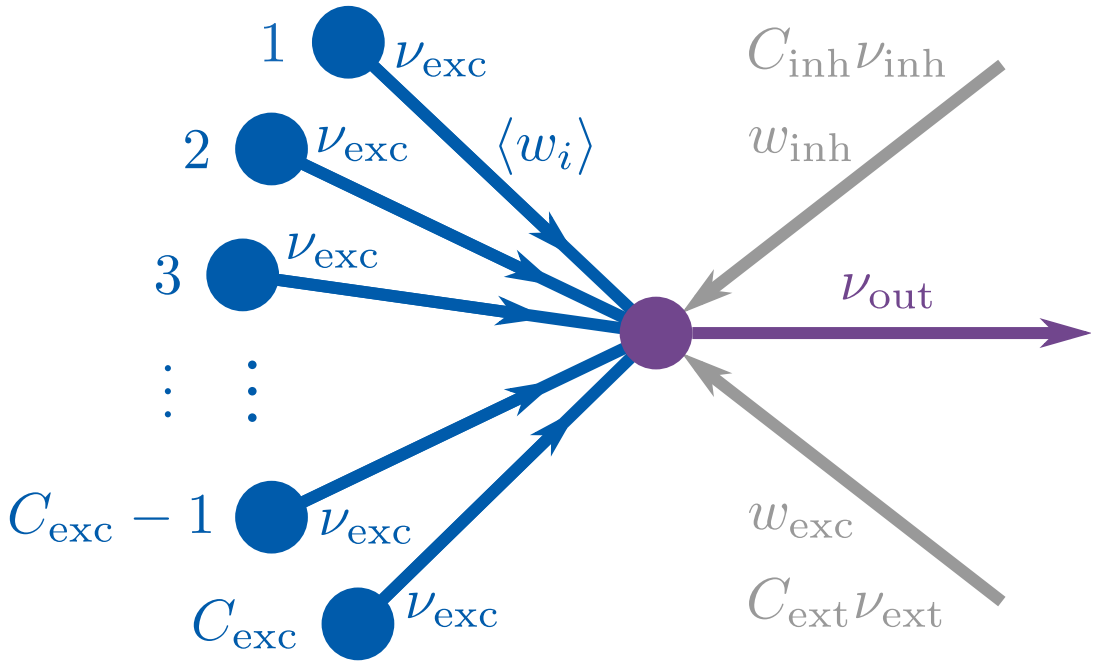


Figure 3.2: An architecture diagram for many to one neural network. The filled circles represent neuron models, blue represents the parrot neurons and purple represents the Leaky integrate-and-fire neuron. The C_{exc}, C_{inh}, and C_{ext} represent connections from the excitatory, inhibitory, and external populations of neurons. The parameter w represents the synaptic weight of the respective connections. The parameter v represents the firing rate of the particular neuron population. Note. Original drawing provided by Susanne Kunkel, using with permission

3.4 Optimizing the α parameter

The firing rate which we got in the model was above 100 spikes/s. The state-of-the-art algorithm was only tested in a network model with a firing rate of 10 spikes/s or lower(Morrison et al. 2007). This is because this is the more biologically plausible firing in the cortical regions, on which the network model is based on (Brunel 2000). For this, we chose to adjust the α parameter[8] as that is the parameter responsible for tuning the depression step. If higher depression occurs by the synapse, then the postsynaptic neuron would be less likely to fire. We increased the alpha value until the exit spike rate is lower than 10 spikes/s, this was done for each case of axonal delay separately.

3.5 Benchmark Performance

To test if our implementation comes with any significant computational cost, our implementation was benchmarked against the current model. Since the current model in NEST Simulator doesn't support the inclusion of axonal delay as a parameter, we modified the model to just accept the parameter[9]. This makes it work well when the axonal delay is lower than the dendritic delay. But, this modified model doesn't properly compute the synaptic weights when the axonal delay is dominating the propagation delay. Also, since it would not be meaningful to compare the performance of networks with different dynamics, we changed our implementation in such a way that the function after performing all the calculations, makes sure that it does not change the original weight. This was done by removing the last part of the adjust_weight function in the model file, where in the weight obtained after the calculation is set to the synapse as its current weight.

The benchmark was done with the same parameters as the analysis of weights without the optimized alpha value. 10 different runs were made for each case to minimize any artifacts. The benchmark was performed on a computer with an i7-9750H CPU, 2.60GHz as its processor, having 8 GB of RAM and 1 TB Solid State Drive for storage. It has 6 cores and all the cores were utilized for benchmarking.[10].

[8]The python code used for parameter optimization: Source Code
[9]Modified version: Source Code, original version: Source Code
[10]The python code used for benchmarking: Source Code

Chapter 4

Results

The first result was to test if the model was working as it was intended to. So, testing and verifying that all the test cases pass without any errors was the first step. Once it was verified the model passed testing, analysis of synaptic weight distribution in the simulated network and benchmarking of the model were performed.

Distribution of Synaptic Weights

The results of the analysis of weight distribution in a many-to-one network model are as shown in Figure 4.1 and **??**. The trend can be clearly seen as predicted in theory that the graphs gradually shift towards the right as the delay value is increased. A more profound effect is seen when the delay value is fully dominated by axonal delay, and the total propagational delay value is also larger (10.0 ms). In this case, the weights apart from shifting to a higher value, the weights separate more between our implementation and the existing implementation. in the case with equal propagation delays, there seems to be only one graph, this is because the graphs are overlapping having the same value. This is the reason why only cases of axonal delay being greater than or equal to dendritic delay are shown, otherwise there will be more overlapping graphs. This agrees with what was predicted by the theory. The obtained weights for both implementations (for every delay case) were statistically significant from each other except for the case of equal propagational delay. We performed Kolmogorov–Smirnov test(Smirnov 1948) for the statistical testing.

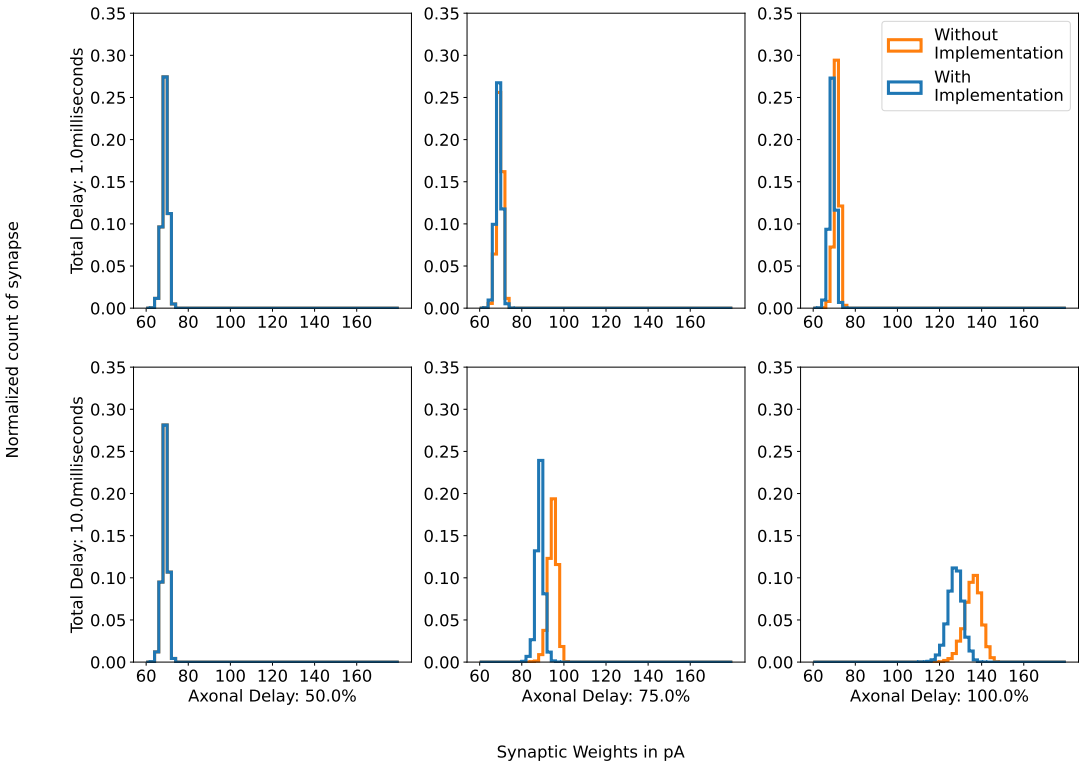


Figure 4.1: The plots show the histogram of weight distribution after the simulation of the many-to-one network model. With implementation here implies our implementation of the model and without involves the usage of the modified current model. The weights shift more to right and from each other as both the axonal and total propagation delay increase.

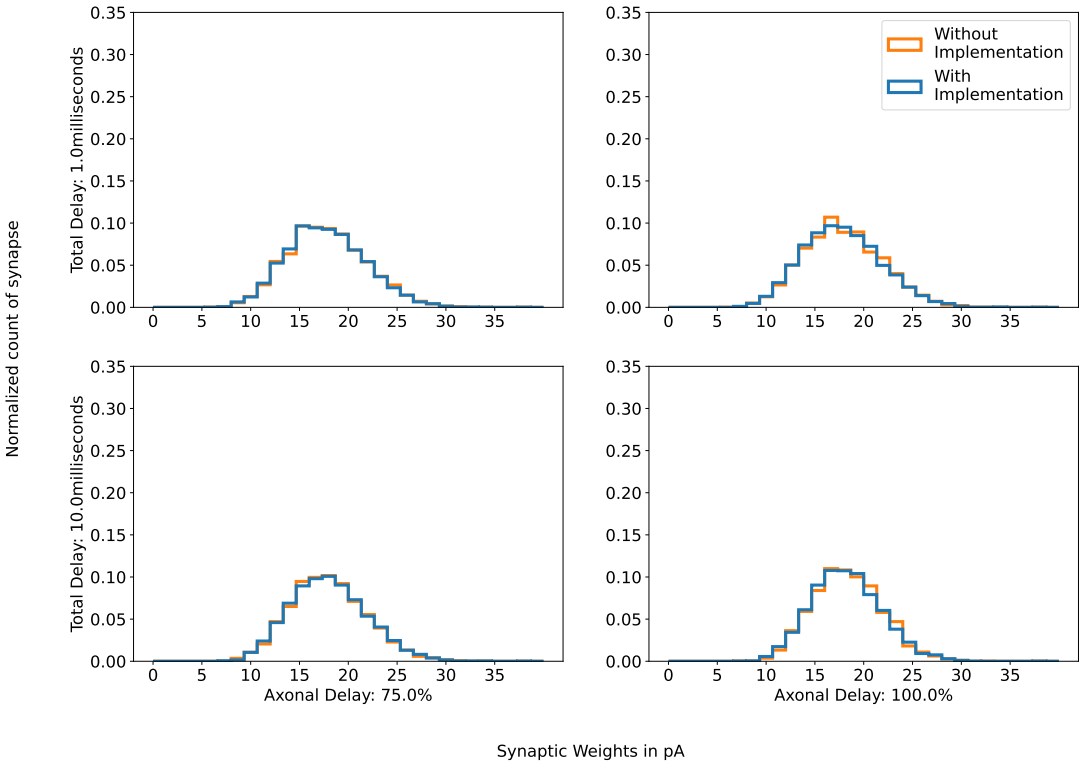


Figure 4.2: The plots show the histogram of weight distribution after the simulation of the many-to-one network model with optimized α parameter. Only cases for dominating axonal delay are shown here, as seen previously lower axonal delay does not have any difference. The variation of weight for different cases seems ever so slight that no noticeable trend can be observed

Distribution of weights after optimization

After finding the α value for which the postsynaptic neuron had a low spiking rate. We used this parameter and again ran the simulations to analyze the effect on the weights. The results for the parameter-optimized model are shown in Figure 4.2. There doesn't seem to be any significant trend in the weights, even for longer cases. There was no statistically significant difference between the weights obtained from the current model and our implementations. This could be explained because the firing rate was lowered by an order of magnitude, and the amount of weight update calculations done by the synapse would also be lowered, this means we should have increased the total simulation time to compensate. As we did not do that here, it could be the reason we don't see any trend.

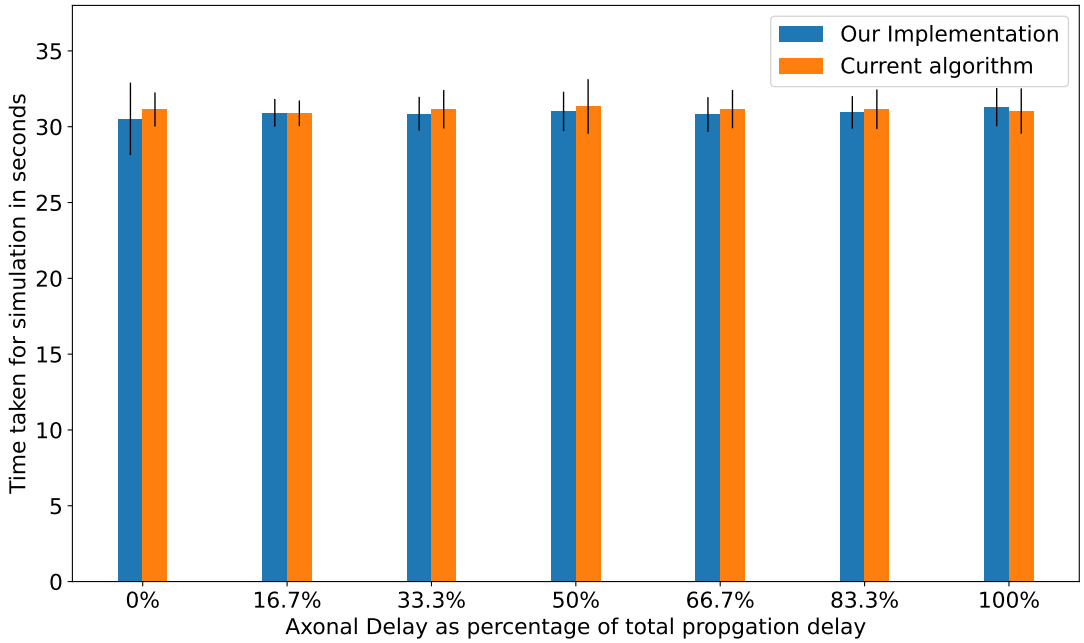


Figure 4.3: The plots depict the mean performance of various models of STDP synapse. It has the models with our new algorithm and previously used algorithm, for total delays of 1.0 ms. The x-axis represents the percentage of axonal delay w.r.t. total delay. The y-axis is in seconds, representing the average time taken for the simulation. The black lines represent the errorbar for each ben

Benchmark perfomrance

The average time taken for each scenario is plotted and shown in the Figure 4.3 and Figure 4.4. The benchmark results show that the time taken for the algorithm almost remains the same with no significant trend. You can see that the average time for every case falls within the error region of the others. This is the case for both 1 ms and 10 ms propagational delays. From this, we can infer that our algorithm does not bring any extra overhead to the model.

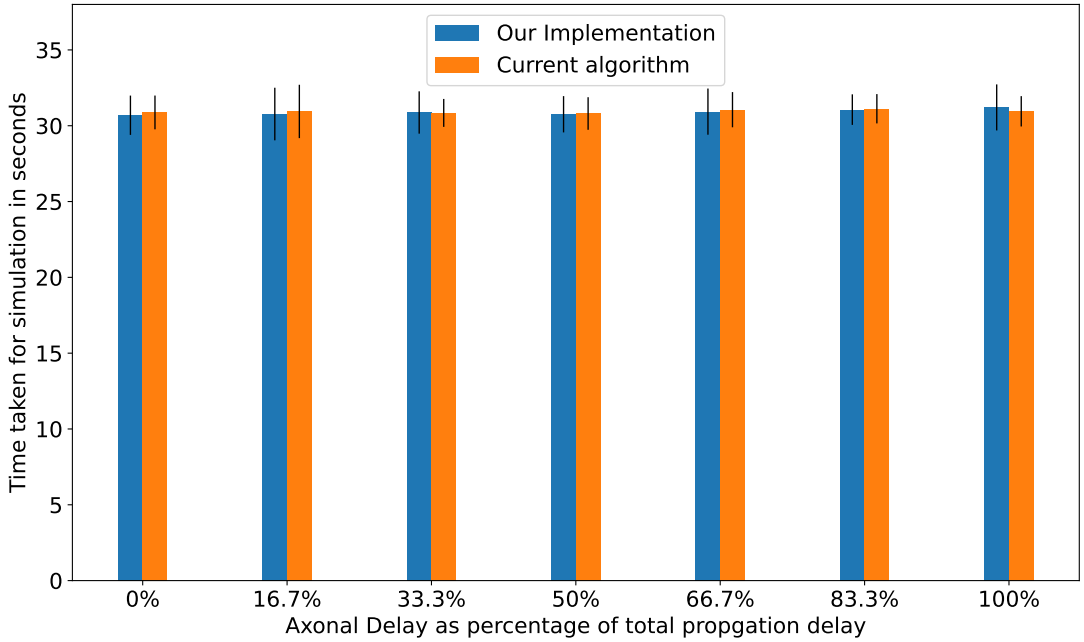


Figure 4.4: The plots depict the mean performance of various models of STDP synapse. It has the models with our new algorithm and previously used algorithm, for total delays of 10.0 ms. The x-axis represents the percentage of axonal delay w.r.t. total delay. The y-axis is in seconds, representing the average time taken for the simulation.

Chapter 5

Discussion

The results gathered from analyzing the synaptic weights are as expected. The weights are shifted towards the left when the axonal delay is increased compared to that of older implementation. We also see the weights have shifted right when the total delay value is increased, this is in line with our theoretical prediction. The results of the benchmark indicate that our model does not have any additional computational costs and runs reliably fast in simulations.

Future enhancements and limitations

Although we have performed the analysis of weights, we have not performed the analysis of weights in a full-scale balanced random network. We only localized to one neuron and simulated the effect. Further analysis of weights in a fully configured balanced random network (Brunel 2000). The bench-marking was not done within a dedicated framework such as the beNNch framework (Albers et al. 2021). A specialized framework for benchmarking ensures that there are no artifacts while performing benchmarks. Since the process of benchmarking involves the calculation of the time taken for a particular task to be completed by the computer. It is sensitive if the computer is occupied by another task which would thereby slow down the calculations performed by the task. When performing in a framework though, it takes care of this issue by creating a virtual shell around the task and running the task without being affected by other computer activities. This is also greatly enhanced if performed on a dedicated computer instead of a general-purpose computer. We also did not see the expected result with a relevant firing rate, the network model can be analyzed with our synaptic model which is

simulated for a longer period of time to see the effect.

The model in its current form only works for one type of neuron model available in the NEST simulator extending the support to more models extends its reach and usability. Our implementation only considers the spike time which is forced to be constrained to that of simulation resolution in NEST. Although NEST stores the precise spike time, which is a spike time in float value is not a multiple of simulation resolution. Future enhancements can be done to implement a precise spike time version of the same. Although precise spike time calculation can have its constraints (Banerjee et al. 2008), they tend to be quite useful for modeling the neurons in visual cortex, which have shown to have high temporal sensitivity (Tiesinga et al. 2008).

Conclusion

To summarize, the STDP synaptic model explains the functional plasticity experienced in the synapse present in the brain. This plasticity is responsible for the abilities of learning and adaptation exhibited by the brain. We chose an implementation a specific implementation of this model based on power law, to be adapted to include a new feature. This feature is the inclusion of axonal propagational delay of the spike signal from a postsynaptic neuron to the synapse in the model. We were motivated to perform the implementation based on mathematical evidence showing a shift in synaptic weights, which was shown by Morrison et al. 2008.

We performed this implementation in NEST simulator as it already contains the implementation of STDP model based on power law, along with numerous other features and reasons. After implementation, we tested the model under many scenarios to see if it performs as intended. After the model passed all testing, we created a network model, with a many-to-one architecture. We analyzed the effects of our implementation on this network, by comparing the synaptic weights before and after implementation. The results of the analysis matched with theoretical predictions. Then we performed benchmarks to see the performance difference between the models with and without the implementation of our algorithm. We did not observe any significant difference in the benchmark. Thereby it can be implied that our model runs properly without any significant computational cost.

All the programming code behind this scientific work can be found here [1], [2]. With that we hope the outcome of this scientific work, finds its usefulness among other researchers working in the same domain.

[1]For NEST related code: https://github.com/sanjay270597/nest-simulator/tree/stdp_axonal_delay

[2]For other code used in this project: https://github.com/sanjay270597

Bibliography

Albers, Jasper et al. (2021). *A Modular Workflow for Performance Benchmarking of Neuronal Network Simulations*. DOI: 10.48550/ARXIV.2112.09018. URL: https://arxiv.org/abs/2112.09018.

Arslan, Salim et al. (2018). 'Human brain mapping: A systematic comparison of parcellation methods for the human cerebral cortex'. In: *Neuroimage* 170, pp. 5–30.

Banerjee, Arunava, Peggy Seriès and Alexandre Pouget (2008). 'Dynamical constraints on using precise spike timing to compute in recurrent cortical networks'. In: *Neural Computation* 20.4, pp. 974–993.

Bi, Guo-qiang and Mu-ming Poo (1998). 'Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type'. In: *Journal of neuroscience* 18.24, pp. 10464–10472.

Blundell, Inga et al. (2018). 'Code generation in computational neuroscience: a review of tools and techniques'. In: *Frontiers in neuroinformatics* 12, p. 68.

Brunel, Nicolas (2000). 'Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons'. In: *Journal of computational neuroscience* 8.3, pp. 183–208.

Brunel, Nicolas and Vincent Hakim (1999). 'Fast global oscillations in networks of integrate-and-fire neurons with low firing rates'. In: *Neural computation* 11.7, pp. 1621–1671.

Brunel, Nicolas and Xiao-Jing Wang (2003). 'What determines the frequency of fast network oscillations with irregular neural discharges? I. Synaptic dynamics and excitation-inhibition balance'. In: *Journal of neurophysiology* 90.1, pp. 415–430.

Costello, Mark J, Robert M May and Nigel E Stork (2013). 'Can we name Earth's species before they go extinct?' In: *science* 339.6118, pp. 413–416.

Craver, Carl F (2007). *Explaining the brain: Mechanisms and the mosaic unity of neuroscience*. Clarendon Press.

Diesmann, Markus et al. (2001). 'State space analysis of synchronous spiking in cortical neural networks'. In: *Neurocomputing* 38, pp. 565–571.

Fokker, Adriaan Daniël (1914). 'Die mittlere Energie rotierender elektrischer Dipole im Strahlungsfeld'. In: *Annalen der Physik* 348.5, pp. 810–820.

Freeland, Elana (2014). *Chemtrails, HAARP, and the full spectrum dominance of planet earth*. Feral House.

Garrett, Bob and Gerald Hough (2017). *Brain & Behavior: An Introduction to Behavioral Neuroscience*. Sage Publications.

Gross, Charles G (2012). *A hole in the head: more tales in the history of neuroscience*. MIT Press.

Hartline, DK and DR Colman (2007). 'Rapid conduction and the evolution of giant axons and myelinated fibers'. In: *Current Biology* 17.1, R29–R35.

Hebb, Donald Olding (1949). *The organization of behavior: A neuropsychological theory*. Wiley and Sons.

Jerison, Harry (2012). *Evolution of the brain and intelligence*. Elsevier.

Kriener, Birgit et al. (2008). 'Correlations and population dynamics in cortical networks'. In: *Neural Computation* 20.9, pp. 2185–2226.

Lucas, Keith (1909). 'The "all or none" contraction of the amphibian skeletal muscle fibre'. In: *The Journal of Physiology* 38.2-3, p. 113.

Madadi Asl, Mojtaba, Alireza Valizadeh and Peter A Tass (2017). 'Dendritic and axonal propagation delays determine emergent structures of neuronal networks with plastic synapses'. In: *Scientific reports* 7.1, pp. 1–12.

Markram, Henry (2012). 'The human brain project'. In: *Scientific American* 306.6, pp. 50–55.

Markram, Henry et al. (1997). 'Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs'. In: *Science* 275.5297, pp. 213–215.

Morrison, Abigail, Ad Aertsen and Markus Diesmann (2007). 'Spike-Timing-Dependent Plasticity in Balanced Random Networks'. In: *Neural Computation* 19, pp. 1437–67. DOI: 10.1162/neco.2007.19.6.1437.

Morrison, Abigail, Markus Diesmann and Wulfram Gerstner (2008). 'Phenomenological models of synaptic plasticity based on spike timing'. In: *Biological cybernetics* 98.6, pp. 459–478.

Planck, VM (1917). 'Über einen Satz der statistischen Dynamik und seine Erweiterung in der Quantentheorie'. In: *Sitzungberichte der*.

Potjans, Tobias C and Markus Diesmann (2014). 'The cell-type specific cortical microcircuit: relating structure and activity in a full-scale spiking network model'. In: *Cerebral cortex* 24.3, pp. 785–806.

Purves, Dale et al. (2008). *Neuroscience. 4th*. Vol. 857.

Rotter, Stefan and Markus Diesmann (1999). 'Exact digital simulation of time-invariant linear systems with applications to neuronal modeling'. In: *Biological cybernetics* 81.5, pp. 381–402.

Smirnov, Nickolay (1948). 'Table for estimating the goodness of fit of empirical distributions'. In: *The annals of mathematical statistics* 19.2, pp. 279–281.

Somogyi, Peter and Thomas Klausberger (2005). 'Defined types of cortical interneurone structure space and spike timing in the hippocampus'. In: *The Journal of physiology* 562.1, pp. 9–26.

Spreizer, Sebastian et al. (Mar. 2022). *NEST 3.3*. Version 3.3. DOI: 10.5281/zenodo.6368024. URL: https://doi.org/10.5281/zenodo.6368024.

Sterratt, David et al. (2011). *Principles of computational modelling in neuroscience*. Cambridge University Press.

Tchumatchenko, Tatjana et al. (2011). 'Spike Correlations – What Can They Tell About Synchrony?' In: *Frontiers in Neuroscience* 5. ISSN: 1662-453X. DOI: 10.3389/fnins.2011.00068. URL: https://www.frontiersin.org/articles/10.3389/fnins.2011.00068.

Teramae, Jun-nosuke, Yasuhiro Tsubo and Tomoki Fukai (2012). 'Optimal spike-based communication in excitable networks with strong-sparse and weak-dense links'. In: *Scientific reports* 2.1, pp. 1–6.

Tiesinga, Paul, Jean-Marc Fellous and Terrence J Sejnowski (2008). 'Regulation of spike timing in visual cortical circuits'. In: *Nature reviews neuroscience* 9.2, pp. 97–107.

Zhang, Yang et al. (2020). 'Brain-inspired computing with memristors: Challenges in devices, circuits, and systems'. In: *Applied Physics Reviews* 7.1, p. 011308.

Appendix A

Detailed overview of NEST objects

NEST has its own terminologies being used. Here is the description of some of the objects used within the framework which have been modified or utilized in this work.

Spike Generator

These are models which can be imagined as the origin of the spikes in the system. Since a neuron by itself does not create any signal unless it is a sensory neuron, these spike generators are responsible for creating signals in the connected network. A Poisson generator is a specific spike generator that uses Poisson distribution to produce spikes.

Parrot Neuron

A spike generator in NEST, should not be directly connected to the model of a neuron but rather connected through a specific neuron solely for this purpose called the parrot Neuron. The function of the parrot neuron is to repeat the signals from the spike generator to the neuron it is connected to. Since it repeats everything it receives, the neuron is aptly named as parrots tend to repeat what it hears. If multiple parrot neurons are connected to a spike generator, each will produce a unique spike pattern from the generator.

Table A.1: Connection Rules available in NEST

Connection Rule	Description
`All-to-All`	All the neurons from the source are connected to the target
`Fixed Indegree`	The neurons are connected randomly with target neurons having fixed incoming connections
`Fixed Outdegree`	The neurons are connected randomly with source neurons having fixed outgoing connections
`Fixed Total Number`	The neurons are connected randomly with a limit on the total number of connections
`One-to-One`	Source and target neurons are connected in one on one correspondence
`Pairwise Bernoulli`	Number of connections is dependant on the specified probability of total connections
`Symmetric Pairwise Bernoulli`	Similar to Pairwise Bernoulli, with the slight difference that the connections are made in both the directions, that is both from source to target and target to source

Connection types

For a neural network to work as a network, it should all be connected properly. The connection is managed in NEST by specifying the source, target, connection properties, and finally synaptic properties. The connection properties in NEST currently support seven different types of connection rules. The connection rules and their descriptions are provided in Table A.1

The source and target for a connection can be a single neuron or a group of neurons. Except for the case of a One-to-One connection in which there should be an equal number of source and target neurons. NEST maintains all the connections in a connection matrix. With spatial class, NEST also supports the use of spatial placement of neurons and their connections.

Random Generators

NEST comes with in-built random generators, which can provide a list of random numbers and are also used to set the random seed for the simulation. Running a simulation with a different random seed is useful for researchers to properly analyze the result from the simulation. Also, running with different random seeds will give results that can be eliminated as a random fluke.

Recording Devices

NEST recording devices mimic physical recorders, used for recording the physical properties or events occurring during the simulation. Voltmeter, Multimeter, and SpikeRecorder are examples of such devices. Recorders are useful devices to measure the actual data from the simulation. If no recording device is connected to the simulation, then only the final state of the simulation can be obtained.

NEST kernel

The NEST Kernel is an important part of the NEST framework. All simulations run in the kernel and the kernel changes its state when running the simulation. Internally all communications are handled by the Kernel and there is a kernel manager subroutine that is responsible for managing the activities of the kernel. General simulation properties such as simulation time resolution, minimum delay, and maximum delay are configured as Kernel attributes.

Although NEST support parallel simulation for large network of neurons, NEST Kernel implementation currently does not support running multiple independent simulations parallelly in a single NEST kernel.

Appendix B

Network Parameters for testing

The tables below give the list of network parameters, that were utilized for testing the implementation.

Table B.1: Parameters of the Poisson spike generator for testing

Parameter	Description	Value
Inhibitory firing rate	Firing rate of the generator representing inhibitory population	20,000 spikes/s
Excitatory firing rate	Firing rate of the generator representing excitatory population	93,600 spikes/s
External firing rate	Firing rate of the generator representing external population	13,600 Hertz spikes/s

Table B.2: Parameters of the neuron models for testing

Parameter	Description	Value
C_m	Membrane capacitance	250.0 pF
E_L	Resting Membrane potential	-70.0 mV
I_e	Resting membrane current	0.0 mV
tau_m	Membrane time constant	15.0 ms
tau_syn_ex	Rise time for the excitatory alpha signal	2.0 ms
tau_syn_in	Rise time for the inhibitory alpha signal	2.0 ms
t_ref	Refractory time period	2.0 ms
V_reset	Reset potential of the membrane	0.0 mV
V_th	Spike threshold voltage	-55.0 mV
V_m	Membrane potential	-70.0 mV

Table B.3: Parameters of the synaptic models for testing

Parameter	Description	Value
Inhibitory connection weight	Weight of the synapse, connecting the inhibitory population to the postsynaptic neuron	-225.0 pA
Excitatory connection weight	Weight of the synapse, connecting the excitatory population to the parrot neuron	45.0 pA
External connection weight	Weight of the synapse, connecting the external population to the postsynaptic neuron	45.0 pA
STDP synaptic weight	Weight of the synapse, connecting presynaptic parrot neuron to the postsynaptic neuron	45.0 pA
α	The alpha value represents the depression factor, only for STDP synapse	0.057
λ	The lambda value represents the weight update factor for STDP synapse	0.1
τ	The tau value represents the time window for the facilitation and depression in STDP synapse	15.0 ms
μ	The mu value represents the exponentiation factor for facilitation in STDP synapse	0.4
delay	The total propagation delay, depicting the sum of axonal and dendritic delay for STDP synapse	10.0 ms
axonal delay	The axonal delay for STDP synapse increased in steps of 0.1 ms	0.0 to 10.0 ms

Appendix C

Parameters for analysis of weight distribution

The following tables contain the parameter values used for analysing the effect of axonal delay, on synaptic weight distribution

Table C.1: Parameters of the Poisson spike generator for analysis

Parameter	Description	Value
Inhibitory firing rate	Firing rate of the generator representing inhibitory population	7,680 Hertz
Excitatory firing rate	Firing rate of the generator representing excitatory population	8 Hertz
External firing rate	Firing rate of the generator representing external population	16,800 Hertz

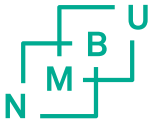
Table C.2: Parameters of the neuron models for analysis

Parameter	Description	Value
C_m	Membrane capacitance in picoFarad	250.0 pF
E_L	Resting Membrane potential	0.0 mV
I_e	Resting membrane current	0.0 mV
tau_m	Membrane time constant	10.0 ms
tau_syn_ex	Rise time for the excitatory alpha signal	0.3258 ms
tau_syn_in	Rise time for the inhibitory alpha signal	0.3258 ms
t_ref	Refractory time period	0.5 ms
V_reset	Reset potential of the membrane	0.0 mV
V_th	Spike threshold voltage	20.0 mV
V_m	Membrane potential	5.7 mV

Table C.3: Parameters of the synaptic models for analysis

Parameter	Description	Value
Inhibitory connection weight	Weight of the synapse, connecting the inhibitory population to the postsynaptic neuron	-192.5 pA
Excitatory connection weight	Weight of the synapse, connecting the excitatory population to the parrot neuron	38.5 pA
External connection weight	Weight of the synapse, connecting the external population to the postsynaptic neuron	38.5 pA
STDP synaptic weight	Weight of the synapse, connecting presynaptic parrot neurons to the postsynaptic neuron	38.5 pA
α	The alpha value represents the depression factor, only for STDP synapse	0.057
λ	The lambda value represents the weight update factor for STDP synapse	0.1
τ	The tau value represents the time window for the facilitation and depression in STDP synapse	15.0 ms
μ	The mu value represents the exponentiation factor for facilitation in STDP synapse	0.4
delay	The total propagation delay, depicting the sum of axonal and dendritic delay for STDP synapse	1.0, 10.0 ms
axonal delay	The axonal delay for STDP synapse, increased in steps of 0.25 ms, for total delay of 10.0 ms, axonal delay is multiplied by 10.	0.0 to 1.0 ms

Thank you.



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway