UNIVERSITY OF THE PHILIPPINES

Bachelor of Science in Applied Physics

Maria Tisha C. Bagasala

Dynamics of brain neurons using a one-dimensional probabilistic cellular automata model

Thesis Adviser:

Johnrob Y. Bantang, Ph.D.
National Institute of Physics
University of the Philippines Diliman

Date of Submission:
January 2022

Thesis Classification:
P

Dynamics of brain neurons using a one-dimensional probabilistic cellular automata model
Author: Maria Tisha C. Bagasala
BS Applied Physics Thesis
National Institute of Physics
University of the Philippines Diliman
January 2022

Classification* : P

* I – invention or creation, P – publication, C – confidential information

Available to the general public	Yes
Available only after consultation with author/adviser for thesis	No
Available only to those bound by nondisclosure or confidentiality agreement	No

[illegible]
Maria Tisha C. Bagasala
Student

[illegible]
Johnrob Y. Bantang, Ph.D.
Thesis Adviser

ENDORSEMENT

This is to certify that this undergraduate thesis entitled **DYNAMICS OF BRAIN NEURONS USING A ONE-DIMENSIONAL PROBABILISTIC CELLULAR AUTOMATA MODEL** prepared and submitted by Maria Tisha Canda Bagasala, is hereby endorsed for acceptance as partial fulfillment of the requirements for the degree of Bachelor of Science in Applied Physics (Instrumentation).

[illegible]
JOHNROB Y. BANTANG, Ph.D.
Thesis Adviser

This undergraduate thesis is hereby officially accepted in partial fulfillment of the requirements for the degree of Bachelor of Science in Applied Physics (Instrumentation).

[illegible]
WILSON O. GARCIA, Ph.D.
Director
National Institute of Physics

NATIONAL INSTITUTE OF PHYSICS
COLLEGE OF SCIENCE
University of the Philippines
Diliman, Quezon City 1101 Metro-Manila
Telephone No. 981-8500 Locals 3701/3704

January 11, 2022

Dr. Wilson O. Garcia
Director
National Institute of Physics
C.S., U.P. Diliman, Quezon City

Dear Dr. Garcia:

Below is the official action of the panel members in the undergraduate thesis defense of Ms. Maria Tisha C. Bagasala. The defense of the thesis entitled **"DYNAMICS OF BRAIN NEURONS USING A ONE-DIMENSIONAL PROBABILISTIC CELLULAR AUTOMATA MODEL"** was held via Zoom link: https://up-edu.zoom.us/j/83493909677 on Tuesday, 11 January 2022, 3:30 PM.

	Rating	
Panel Members	PASSED	FAILED
Vince Paul P. Juguilon, B.Sc. Instructor of Physics National Institute of Physics C.S., U.P. Diliman, Quezon City	[illegible]	
Kimver Louie S. Nuñez, M.Sc Instructor of Physics National Institute of Physics C.S., U.P. Diliman, Quezon City	[illegible]	
Johnrob Y. Bantang, Ph.D. (Adviser) Professor of Physics National Institute of Physics C.S., U.P. Diliman, Quezon City	[illegible]	

The rating is therefore passed.

Sincerely yours,

CVillagonzalo

CRISTINE DLR. VILLAGONZALO, Dr.rer.nat.
Deputy Director for Academic Affairs, NIP

NATIONAL INSTITUTE OF PHYSICS
College of Science
University of the Philippines
Diliman, Quezon City

ANNOUNCEMENT OF THE UNDERGRADUATE THESIS SEMINAR

of

MARIA TISHA C. BAGASALA

on

DYNAMICS OF BRAIN NEURONS USING A ONE-DIMENSIONAL PROBABILISTIC CELLULAR AUTOMATA MODEL

In partial fulfillment of the requirements for the degree of
Bachelor of Science in Applied Physics
on Tuesday, 11 January 2022, 3:30 PM

To participate in the online thesis/dissertation defense, please register at:
Zoom link: https://up-edu.zoom.us/meeting/register/tZcpfuqhrjIsHdPQI7_z4mFqSOKZUPLobLuo

THESIS ADVISER

JOHNROB Y. BANTANG, Ph.D.
Professor of Physics
National Institute of Physics
C.S., U.P. Diliman, Quezon City

THESIS PANEL MEMBER

VINCE PAUL P. JUGUILON, B.Sc.
Instructor of Physics
National Institute of Physics
C.S., U.P. Diliman, Quezon City

THESIS PANEL MEMBER

KIMVER LOUIE S. NUÑEZ, M.Sc.
Instructor of Physics
National Institute of Physics
C.S., U.P. Diliman, Quezon City

ENDORSED BY:

CVillagonzalo

CRISTINE DL R. VILLAGONZALO, Dr.rer.nat.
Deputy Director for Academic Affairs
National Institute of Physics
C.S., U.P. Diliman, Quezon City

AUTHORIZED BY:

[illegible]

WILSON O. GARCIA, Ph.D.
Director
National Institute of Physics
C.S., U.P. Diliman, Quezon City

Acknowledgments

I would like to deeply express my thanks and gratitude to the following:

Dr. Johnrob Bantang for his dedication and interest in helping me develop the attitude of a researcher and his continuous support, patience, motivation and knowledge that carried me throughout the stages of doing this research.

Mama Timmy and Papa Bert for their support and understanding of my decisions and choices in my college life.

Complexity Science Group (CSG) for the valuable experiences that I am able to gain and share with the other members. To my seniors: **Kuya Reinier** who really helped me accomplished this thesis by providing helpful insights related to the topic. I can say that he is like my second mentor in the group; **Fritz, Matt and Sarah** for the fun interactions that we've had during the research meetings, workshops and even events outside NIP.

"Part-timerz" Karen and Giselle for all the fun that we've had pre-pandemic.

Almighty Father I would like to thank You for the wisdom, the strength, and good health that You bestowed upon me in order to help me finish this thesis

And for all those people that I've met and shared experience with back in UP, I am thankful for the learnings and fun that we've had.

ABSTRACT

DYNAMICS OF BRAIN NEURONS USING A ONE-DIMENSIONAL PROBABILISTIC CELLULAR AUTOMATA MODEL

Maria Tisha C. Bagasala
University of the Philippines (2022)

Adviser:
Johnrob Y. Bantang, Ph.D.

Cellular automata (CA) models that incorporate probabilistic transitions can be a powerful modelling tool for fast simulation of the dynamics of the brain neurons. Such CA is called a probabilistic CA (pCA) and a more generalized CA model where the designed rules mimic the transmission of the action potential from one neuron to another. Generally, signals that goes from a presynaptic to a postsynaptic neuron will create an action potential once a certain membrane threshold voltage is met. The action potential can be divided into two general states: excitatory and inhibitory, which are based on the measured membrane voltages of the neuron. We used pCA to model this activity by considering a circular one-dimensional CA grid with $N = 100$ neurons for $t = 1000$ timesteps considering two probability rates relative to the state of the action potential: a as our activation rate and b as the deactivation rate. CA states are limited to active or inactive excitatory neurons and the evolution of states are defined by the following set of rules. If a neuron at time t is inactive and at least one of its neighbors are active, then it becomes active with a probability a at $t + 1$. However, if a neuron at time t is active, it will transition to inactive with a probability b. Using this setup, we analyze the dynamics of the brain neurons by looking into: (1) the cellular automata evolution for fixed a and b; and (2) time series analysis plots for the density of the active neurons (Φ) and present its phase transition diagrams. We also compare the simulated phase transition to a mean-field analysis. It has been noted that the mean-field analysis is limited to a single-cell approximation and shows discrepancies from the numerical simulation. Thus, higher order mean-field analysis is recommended to improve the difference in the results. Critical points and directed percolation values are presented and compared to well-known models.

PACS: 87.19.ll (Models of neurons), 87.10.Mn (Stochastic modeling), 68.35.Rh (Phase transitions and critical phenomena)

Table of Contents

List of Figures

List of Symbols

$\Delta u_i(t)$	postsynaptic potential
$u_i(t)$	input membrane potential of neuron i at time t
u_{rest}	resting membrane potential
$t_j^{(f)}$	a presynaptic neuron j fires spikes at fth iteration
x_i	ith neuron activity
c_{ij}	coefficient of neural network equation
S	saturation (gain) function
D_{max}	deterministic operator of a percolation process
R_p	random operator of a percolation process
Φ	mean density
$\mathbb{L}$	lattice set
P	transition function
a	probability activation rate
b	probability deactivation rate
m	slope of the transition curve of the phase diagram
β	universal directed percolation
$\mathbb{P}$	probability distribution
$\mathbb{S}$	state set
$\mathbb{N}$	neighborhood

Chapter 1

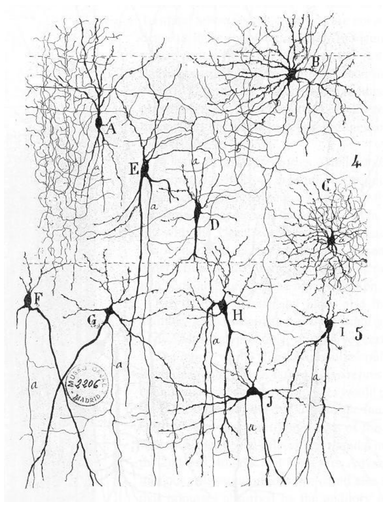
Neuronal Systems

Numerous studies have already contributed to our knowledge of the human brain's anatomy and complexity [36, 47]. Because of the pioneering works on neuroscience from the 1900's [36], we are now able to determine different types of neuronal cell bodies and their synapses. Despite being situated in a single organ, one key difference of the structure of neurons as compared to the other cells of the body is their functional specialization, which when combined in groups results to different functional regions in the brain [7]. In reality, this is just a part of a vast network of neurons with varying forms and functions. We now know that the cortex is also composed of a large number of glia cells which are limited by energy distribution and structural stability of brain tissue, but are not involved in information processing [16]. However, since we are only interested on how signals are being transformed and communicated between neurons, we will look more into what is called the spiking neurons.

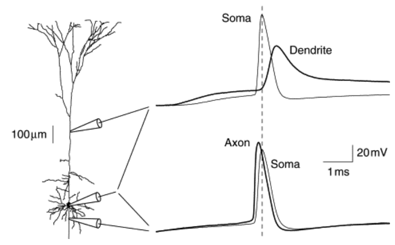
1.1 Spiking neurons

Morphologically, the three regions of a neuron is composed of the dendrites, soma and axon as shown in Figure 1.1. We can summarize their functions by thinking of the dendrites as the "input device", soma as the "central processing unit" and axon as the "output device". Basically, signals from other neurons enters the cells through the dendrites, then it travels to the soma to perform a nonlinear process wherein if the total input is greater than a certain threshold, then an output signal is generated. This output signal will then be delivered by the axon to the other neighboring neurons [16].

A synapse is a gap between two neurons, which is the junction where neurochemi-



(a)



(b)

Figure 1.1: (a) Design of the neuronal morphology by Santiago Ramón y Cajal, distributed under public domain [6] and (b) Voltage spike along the axon and soma of a neuron, distributed under CC BY-SA 4.0 [35]. Reprinted from Wikimedia Commons. These illustrations provide foundations for our modeling and understanding of neuronal anatomy and function.

cals pass through whenever these neurons communicate with each other. On average, 100 billion neurons communicate with tens of synapses each neuron. These signals commonly flow one way from pre-synaptic to post-synaptic receptors [20].

1.2 Communication between neurons

We can quantitatively measure how these neurons communicate by looking at the potential difference or the membrane potential between the interior of the cell and its surroundings. At rest or when there is no electrical input, the membrane potential is around -60 to -75 mV [7, 16, 30]. Once signals pass through the input of the neuron, its potential changes until it goes back to its resting potential as shown in Figure 1.2. In this figure, we can see the detailed variations of the membrane potential depending on the state of the neuron [17]. A positive change in the potential means that the synapse is excitatory while a negative change means that the synapse is inhibitory [16].

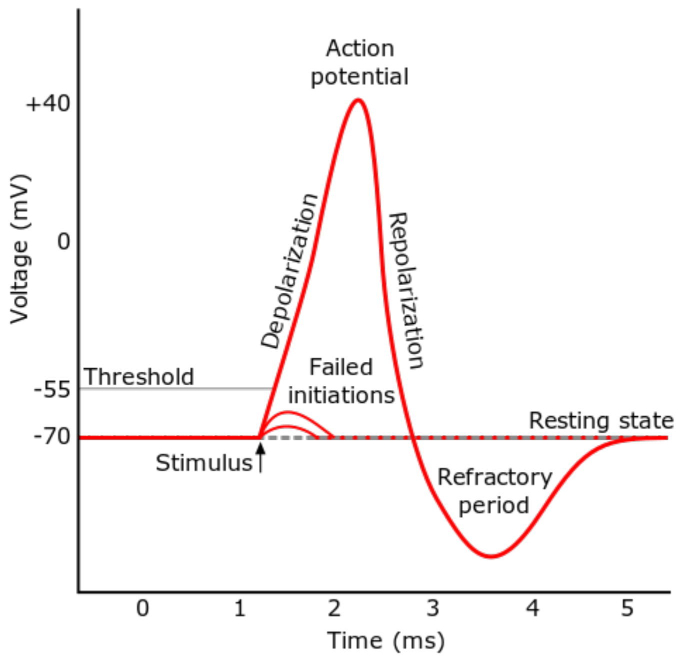


Figure 1.2: Levels of the membrane voltage potential, V_m, as signals travel from the dendrites to the neuron's body. At rest, the neuron's potential is around -70 mV. Synaptic inputs to the neuron cause the membrane potential to increase. When it reaches a particular threshold value, it initiates an action potential, which causes the voltage to rise (depolarization) and then fall (repolarization) until it returns to its resting state. Image is obtained from Wikimedia Commons CC BY-SA 3.0 [39]

1.3 Ionic concentrations in neurons

A more detailed definition of the potential difference can be seen in the ionic concentrations inside and outside of the neurons. There are four ions responsible for the active potential difference of the neuron with its environment: Na^+, K^+, Cl^-, Ca^+. Potassium ions are more concentrated inside the cell, while Sodium, Chloride and Calcium ions are active outside the cell. These ions move down towards their negative concentration gradient, subsequently increasing the electropotential gradient in the opposite region. The equilibrium potentials for these ions were demonstrated in Figure 1.3 [7].

Observed in the illustration Figure 1.3, the Na^+/ K^+ - ATPase pumps releases three Na^+ for every two K^+ ions at a cost of one ATP molecule [3, 7]. The depolarization of the membrane (resting potential exceed the activation threshold) happens when the Na^+ channels open. After some time, these Na^+ channels will close and K^+ ion channels will be activated (hyperpolarization) until the neuron's state stabilizes

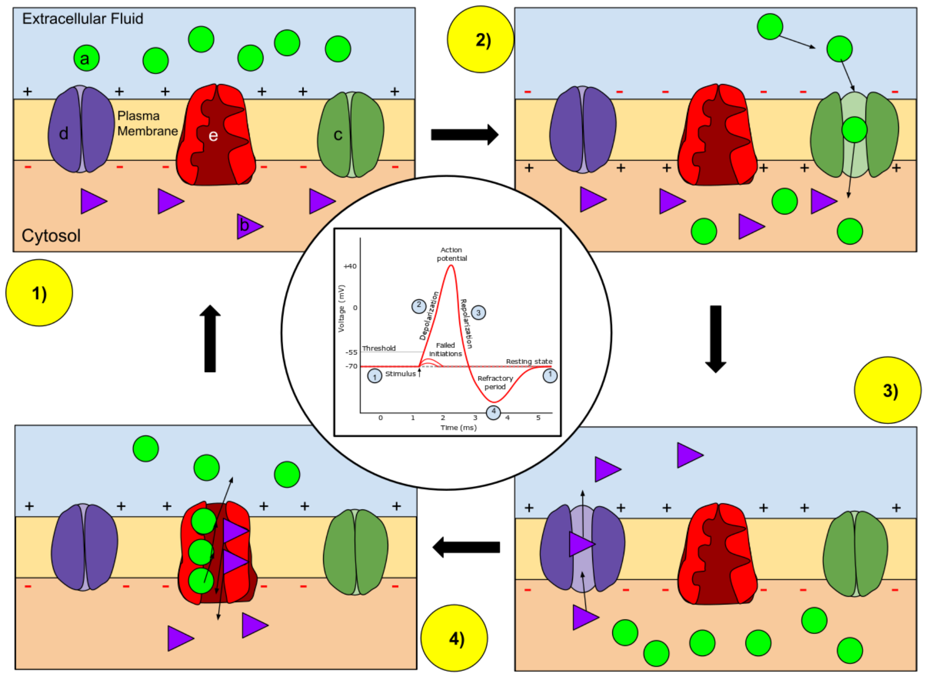


Figure 1.3: Ion concentrations inside and outside the cell that determines the neuron's membrane potential. At rest (1), there are more sodium (Na+) ions (labelled as a) outside the cell than inside the cell, which explains the neuron's initial negative charge. Once the action potential is initiated, depolarization (2) opens the Na+ channel (labelled as c), enabling Na+ ions to flow through the membrane, resulting in a positive charge in the neuron and a negative charge in the extracellular fluid. After reaching the action potential, the neuron undergoes repolarization (3), during which the Na+ channels close and the Potassium (K+) channels (labelled as d) open, allowing K+ ions to cross the membrane and go into the extracellular fluid, creating a positive charge in the extracellular fluid and a negative charge, lower than the neuron's resting potential. Finally, when the K+ channels shut during the refractory period (4),the membrane potential returns to its resting state. Image is created by CThompson02, distributed under CC BY 4.0 via Wikimedia Commons [10].

back to its resting potential.

1.4 Synaptic potential and threshold

We can denote the membrane potential of neuron i at time t to be $u_i(t)$. Before the arrival of the input spike, $u_i(t)$ is equal to the resting potential, u_{rest}. Specifically, at $t = 0$, a presynaptic neuron j will initiate a spike and as $t > 0$, neuron i will have a potential

$$\Delta u_i(t) \equiv u_i(t) - u_{\text{rest}} \tag{1.1}$$

where $\Delta u_i(t)$ is the postsynaptic potential (PSP). In this case, if Eq. (1.1) > 0, then we have an excitatory PSP. If it is negative, then we have an inhibitory PSP[16].

Once this potential reaches a certain threshold value, it will follow an *all-or-none* principle where there is action potential produced and propagated, while none will be initiated if the spikes are below threshold. Threshold values vary per neuron and over time as a function of the previous stimulations [29].

By including the input spike, u_i, from Equation **??**, we now have a neural network model of the neuronal activity with the form

$$\dot{x_i} = -x_i + S\left(u_i + \sum_{j=1}^{n} c_{ij}x_j\right), \qquad x_i \in \mathbb{R}, \qquad i = 1, ..., n \tag{1.2}$$

where each scalar x_i indicates the ith neuron activity; the coefficients c_{ij} where $i \neq j$ denotes synaptic connections, while c_{ii} is the feedback parameter, and S as the saturation (gain) function [13]. An example of the function is

$$S(x) = \frac{1}{1 + e^{-x}} \tag{1.3}$$

This model tells us that the greater the amount to which excitation converges on the neuron, then, the more active it is.

1.5 Excitatory and Inhibitory Neurons

Based on our earlier discussions, we now realized that a neuron may exist in one of two states depending on the degree of its membrane potential: excitatory or inhibitory. The excitatory and inhibitory neurons are referred to as the neural oscillators because the excitatory neurons stimulate the inhibitory neurons, which then inhibit the excitatory neurons reciprocally. This is thought to be one of the fundamental processes behind the generation of oscillatory activity in the brain [13].

The dynamics of these two states can be understand by using Equation 1.2 for only one neuron such that

$$\dot{x} = -x + S(u + cx), \qquad x, u, c \in \mathbb{R} \tag{1.4}$$

We can see that as the input approaches $-\infty$ (towards inhibition), the neuron activity $x \to 0$ (hyperpolarization). However, when the input is very excitatory, $u \to +\infty$, the activity becomes depolarized ($x \to 1$) [13].

1.6 Bifurcations in the neuronal dynamics

In dynamical system analysis, bifurcations are important because they reveal qualitative changes in a system's behavior. The shift from resting state to a single spike is a classic example of a bifurcation in neuron dynamics. In order to describe the bifurcation of a system, we can consider an ordinary differential equation (ODE) such as

$$\dot{x} = F(x, \lambda), \qquad X \in \mathbb{R} \tag{1.5}$$

where F is the function of position x and λ parameter value. A parameter value is said to be regular or nonbifurcational if the defined neighbourhood of the parameter are topologically equivalent. As a result, the qualitative behavior of this dynamical system is consistent for all λ values close to the nonbifurcation value. Therefore, a bifurcation point $\lambda = \lambda_b$ of a dynamical system is defined such that its qualitative behavior is different from some λ_1 in a system. With this, we can go back into Equation 1.4 and note that when the external input, u, takes on intermediate values, the dynamics may show bistability, with the activity of the neuron x exhibiting three equilibrium values if c (nonlinear term) is sufficiently big [13]. Note that when a parameter change affects the stability of an equilibrium (or fixed point), it is called local bifurcation [1].

Thus far, we have explored bifurcations using ODEs in a continuous-time system. Now, we may apply this concept to the description of discrete-time bifurcations by reducing the ODEs to mappings of form

$$x \mapsto F(x) \tag{1.6}$$

or

$$x^{t+1} = F(x^t) \tag{1.7}$$

where x^t denotes the tth iteration of variable x [13].

There are different types of local bifurcations for a discrete-time dynamical system such as: saddle-node, pitchfork and hopf. To be classified for each type of bifurcation, a dynamical system must satisfy specific conditions. For instance, the pitchfork bifurcation is has a nonzero cubic term as its $F(x)$ characterized by system transitions from one fixed point to three fixed points. We can see this in Figure 1.4 where $x = 0$ is always a fixed point of this mapping. Depending on the equation of $F(x)$, there are

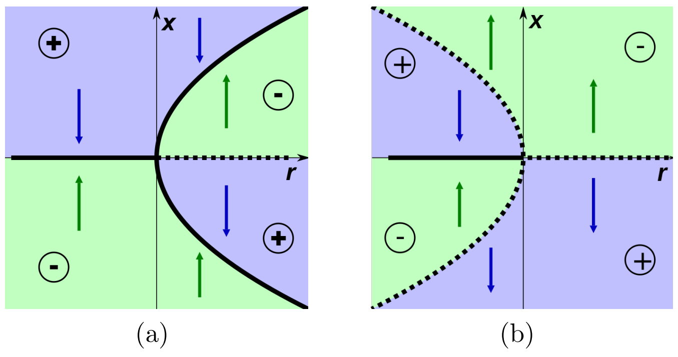


Figure 1.4: A pitchfork bifurcation diagram illustrating a system transition from one fixed point to three fixed points, where (a) represents a supercritical scenario with two stable points at $x \pm \sqrt{r}$ and one unstable point at $x = 0$, and (b) represents a subcritical example with two unstable points at $x \pm \sqrt{-r}$ and one stable point at $x = 0$. Stable points are shown by solid lines. whereas the dotted line indicates an unstable on, distributed by Claudio Rocchini, CC BY-SA 3.0, via Wikimedia Commons [31]

two cases for this bifurcation which are either subcritical or supercritical. Supercritical means that there is one stable equilibrium at $x = 0$. However for $r > 0$, there is unstable equilibrium at $x = 0$ and two stable equilibria at $x = \pm\sqrt{r}$. Meanwhile for subcritical case, the equilibrium is stable at $r < 0, x = 0$ and unstable at $x = \pm\sqrt{-r}$. For $r > 0$, the equilibrium at $x = 0$ is unstable [34].

We have now discussed the basic processes that happens within the neuronal systems, from the way the action potential is formed down to the microscopic activities that contributes to the transmission of signals from one neuron to another. We also include several models that are used in order to analyze such system. In the next chapter, we will discuss a possible model that uses cellular automata for a faster simulation of the dynamics of the neurons.

Chapter 2

Probabistic Cellular Automata

In this chapter, we will introduce a model that will be used for the simulation of the transition of the neurons from quiescent to excited state and vice versa which incorporates randomness in its rules.

2.1 Cellular automata

Cellular Automata (CA) are discrete-time lattices of interconnected cells that iterates according to a defined set of rules and the states of the neighborhood cells [11]. The first use of this algorithm can be seen in the works of von Neumann (1948), Ulam (1952) [27, 38] and even the 1946 paper by Wiener and Rosenbluth for their model on the impulse condition in cardiac systems [41]. Despite having simple local rules, the dynamics of a CA exhibits a wide range of patterns and structures in such a way that it has been used for different applications in various areas. Classifications of one-dimensional CA were documented by Wolfram in the 1980's [42] and cellular automata were even extended to a two-dimensional automaton by Conway's Game of Life in the 1970's [15].

An elementary cellular automata (ECA) is a one-dimensional CA composed of binary states depending on a given model application such as: 1 or 0, dead or alive, etc. For this kind of CA, a neighbourhood would be comprised of three cells, the current cell and its left and right neighbours with $2^3 = 8$ possible patterns. The rules for each pattern will decide the state of the cell in the next generation. According to Wolfram in his book *A New Kind of Science* [45], there are 256 possible configurations or rules in a given 1D lattice.

These rules follow a naming convention proposed by Wolfram called the Wolfram code. Basically, this code is represented by a k-digit number in an $S-$nary positional number system where S is the number of states, $k = S^{(n+1)}$ is the number of neighbourhood patterns and n is the radius of the neighbourhood. Thus, the Wolfram code for a particular rule will be a decimal notation from 0 to $S^k - 1$. In the case of a 1D CA with binary states, we have $S = 2, k = 2^3 = 8$ which will give a decimal representation from 0 to 255. Figure 2.1 shows an illustration of the transition rule of an ECA and the corresponding way to name such rule (via decimal representation). We can see from the figure that the collection of digits $[c_7c_6c_5c_4c_3c_2c_1c_0]_2$ are the binary equivalent of the current state in the center.

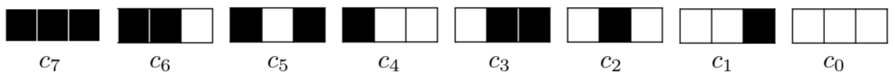


Figure 2.1: Graphical representation of a one-dimensional Wolfram rules where the shaded region is equivalent to state 1 and the opposite is state 0. The combination of digits $[c_7c_6c_5c_4c_3c_2c_1c_0]_2$ below is the binary value of the rule configurations [45]

2.2 Wolfram ECA Classifications

Wolfram categorizes elementary cellular automata (ECA) according to their behavioral characteristics, in order to better grasp their universality and complexity. Wolfram postulated that all ECA fall into one of four behavioral categories [44]. His classification is based on the spatial–temporal patterns generated from a finite initial configuration as shown below[43]

- Class 1: These rules result in a homogeneous state after some finite number of time steps, which also suggests that the majority of these CA are irreversible, since the original configuration information is lost.
- Class 2: Almost all initial configurations exhibit periodic structures in the space–time diagrams that follow these rules. The initial state information is not completely lost, since the density of certain sequences in the initial configuration has an effect on the statistical properties of the final configuration.
- Class 3: These rules result to a chaotic, aperiodic structures in the majority of its initial configurations. The statistical features of the CA attain

an equilibrium state exponentially with time. In general, these statistical features are constant across all beginning configurations.

- Class 4: We may see some stable and propagating structures in the patterns produced by certain initial configurations under these rules. Class IV is defined by the formation of interacting long-lived structures throughout its CA evolution, a behavior known in 2D CAs (e.g. Game of Life) as "glider dynamics". [40, 43, 46].

2.3 Basic stochastic models

A Markov process describes a series of potential occurrences in which the probability of each event is solely determined by the state obtained in the preceding event. A countably infinite sequence with discrete time steps results in a discrete-time Markov chain (DTMC) [14]. As an example of this, we will look into a basic stochastic model with absorbing states and understand some of the properties and techniques used in studying discrete-time processes.

Percolation operators (PO) are Markov processes that consists of units with either one of two individual states [2]. The states transition which can be: "death" ($1 \to 0$) or "birth" ($0 \to 1$) are independent given the previous timesteps and transition ($0 \to 1$) can happen given that a neighbor of a previous step has a state of 1 [2, 32].

One feature of the POs is their non-ergodicity [22] which means that they can allow two different stationary states in the system, which can be: (1) 'all 0s' and (2) 'mixture of 0s and 1s'. All zero states are also called the absorbing state and the final state will depend on the initial state and the parameters of the system.

The transition probabilities in this process can be generalized into a deterministic operator $D_{\max}$ and a random operator R_p. In his paper, Slowinski (2018) described $D_{\max}$ to be the maximum function $f\colon \{0,1\}^n \to \{0,1\}$ of site x situated in the neighbourhood $\mathbb{N} = \{x-1, x, x+1\}$. Meanwhile, R_p assigns a certain probability of turning each 1 into 0. Percolation operators for different are autonomous, conditionally of the past behavior. In other words, at each time step, each site observes the greatest value in its neighborhood and either changes its state in accordance with the output of the deterministic operator or produces an error with probability p [32].

In a given neighbourhood where the immediate neighbors were the left and right cells, we can visualize the schematic action of a 1D $PO_{n=3}$ as shown in Figure 2.2. The description of the deterministic component of the percolation operator, $D_{\max}$, may be interpreted in terms of the neighbourhood's action on the site (as represented by the red triangle) or the site's action on its neighbourhood (as illustrated by the blue triangle). [32].

A description of stochastic models as a combination of deterministic and stochastic operators is quite useful for comprehending their evolution [2]. We can look on the separate actions of the operators $D_{\max}$ and R_p to see how PO acts on the whole grid. $D_{\max}$ alone is an *eroder* if any site in state 1 becomes 'all 1s' as time goes to infinity. Meanwhile, operator R_p alone will turn the initial state to 'all 0s' as time tends to infinity. So, the mixture of both the actions of R_p and $D_{\max}$ will lead to an evolution that depends on the noise intensity and the initial state [32].

Therefore, any neighbourhood of $n \geq 2$ have a critical value $p^* > 0$ for which if $p^* > p$, then the lattice is ergodic. Else, it is considered to be non-ergodic [37]. That is, any combinations of the initial state will get absorbed as 'all 0s' state if p is greater than the critical value. Slowinski defined the upper and lower bounds for p^* as:

$$0.09 < p^* \leq 1 - \frac{1}{n} \tag{2.1}$$

where n would be the size of the neighbourhood. The author defined the lower bound to be based on the probability of percolation in the oriented site percolation process while the upper bound is based on the structure of tree-like graphs [2, 32, 37]. For example, a percolation operator with a neighbourhood of size 3 has a theoretical bounds ranging from $0.09 < p^*_{PO,n=3} \leq 0.495$

2.4 Probabilistic cellular automata

Probabilistic cellular automata (pCA) are generalized CA described as Markov chains that allow for random updating of the cell states. Probability distributions define the new values of each automaton based on the configurations of the cell's given neighborhood. In general, updates in each automaton are in parallel or synchronous with each other and neighborhoods are within finite sets [11].

The probabilistic component provides a realistic simulation to some of the complex activities in the real-world system like biological systems, neuronal networks and large systems of interacting automata [2]. PCAs are useful if we want to focus on looking

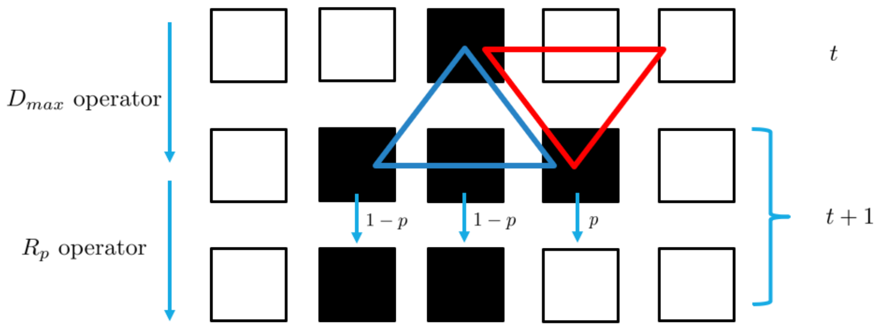


Figure 2.2: Time evolution of a one-dimensional Percolation Operator with n = 3 neighborhood. The deterministic operator, D_{max} can be defined by either the action of the site to its neighborhood (represented by the blue triangle) or the action of the neighborhood on the site (represented by the red triangle). Meanwhile, the random operator, R_p converts each 1 into 0 with probability p and has no effect on sites that are in the state 0. Figure is reconstructed from [33]

into the statistical mechanical and mathematical physics of these systems such as its phase transition and critical point. This is because the dynamics of the PCA is related to a *non-equilibrium lattice model*. In statistical physics, a lattice can be interpreted as a network of interconnected cells wherein the strength of an interaction is inversely proportional to the number of mediating links. This is similar to the notion of neighbourhood in PCA where two vertices are the nearest neighbours if they are at the end point of the link. As such, PCA can be used to model the *non-equilibrium* phenomena taking place during the evolution of this lattice towards the end equilibrium [11].

Chapter 3

Modelling action potential propagation

The purpose of this study is to provide a straightforward framework for modeling the dynamics of brain neurons using probabilistic cellular automata. We aimed to describe the overall characteristics of a linked collection of “neuron-like” elements using numerical simulations. All simulations are done using the PYTHON programming language[1]

3.1 Proposed model

A one-dimensional elementary probabilistic cellular automata can just include a few hundred sites but still can evolve into a steady-state with a given asymptotic mean density, Φ[28]. Our binary model consists of $\mathbb{L} = 100$ sites with periodic boundary conditions iterating over 1000 timesteps. In mathematical terms, let us denote the state of neuron i at time t to be $x_i(t)$. Since this is a two-state PCA, $x_i(t) \in \{0, 1\}$ such that neuron is active at time t if $x_i(t) = 1$, otherwise, inactive if $x_i(t) = 0$. Our lattice will be under periodic boundary conditions $i + \mathbb{L} \equiv i$ with 1D Von Neumann neighbourhood (immediate neighbors were the left and right neurons) as shown in Figure 3.1.

The evolution of the automaton is defined such as each neurons were updated synchronously by a given transition function, P where

$$x_i^{t+1} = P_i(\mathbf{x}^t) = P(x_{i-1}^t, x_i^t, x_{i+1}^t) \tag{3.1}$$

[1]Python Software Foundation. Python Language Reference, version 3.8. The program was run in a laptop PC with Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz (8 core), 2.6GHz and 8GB DRR4 RAM

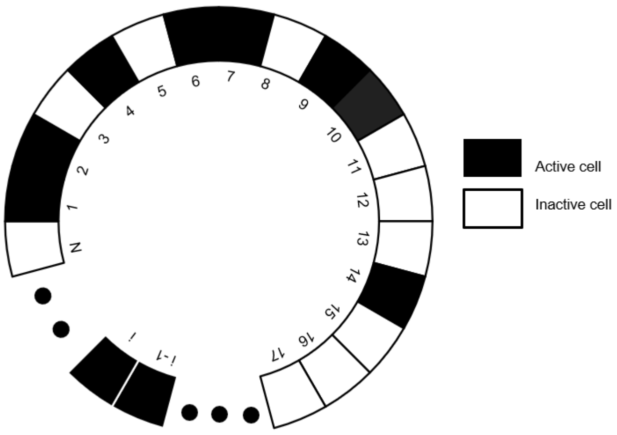


Figure 3.1: A single neural grid defined by a circular neighbourhood in which the shaded boxes correspond to active (excitatory) neurons and the unshaded boxes correspond to inactive (inhibitory) neurons. The label i represents the neuron id.

with the following probabilities a as the activation rate and b as the deactivation rate $\in [0, 1]$ as defined by the following rules:

- if the current state of the neuron at time t is active, $x_i^t = 1$, and at least one of its immediate neighbours are also active, then we have a probability a for $x_i^{t+1} = 0$
- if the current state of the neuron at time t in inactive, $x_i^t = 0$, then we have a probability b for $x_i^{t+1} = 1$

In terms of conditional probabilities, $W(x_i'|x_{i-1}, x_i, x_{i+1})$, we have $b = W(0|0,1,1) = W(0|0,1,0) = W(0|1,1,1) = W(0|1,1,0)$ and $a = W(1|0,0,1) = W(1|1,0,0) = W(1|1,0,1)$. We can further visualize our rules in Table 3.1 and Figure 3.2. The figure shows that for each rule configuration for our one dimensional lattice, the cell at the center maps to a certain state with a given probability a or b. This mapping is dependent on the actual activities of the neurons in our brain. The initial state of the lattice for each run is set to have a random distribution with probability $P = 1/2$ for a neuron to be active. Later on, this will be modified such that the initial P will be varied from [0,1].

	0	1	2
0	0	a	a
1	$1-b$	$1-b$	$1-b$

Table 3.1: Rule table for the 1D pCA. The left column gives the state (0 or 1) of a cell at time t, and the upper row gives the neighboring cells that are alive at time t. The probability rates inside are the probability for a given cell to become active at $t+1$.

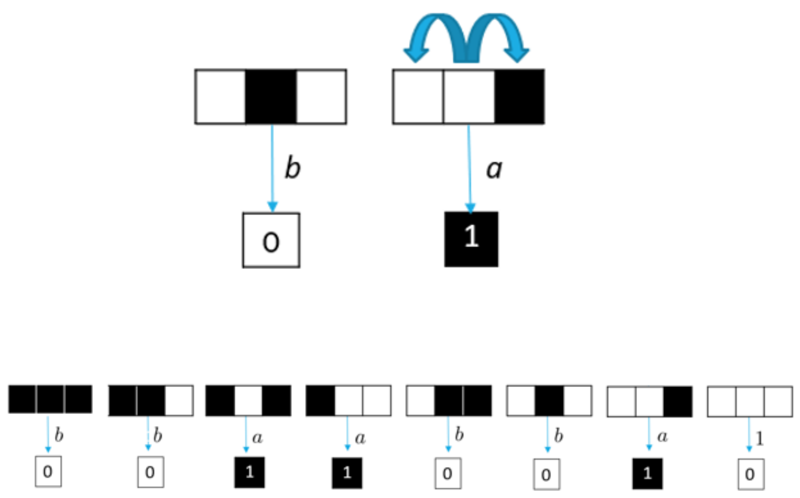


Figure 3.2: Graphical representation of the transition rules from state x_i^t to x_i^{t+1} where probability rates: a is applied for a current inactive neuron (unshaded box) to transition to an active neuron (shaded box) given the state of its neighbours (represented by the curved arrows), and b is applied for an active neuron to transition to an inactive neuron, independent of its neighbors. Meanwhile, a more generalized illustration below shows 8 possible configurations for a cell and its immediate neighbours and the probability rates for the next transition.

3.2 Time evolution of a pCA model

Following the rules set for our model, we can analyze the time series evolution of our model by considering the fraction of active neurons at a given probability rate.

3.2.1 Density of active neurons

Generally, a 1D CA quickly evolve into a state with density fluctuations about a given mean, Φ. Now that we have an overview of the time evolution of the CAs in our given set of rules, we now want to get the mean density of states as our parameter for the last $t = 20$ timesteps of the CA grid. Figure 3.10 shows the time evolution of the mean density with initial density equal to $\Phi_0 = 0.5$ (i.e., initial configuration with a random probability of 0.5 for a cell to be alive). Initially, we can see from the plot $b = 0, a \in [0, 1]$ that almost all of the lines have a $\Phi \geq 0.5$ However, as b increases, the density decreases to lower values until all of them will die at $b = 1.0$ even with the contribution of a.

3.2.2 Extreme cases

Depending on the extreme values of a and b, we can obtain four extreme ECA rules with their known properties. So, when $a = 0$ and $b = 0$, we get ECA Rule 204. It has been noted that ECA 204 is very sensitive towards perturbations, that is, active and inactive neurons will remain the same regardless of the state of its neighbours. Furthermore, when $a = 0$ and $b = 1$, we get ECA Rule 0. ECA Rule 0 leads to the absorption of the initial state to 'all 0s', meaning that the neurons will eventually die from just iterating on a few timesteps. When $a = 1$ and $b = 0$, we obtain ECA Rule 254. This rule is the opposite of ECA 0, with neurons becoming active with at least one neighbour. Lastly, when $a = 1$ and $b = 1$, we get ECA Rule 50. ECA Rule 50 shows that all alive cells in the initial configuration will die (become inactive). All dead cells with at least one neighbor will become alive.

a	b	Wolfram ECA Rule
0	0	204
0	1	0
1	0	254
1	1	50

Table 3.2: Four Wolfram ECA rules derived from the extreme cases of the probability rates a and b following the rules for our pCA model.

The summary of these rules are found in Table 3.2 while the evolution of the ECAs are shown in Figure 3.3.

3.3 Phase transition

To examine the transition from a phase with absorbing states ($\Phi = 0$) to a phase with complex properties ($\Phi > 0$), we consider the average of the last 20 time points corresponding to the steady state values of the generated time plots (i.e. $t = 980$ to

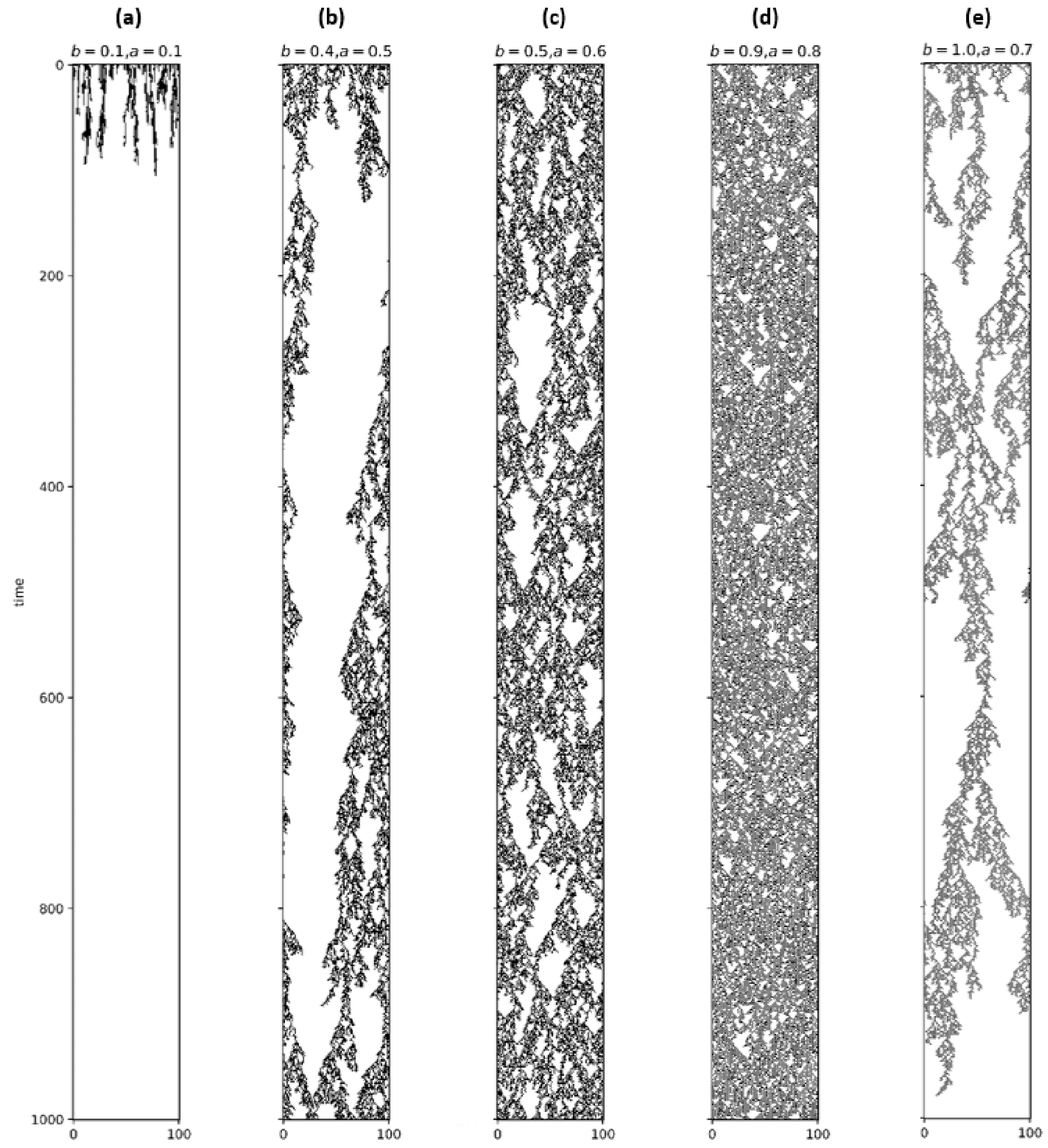


Figure 3.3: Time evolution of the cellular automata for a given value of the activation rate a and deactivation rate b. Simulation is set to have $N = 100$ neurons, $t = 1000$ timesteps

$t = 1000$). From this, we are able to get a phase diagram showing two regions where the transition happened as shown in Figure 3.4. To get the fitting equation, we set a threshold value, $\Phi \geq 0.1$, for non-zero densities ($\Phi > 0$). In Figure 3.4, we can see a quadratic transition curve which will separate the two observed phases. From this curve, we derived a numerical fitting equation to be

$$a = m(b-1)^2 + a_0 \tag{3.2}$$

where $a_0 = 0.71$ and $m = -0.68$. With the exception of CA 204, we can observe a continuous transition from $\Phi = 0$ to $\Phi > 0$. Using Equation 3.2, we can now

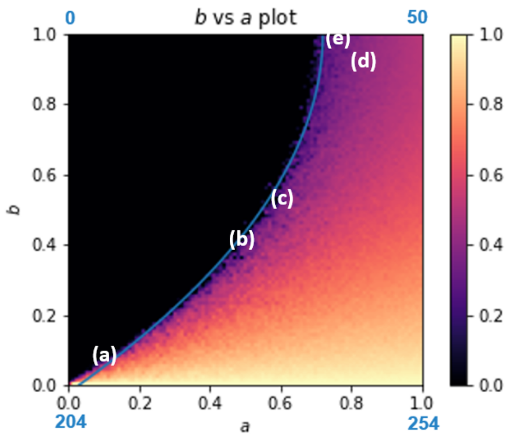


Figure 3.4: Phase diagram of the pCA model showing a phase transition from the inactive (dark region) to active state (colored region). Corner ECAs for special values of a and b are determined to be rules 0, 50, 204 and 254. The labels (a), (b), (c), (d) and (e) represents the single point average of the last 20 densities from Figure 3.3.

compute for the critical points, a_c, at constant b or critical points, b_c, at constant a. For example, when $b = 1$, the critical point is determined to be $a_c = 0.71$. This value is the maximum critical point for the series of a_c observed when b is constant as shown in Figure 3.5. Additionally, we noted that the figure resembles a bifurcation in neural dynamics. Consider this as a pitchfork supercritical bifurcation, in which density values tend to gravitate toward the stable equilibrium points at $x = 0$ and $x_{critical}$, which changes according to alpha and beta rates. We may begin by observing how $\phi = 0$ loses stability at varying critical points, resulting in the emergence of a new equilibrium that is equal to a quadratic equation's root. This is equivalent to a soft loss of stability [13].

We can now explain the activity of a single neuron or a limited group of neurons by defining the soft bifurcations in neural systems. When the neuron's equilibrium state, which corresponds to its resting potential, loses stability due to soft bifurcation, it is claimed that the neuron does not fire. It has no macroscopic effect on other neurons. Therefore, variations in the activity of such a neuron are graded. It is worth noting that when neurons communicate by action potentials, the network activity is insensitive to graded (nonspiking) neuronal activity, regardless of how strong the synaptic connections between neurons are. Thus, the soft bifurcation has an effect on the dynamics of a single neuron but not on the dynamics of the whole network. When neurons interact through electrical synapses, however, graded activity is critical. Even when the connections between neurons are weak, soft bifurcations may alter the dynamics of other neurons [13].

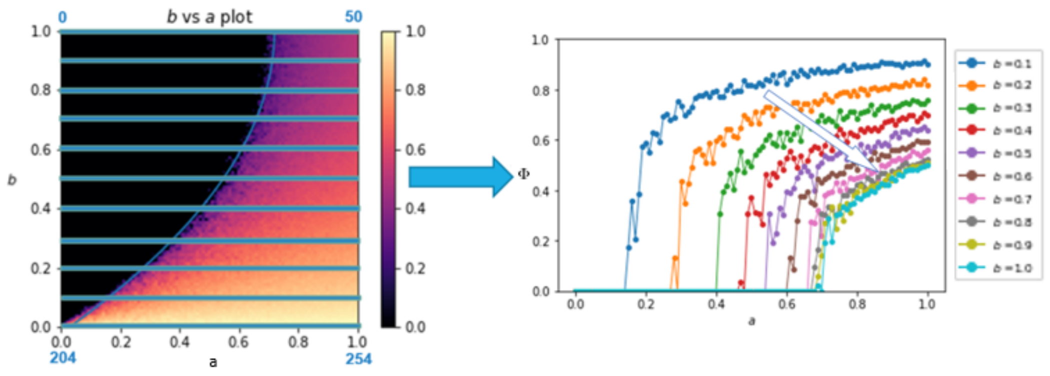


Figure 3.5: Density phase transition plot for constant b and variable a values $\in \{0, 1\}$. The arrow indicates that as b increases, the critical points also increase until they reach their maximum value at $a_c = 0.71$.

Since we already know how to determine the critical point from the fitting equation, we can focus specifically on the CA 0-50 transition ($b = 1$, varying a), with $a_c = 0.71$. With this value, a power law fit has been made as shown in Figure 3.6. It is found that $\Phi \propto (a - a_c)^\beta$ has an experimental directed percolation of $\beta = 0.319$ which is slightly greater than the results of Kinzel (1983) [21], $\beta = 0.2766$. It is noted that some of the papers that derived the numerical directed percolation for their specific systems has values of at least 10% deviation from value derived by Kinzel [8, 28]. Based on our results, we found that the computed β exponent can be considered to belong to the 1-dimensional directed percolation universality class of critical behavior [19].

We carried out similar analysis for the constant a with varying b values from $b \in [0, 1]$, which we can see in Figure 3.7. We divide the plot into two parts: (1) $a \in [0.1, 0.68]$, where there is an observed transition from a phase where $\Phi > 0$ to a phase where $\Phi = 0$ and (2) $a \in (0.68, 1.0]$ where there the value Φ decreases monotonically towards some constant Φ. For (1), we can see that there is a transition from active ($\Phi > 0$) to inactive ($\Phi = 0$) where the maximum critical point, $b_c = 0.78$

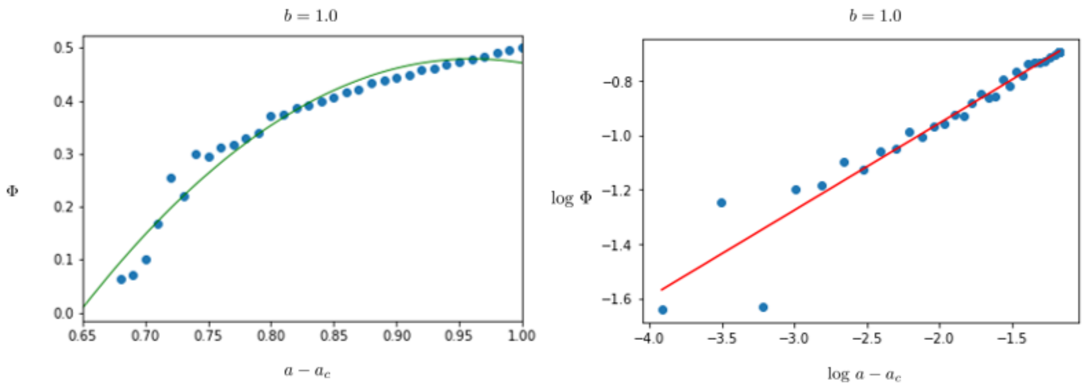


Figure 3.6: (Left) Density of active neurons (Φ) versus the activation rate (a) for $b = 1$ ($a \in [0.65, 1]$) where it focuses only on the active region of the phase diagram. (Right) Log-log plot of the active region where the computed $\beta = 0.319$

for $a = 0.68, b \in [0, 1]$. Meanwhile, (2) shows a continuous phase transition from a phase with constant density to a phase with density $\Phi > \Phi_{\text{constant}}$. Considering only the transition in the extreme ECAs 254-50 ($a = 1.0$ and b is varied), Φ decreases monotonically towards $\Phi \approx 0.5$.

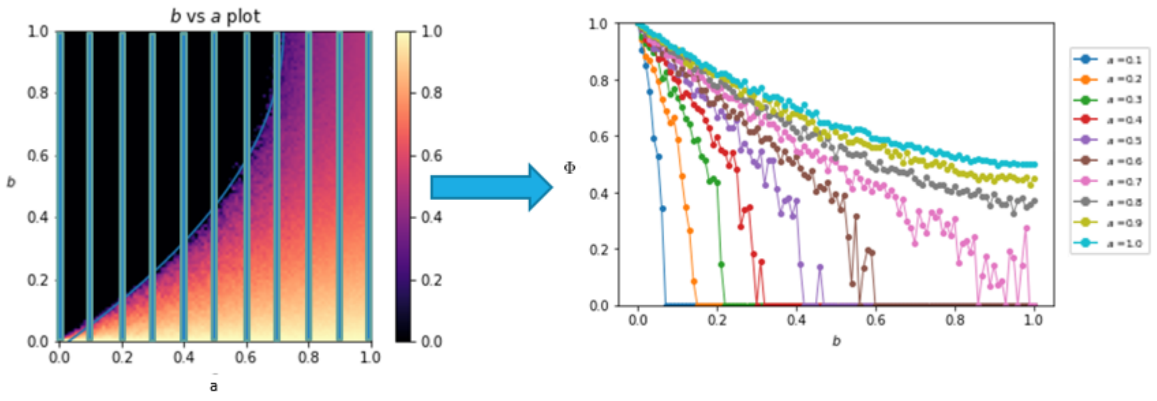


Figure 3.7: Density phase transition plot for constant a with variable b values $\in \{0, 1\}$.

Since we are more interested in the transition from inactive to active state, we go back to part (1) and determine its numerical percolation value from its plot as shown in Figure 3.8. By using linear regression, slope from $\Phi = A(b - b_c) + \Phi_0$ is computed to be equal to $A = 1.06$.

The system's general structure is seen in Figure 3.9. The previously discussed phase diagram is shown in this image, along with the temporal evolution patterns for

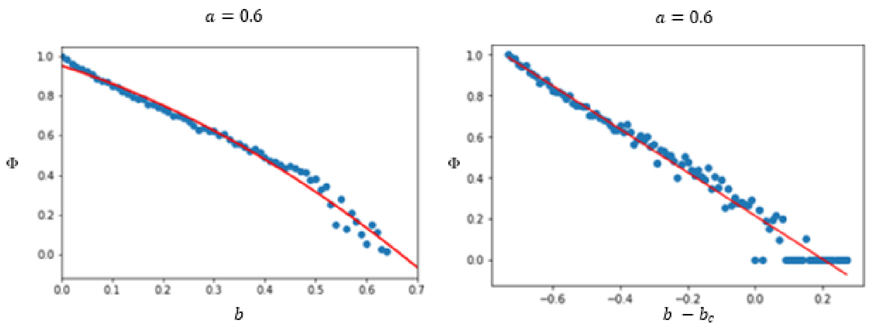


Figure 3.8: (Left) Density of active neurons (Φ) versus the deactivation rate (b) for $a = 1$ ($b \in [0, 0.7]$) where it focuses only on the active region of the phase diagram. The red curve follows the fitting equation presented in Equation 3.2 (Right) $\Phi versus$b-b$_c$ where the computed slope is 1.06.

random a and b. We can observe that there are certain interesting locations in the diagram (e.g., the plots around critical points) where the spatial length extends to the last iteration. However, there are times when the pattern stays constant throughout the simulation. This demonstrates that the pCA is a composite of the several Wolfram classes as explained before.

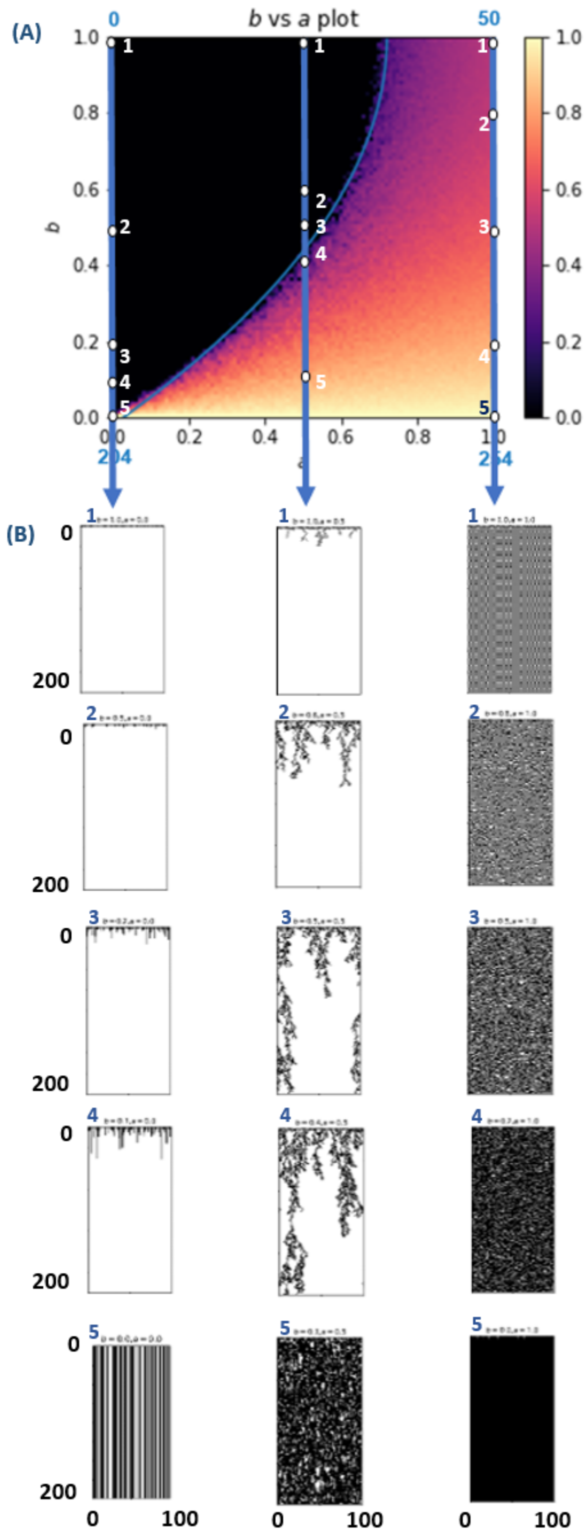


Figure 3.9: (A) Phase diagram of our pCA model with (B) random b and a values (as shown by the points in the arrow). Describe here are the extents of x and t in the space-time plot with initial neurons, $N = 100$, iterated over $t = 200$ timesteps.

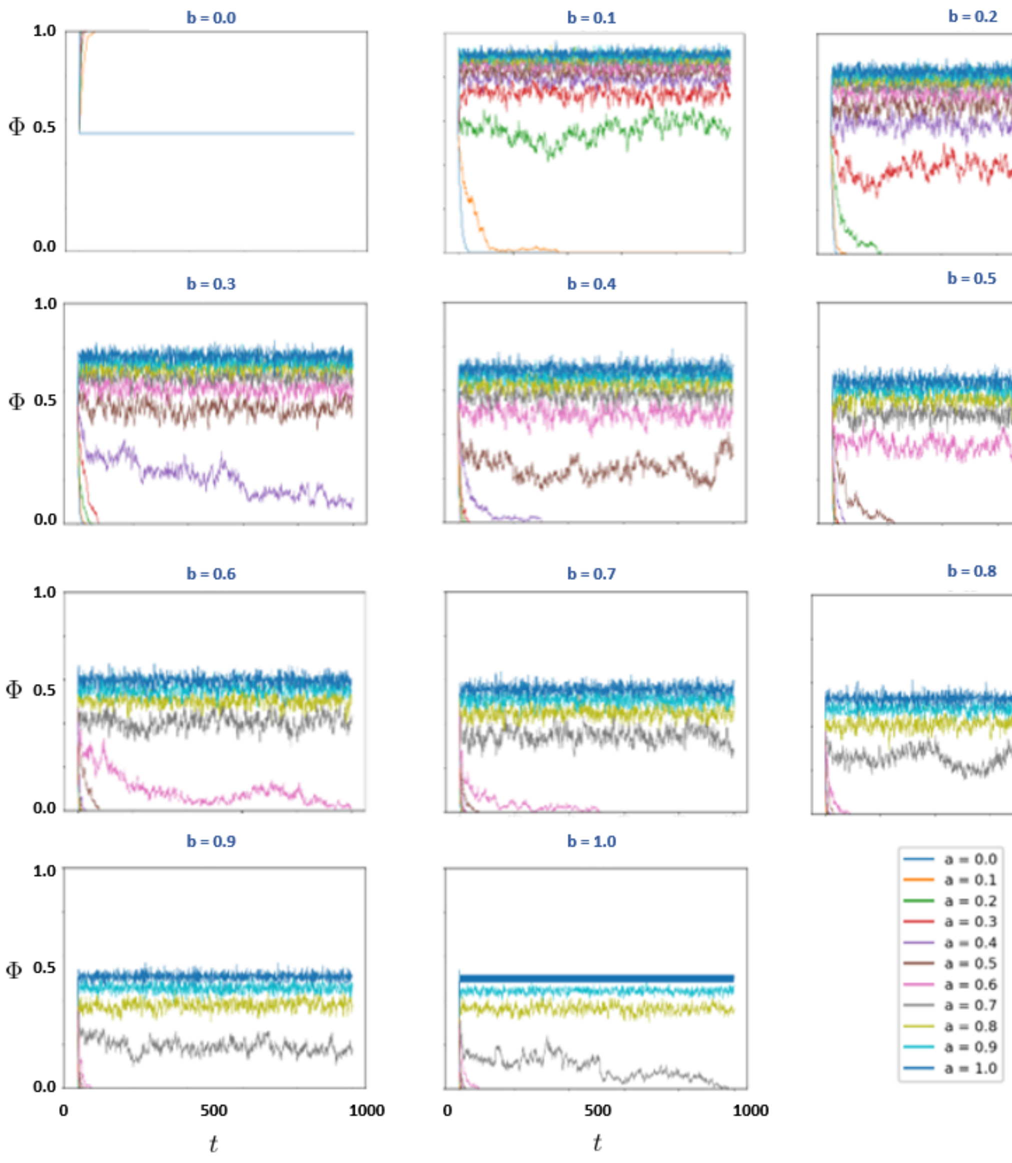


Figure 3.10: Density of active neurons over iteration numbers up to iteration $t = 1000$ for $N = 100$ neurons. Plots are selected where $a, b \in [0, 1]$ with a 0.1 increment.

Chapter 4

Mean Field Analysis

The mean field approximation, as defined by Balister et al. (2003), can be restricted into a 2-state process {0,1} with a semi-totalistic case such that p_Φ depends only on the cardinality of the active neighbors and on the state of the cell. Additionally, they noted that altering the initial density value might result in a mixture of stable fixed points, stable limit cycles, and chaotic behavior [4].

4.1 Application to Neuronal pCA

By following the derivation for the mean-field approximation by Mendonça [26], we can start by applying the Markovian dynamics over a probability distribution $\mathbb{P}_t(\overrightarrow{\mathbf{x}})$ where $\overrightarrow{\mathbf{x}} = \langle x_1, x_2, ..., x_i \rangle$ with $x_i \in \{0, 1\}$ so that

$$\mathbb{P}_{t+1}(\mathbf{x}') = \sum_{x} W(\mathbf{x}'|\mathbf{x})\mathbb{P}_t(\mathbf{x}) \tag{4.1}$$

where $W(x'|x)$ is the conditional probability for the transition $\mathbf{x} \to \mathbf{x}'$. If the CA is updated simultaneously and independently,

$$W(\mathbf{x}'|\mathbf{x}) = \prod_{i=1}^{L} W(x_i'|\mathbf{x}) \tag{4.2}$$

with

$$\sum_{x'} W_i(x'|\mathbf{x}) = 1 \tag{4.3}$$

The marginal probability distribution is now derived from Eqs [4.1,4.2,4.3] as[26]

$$\mathbb{P}_{t+1}(x_1', ..., x_n') = \sum_{x_1,...,x_{n+1}} \left[\prod_{j=i}^{L} W(x_j'|x_{j-1}, x_j, x_{j+1}) \right] \mathbb{P}_t(x_1, ..., x_{n+1}) \tag{4.4}$$

So, if the alive neurons have a given density, Φ at time t, then Φ_{t+1} will be the sum over the probabilities of combinations of alive cells as represented by Φ_t given by

$$\Phi' = (\Phi^3 + 2\Phi^2(1-\Phi) + \Phi(1-\Phi)^2)(1-b) + a(\Phi^2(1-\Phi) + 2\Phi(1-\Phi)^2) \qquad (4.5)$$

which is based from the rule table shown in Figure 3.2. To account for the steady-state densities, we set $\Phi' = \Phi = \Phi^*$ such as Eq 4.5 will be

$$\Phi^* = (\Phi^{*3} + 2\Phi^{*2}(1-\Phi^*) + \Phi^*(1-\Phi^*)^2)(1-b) + a(\Phi^{*2}(1-\Phi^*) + 2\Phi^*(1-\Phi^*)^2). \quad (4.6)$$

which can be simplified to this

$$\Phi^* = a\Phi^{*3} - 3a\Phi^{*2} + 2a\Phi^* + \Phi^* - b\Phi^* \qquad (4.7)$$

Rearranging the variables, we now have a polynomial equation such as

$$a\Phi^{*3} - 3a\Phi^{*2} + 2a\Phi^* - b\Phi^* = 0 \qquad (4.8)$$

Then, we factor out Φ^* to have

$$\Phi^*(a\Phi^{*2} - 3a\Phi^* + 2a - b) = 0 \qquad (4.9)$$

So, we now have two factors equating to zero: $\Phi^* = 0$ and $a\Phi^{*2} - 3a\Phi^* + 2a - b = 0$. Therefore, the solutions for Equation 4.9 are

$$\Phi^* = 0 \text{ or } \Phi^* = \frac{3a \pm \sqrt{a^2 + 4ab}}{2a} \qquad (4.10)$$

4.2 Density for varying a and b

Solution $\Phi^* = 0$ corresponds to a single state $\{\mathbf{0}\} = (0, ..., 0)$ or the inhibitory state of neurons. Meanwhile, $\Phi^* = \frac{3a \pm \sqrt{a^2+4ab}}{2a}$ has two possible solutions. The first solution, $\Phi^* = \frac{3a+\sqrt{a^2+4ab}}{2a}$, is greater than one for all probability rates a and b which is unphysical while the second one, $\Phi^* = \frac{3a-\sqrt{a^2+4ab}}{2a}$ is the solution we used for plotting the density for varying probability rates.

We used Eqs 4.10 to plot similar phase space diagrams for the mean field approximation and compare it with the numerical solutions as shown in Figure 4.1. We also extend the analysis to the density versus probability plots that we used earlier.

We extend the solution for the analysis of mean-field equations using varying initial densities. This was shown in Figures 4.2 and 4.3 for both a and b values.

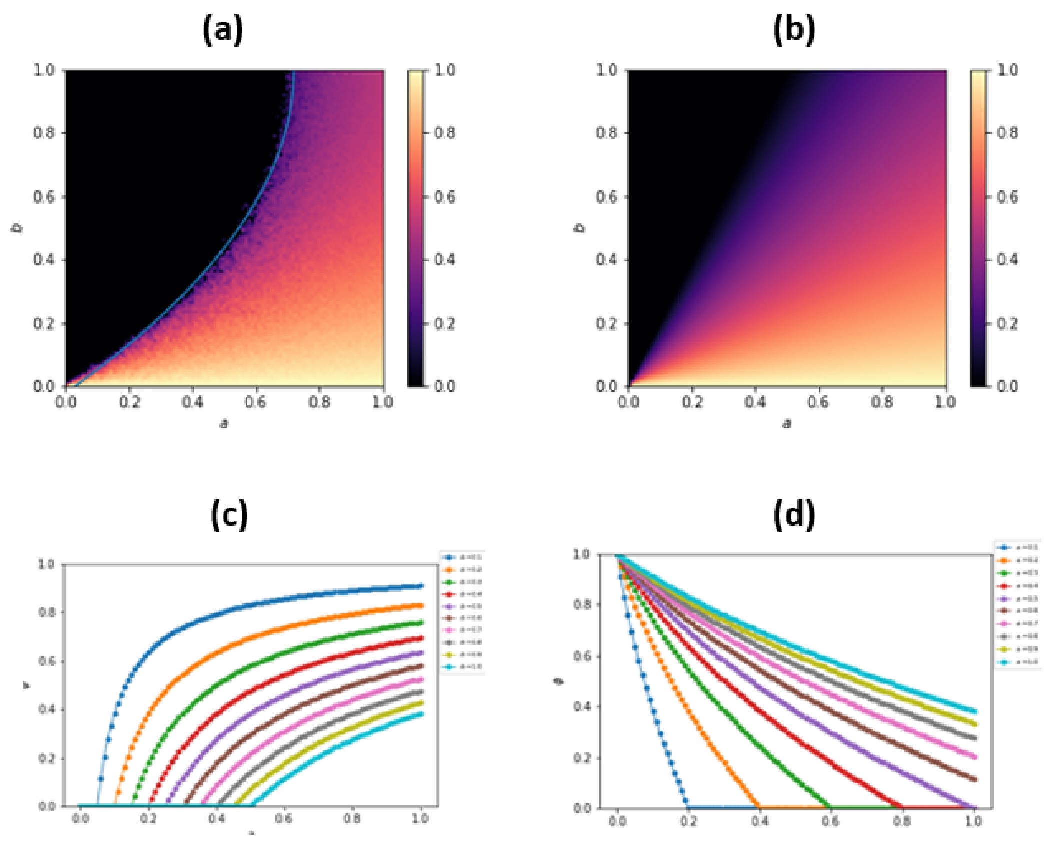


Figure 4.1: Comparison of the phase diagram through (a) pCA simulation and (b) mean-field analysis using the solutions derived from Eq 4.10. Similarly, (c) shows the density versus a for constant b and (d) shows the density versus b for constant a from these mean-field solutions.

4.3 Deviation from the numerical model

It has been highlighted that our resulting mean-field equations are analogous to the solution of Mendonça (2015)'s one-cell approximation, where they are able to derive a generalized approximation from the Markovian dynamics of the probability distribution $P_t(\mathbf{x})$ of the PCA states $x \in \mathbb{S}$ [26].

Since there are no significant differences between utilizing alternative sizes for the initial array of neurons ($\mathbb{L} = 50$, $\mathbb{L} = 100$, $\mathbb{L} = 500$), we chose $\mathbb{L} = 100$ for our numerical simulations. As seen in Figure 4.2, the mean-field findings are quite near to the actual simulation values for smaller values of the constant b (deactivation rate). Additionally, we included pCA with randomized neighbor to better approximate the mean-field assumption. However, discrepancies appear with high probability rates. This is in contrast to what we saw with constant a in Figure 4.3, when numerical simulations converged closer to the mean-field line as the activation rate was increased.

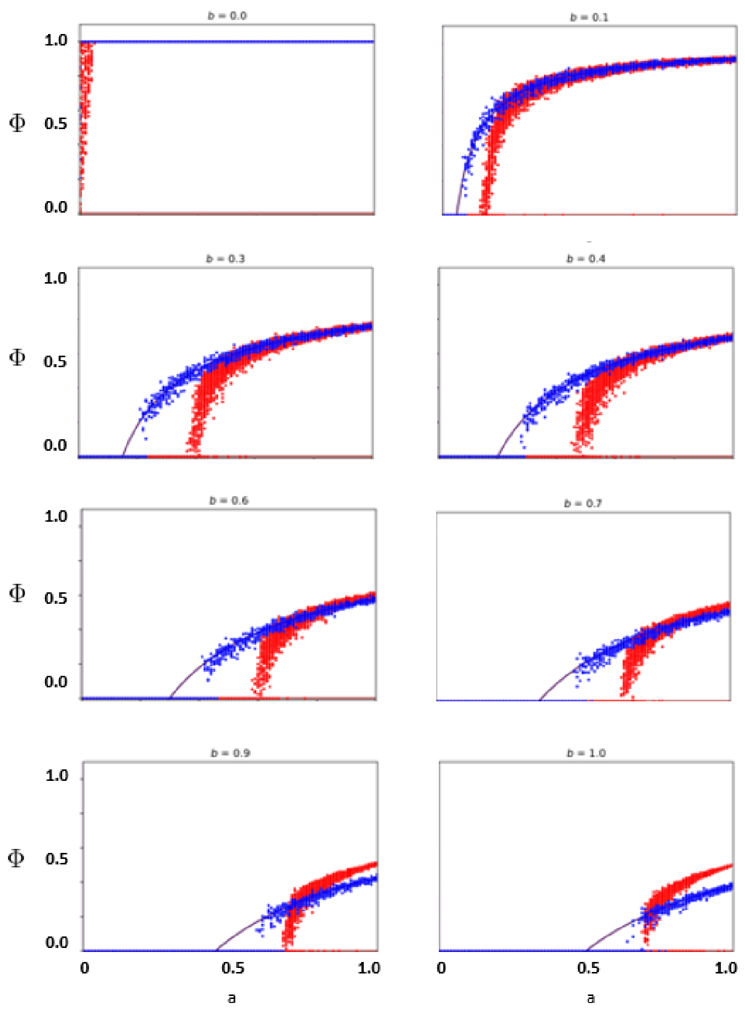


Figure 4.2: Density versus activation rates for fixed $b \in [0, 1]$ using varying initial density $\Phi_0 \in [0, 1]$ (in red scattered points) as compared to the solutions of the mean-field results (black line) and the density with random neighbourhood (blue scattered points)

Moreover, it has been observed that the deviation in the critical point from the mean-field approximation (using Eq 4.10 to the numerical value (using Eq 3.2) decreases as b increases. We can see that for $b = 1$, the ratio of the theoretical critical point to the empirical value is determined to be $\sim 0.5/0.71 \sim 0.704$, i.e. by just

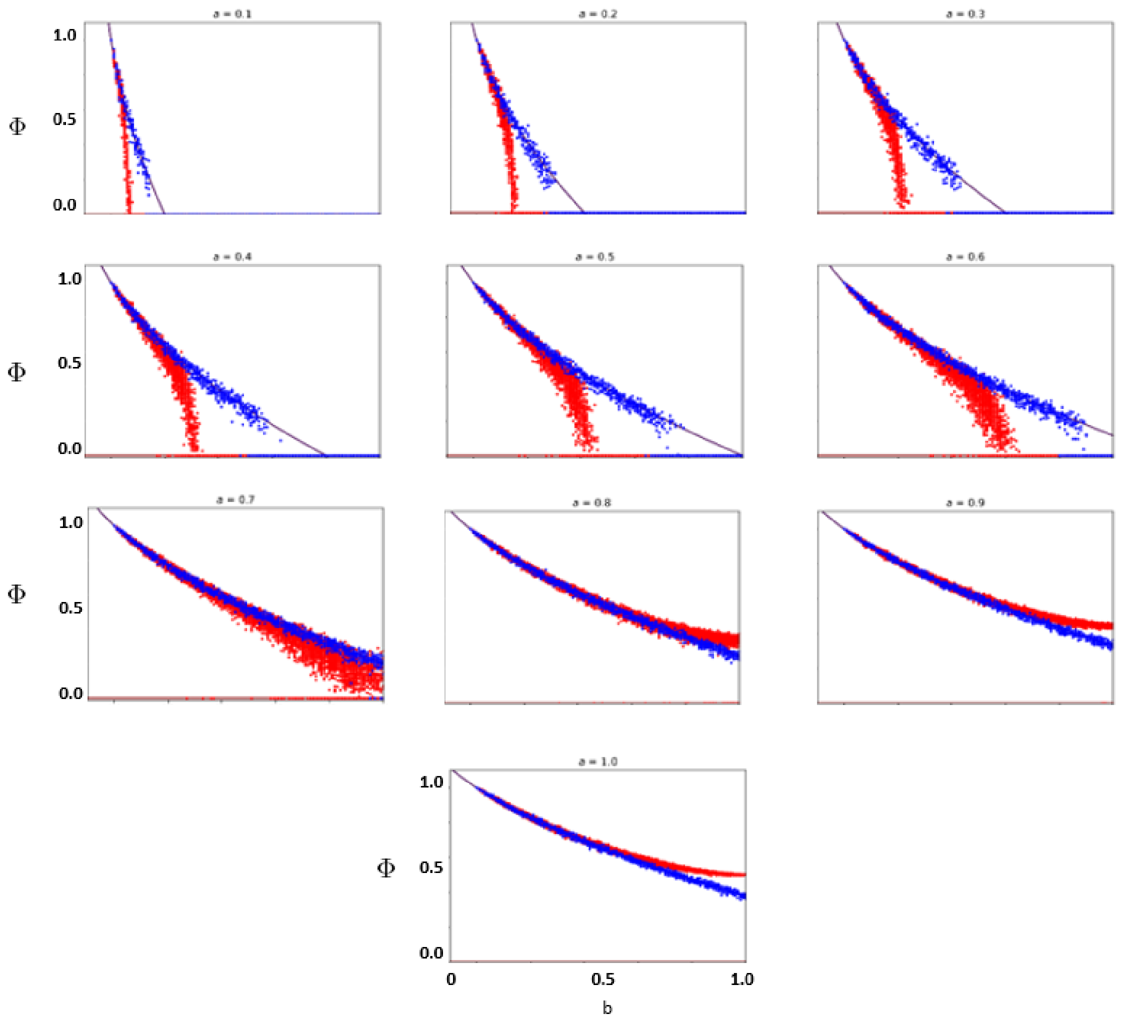


Figure 4.3: Density versus deactivation rates for fixed $a \in [0, 1]$ using varying initial density $\Phi_0 \in [0, 1]$ (in red scattered plot) as compared to the solutions of the mean-field results (black line) and the density with random neighbourhood.

$\sim 1/3$ of the empirical value.

Notice in Figures 4.2 and 4.3, we can see similar patterns of bifurcation in the mean-field curve as observed in the numerical results. Since we have a quadratic and a linear solution in our derivation, it is expected that if we extend the range of densities, there would be three "arms" in our plot that would look like a pitchfork bifurcation.

However, several studies indicate that it is difficult to rely only on the mean field equations in the determination of the critical exponents or phase diagrams with more than one parameter since the multicritical points and critical manifolds can differ from the ones provided by the approximations[18, 23–25].Furthermore, deviation from mean-field indicates some level of ordering in the spatio-temporal pattern resulting from the dynamics of the system. This means that the discrepancy that we observed between the numerical and mean-field results returns a difference in the transition point from the absorbing states to densities greater than zero. Therefore, we cannot fully use the analytic portion of the mean-field for approximating the transition property of our phase diagram.

Chapter 5

Summary and Conclusions

In this paper, we have studied a stochastic modelling approach on the dynamics of the brain neurons by using pCA. We can find similar well-known models such as the one proposed by Benayoun et al. [5] where they treat the neurons as a coupled, continuous-time two-state Markov process wherein each neuron can exist in either active (neuron firing an action potential) or resting state (refractory period or quiescent state).

To apply randomness in a general deterministic CA, we used transition probability a from resting to firing given a neighbourhood with at least one active neuron. Meanwhile, active neurons become inactive with probability b regardless of the state of its neighbourhood. With these set of rules, we are able to generate several analysis on the dynamics of the neurons such as: signal spreading (transmission) for a given timesteps, phase transition from the inactive to active state and the mean-field analysis for the phase diagram.

More precisely, we obtained the density of excited neurons at a particular time point after iterating for $t = 1000$ timesteps. Our findings are consistent with our expectation that the density would decrease as beta approaches one and increase as alpha approaches one. A combination of these two will eventually result in a complicated behavior that may be further examined by checking their individual time evolution graphs.

Because we were interested in determining if there is a transition between two major states of neurons, we attempted to utilize Petersen and Alstrøm's [28] methodologies to get plots that quantitatively depict the crucial sites of transition from inhibitory to excitatory states. We discovered that when b is fixed and a is varied, the critical points for various probability rates also varies, but they eventually approach the maximum value equal to $a_c = 0.71$. This critical point was utilized to get the experimental directed percolation (DP) value, which may belong to the DP universality class in one dimension.

Finally, we applied mean-field analysis to compare the numerical findings to the computational derivations using the proposed set of rules. As expected, the numerical results follow the mean-field approximation's behavior. We can observe pitchfork bifurcation in the simulation and analytical approximation up to a certain range of density values. Our mean-field solutions provide us the three branches of the bifurcation, but physically, the negative solutions are not possible in the real-world simulations.

Large variations between these two are also observed. This discrepancy may be attributed to the spatial correlation detected in the pCA's time evolution. If we are going to look at the critical parameter value and from that obtain the phase transition by obtaining the exponent of the power law, we will get different transition points from the mean-field analysis and numerical simulations because of the spatio-temperoral effects. Thus, we cannot fully use the analytic portion of the mean-field for approximating the transition property. However, this initial approximation can provide us the extent in which this effect should be taken into account.

Because of this, aside from the possible bidirectionality of the neurons, the main limitation of the mean-field approach is its secondary effects on the system. One possible solution for this limitation is the usage of higher order approximation for the mean-field analysis that incorporates the spatial parameter.

Ultimately, this model is used because of its simplicity and efficiency in modelling thousands of neurons as compared to using the networks framework. Similar to the Game of Life approach by Fraile et al. [12], the pCA model can be used to understand the propagation of the signal or activity in the brain dynamics. As such, one possible direction of this research is to look for the first passage time. This means that we can measure the time it takes before it reach a certain threshold of Φ. However, it needs a corresponding empirical data as our input and then use our model as the basis.

Although the rules are an oversimplification of the connection of the neurons in our brain, this model can be used as a base framework for further development of much more advanced and complex CAs and networks.

Appendix A

Appendix

A.1 Equations

Cowan et.al derived the final equation that includes the dynamics of the excitatory and the inhibitory neurons [9] as follows:

$$\begin{aligned}\frac{dP_{(n_E,n_I,t)}}{dt} =&\alpha_E[(n_E+1)P(n_E+1,n_I,t)-n_EP(n_E,n_I,t)]\\&+(N_E-n_E+1)f_E[s_E(n_E-1,n_I)]P(n_E-1,n_I,t)\\&-(N_E-n_E)f_E[s_E(n_E,n_I)]P(n_E,n_I,t)\\&+\alpha_I[(n_I+1)P(n_E,n_I+1,t)-n_IP(n_E,n_I,t)]\\&+(N_I-n_I+1)f_I[s_I(n_E,n_I-1)]P(n_E,n_I-1,t)\\&-(N_I-n_I)f_I[s_I(n_E,n_I)]P(n_E,n_I,t)\end{aligned} \tag{A.1}$$

A.2 PYTHON codes

```
# -*- coding: utf-8 -*-
"""
Created on Thu Nov 11 13:13:12 2021

@author: PC
"""

import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import pickle
import pandas as pd

time = 1000
increments = 0.01
neurons = 100
neuronList = np.arange(0,neurons + 1,1)
timesteps = np.arange(0,time+1,1)

def loadGrid(file:str):
    """
    Loads the grid generated from init.py

    Args
    ----
    file: str
        A text file that contains the saved grid

    Returns
    -------
    length: int
```

```
        Number of length of the grid
    initialGrid: ndarray
        An initial array of the CA (previously generated in init.py)

    """

    initialGrid = np.loadtxt(file).astype(int)
    length = len(initialGrid)

    loadGrid = {"initialGrid": initialGrid, "length": length}

    return loadGrid

def probabilityList(name, start = 0.0, end = 1.01, increments = increments):

    """
    Creates a dictionary of probabilities
    Size: Depending on the increments. For 0.01 -> size = 100

    Args
    ----
    start: float
        starting probability value
    end : float
        ending probability value
    increments: float
        no. of increments from the start of the probability to the end
    name: string
        name of the probability list
    probList: array
        array of the desired probabilities

    Returns
    -------
    probDict: dict
        dictionary of the probability list with their names

    """
    probList = np.arange(start,end,increments)
    probDict = {name: probList, "size": len(probList)}

    return probDict

def checkNeighbor(grid, state):
    """
    Checks the left and right neighbors of the current state

    Args
    ----
    state: int
        notes the index of the current state

    grid: array
        takes the desired CA grid (matrix)

    neurons: int
        gets the number of neurons per row

    neighbors: list
        creates a list of the neighbors of the current state

    Returns
    -------
    sum of the neighbors
    """
    neurons = 101
    neighbors=[grid[state-1], grid[state-(neurons-1)]]
    return sum(neighbors)

def PCA(gridFile,neurons:int, maximumTime = time, gridIndex = 0):

    """
    Size:
    Args
    ----
    maximumTime: int
        Sets the maximum time of the PCA
    gridFile: array
        Text file of grid generated from init.py
    neurons: int
        Number of initial neurons (width of the grid)

    Returns
    -------
    finalCA: ndarray
        Returns an array of the 4D CA

    """


    alphaList = probabilityList("alpha")["alpha"]
    betaList = probabilityList("beta")["beta"]
    betaLen = probabilityList("beta")["size"]
    time = range(int(maximumTime))
    if loadGrid(gridFile)["initialGrid"].shape == (101,101) :
        initialGrid = loadGrid(gridFile)["initialGrid"][gridIndex]
    else:
        initialGrid = loadGrid(gridFile)["initialGrid"]
    final_CA = []
    for b in range(betaLen):
        list_CA = []
        beta = betaList[b]
        for a in alphaList:
            alpha = round(a,2)
            CA = [initialGrid]
            grid = initialGrid
            for t in time:
                Next = grid.copy()

                for g in range(neurons):
                    if grid[g] == 1:
                        temp = np.random.random()
                        if temp<=beta:
                            Next[g] = 0
                    else:
                        if checkNeighbor(grid,g) != 0:
                            temp2 = np.random.random()
                            if temp2<=alpha:
                                Next[g] = 1

                grid = Next.copy()
                CA.append(grid)

            list_CA.append(CA)
        final_CA.append(list_CA)
    return final_CA

def averageValue(array, neurons, timesteps, axis, lastvalues = 20,purpose = None):
    """
    Args
    ----
    array:str or ndarray
        inputs the file name that contains the grid
    neurons:int
        number of initial neurons
    lastvalues:int
        gets the number of last values to be averaged
    timesteps: int
        number of timesteps for every iteration
    axis: int
        is equal to the number of dimensions - 1

    Return
    ------
    xvalarr: ndarray
        array of the x values
    mean : ndarray
        array of the mean for the last n values
    """


    listPCA = None
    if type(array) is str and purpose == None:
        listPCA = np.asarray(PCA(array, neurons = 101))
        mean = np.mean(listPCA,axis = axis,dtype = np.float64)
        xval = np.arange(timesteps + 1)
        xvalarr = np.repeat(xval[np.newaxis, :], probabilityList("alpha")["size"], axis = 0)
        return xvalarr,mean


    elif type(array) is np.ndarray and purpose == None:
        y = array
        lastvals = y[:,:,(timesteps-lastvalues):]
        mean = np.mean(lastvals,axis = axis)
        xval = np.arange(timesteps + 1)
        xvalarr = np.repeat(xval[np.newaxis, :], probabilityList("alpha")["size"], axis = 0)

        return xvalarr, mean

    elif type(array) is np.ndarray and purpose == "timeplot":
        listPCA = array
        mean = np.mean(listPCA,axis = axis,dtype = np.float64)
        xval = np.arange(timesteps + 1)
        xvalarr = np.repeat(xval[np.newaxis, :], 11, axis = 0)
        mean = mean[0::10, 0::10]
        return xvalarr,mean

def timeplot(file, neurons = 101, timesteps = 1000,axis = 3, purpose ="timeplot"):
    """
    Return
    ------
    1-D plot of the array

    """
    #x and y values coming from a PCA file
    if type(file) == str:
        x,y = averageValue(file, neurons = 101, timesteps =1000,axis = 3, purpose = "timeplot")
    else:
        x,y = file[0], file[1]


    for i in range(len(x)):
        plt.figure(figsize=(8, 6), dpi=300)
        plot = plt.plot(x.T,y[i].T, "-", linewidth=1, markersize=4)
        leg = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]
        plt.legend(plot, ["a = " + str(i) for i in leg ], prop={'size': 6}, bbox_to_anchor=(1.12, 1.05), loc='upper right')
        plt.title(r'$b = $' + str(i/10))
        plt.xlabel(r'$t$')
        plt.ylabel(r'$\phi$')
        plt.ylim(ymin = 0, ymax = 1)
def phaseplot(array, cmap, kind,title, lasttime = 0, time = time):
    if kind == "phasespace":
        if lasttime > 0:
            plt.imshow(array[-lasttime:, :], cmap = cmap, origin = "upper", extent = [0,100,time, time-lasttime])
            plt.ylabel("time")
            #plt.yticks(np.arange(lasttime, time + 10, 10))
            plt.title(title)
            plt.figure(figsize = (2.5,15))
            plt.savefig(str(title) + ".png")
        else:
            plt.imshow(array[-lasttime:, :], cmap = cmap, origin = "upper")
            plt.ylabel("time")
            #plt.yticks(np.arange(lasttime, time + 10, 10))
            plt.title(title)
            plt.figure(figsize = (2.5,15))
            plt.savefig(str(title) + ".png")


    else:
        plt.figure()
        plt.imshow(array, cmap = cmap , extent=[0.0,1.0,0.0,1.0], origin = "lower")
        plt.xlabel(r'$a $')
        plt.ylabel(r'$b$')
        plt.title(r'$b $' + ' vs ' + r'$a $' + ' plot')
        plt.colorbar()

def criticalPoint():
    rowindex = []
    for i in range(101):
        rowindex.append(np.amax(np.where(yphase[...,i] >= 0.1)))
    colindex = np.arange(0,101,1, dtype = int)

    indextuple = np.column_stack((rowindex,colindex))
    linephase = np.zeros((101,101))
    linephase[indextuple[:,0], indextuple[:,1]] = 1
    plt.imshow(linephase, cmap = "gray")


    rowindex2 = []
    for i in range(110):
        rowindex2.append(np.amax(np.where(yphase[...,i] >= 0.5)))

    indextuple2 = np.column_stack((rowindex2,colindex))
    linephase[indextuple2[:,0], indextuple2[:,1]] = 2
    plt.imshow(linephase, cmap = "gray")


    rowindex3 = []
    for i in range(110):
        rowindex3.append(np.amin(np.where(yphase[...,i] <= 0.6)))

    indextuple3 = np.column_stack((rowindex3,colindex))
    linephase[indextuple3[:,0], indextuple3[:,1]] = 3
    plt.imshow(linephase, cmap = "gray")


def alphaPhiplot(array, beta, alpha):
    alphaList = probabilityList("alpha")["alpha"]


    if beta is None:
        None
    else:
        if type(beta) is np.ndarray and len(beta) == len(alphaList):
            fig,ax = plt.subplots()
            for i in range(len(beta)):
                ax.plot(alphaList,array[i], "o", linewidth=1, markersize=4, label = r'$\beta$' + ' =' + str(i))
            #leg = ax.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
            plt.ylim(ymin = 0, ymax = 1)
            plt.xlabel(r'$a$')
            plt.ylabel(r'$\phi$')
            plt.figure()

        elif type(beta) is np.ndarray and len(beta) < len(alphaList):
            fig,ax = plt.subplots()
            for i in beta:
                #i = int(i * (len(beta) - 1))
                ax.plot(alphaList,array[i], "o", linewidth=1, markersize=4, label = r'$b$' + ' =' + str(i/(len(alphaList) - 1)))
            leg = ax.legend(bbox_to_anchor=(1.14, 0.80), loc='right', prop=fontP)
            plt.ylim(ymin = 0, ymax = 1)
            plt.xlabel(r'$a$')
            plt.ylabel(r'$\phi$')
            plt.figure()


        else:
            plt.plot(alphaList,array[beta], "o", linewidth=1, markersize=4)
            plt.ylim(ymin = 0, ymax = 1)
            plt.title("Beta = " + str(beta) + " varying alpha")
            plt.xlabel(r'$a$')
            plt.ylabel(r'$\phi$')
            plt.figure()


    if alpha is None:
        None
    else:
        if type(alpha) is np.ndarray:
            fig,ax = plt.subplots()
            for i in alpha:
                ax.plot(alphaList,array.T[i], "o", linewidth=1, markersize=4, label = r'$a$' + ' =' + str(i/100))
            leg = ax.legend(bbox_to_anchor=(1.14, 0.80), loc='right', prop=fontP)
            plt.ylim(ymin = 0, ymax = 1)
            plt.xlabel(r'$b$')
            plt.ylabel(r'$\phi$')
            plt.figure()

        elif type(alpha) is int:
            plt.plot(alphaList,array.T[alpha], "o", linewidth=1, markersize=4)
            plt.ylim(ymin = 0, ymax = 1)
            plt.title(r"$\alpha = $" + str(alpha) + " ,varying beta")
            plt.xlabel(r'$b$')
            plt.ylabel(r'$\phi$')
            plt.figure()
```

Bibliography

[1] Local Bifurcation. In *Ordinary Differential Equations with Applications*, pages 545–602. Springer New York, New York, NY, 2006.

[2] O. Stavskaya L. Mityushin G. Kurdyumov S. Pirogov A. Toom, N. Vasilyev. Discrete local markov systems. In Kryukov R. Dobrushin, V and A. Toom, editors, *Stochastic Cellular Systems: ergodicity, memory, morphogenesis*, pages 1–182. Manchester University Press, Manchester, 1990.

[3] Ashley Allshire. Electrogenic ion pumps; by p. lauger, sinauer assoc.; sunderland, ma; 1991;(distributed by wh freeman & co.) xii+ 313 pages; e34. 95. isbn 0-87893-451-o. *FEBS LETTERS*, 1993.

[4] P Balister, B Bollobás, and R Kozma. Mean field models of probabilistic cellular automata. 2004.

[5] Marc Benayoun, Jack Cowan, Wim van Drongelen, and Edward Wallace. Avalanches in a stochastic model of spiking neurons. *PLoS computational biology*, 6:e1000846, 07 2010.

[6] Santiago Ry Cajal. *Texture of the Nervous System of Man and the Vertebrates*, volume 1. Springer Science & Business Media, 1999.

[7] Pavel Cejnar, Oldřich Vyšata, Jaromír Kukal, Martin Beránek, Martin Vališ, and Aleš Procházka. Simple capacitor-switch model of excitatory and inhibitory neuron with all parts biologically explained allows input fire pattern dependent chaotic oscillations. *Scientific Reports*, 10(1):7353, April 2020.

[8] Hugues Chaté and Paul Manneville. Criticality in cellular automata. *Physica D: Nonlinear Phenomena*, 45(1):122–135, 1990.

[9] Jack Cowan, Jeremy Neuman, and Wim van Drongelen. Wilson-cowan equations for neocortical dynamics. *Journal of mathematical neuroscience*, 6:1, 01 2016.

[10] CThompson02. Membrane Permeability of a Neuron During an Action Potential, May 2018.

[11] Roberto Fernández, Pierre-Yves Louis, and Francesca R. Nardi. Overview: PCA Models and Issues. In Pierre-Yves Louis and Francesca R. Nardi, editors, *Probabilistic Cellular Automata: Theory, Applications and Future Perspectives*, pages 1–30. Springer International Publishing, Cham, 2018.

[12] Alberto Fraile, Emmanouil Panagiotakis, Nicholas Christakis, and Luis Acedo. Cellular automata and artificial brain dynamics. *Mathematical and Computational Applications*, 23(4), 2018.

[13] Eugene M. Izhikevich (auth.) Frank C. Hoppensteadt. *Weakly Connected Neural Networks*. Applied Mathematical Sciences (Switzerland) №126. Springer-Verlag New York, 1 edition, 1997.

[14] Paul A Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[15] Martin Gardner. Mathematical games. *Scientific American*, 223(4):120–123, 1970.

[16] Wulfram Gerstner and Werner M. Kistler. Introduction. In *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, page 1–28. Cambridge University Press, 2002.

[17] Constance Hammond. Chapter 3 - Ionic gradients, membrane potential and ionic currents. In Constance Hammond, editor, *Cellular and Molecular Neurophysiology (Fourth Edition)*, pages 39–54. Academic Press, Boston, fourth edition, 2015.

[18] Haye Hinrichsen. Non-equilibrium critical phenomena and phase transitions into absorbing states. *Advances in Physics*, 49(7):815–958, Nov 2000.

[19] Haye Hinrichsen. Observation of directed percolation—a class of nonequilibrium phase transitions. *Physics*, 2:96, 2009.

[20] George M. Kapalka. Chapter 3 - Pharmacodynamics. In George M. Kapalka, editor, *Nutritional and Herbal Therapies for Children and Adolescents*, Practical Resources for the Mental Health Professional, pages 47–70. Academic Press, San Diego, 2010.

[21] Wolfgang Kinzel. Percolation structures and processes. *Ann. Isr. Phys. Soc*, 5:425, 1983.

[22] Thomas M. Liggett. *Interacting Particle Systems.* Classics in Mathematics. Springer-Verlag, Berlin Heidelberg, 2005.

[23] Joaquin Marro and Ronald Dickman. *Nonequilibrium Phase Transitions in Lattice Models.* Collection Alea-Saclay: Monographs and Texts in Statistical Physics. Cambridge University Press, 1999.

[24] ML Martins, HF Verona de Resende, Constantino Tsallis, and ACN de Magalhes. Evidence for a new phase in the domany-kinzel cellular automaton. *Physical review letters*, 66(15):2045, 1991.

[25] J. Ricardo G. Mendonça. Monte carlo investigation of the critical behavior of stavskaya's probabilistic cellular automaton. *Phys. Rev. E*, 83:012102, Jan 2011.

[26] J. Ricardo G. Mendonça. The inactive–active phase transition in the noisy additive (exclusive-or) probabilistic cellular automaton. *International Journal of Modern Physics C*, 27(02):1650016, 2016.

[27] János Neumann, Arthur W Burks, et al. *Theory of self-reproducing automata*, volume 1102024. University of Illinois press Urbana, 1966.

[28] Niels K. Petersen and Preben Alstrøm. Phase transition in an elementary probabilistic cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 235(3):473–485, 1997.

[29] Jonathan Platkiewicz and Romain Brette. A Threshold Equation for Action Potential Initiation. *PLOS Computational Biology*, 6(7):e1000850, July 2010. Publisher: Public Library of Science.

[30] A. A. Ramahi and R. L. Ruff. Membrane Potential. In Michael J. Aminoff and Robert B. Daroff, editors, *Encyclopedia of the Neurological Sciences (Second Edition)*, pages 1034–1035. Academic Press, Oxford, second edition edition, 2014.

[31] Claudio Rocchini. Scheme of pitchfork bifurcation supercritical, May 2007.

[32] Piotr S\lowiński. Percolation Operators and Related Models. In Pierre-Yves Louis and Francesca R. Nardi, editors, *Probabilistic Cellular Automata: Theory, Applications and Future Perspectives*, pages 197–214. Springer International Publishing, Cham, 2018.

[33] G. Steinmeyer, D. Sutter, L. Gallman, N. Matuschek, and U. Keller. Frontiers in ultrashort pulse: Pushing the limits in linear and non-linear optics. *Science*, 286(5544), 1999.

[34] Steven H Strogatz. Nonlinear dynamics and chaos: with applications to physics. *Biology, Chemistry and Engineering*, page 1, 1994.

[35] Greg Stuart, Nelson Spruston, Bert Sakmann, and Michael Häusser. Action potential initiation and backpropagation in neurons of the mammalian CNS. *Trends in Neurosciences*, 20(3):125–131, 1997.

[36] Larry W Swanson and Mihail Bota. Foundational model of structural connectivity in the nervous system with a schema for wiring diagrams, connectome, and basic plan architecture. *Proceedings of the National Academy of Sciences*, 107(48):20610–20617, 2010.

[37] A.L Toom. Contours, Convex Sets, and Cellular Automata - Course notes from the 23th Colloquium of Brazilian, 2004.

[38] S. Ulam. Random processes and transformations. In *In Sets, Numbers, and Universes*, pages 326–337. MIT Press, 1974.

[39] Updated by tiZom, Converted to SVG by en:User:Diberri, Original by en:User:Chris 73. Illustration of the schematic of an action potential.

[40] Milan Vispoel, Aisling J. Daly, and Jan M. Baetens. Progress, gaps and obstacles in the classification of cellular automata. *Physica D: Nonlinear Phenomena*, page 133074, 2021.

[41] N Wiener and A Rosenbluth. The mathematical formulation of the problem of conduction of impulses in a network of connected excitable elements, specifically in cardiac muscle. *Archivos del Instituto de Cardiologia de Mexico*, 16(3):205—265, July 1946.

[42] Stephen Wolfram. Statistical mechanics of cellular automata. *Rev. Mod. Phys.*, 55:601–644, Jul 1983.

[43] Stephen Wolfram. Computation theory of cellular automata. *Communications in mathematical physics*, 96(1):15–57, 1984.

[44] Stephen Wolfram. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):1–35, 1984.

[45] Stephen Wolfram. *A New Kind of Science.* Wolfram Media, 2002.

[46] Andrew Wuensche. Classifying cellular automata automatically: Finding gliders, filtering, and relating space-time patterns, attractor basins, and the z parameter. *Complexity*, 4(3):47–66, 1999.

[47] Santiago Ramón y Cajal. *Sobre las fibras nerviosas de la capa molecular del cerebelo.* 1888.