SIMULATION AND DESIGN OF BIOLOGICAL AND BIOLOGICALLY-MOTIVATED COMPUTING SYSTEMS

A Dissertation

by

YONG ZHANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Peng Li
Committee Members,	Paul Gratz
	Garng Huang
	Yoonsuck Choe
Head of Department,	Chanan Singh

May 2014

Major Subject: Computer Engineering

ABSTRACT

In life science, there is a great need in understandings of biological systems for therapeutics, synthetic biology, and biomedical applications. However, complex behaviors and dynamics of biological systems are hard to understand and design. In the mean time, the design of traditional computer architectures faces challenges from power consumption, device reliability, and process variations. In recent years, the convergence of computer science, computer engineering and life science has enabled new applications targeting the challenges from both engineering and biological fields. On one hand, computer modeling and simulation provides quantitative analysis and predictions of functions and behaviors of biological systems, and further facilitates the design of synthetic biological systems. On the other hand, bio-inspired devices and systems are designed for real world applications by mimicking biological functions and behaviors. This dissertation develops techniques for modeling and analyzing dynamic behaviors of biologically realistic genetic circuits and brain models and design of brain-inspired computing systems. The stability of genetic memory circuits is studied to understand its functions for its potential applications in synthetic biology. Based on the electrical-equivalent models of biochemical reactions, simulation techniques widely used for electronic systems are applied to provide quantitative analysis capabilities. In particular, system-theoretical techniques are used to study the dynamic behaviors of genetic memory circuits, where the notion of stability boundary is employed to characterize the bistability of such circuits. To facilitate the simulation-based studies of physiological and pathological behaviors in brain disorders, we construct large-scale brain models with detailed cellular mechanisms. By developing dedicated numerical techniques for brain simulation, the

simulation speed is greatly improved such that dynamic simulation of large thalamocortical models with more than one million multi-compartment neurons and hundreds of synapses on commodity computer servers becomes feasible. Simulation of such large model produces biologically meaningful results demonstrating the emergence of sigma and delta waves in the early and deep stages of sleep, and suggesting the underlying cellular mechanisms that may be responsible for generation of absence seizure. Brain-inspired computing paradigms may offer promising solutions to many challenges facing the main stream Von Neumann computer architecture. To this end, we develop a biologically inspired learning system amenable to VLSI implementation. The proposed solution consists of a digitized liquid state machine (LSM) and a spike-based learning rule, providing a fully biologically inspired learning paradigm. The key design parameters of this liquid state machine are optimized to maximize the learning performance while considering hardware implementation cost. When applied to speech recognition of isolated word using TI46 speech corpus, the performance of the proposed LSM rivals several existing state-of-art techniques including the Hidden Markov Model based recognizer Sphinx-4.

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my advisor, Dr. Peng Li, for the continuous guidance of my Ph.D. research, for introducing me to the frontier of interdisciplinary research areas, and for spending a great amount of time and effort on numerous insightful discussions, for patiently correcting my writings, and for providing financial support in the form of graduate research assistantship. This dissertation would not have been possible without these. Therefore, my deepest appreciation goes to Dr. Peng Li for his guidance, support and help throughout these years.

I would like to thank Dr. Garng Huang, Dr. Boyuan Yan, Mingchao Wang, Jingzhen Hu and Haokai Lu for research collaborations. Without these collaborations, many parts of this dissertation would not have been possible.

I would also like to express my gratitude to Dr. Paul Gratz and Dr. Yoonsuck Choe for serving on my Ph.D. committee. I have greatly benefited from their insightful comments and suggestions.

I am also indebted to many friends who helped me during these years of my Ph.D. study. Special thanks go to senior students in Dr. Peng Li's research group. Dr. Wei Dong and Dr. Zhuo Feng helped me to learn the basics of simulation. Dr. Xiaoji Ye helped me to use sequential quadratic programming optimization package. Dr. Guo Yu provided circuits as benchmarks when I was studying circuit simulations. I also benefited from discussions with Dr. Zhiyu Zeng and Dr. Leyi Yin.

Last but not least, I would like to thank my wife Jieyu and my parents for supporting and encouraging me all the time.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

In recent years, the development of traditional computer architectures faces increasing challenges in power consumption, device variability, process variations and so on. In the mean time, the advancement of biotechnology has create an exciting new interdisciplinary applications in therapeutics, biomedicine, biomaterial and bio-inspired computing. Development of these applications requires understanding and design of biological and biologically motivated systems. However, complicated behaviors and dynamics of biological systems are hard to understand and design.

To this end, great research efforts have been devoted to quantitative studies of biological systems and bio-inspired engineering. On one hand, targeting a variety of biomedical applications, behaviors, functions and various design issues of biological systems have been studied quantitatively by mathematical modeling and computer simulations [1][2][3][4][5][6]. On the other hand, bio-inspired engineering systems have emerged in the past. For example, a number of bionic devices and systems have been demonstrated [7][8][9]. There has also been ambitious work to build neuromorphic computers by mimicking functions of the brain [10][11]. In the process of working toward this goal, spiking neural networks have been constructed for various applications [12].

As illustrated in Fig. 1.1, this dissertation builds network models and develops numerical simulation techniques to quantitatively study dynamic behaviors of biological and biologically-motivated systems for a number of applications. Leveraging the capabilities of developed techniques for quantitative analysis, we study the impacts of key parameters on system behaviors to facilitate the design of these systems or their components. This dissertation has three main contributions summarized as

follows.

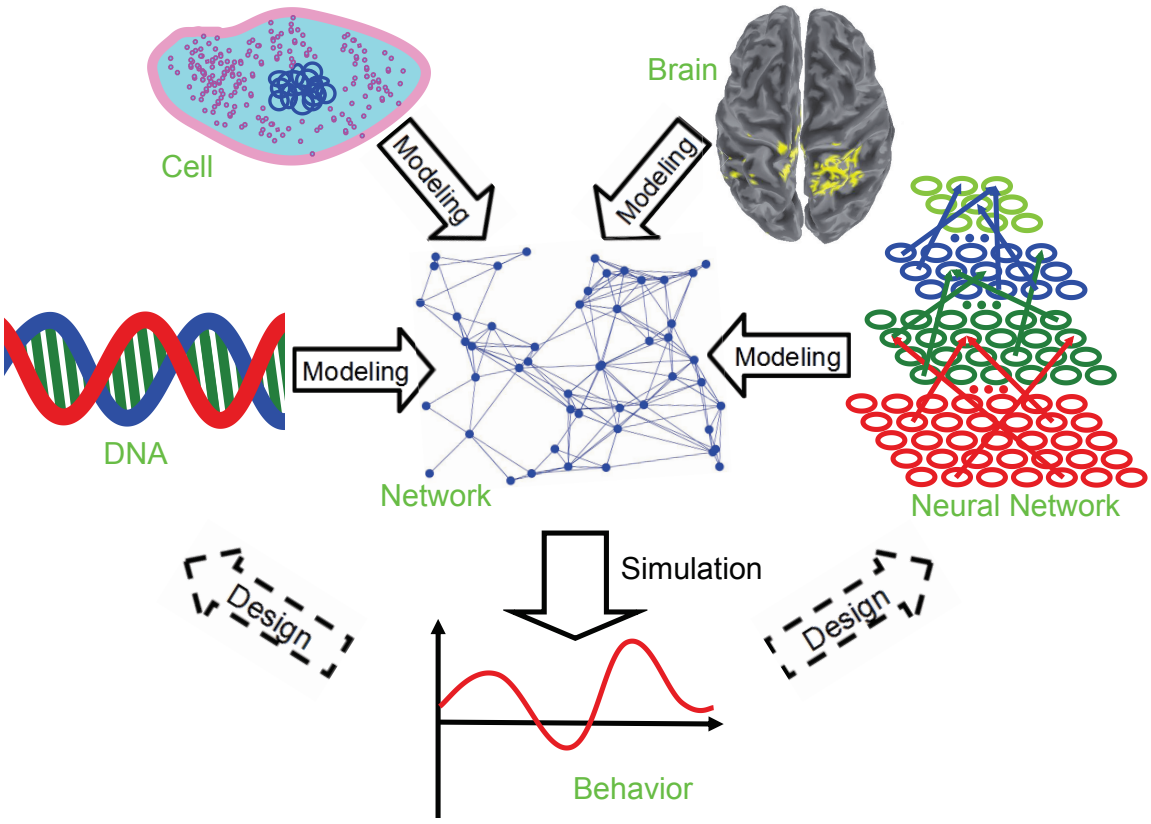

Figure 1.1: Quantitative study of biological and biologically-motivated systems by network modeling and numerical simulations, and its role in the system design.

First, we study dynamic behaviors of genetic memory circuits, which are promising candidate for information storage in synthetic biological systems. As an information storage unit, a genetic memory circuit has two (or more) stable states. Its biological functions critically depend on its dynamics, i.e., the retention of certain states and the transition dynamics between different stable states, corresponding to data retention and writing, respectively. However, the desired stability and dynamics

are subjective to noise, cell-to-cell variation and other design issues. To this end, we quantitatively study the impacts of these factors on behaviors of genetic memory circuits. To facilitate the quantitative study, we build electrical-equivalent models for biochemical reactions involved in the memory circuits and use a network comprised of biochemical reactions to model the the entire circuit. Simulation techniques widely used for electronic systems are applied to provide quantitative analysis capabilities. We further demonstrate an optimization-based parameter identification procedure with the proposed simulation engine. To rigorously and efficiently characterize stability, system-theoretical techniques developed for semiconductor memory cells are extended to study the dynamic behaviors of genetic memory circuits. The notions of stability boundary and dynamic noise margin are employed to characterize the bistability of such circuits.

Second, we develop techniques to model and analyze physiological and pathological behaviors in the brain. Due to the high complexity of neuronal systems, the existing work on brain simulation focuses either on small network models or large networks of simplified neuron models without detailed biophysical mechanisms. To gain better understanding of brain behaviors, we bridge this gap by constructing large-scale brain models with detailed cellular mechanisms and develop dedicated numerical simulation techniques. With greatly improved simulation speed, dynamic simulation of large thalamocortical models with more than one million multi-compartment neurons becomes feasible on commodity computers. Simulation of the adopted models demonstrates the emergence of sigma and delta waves in the early and deep stages of sleep, and suggests the underlying cellular mechanisms that may be responsible for generation of absence seizure.

The third main contribution of this dissertation aims to address the limitations of the mainstream Von Neumann computer architecture in terms of power consumption

and susceptibility to device failure and process variations. To this end, brain-inspired computing paradigms may offer promising solutions. We design a fully biologically-inspired and hardware-friendly learning system, consisting of a digitized liquid state machine and a spike-based learning rule. Taking into account hardware implementation cost, the learning performance of the liquid state machine is maximized by optimizing key design parameters. Tested by speech recognition task using TI46 speech corpus, the liquid state machine produces top-notch performance comparable to existing state-of-art techniques including the Hidden Markov Model based recognizer Sphinx-4 [13].

These three main contributions are discussed with more details below.

1.1 Modeling, Simulation and Dynamic Stability Analysis of Genetic Memory Circuits

Bistability, or more generally, multistability, has been found in many biological networks. In such systems, there exist multiple stable equilibria and biological functions critically depend on retaining certain desired stable equilibria, corresponding to memory, or proper transitions between equilibria. In the field of synthetic biology, for example, there exist research works that attempt to modify biological functions of organisms and construct new biological systems to satisfy human needs [14, 15]. Many basic engineered gene circuits have been designed and implemented to comprise more complex synthetic biological systems. These basic circuits include biological toggle switches, oscillators, and logic gates and so on. Among these basic engineered gene circuits, biological memories such as toggle switch [16, 17] and conditional memory [18] are most well known. As illustrated in Fig. 1.2, these circuits are designed to possess bistability; The gene regulatory network can hold one of its two stable states through a feedback loop even under small perturbations. Under

large external excitations, it is possible to overwrite the stored state and force the system to transit to the other stable equilibrium. These attributes make the system memorize information, and qualify it to serve as a storage device in larger and more complex biological systems. In designing this kind of gene circuits, there are several important issues to be considered, such as dynamics, bistability, cell-to-cell variations and noises.

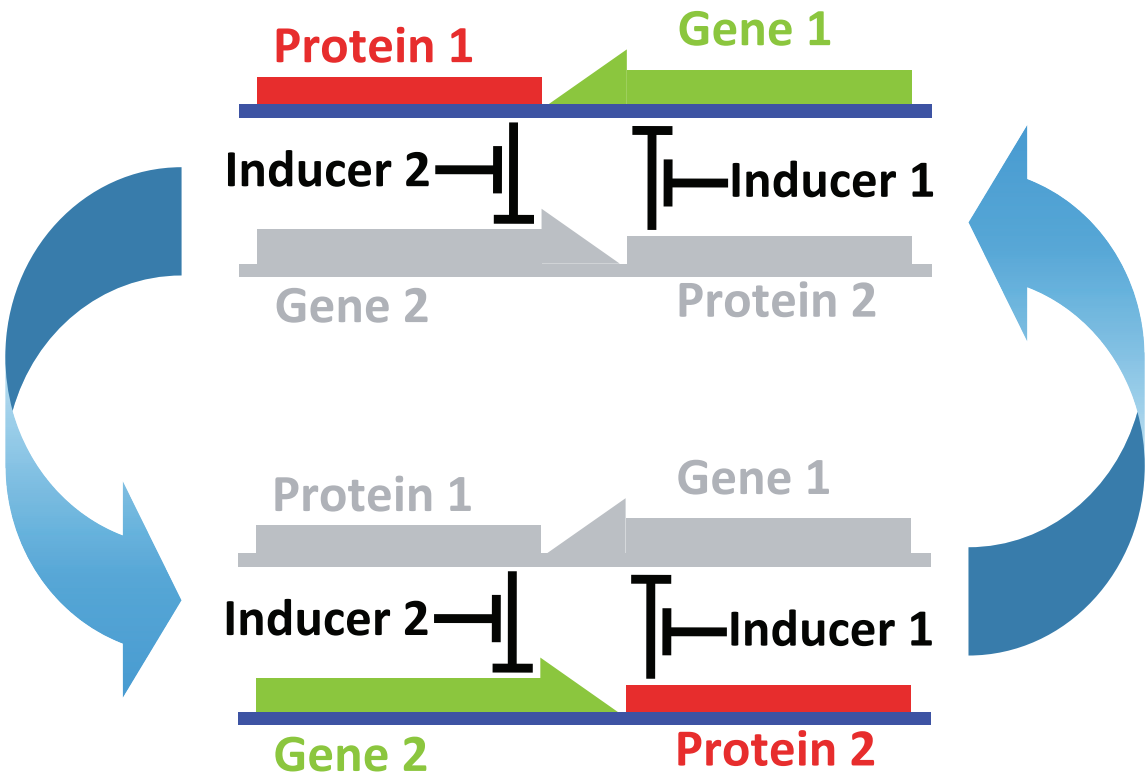

Figure 1.2: Quantitative study of bistability of genetic memory circuits.

Dynamics of biological systems and their building blocks are of great importance to their behavior [19–22], while existing works on toggle switch and conditional memory [16, 18, 23, 24] do not quantitatively analyze bistability from a dynamic point of

view.

Both stochastic models [25] and deterministic models [18] may be used to study behaviors of genetic circuits. The former captures the stochasticity of circuit behaviors caused by low molecular counts, but requires quite a few simulations to reflect the average behavior of each specific circuit. The later reflects the average behavior in the limit of large molecular counts, but does not capture stochastic fluctuations that act as noisy inputs to the circuits. Under this situation, the stability of a network must be examined while considering its susceptibility to noise. In the past decade, research has been geared to understanding cellular noises from different perspectives [25–28]. In view of all these works, it is not surprising to expect that noises may render a memory circuit to lose its stability. Hence, to understand how noise may influence the stability of a multistable system in a dynamic fashion is of great interest.

Individual differences are common in the biological world [29]. Therefore, in genetic regulatory networks, cell-to-cell variation is an important issue and shall be carefully examined. The same shall also be the case when studying dynamic stability.

To address the problems mentioned above, the ability in detailed modeling and simulation of complex genetic networks is not only attractive, but also essential. First-principle based computer models and simulations can provide quantitative analysis and prediction of the behaviors and functions of genetic networks, thereby providing valuable verification and design guidance. If successful, computer simulation or *in silico* analysis may offer accurate predictions that do not involve time consuming or expensive lab (*in vitro* or *in vivo*) experiments. And these simulations may provide deep understanding and insights that are not accessible from in *vivo* or *in vitro* experiments. In addition to design guidance for specific genetic circuit design, calibrated simulation models and analysis can assist the basic understanding

of fundamental biological pathways; they may also enable the understanding of large genetic networks by superimposing verified component models.

We propose to model biological genetic memory circuits using electrical-equivalent modeling and develop biological simulation capability by leveraging the resulting SPICE-like electronic circuit simulation environment. With the developed simulation engine, we propose a Bayesian framework for model parameter identification to avoid the difficulty of directly measuring model parameters. We introduce and apply dynamic stability concepts, namely, dynamic noise margins, developed in [30] to characterize the stability of a conditional memory circuit [18] and its immunity to injected noises. Static noise margins are first presented to capture the effects of the amplitudes of the noise perturbations. Then, this static view of stability is extended to dynamically characterize the memory circuit, leading to dynamic noise margins that further capture nonlinear dynamics of the network.

1.2 Large-Scale Modeling and Simulation of Brain with Detailed Cellular Mechanisms

To understand brain behaviors, it is important to directly associate the network level activities to the underlying biophysical mechanisms. While detailed network models are useful for the studies of physiological and pathological behaviors of the brain, existing computational works towards this direction are typically restricted to small-scale networks due to limitations in computing power. The reason is that, in order to take into consideration complex cellular and molecular mechanisms, the neurons in the network have to be described by biophysically realistic and computationally expensive models such as Hodgkin-Huxley (HH) models[31]. In addition, most simulation works that are based on biophysically realistic models are only limited to local cortical circuitry and do not take into consideration the global structure of the

brain, which includes multiple brain regions, corticocortical connections, and so on. However, those ignored elements can play important roles in generating physiological and pathological behaviors of the brain. For example, to study the initialization, propagation, and termination of seizure, as shown in Fig. 1.3, the topology of the brain network needs to be considered.

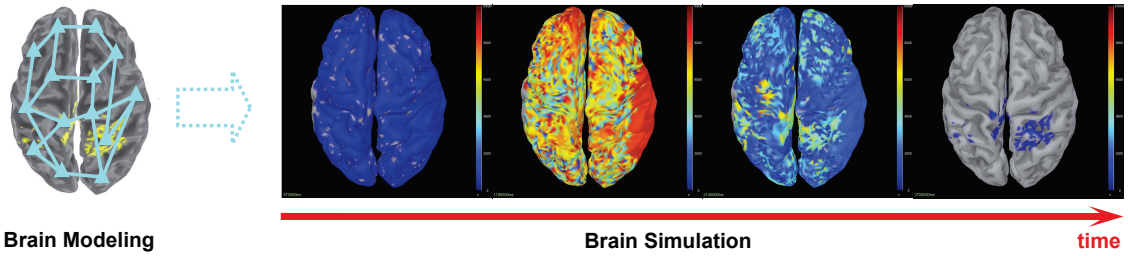

Figure 1.3: Simulation of a brain model with global brain structure and cellular mechanisms demonstrates the emergence and termination of seizure.

On the other hand, with the emergence of more powerful processors and supercomputing platforms, it is not surprising for a lot of recent works to use parallel computing to simulate large scale brain models. For example, in the recent work from IBM DARPA SyNAPSE project[3], a cortical network comprised of more than one billion neurons is simulated on a supercomputer. However, as those works are based on simple phenomenological models[32], they cannot be used to directly associate network level behavior with underlying cellular and molecular mechanisms. As a result, while some interesting brain behaviors can be demonstrated, their applicability is limited. More recently, massively parallel graphic processors or GPUs, have also been employed to simulate brain models [33, 34] to achieve high computational efficiency. However, the models in those works are highly simplified and do not in-

clude sufficient biophysical details to study the biophysical mechanisms underlying the brain function.

While noticeable efforts are ongoing towards modeling the whole brain down to the molecular level, the goal cannot be easily achieved owning to limited computational power and lack of detailed knowledge of neuronal systems. To bridge the gap in the current stage, we take an intermediate step by constructing a large-scale biophysically realistic brain model with sufficient biophysical details. In our brain model, there are a six-layered cerebral cortex with seventy regions and multiple thalamic nuclei. Both local cortical microcircuits and global corticocortical connections are constructed from a careful literature review[35–42]. There are twenty-two types of neurons being modeled, and each type is modeled by a biophysically detailed multicompartment Hodgkin-Huxley model. The dynamic models of different ion channels and synaptic receptors are adopted from various available literatures to best reflect their roles[1, 43–48]. The brain model is scalable to different sizes by changing the number of neurons while maintaining the spatial density. Most experimental results in this dissertation are obtained with a model with one million neurons.

While general purpose simulation tools such as GENESIS and NEURON are available for neural systems [49, 50], to address the associated computational challenges in simulation of networks of such complexity, we develop a customized simulator, in which parallel computing is leveraged and efficient techniques are proposed and adopted from different angles. Our simulator has the capability of simulating brain networks composed of more than one million neurons and hundreds of millions of synapses with biophysically detailed models. As a first step towards using computer simulation to understand brain behavior, we reproduce sigma and delta waves emerging in the early and deep stages of sleep, respectively, by simulating the detailed model of the thalamocortical network. We also demonstrate the role of

$GABA_A$ receptors in the generation of epileptic seizure: the typical spike-and-wave (SW) pattern in absence seizures can be initialized from both sigma and delta waves by globally suppressing the $GABA_A$-mediated inhibition and can be subsequently terminated by recovering the $GABA_A$-mediated inhibition.

Those initial results are interesting because they show the possibility to determine underlying causes of diseases by simulating the biologically realistic brain model. At the same time, as the results are consistent with the basic understanding of seizure generation and existing experimental results[47, 51–53], the effectiveness of the modeling and simulation work has been initially validated. To the best of our knowledge, little work has been done to link the transition between physiological and pathological states of the brain to underlying cellular mechanisms with a biologically realistic brain model of such complexity. With further development, the work is geared to assisting the clinicians in determining underlying causes of brain disorders and selecting the optimal treatment on an individual basis in the future.

1.3 A Hardware-Oriented Liquid State Machine with Biologically Inspired Learning

Spiking neural networks are proven to be computationally more powerful than previous generations of neural networks based on McCulloch-Pitts neurons and threshold gates [12]. Therefore, many recent research works on neural networks are geared to more biologically-inspired learning algorithms, network structures and applications of spiking neural networks [54][55][56][57][58][59]. On the other hand, as spiking neurons more closely resemble the behavior of neurons in nervous systems, they may also consume lower power than previous generations of neural networks when implemented in hardware [60][11][10]. In addition, it has been shown that spiking neural networks are error resilient [61], a very appealing property for implementa-

tion in modern VLSI techniques in which device reliability and process variation are growing challenges. With these inherent advantages, spiking neural networks become promising for designing new hardware architectures. In recent years, various neuromorphic chips have been designed or fabricated to demonstrate their computational capability and low power consumptions for certain applications [11][10][61].

Inspired by the fact that neocortex processes wide spectrum of information by stereotypical neural microcircuitry, the network model of liquid state machine (LSM) is proposed and subsequently proven to be efficient for various tasks [55][62][63][64]. Structurally, the LSM consists of a reservoir receiving input spike trains and a group of readout neurons receiving signals from the reservoir. With a group of neurons randomly connected by fixed synapses, the reservoir has a recurrent structure. This leads to decaying transient memories represented by the dynamic response of the reservoir to input spike trains. For this reason, the LSM is specially competent for processing temporal patterns such as speech signal [65][66]. For readout neurons, [65] uses ridge regression to calculate synaptic weights between the reservoir and readout neurons for classification tasks. [66] trains output neurons using backpropagation-based multi-layer perceptrons.

Comparing with other standard methods of recognizing isolated utterance such as HMM based approach[13], template based approach [67] and acoustic phonetic approach [68], the liquid state machine is more biologically plausible and is a more general model for speech recognition. The internal model parameters are learned by extracting statistical information from the data without using acoustic models and language models that are specific to languages and data sets. And LSM is computationally powerful such that its performance on isolated word recognition can be comparable to state-of-art signal processing methods. Comparing with other neural network based methods such as LSTM nets[69] and multi-layer perceptron based

classifiers[66], LSM is either superior in performance or more hardware-friendly and biologically plausible, as shown in Fig. 1.4. However, the off-line learning in [65] requires a large amount of storage for intermediate results during training. And the learning rule used in backpropagation-based multi-layer perceptrons [66] is arithmetically complicated and not local. Both of these increase the complexity of the network model. These drawbacks limit the potential application of the network model to real-time low-power hardware implementation.

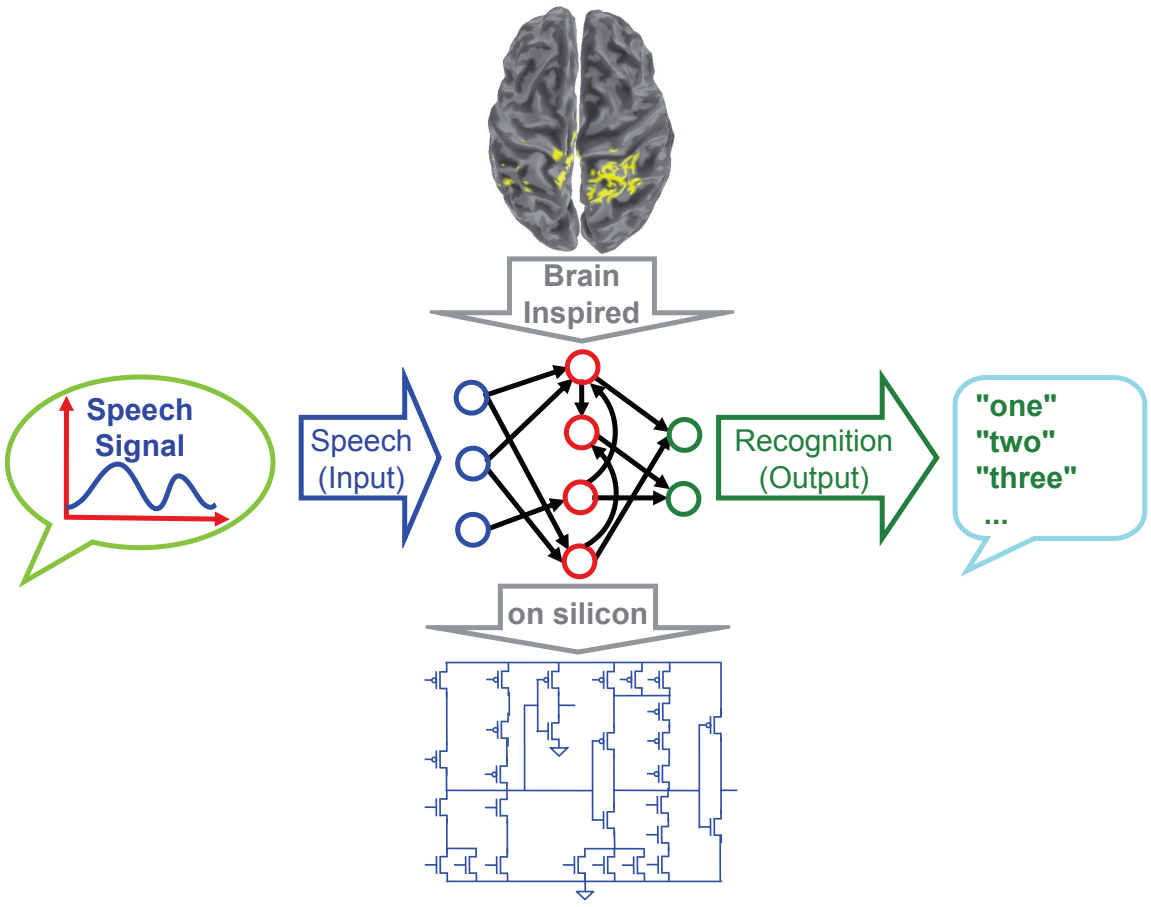

Figure 1.4: Hardware friendly biologically inspired spiking neural network for speech recognition.

In this dissertation, we apply a bio-inspired online learning rule to liquid state machine for solving the problems mentioned above. All model parameters are digitized for VLSI implementation. With the purpose of hardware-friendly network design while keeping the recognition performance, we study and simplify synaptic dynamics and properly choose key parameter values for arithmetic customization. Our simulation results show that in terms of isolated word recognition evaluated using the TI46 speech corpus, the performance of the proposed digital liquid state machine rivals the state-of-the-art Hidden Markov Model (HMM) based recognizer Sphinx-4 [13].

1.4 Organization of the Dissertation

Focusing on the three main contributions introduced above, the remaining part of this dissertation is organized as follows, as illustrated in Fig. 1.5. The modeling, simulation and dynamic stability analysis of genetic memory circuits are elaborated in Section 2. The large-scale modeling and parallel simulation of brain models for the study of epilepsy are in Section 3. Section 4 discusses the design of the hardware friendly liquid state machine for speech recognition. And we draw conclusions in Section 5.

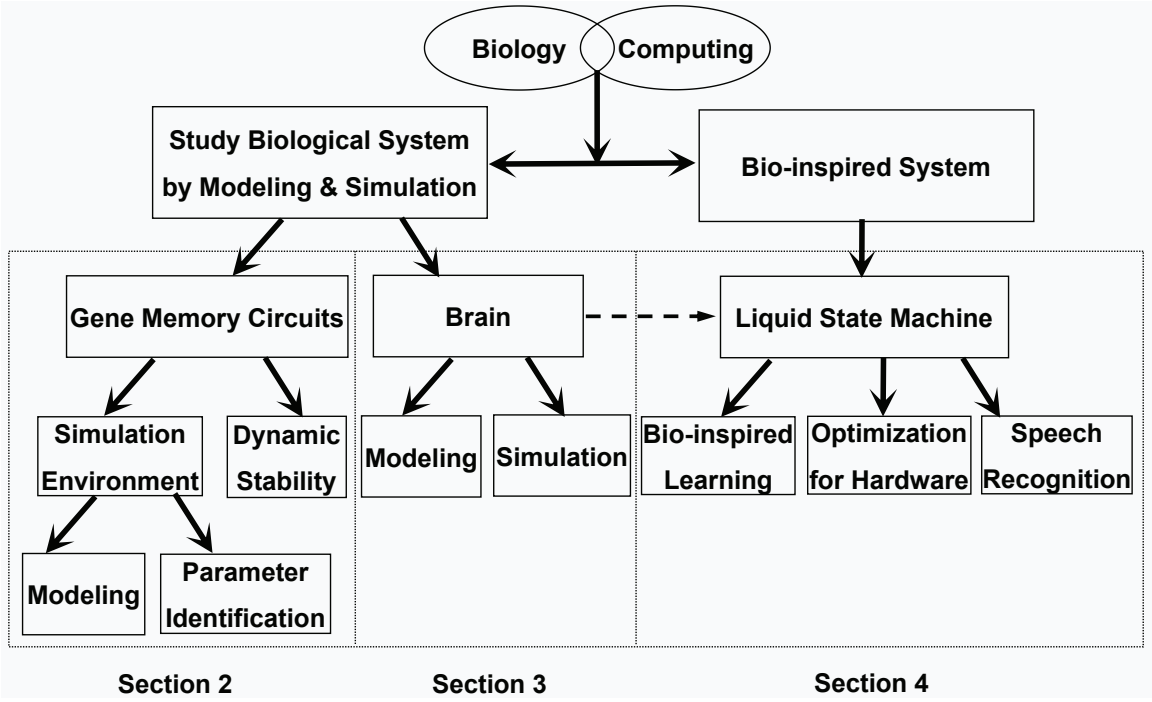

Figure 1.5: Organization of the dissertation.

2. MODELING, SIMULATION, AND DYNAMIC STABILITY ANALYSIS OF GENETIC MEMORY CIRCUITS [1]

In this section, we target dynamic stability analysis of genetic memory circuits. As a prerequisite, quantitative analysis capability is developed leveraging mathematical modeling and simulation techniques.

To begin with, we propose to model biological genetic memory circuits using electrical-equivalent modeling and develop biological simulation capability leveraging a SPICE-like electronic circuit simulation environment. A similar effort has been recently pursued at the Sandia National Lab [2]. In our approach, we specifically target gene-regulatory memory networks and model the nonlinear dynamics associated with transcription, translation, dimerization, binding and degradation of various species using chemical reaction equations, which is appropriate when the system is at the thermodynamical limit [70]. We map the involved nonlinear kinetics into equivalent circuit models and employ a SPICE-engine to perform time-domain dynamic simulation. The composability of these equivalent circuit models allows flexible construction of large gene networks via straightforward model stitching. As such, under our modeling and simulation environment, a gene network is described by an input netlist and the dynamical behavior of the network is simulated using an established nonlinear ODE solver developed for integrated circuit applications.

While the aforementioned simulation engine provides appealing quantitative un-

[1] Section 2.1, 2.2, 2.3, 2.7.1, and 2.7.2 are reprinted with permission from Y. Zhang and P. Li. Gene-regulatory memories: Electrical-equivalent modeling, simulation and parameter identification. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, pages 491-496, ©2009 ACM, Inc. http://doi.acm.org/10.1145/1687399.1687492. Section 2.4, 2.5, 2.6, 2.7.3, 2.7.4, 2.7.5, and 2.8 are reprinted with permission from Y. Zhang, P. Li, and G. M. Huang. Quantifying dynamic stability of genetic memory circuits. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 9(3):871-884, 2012. ©2012 IEEE.

derstanding of the behavior of gene networks, the fidelity of such analysis is contingent upon the availability of detailed network model parameters such as various chemical reaction rates. In practice, it is difficult to directly measure model parameters. To address this difficulty, we propose a Bayesian framework for model parameter identification. By properly formulating likelihood functions, constrained nonlinear optimizations are solved to find the optimal posterior parameter estimates for given measured noisy gene circuit outputs. Our simulation engine is employed at the inner loop of the iterative procedure to search for the best parameter estimate. Since the parameter identification is a numerical procedure, it is critical to ensure that key network properties shall be inferred correctly. For memory circuits, one of such key network properties is bistability, in the sense that only a bistable (or more precisely, multi-stable) network can behave as a memory device. To achieve the realizable identification of bistability vs. monostability, we propose a two-step structure-preserving parameter identification approach. In the first step, measurements that are strongly correlated to bistability are employed to reliably determine bistability. If succeeded, in the subsequent step, model parameters are further optimized to match the measured dynamical characteristics while preserving the identified bistability. This is achieved through a constrained nonlinear optimization formulation.

We introduce and apply dynamic stability concepts, namely, dynamic noise margins, developed in [30] to characterize the stability of a conditional memory circuit [18] and its immunity to injected noises. Static noise margins are first presented to capture the effects of the amplitudes of the noise perturbations. Then, this static view of stability is extended to dynamically characterize the memory circuit, leading to dynamic noise margins that further capture nonlinear dynamics of the network.

Using rigorous system theory, we develop computationally efficient algorithms to compute the defined dynamic noise margins of the conditional memory circuit.

These algorithms operate on the ODE based kinetic models and have their roots in the notion of stability boundaries, or separatrices, which provides a systemic view of system dynamic stability properties [30]. We discuss the issues involved in computing separatrices in low and high dimensional state spaces of the entire genetic memory circuit, and present approaches that are more amenable to high-dimensional models. The tangent approximation of separatrices is also developed as a more efficient way of characterizing dynamic noise margins.

Using the developed computational techniques, dynamic noise margins of the conditional memory circuit have been analyzed for both hold and write operations under constant parameters which are not time varying. Cell-to-cell differences are modeled as parametric variations of the memory circuit model and their impacts on dynamic stability are analyzed. It is shown that the sensitivities of dynamic noise margins to different parametric variations vary significantly, revealing important differences in structure and time scales, and reflecting nonlinear interactions between various biochemical reactions. Such knowledge is expected to be able to facilitate robust memory circuit design.

While focusing mostly on bistable memory circuits, the presented dynamic stability and computational algorithms are quite general in nature. With proper extensions, they may be employed to study dynamic stability of a broad range of multistable biological networks.

2.1 Genetic Memories

2.1.1 Genetic Toggle Switch

The genetic toggle switch design and its functional circuit equivalent, an RS latch of [16], are shown in Fig. 2.1. The toggle switch is implemented in Escherichia coli plasmids and consists of two repressors and two constitutive promoters. The

structure is designed to possess bistability. Without any inducer, there exist two stable states in this mutually inhibitory genetic network. In the first one (ON-state), gene A maximally transcribes and represses promoter B (hence the transcription of gene B). In the second stable state (OFF-state), gene B maximally transcribes and represses Gene A. The promoters and repressors can be Lac repressor (lacI) with Ptr-2 promoter and PLs1con promoter with a temperature-sensitive repressor (cIts). The state can be flipped by introducing a pulse of one of the two inducers: isopropyl-β-D-thiogalactopyranoside (IPTG) or a thermal pulse. With the presence of an inducer, the inactive repressor is allowed to be maximally transcribed until it stably represses the original active promoter. The two inducers are analogous to the R and S inputs to an RS-latch, or the write-one and write-zero events to an SRAM cell.

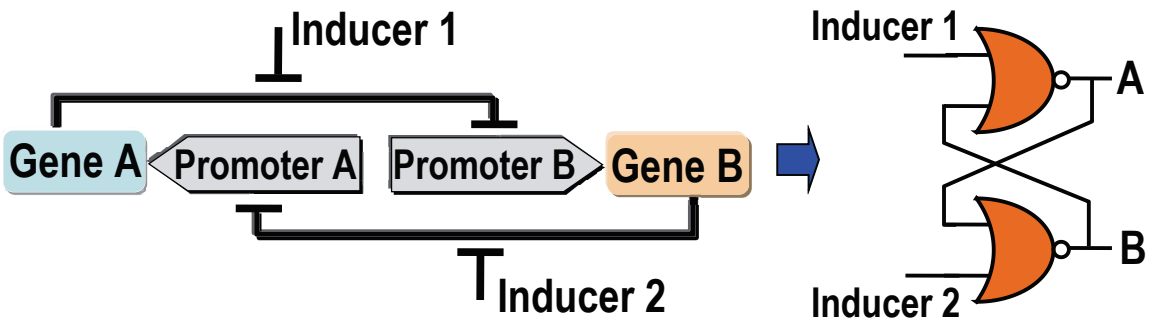

Figure 2.1: Genetic toggle switch and its functional model.

2.1.2 Genetic Conditional Memory

In [18], a regulatory front end is added through protein heterodimerization to realize a conditional memory, as shown in Fig. 2.2. A more detailed model encom-

passing the dynamics involved in all steps of transcription, translation, dimerization, binding and degradation in Fig. 2.3.

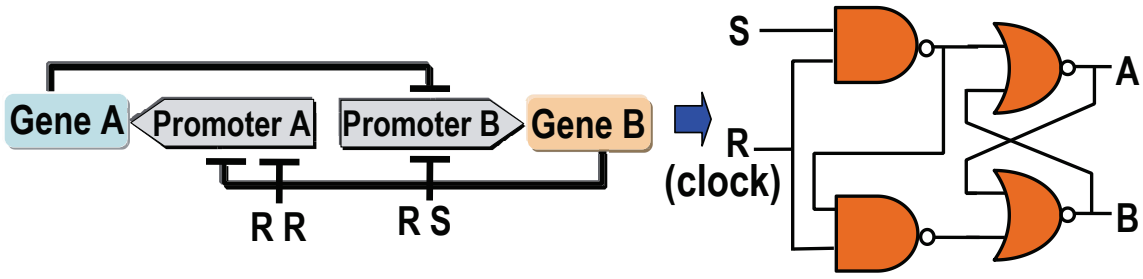

Figure 2.2: Genetic conditional memory and its functional model.

Two combinational control genes R and S are the inputs. This conditional memory behaves like an electrical D-latch. In the regulatory front end, the transcription of gene A can be repressed if protein R is present and forms homodimers (R2). With the presence of protein R, if S is highly expressed, mostly heterodimers (RS) will form (instead of R2) and gene B will be repressed. As a result, when R is absent, neither R2 nor RS forms and the latch maintains its current state regardless of the level of S. Otherwise, the state will reflect S. R and S are analogous to the clock and input signal to an electrical D-latch. Detailed chemical reaction models can be established to quantitatively describe the processes of transcription (DNA $\rightarrow$ mRNA), mRNA degradation, translation (mRNA $\rightarrow$ protein), protein degradation, and protein dimerization for both species R and S. Quantitative models also exist to describe the binding and unbinding of homodimers (R2) and heterodimers (RS) to the corresponding binding sites of Genes A and B, hence characterizing gene regulation.

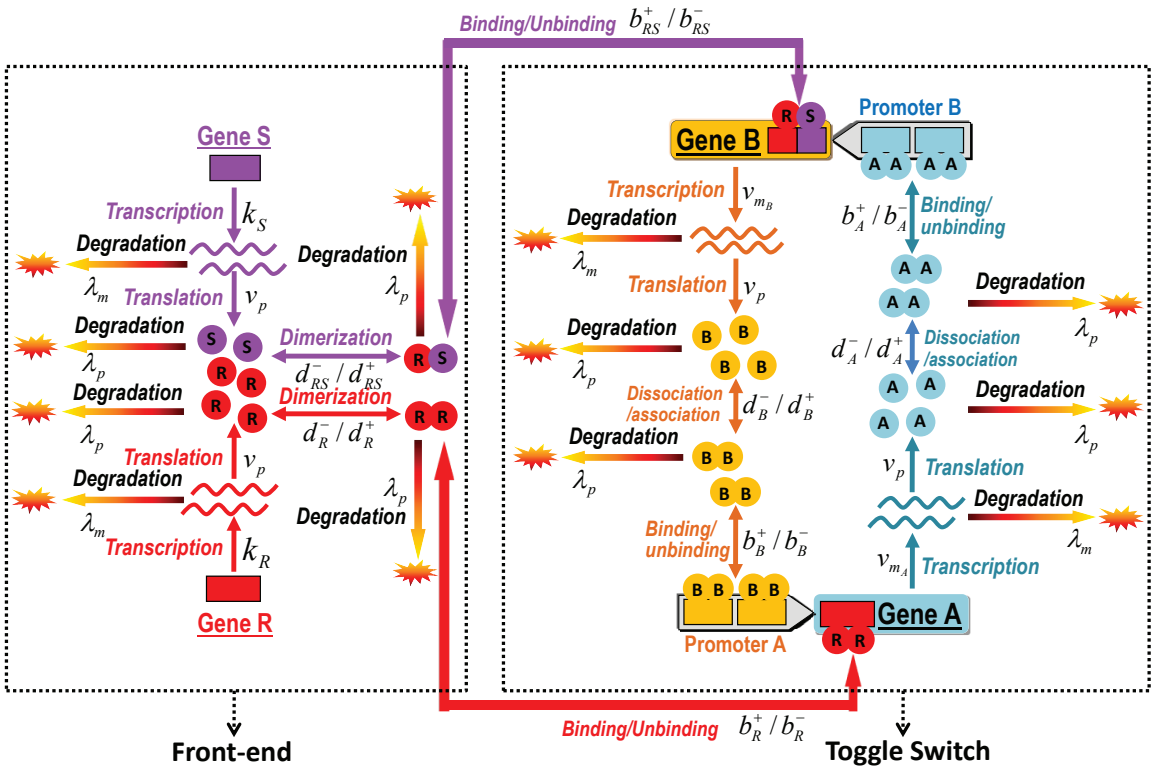

Figure 2.3: The complete biochemical model of the conditional memory circuit.

The back end of the network is similar to the toggle switch design [16]. Depending on the presence of input signals R and S, one of genes A and B will maximally transcribe. Detailed chemical reaction equations can be used to describe the production of the corresponding mRNA (DNA $\rightarrow$ mRNA), protein (mRNA $\rightarrow$ protein), protein dimerization, mRNA and protein degradations, and the binding of the protein on the opposite genes promoter site.

2.2 Electrical-Equivalent Modeling and Simulation

In this work, we have developed an electrical-like simulation environment for analyzing biological genetic networks. We specially target genetic memories such as the ones discussed in the previous section. In this section, we first present the basic

biochemical models for various biological processes involved in a genetic memory. We also present our proposed electrical-equivalent modeling, and discuss specific network modeling and simulation issues.

In the biochemical domain, there exist a lot of correlated chemical reactions; the rate of each reaction is fully determined by the concentration of its reactants and resultants if the reaction is invertible [18]. In the electrical domain, circuit behaviors follow the law of conservation of charge, or Kirchhoffs Current Law (KCL). When applied to each circuit node, KCL states that the net sum of currents flowing into the node is zero. A similar balance exists for each chemical substance in chemical reactions, which often take place in a container, in our case, within the cell volume. For a specific substance, it can react as a reactant, be produced as a resultant, and stored in the container. So the balance is among the substances consumed, produced and the incremental mass stored in the container. In an electrical circuit, the charge stored in a linear capacitor is proportional to the capacitor voltage. For chemical substances, the mass stored in the container is proportional to its concentration, which will regulate the rate of reactions that are related to this substance. As shown in Fig. 2.4, conceptually, each chemical substance is modeled as an equivalent circuit node with a grounded a capacitor. The capacitance is determined by the volume of the container.

2.2.1 Degradation

mRNAs and proteins have finite life time. In chemistry, degradation of these species is characterized by half-life time $T_{half-life}$. The degradation rate is proportional to its mass. As a result, the equivalent RC circuit in Fig. 2.4 is used. Since

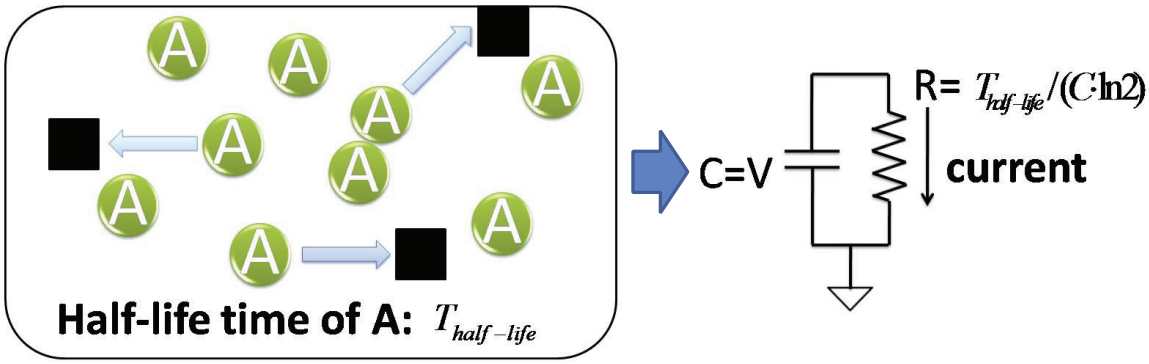

Figure 2.4: Modeling degradation.

the capacitance is determined by the cell volume, the resistance is given by

$$R = \frac{T_{half-life}}{C \cdot ln2}. \tag{2.1}$$

2.2.2 Chemical Reactions

Mass conservation takes place in each reaction. This is guaranteed by stoichiometric balance on chemical reactions, from which we construct the circuit-like model. For example, the heterodimer RS in the genetic conditional memory [18] is produced by the following invertible reaction

$$R + S \underset{\Longrightarrow}{\overset{k^-_{RS}}{\Longleftarrow}} \overset{k^+_{RS}}{} RS, \tag{2.2}$$

where k^+_{RS} and $k^-_{RS}s$ are reaction rate constants for combination and decomposition reactions, respectively. This reaction can be viewed as two separate ones:

$$R + S \overset{k^+_{RS}}{\Longrightarrow} RS, \tag{2.3}$$

$$RS \overset{k^-_{RS}}{\Longleftarrow} R + S, \tag{2.4}$$

These two reactions can be described with more details as

$$\frac{d[RS]}{dt} = -\frac{dR}{dt} = -\frac{dS}{dt} = k^+_{RS} \cdot [R] \cdot [S] \tag{2.5}$$

$$-\frac{d[RS]}{dt} = \frac{dR}{dt} = \frac{dS}{dt} = k^-_{RS} \cdot [RS]. \tag{2.6}$$

The overall effects are

$$\frac{d[RS]}{dt} = -\frac{dR}{dt} = -\frac{dS}{dt} = k^+_{RS} \cdot [R] \cdot [S] - k^-_{RS} \cdot [RS]. \tag{2.7}$$

The above equation can be modeled as a three-terminal device, shown in Fig. 2.5. And the mass flux through each terminal is a function of the concentrations of reactants and resultants. The model can also be viewed as three concentration-controlled flux sources. When stitching them together in a circuit, we treat the device as a black box.

2.2.3 Transcriptions and Translations

In transcriptions and translations, RNAs and proteins are produced from ribonucleotides and amino acids, respectively. The information carried by DNA and RNA is essential in these chemical processes. Take translation as an example. A protein molecule is assembled from amino acid according to the information contained in the corresponding mRNAs. The process can be described as

$$\text{amino acid} \overset{RNA}{\Longrightarrow} \text{protein}. \tag{2.8}$$

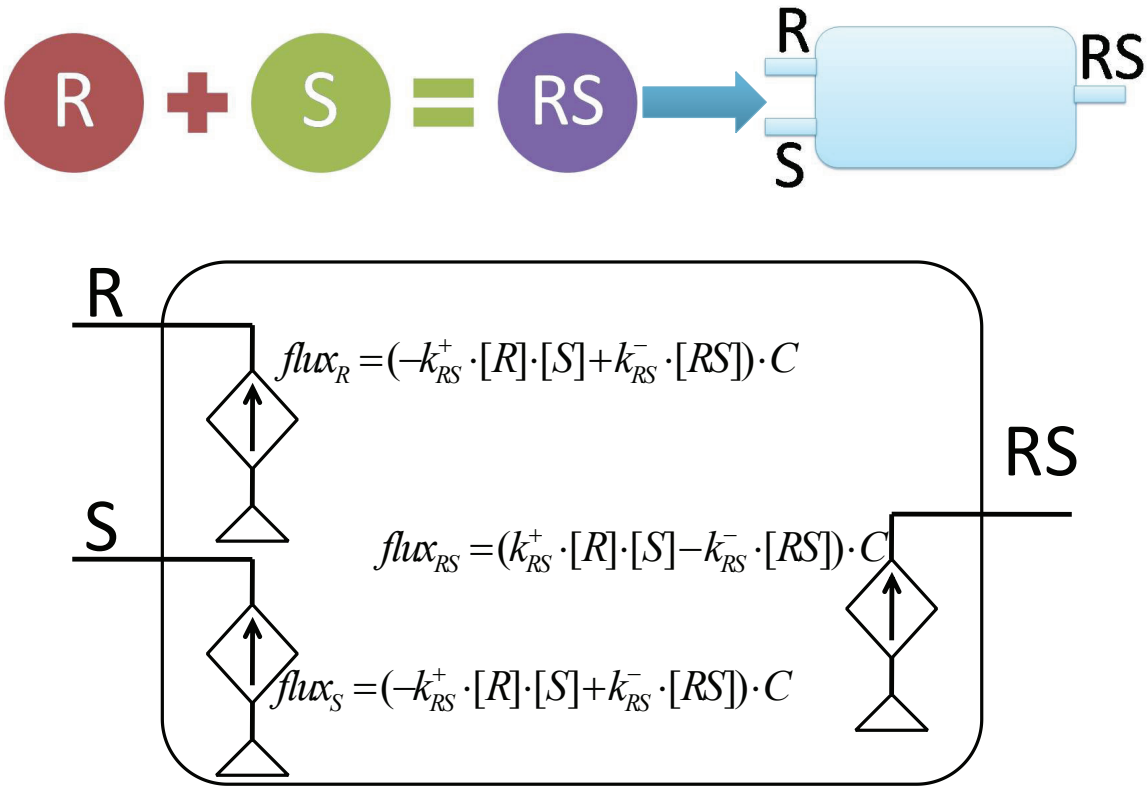

Figure 2.5: Modeling chemical reactions.

One difference between this process and the chemical reactions we have talked about is that this translation process is unidirectional. Another difference is that the reaction rate is influenced by RNAs, which is neither a reactant nor resultant. In this work, we assume that there is sufficient amount of amino acid so that the influence of its concentration on the reaction rate is insignificant. The dominant factor for the reaction rate is the concentration of the RNA. The rate at which proteins are produced is proportional to the mass of the RNA. So if protein A is produced by the corresponding RNA m_A, and the translation rate is , this process is described as

$$m_A \xrightarrow{\nu} A. \tag{2.9}$$

We model a translation as a concentration-controlled flux source, whose electrical equivalent is shown in Fig. 2.6. The same approach is applied to transcriptions.

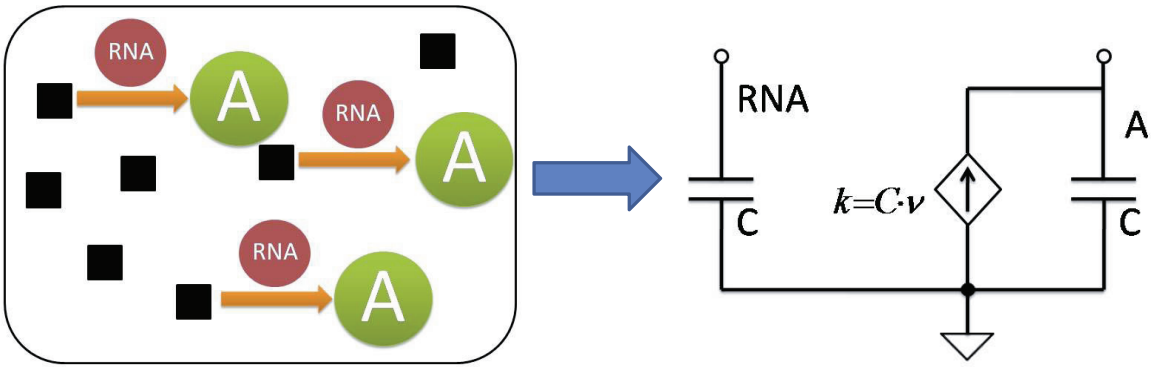

Figure 2.6: Modeling translation.

2.2.4 Gene Regulations

The transcription of a gene can be regulated. There exist multiple promoter sites on a gene. If a promoter site is bounded, or blocked, by a small molecule, the process of transcription is stopped. As an example, in the genetic memory circuit, gene A has three promoter sites, of which one is for protein dimer and two are for dimer . Therefore, gene A can be in one of the following six states:$geneA$, $geneA \cdot B_2$, $geneA \cdot B_2 \cdot B_2$, $geneA \cdot R_2$, $geneA \cdot R_2 \cdot B_2$ and $geneA \cdot R_2 \cdot B_2 \cdot B_2$, which involve seven reactions:

$$geneA + B_2 \underset{}{\overset{k_{gB}^{-}}{\underset{}{\rightleftharpoons}}} \overset{2 \cdot k_{gB}^{+}}{} geneA \cdot B_2, \tag{2.10}$$

$$geneA \cdot B_2 + B_2 \underset{}{\overset{k_{gB}^{-}}{\underset{}{\rightleftharpoons}}} \overset{2 \cdot k_{gB}^{+}}{} geneA \cdot B_2 \cdot B_2, \tag{2.11}$$

$$geneA \cdot R_2 + B_2 \underset{}{\overset{k_{gB}^{-}}{\underset{}{\rightleftharpoons}}} \overset{2 \cdot k_{gB}^{+}}{} geneA \cdot R_2 \cdot B_2, \tag{2.12}$$

$$geneA \cdot R_2 \cdot B_2 + B_2 \overset{k_{gB}^{-}}{\Longleftarrow} \overset{2 \cdot k_{gB}^{+}}{\Longrightarrow} geneA \cdot R_2 \cdot B_2 \cdot B_2, \tag{2.13}$$

$$geneA + R_2 \overset{k_{gR}^{-}}{\Longleftarrow} \overset{k_{gR}^{+}}{\Longrightarrow} geneA \cdot R_2, \tag{2.14}$$

$$geneA \cdot B_2 + R_2 \overset{k_{gR}^{-}}{\Longleftarrow} \overset{k_{gR}^{+}}{\Longrightarrow} geneA \cdot R_2 \cdot B_2, \tag{2.15}$$

$$geneA \cdot B_2 \cdot B_2 + R_2 \overset{k_{gR}^{-}}{\Longleftarrow} \overset{k_{gR}^{+}}{\Longrightarrow} geneA \cdot R_2 \cdot B_2 \cdot B_2, \tag{2.16}$$

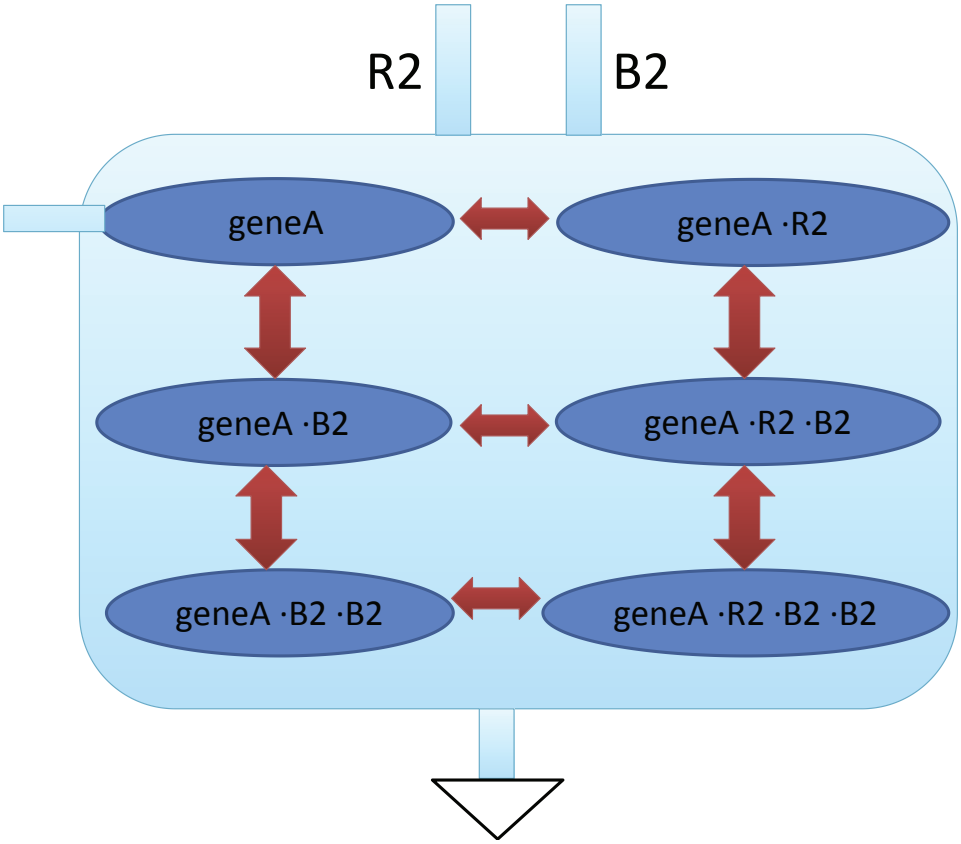

Figure 2.7: Modeling gene regulation.

We use the macromodel in Fig. 2.7 to capture gene regulation. Because genes do not degrade, the summation of the above six gene concentrations is constant. So

the number of independent variables in simulation is five. Since transcription only works for unblocked genes, we treat the variables corresponding to blocked genes as internal nodes in the model, as shown in Fig. 2.7.

Biological system may be affected by some external interference, which serves as inputs. For example, in the gene memory circuit considered in this section, the transcription rates of geneR and geneS are controlled by external signals [18]. We model them as two time-varying current sources connected to the corresponding mRNAs.

2.3 Simulation-Driven Bayesian Parameter Identification

While the modeling and simulation framework presented in the previous section provides powerful analysis capabilities for quantitative prediction of biological network behaviors, practical difficulties arise when coming to determine the detailed model parameter values such as chemical reaction rates. These parameters are not easily directly exposed in lab experiments. To address this difficulty, we propose a Bayesian based framework to identify model parameters based upon measured genetic circuit outputs. Our approach takes into consideration unavoidable lab measurement noises and prior statistical distributions of model parameters while optimizing certain likelihood functions. The optimizations are driven by the developed simulation infrastructure.

Unlike existing Bayesian approaches [71][72], we propose a two-step structure-preserving parameter identification approach in a way relevant to genetic memories. Our specific emphasis is to reliably identify one of the most important properties of a memory device, bistability, and maintain such property if identified in the presence of measurement and numerical noises. As shown in Fig. 2.8, the first identification step utilizes bistability specific output measurements to drive the initial model parameter

identification. If a memory design is identified to be bistable, in the following step, a constrained parameter identification process is performed that furthers matches the measured dynamic behaviors while maintaining bistability.

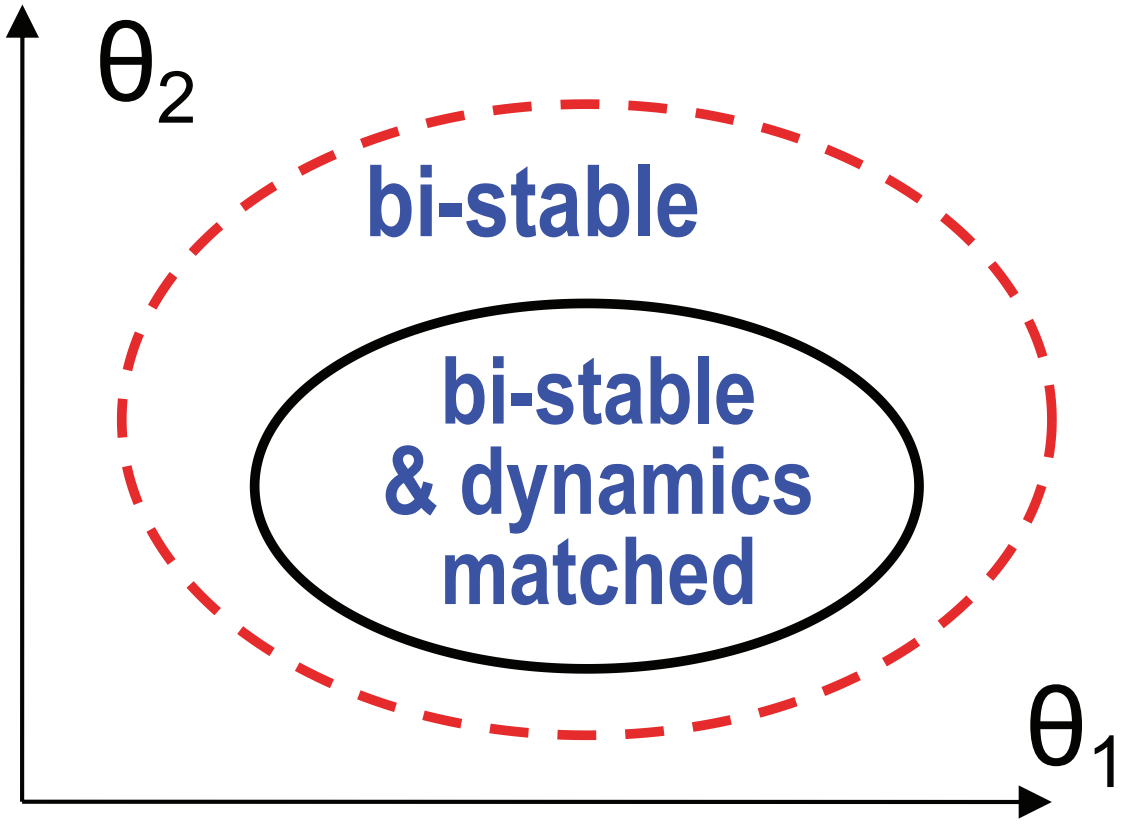

Figure 2.8: Two-step bistability preserving parameter identification.

2.3.1 Bayesian Model Parameter Identification

Let us assume that under a given input $u(t)$, the outputs of a genetic network, say, the transcription levels measured by using green fluorescent protein (GFP)mut3 as a reporter [16], are sampled at N time points: $t_1, t_2, \cdots, t_N$. Denote the corresponding

sampled outputs as $y_m = [y_m(t_1), y_m(t_2), \cdots, y_m(t_N)]^T$. Now consider an additive Gaussian noise output model

$$y_m = f(\theta, u) + \epsilon, \tag{2.17}$$

where $\epsilon = [\epsilon(t_1), \epsilon(t_2), \cdots, \epsilon(t_N)]^T$ is the additive measurement noises and is assumed to be Gaussian: $\epsilon \sim N(0, C_\epsilon)$. Denote the true outputs as $y = [y(t_1), y(t_2), \cdots, y(t_N)]^T$, which are given by $y = f(\theta, u) = [f(\theta, u, t_1), f(\theta, u, t_2), \cdots, f(\theta, u, t_N)]^T$ in (17), where $f(\cdot)$ is a functional that maps the network input to the network outputs, and $\theta = [\theta_1, \theta_2, \cdots, \theta_P]^T$ is the unknown network parameters to be identified. In our case, $f(\cdot)$ is defined by the network model and the noise-free outputs can be obtained through model simulation. Furthermore, it is assumed that the prior statistical distributions for θ are available and θ is considered to be multivariate Gaussian: $\theta \sim N(\mu_\theta, C_\theta)$. Under the framework Bayesian inference, we define the posterior probability density for θ

$$p(\theta|y_m) = \frac{p(y_m|\theta)p(\theta)}{p(y_m)}, \tag{2.18}$$

where $p(y_m)$ is the marginal probabilistic density for y_m, which is independent of θ. Hence, θ is identified by maximizing the following likelihood [72]

$$L(\theta) = p(y_m|\theta)p(\theta) \propto p(\theta|y_m) \tag{2.19}$$

From (17) we get: $\epsilon = y_m - f(\theta, u)$, which implies that

$$p(y_m|\theta) = exp\Big(-\frac{1}{2}\big(y_m - f(\theta, u)\big)^T C_\epsilon^{-1}\big(y_m - f(\theta, u)\big)\Big) \tag{2.20}$$

Plugging $p(\theta) = exp\big(-\frac{1}{2}(\theta - \mu_\theta)^T C_\theta^{-1}(\theta - \mu_\theta)\big)$ and (20) into (19) and computing

the logarithm of the likelihood gives

$$\log\{L(\theta)\} = -\frac{\big(y_m - f(\theta,u)\big)^T C_\epsilon^{-1}\big(y_m - f(\theta,u)\big) + (\theta-\mu_\theta)^T C_\theta^{-1}(\theta-\mu_\theta)}{2} \quad (2.21)$$

In this work, the above log likelihood is maximized to estimate θ. This is achieved by interfacing a sequential quadratic programming optimization package [73] with our biological simulator. The evaluation of $f(\theta, u)$ for a given θ in (21) is facilitated by running our biological simulator.

2.3.2 Bistability Check

The previous section lays out the classical framework of Bayesian interference. In this subsection, we address the stability issues. Bistability is one of the most important properties of a memory device. Typical memory cells, including the biological ones we focus on in this section, posses two stable equilibriums (hence bistable), representing the ON and OFF states, respectively. The third equilibrium is unstable and is also referred to as the saddle. Under certain design parameters, gene memory circuits may lose bistability and become monostable, hence stopping operating as a memory device [16][18].

To reliably identify bistability in the first identification step, we propose to employ the presented Bayesian approach to identify the model parameters under lab experiments that are geared to expose the stability property. Once the complete mathematical model is available, bistability can be checked rather straightforwardly by examining the model. An example of a suitable lab measurement setup is shown in Fig. 2.9 (a). Inputs R and S are set up in a way such that the ON-state is being written into the memory circuit with sufficient time margin. After that, input R is disabled and the experiment continues for a sufficiently long period to expose the DC characteristics of the design. The protein As concentration is sampled at the late

part of the experiment and is used to drive the parameter identification. Intuitively, the sampled outputs manifest whether or not the memory circuit can maintain the ON-state once it is written. A similar measurement can be set up to check whether a written OFF-state can be maintained.

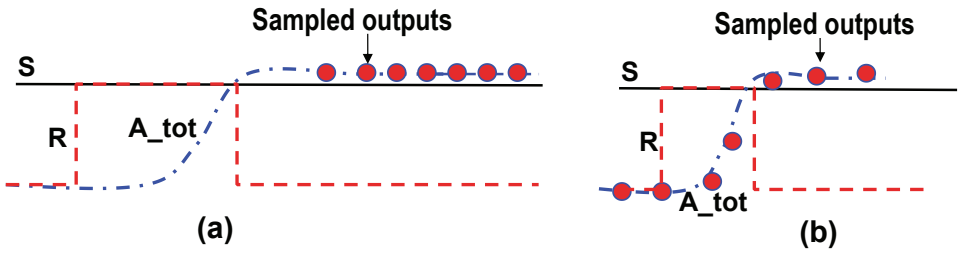

Figure 2.9: Measurement setups for the two-step identification: a) first step (bistability check), and b) second step.

2.3.3 Bistability-Preserving Parameter Identification

If a memory design is identified to be bistable, the second identification step is conducted to further match the measured dynamical behaviors, hence increasing the accuracy of the identified model. Various dynamic measurements can be done. One example is shown in Fig. 2.9 (b). Note that in the case, the entire dynamic output response is sampled. Additionally, the measurement experiment needs not to be taken over an excessively long period of time to expose "DC" characteristics. However, the model identified by this process should maintain bistability.

Fig. 2.10 illustrates how continuous circuit parameter changes may convert a bistable system via saddle-node bifurcation to mono-stable. At bifurcation, one stable equilibrium coalesces with the saddle, and then both equilibriums disappear, leaving only one stable equilibrium [74]. Let us describe the dynamical model of the

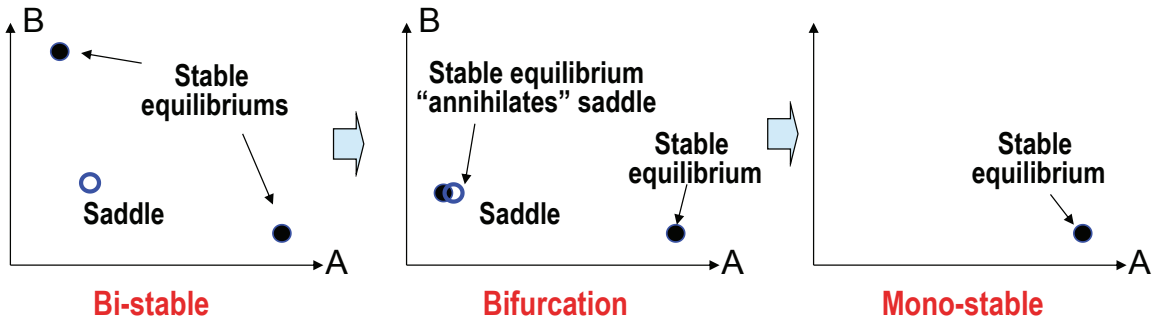

Figure 2.10: Bistability, bifurcation and mono-stability.

entire gene circuit in the following general form

$$\dot{q}\big(x(t)\big) + g\big(x(t)\big) + u(t) = 0, \tag{2.22}$$

where $x(t) = [x_1(t), x_2(t), \cdots, x_M(t)]^T$ is the state vector, and

$$g(\cdot) = [g_1(\cdot), g_2(\cdot), \cdots, g_M(\cdot)]^T. \tag{2.23}$$

The saddle-node bifurcation theory [74] indicates that saddle-node bifurcation happens if the Jacobian matrix g_x, which is given as follows, becomes singular at any equilibrium

$$g_x = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_M} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_M}{\partial x_1} & \frac{\partial g_M}{\partial x_2} & \cdots & \frac{\partial g_M}{\partial x_M} \end{bmatrix} \tag{2.24}$$

To maintain bistability, we solve the following constrained Bayesian inference

problem

$$\min_{\theta} \log\{L(\theta)\} = -\frac{\big(y_m - f(\theta,u)\big)^T C_\epsilon^{-1}\big(y_m - f(\theta,u)\big) + (\theta-\mu_\theta)^T C_\theta^{-1}(\theta-\mu_\theta)}{2} \tag{2.25}$$

$$\text{s.t. } |\lambda_{i,j}| > \delta,\ i = 1,2,\ j = 1,2,\cdots,M, \tag{2.26}$$

where we denote the two stable equilibriums as x_1^e and x_2^e, $\lambda_{i,j}$ is an eigenvalue of the Jacobian g_x evaluated at x_i^e: $g_x(x_i^e)$, and $\delta > 0$ is a user-defined constant. The bistable model parameters identified in the prior step are used as the initial guess for the optimization problem of (23). By adding the constraint in (25), we ensure that all the eigenvalues of each evaluation of the Jacobian is kept away from zero by a certain margin. This constraint ensures that each Jacobian is nonsingular, and hence from the saddle-node bifurcation theory, the model is guaranteed to be bistable.

2.4 Static Noise Margins

As a widely accepted way of analyzing the susceptibility of a system to noise, *noise margins* are commonly defined for electric circuits [75]. More specifically, two types of noise margins are possible: static and dynamic. While static noise margins are widely used for electronic circuits, dynamic noise margins are less well studied even though they are strongly desired [30, 76]. This section first extends the concept of static noise margins for understanding the stability of genetic memory circuits. In the following discussion, each input signal is held at a constant level indefinitely.

To illustrate the static noise margins, we take the state flip from *zero* to *one* as an example. To write *one*, we assume that the $gene_R$ is activated for long enough and the gene expression level of $gene_S$ is maintained at a low level k_S^{low}. A successful write operation is shown in Fig. 2.11 (Left). Let k_R^s denote the transcription rate of $gene_R$ in this successful write process.

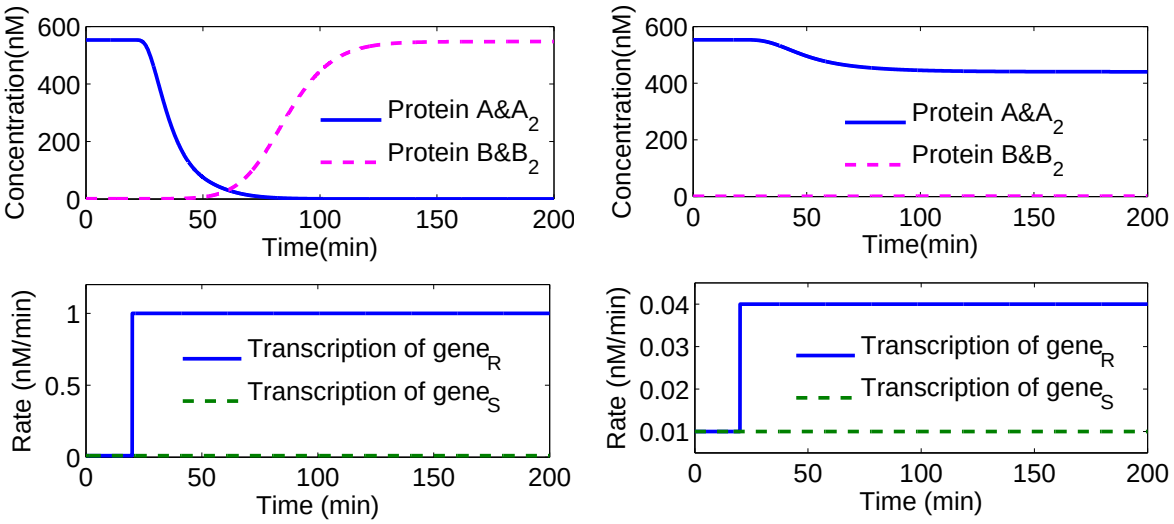

Figure 2.11: (Left) Write *one* successfully with high transcription rate of $gene_R$. The bottom plot shows the input to the conditional memory. The transcription rate of $gene_S$ is kept at low level (0.01 nM/min) while $gene_R$ is activated to transcribe at a high rate (1 nM/min). The top plot shows the response of the memory: the system flips its state from *zero* to *one*. The concentration of $A\&A_2$ decreases from more than 500 nM to almost 0. The concentration of $B\&B_2$ increases from almost 0 to more than 500 nM. (Right) Fail to write *one* with low transcription rate of $gene_R$. Similar to the left panel, the transcription rate of $gene_S$ is also kept at low level (0.01 nM/min). $gene_R$ is activated to transcribe at a rate (0.04 nM/min) higher than that of $gene_S$. However, the transcription rate of $gene_R$ is not high enough to flip the state of the system: the concentration of $B\&B_2$ has little change, and that of $A\&A_2$ is kept high even though it decreases slightly. In this case, the state of the memory circuit is only slightly pushed away from the initial stable state.

In the write process mentioned before, if the transcription rate of $gene_R$ is not activated as high as k_R^s, it may fail to flip the state of the memory. Fig. 2.11 (Right) shows a failing example. Let k_R^f denote the transcription rate of $gene_R$ in this failing write process. Between k_R^f and k_R^s there exists a transcription rate of $gene_R$, which is referred to as threshold k_R^{th}. Only above this value, the activation of the $gene_R$ can write *one* successfully. For the write operation, the *static noise margin* (SNM)

is defined as

$$k_{R,SNM}^{write}(k_R^{write}) = k_R^{write} - k_R^{th}, \tag{2.27}$$

where k_R^{write} is the transcription rate of $gene_R$, $k_{R,SNM}^{write}$ is a function of k_R^{write}. $k_{R,SNM}^{write}$ is a metric of noise tolerance of the memory circuit on k_R^{write}. Similarly, when the memory circuit is holding its state, noises on k_R^{hold} may undesirably perturb the circuit such that the state of the memory flips unexpectedly. Under this circumstance, the static noise margin for the transcription rate of $gene_R$ k_R^{hold} is defined as

$$k_{R,SNM}^{hold}(k_R^{hold}) = k_R^{th} - k_R^{hold}. \tag{2.28}$$

In the above definitions of the static noise margins, the other input k_S is assumed to be fixed at k_S^{low}. In practice, the value of k_S may vary. For a larger value of k_S, after activating $gene_R$, the quantity of protein RS produced would be more than the case with smaller k_S. Therefore, more R molecules are consumed by producing RS. As a result, the concentration of protein R_2 would decrease. This way, a larger k_S results in more RS and less R_2 such that the activation of $gene_R$ is less capable of flipping the state from *zero* to *one*. In other words, the threshold of k_R to flip the state increases with k_S.

As discussed above, k_R^{th} is a function of k_S, as shown in Fig. 2.12. Furthermore, the static noise margin can be more generally defined as

$$k_{R,SNM}^{write}(k_R^{write}, k_S) = k_R^{write} - k_R^{th}(k_S), \tag{2.29}$$

and

$$k_{R,SNM}^{hold}(k_R^{hold}, k_S) = k_R^{th}(k_S) - k_R^{hold}. \tag{2.30}$$

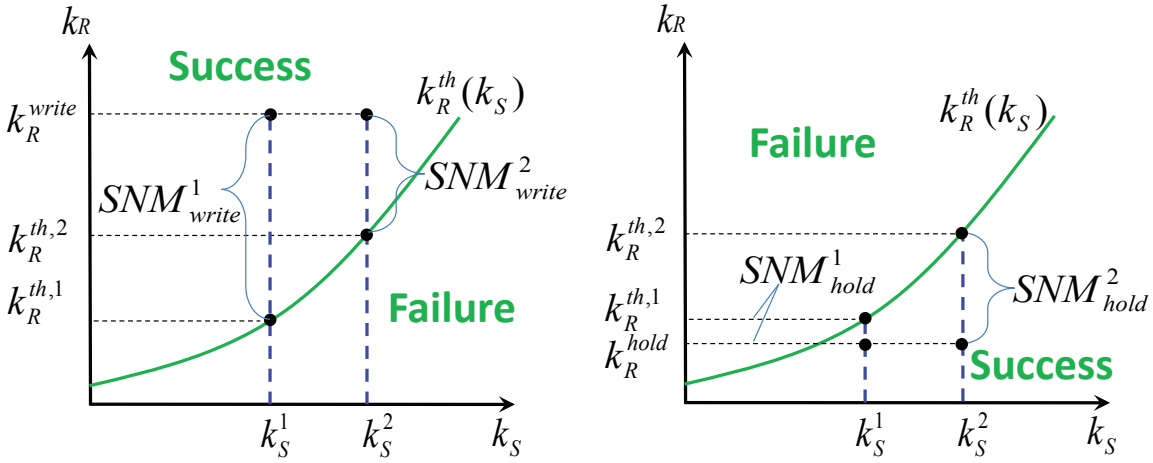

Figure 2.12: SNMs of the conditional memory for varying k_S. (This figure is an illustration.) The left and right figures show the SNMs for write and hold, respectively. In each of the two figures, the curve $k_R^{th}(k_S)$ stands for the k_R^{th} value as a function of k_S. For the write, the regions above and below the curve correspond to successful writes and write failures, respectively. In contrast, for the hold, these two regions correspond to hold failures and successful holds. For fixed k_S values: $k_R^{th,i} - k_R^{hold}$ and $k_R^{write} - k_R^{th,i}$ are defined as SNM^i_{hold} and SNM^i_{write}, respectively, where $i = 1, 2$. As k_R^{th} varies with k_S, the static noise margins for holds and writes are functions of k_S.

In reality, as inputs to the conditional memory circuit, noises may be injected onto both k_R and k_S. For this reason, the static noise margins need to be extended to consider these noises simultaneously. When the memory circuit holds *zero*, in Fig. 2.13, the more general concepts of static noise margins are defined as the distance between (k_S, k_R) and the curve $k_R^{th}(k_S)$ in the $k_S - k_R$ plane. These noise margins account for concurrent noises on k_R and k_S for both hold and write.

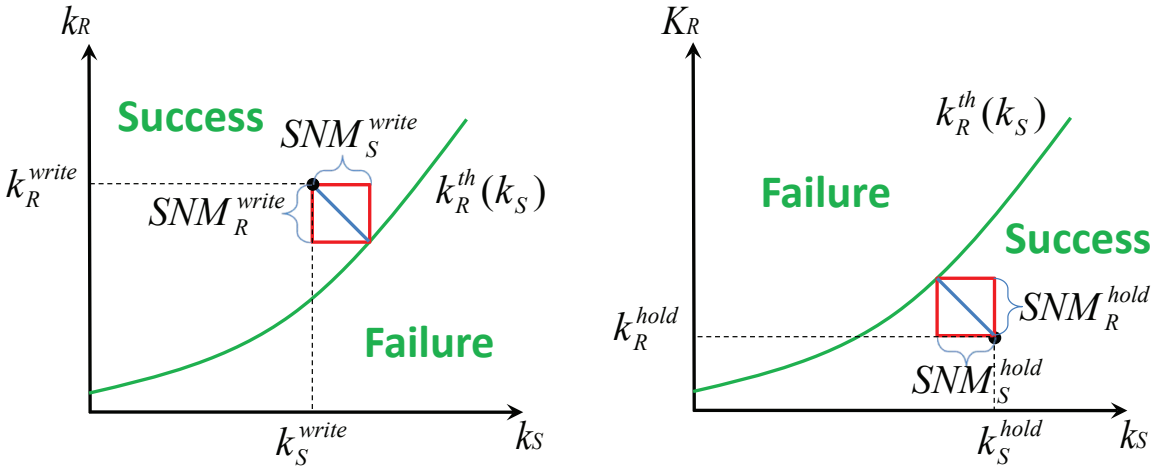

Figure 2.13: SNMs of the conditional memory for k_R and k_S. (This figure is an illustration.) As an extension of situations with fixed k_S values, this definition of SNM takes noises on k_S into consideration. The static noise margin for hold is the distance between the point (k_S^{hold}, k_R^{hold}) and the curve $k_R^{th}(k_S)$. A noise margin which is big enough is supposed to tolerate noises on both k_R and k_S such that the inputs of the memory do not move beyond the curve $k_R^{th}(k_S)$. Similarly, for inputs writing *one* to the memory, the SNM is the distance between the point $(k_S^{write}, k_R^{write})$ and the curve $k_R^{th}(k_S)$.

2.5 Dynamic Noise Margins

2.5.1 Limitation of Static Noise Margins

Conceptually, the static noise margins are effective metrics for characterizing a bistable system. For the genetic conditional memory, they specify the maximum tolerable noise magnitudes. However, the flipping of a system is a dynamic process. Even if the magnitude of noise exceeds the static noise margin, it still takes considerable period of time for the memory to flip.

In practice, every biological or biochemical process takes finite period of time. For instance, if this period is shorter than the time required to flip the memory, the write process would still be a failure even if in a static way the write is predicted to be successful. Fig. 2.14 shows a successful write and a write failure for the same level of $gene_R$ activation with different durations. Therefore, in the write process, the static noise margin may optimistically predict flip of the system under an activation of $gene_R$ with k_R^{write} exceeding the threshold. Similarly, when the memory is holding its state, the static noise margin for hold may pessimistically predict that noises with a magnitude exceeding the static noise margin would destroy the information held.

To this end, the dynamics of the memory circuit must be considered when defining noise margins. As a powerful tool, the concept of state space is used to help to understand the flipping dynamics of a bistable system. To simulate the dynamical behaviors of a gene network, we use the simulator in [23].

2.5.2 State Space of Memory Circuits

Since the dynamics is described as how the concentration of each species varies with time, we model the entire system by the following dynamic equation

$$\dot{x} = f(x), \tag{2.31}$$

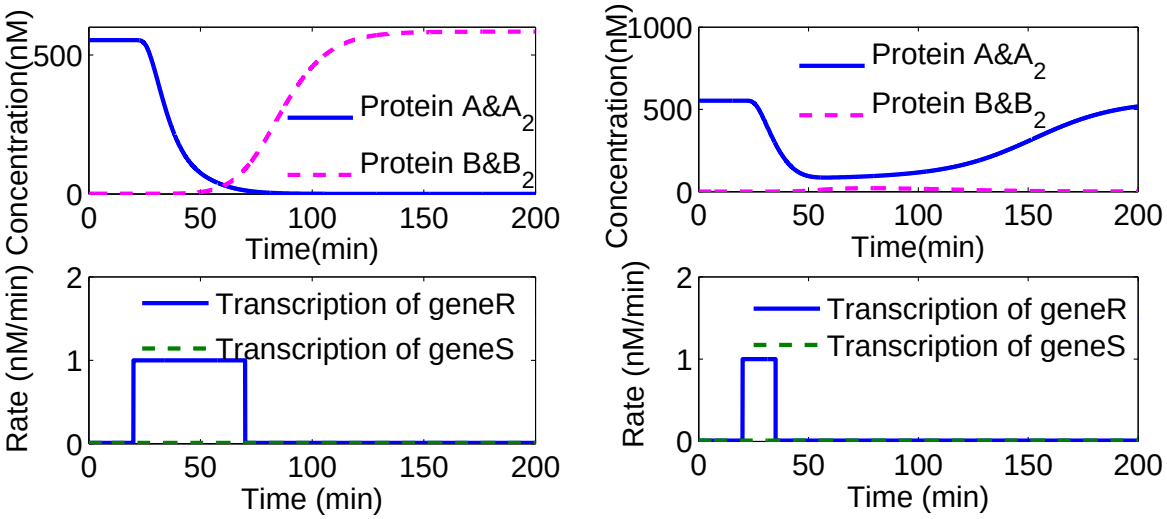

Figure 2.14: (Left) Write *one* successfully by long time activation of $gene_R$. Initially, the memory circuit holds the stable state *zero*. The transcription rate of $gene_S$ is kept at a low level. $gene_R$ is activated for a period of time to flip the state from *zero* to *one* successfully. The duration of $gene_R$ activation is limited to resemble a more realistic activation. (Right) Fail to write *one* by short time activation of $gene_R$. The transcription rate of $gene_S$ is always low. The memory circuit holds *zero* and the transcription rate of $gene_R$ is low at the beginning. Then $gene_R$ is activated to a level which is high enough to write *one* if the duration is long enough. After the activation of $gene_R$, there is a significant decrease of the concentration of $A\&A_2$. The concentration of $B\&B_2$ also increases considerably. However, before $gene_B$ is capable of inhibiting $gene_A$ independently, $gene_R$ is deactivated and its transcription rate comes back to its initial low level ($0.01\ nM/min$).

where $x \in R^N$ is the vector of species concentrations. $f(\cdot)$ represents chemical laws which relate the concentrations of all species and their rates of change. For a bistable system, there are three equilibrium states that satisfy

$$f(x) = 0. \tag{2.32}$$

Two of these equilibrium states are stable and represent the *zero* state and *one* state. The third one is unstable, which is called the *saddle*. As illustrated in Fig. 2.15 (Left).

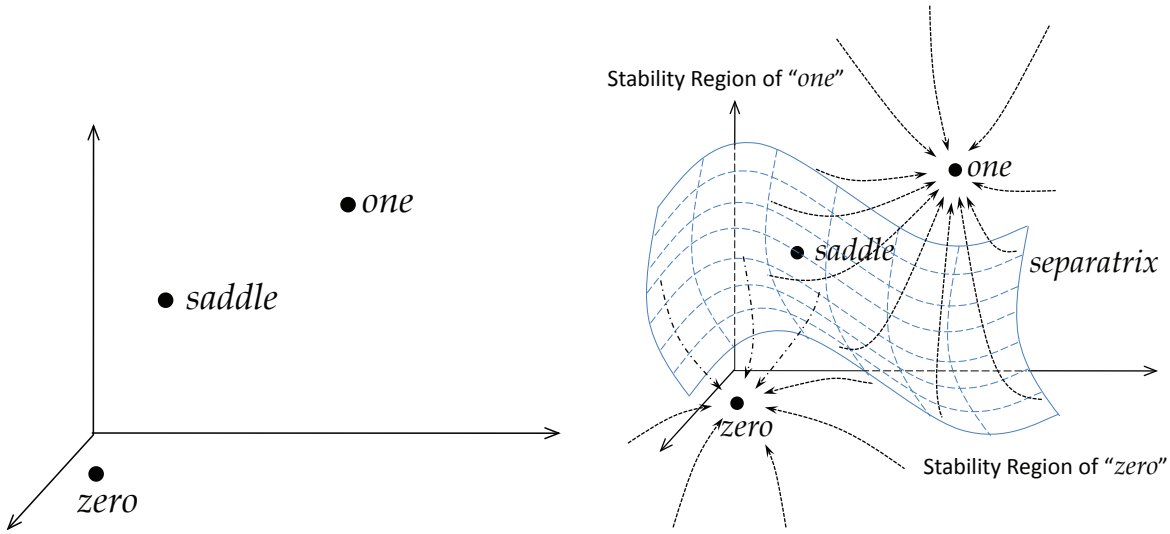

Figure 2.15: (Left) Equilibria of a bistable system. A bistable system has three equilibria, among which two are stable. The third one is called the saddle point, which is unstable. (Right) The state space of a bistable system. A hypersurface separates the state space into two parts, of which each one is the stable region of a stable equilibrium. That is, if the state of the system is within a stable region, it will eventually converge to the corresponding stable equilibrium. The saddle point is on the separatrix and the separatrix is the stable region of the saddle point.

Since each point in the state space of the entire memory circuit stands for a system state, we use trajectories of moving states to describe the transient behavior of the bistable system.

For an equilibrium e, the stable manifold or region of attraction is defined as [74, 77]

$$M^s(e) = \{x \in R^N | \lim_{t\to\infty} T(x,t) = e\}, \tag{2.33}$$

where $T(x,t)$ represents the trajectory of the system state starting from an initial state x, and t is the time variable. Fig. 2.15 (Right) shows the stable manifolds of the two stable equilibria of a bistable system. Similarly, the unstable manifold of the

equilibrium e is defined as

$$M^u(e) = \{x \in R^N | \lim_{t \to -\infty} T(x,t) = e\}. \tag{2.34}$$

From (2.33) and (2.34), it is clear that if the vector field (defined as the derivative of the system state with respect to time) is reversed, we have a new system

$$\dot{x} = -f(x), \tag{2.35}$$

in which all trajectories flow in the reverse direction and the stable manifold of the equilibrium e is the unstable manifold for the original system [30].

For a bistable system, as shown in Fig. 2.15 (Right), there are two stable equilibria and two corresponding stable regions as mentioned before. The stability boundary of the two stable regions is referred to as *separatrix*. Importantly, the *separatrix* is the stable region of the unstable equilibrium or the saddle point [77, 78].

Intuitively, if the state vector of the system falls in the stable region of a stable equilibrium, the dynamics of the system would drive the trajectory towards the corresponding stable equilibrium. If the initial state is on the separatrix, in principle, the system state would end up staying at the saddle. This is the case of metastability. In reality, perturbations may push the state away from the saddle towards a stable equilibrium. Whenever the system is perturbed by certain excitations and/or noises, the system state may be pushed away from a stable equilibrium in the state space. To see whether the system can come back to the original state autonomously, one needs to check whether the state goes beyond the stability boundary/separatrix. If the perturbed state is within the same stable region, the system state would come back spontaneously. Otherwise, the state would be attracted to the other stable

equilibrium and a state flip results.

2.5.3 Dynamic Noise Margins

We define dynamic noise margins (DNMs) to quantify the memory circuit's susceptibility to injected noises. As being dynamic, these dynamic noise margins not only take the magnitude of the noise into consideration as in the case of static noise margins, importantly, they also capture the effects of duration of the noise. In later sections, we further consider the variability of DNMs due to parameter changes within the memory circuit.

The same example of flipping the state from *zero* to *one* in the section of static noise margins would also be used here to show how dynamic noise margins extend SNMs. Therefore, as inputs, the write signals have constant amplitudes in the example. We study the dynamic noise margin in the state space of the memory circuit as illustrated in Fig. 2.16. Besides the amplitude, the timing information of input signals is also taken into consideration in the concepts of dynamic noise margin. In static cases, when inputs of the memory circuit are in the region of *hold "zero"*, the steady-state solutions of the circuit would be in the stable region of *zero* in the state space. Under these circumstances, the inputs push the state away from *zero* but the state does not cross the separatrix. Therefore, the time used to cross the separatrix under these inputs is denoted by ∞ in the following context. For inputs located in the region of *write "one"*, the steady-state solution of the memory circuit is *one*. In this case, the time for crossing the separatrix must be a finite value.

Fig. 2.11 (Left) shows a successful write by infinitely long time of $gene_R$ activation, while for the same magnitude of inputs, the simulation results in Fig. 2.14 demonstrate that finite duration of the $gene_R$ activation may also cause write failure. To this end, we represent a successful write and hold by state trajectories in state

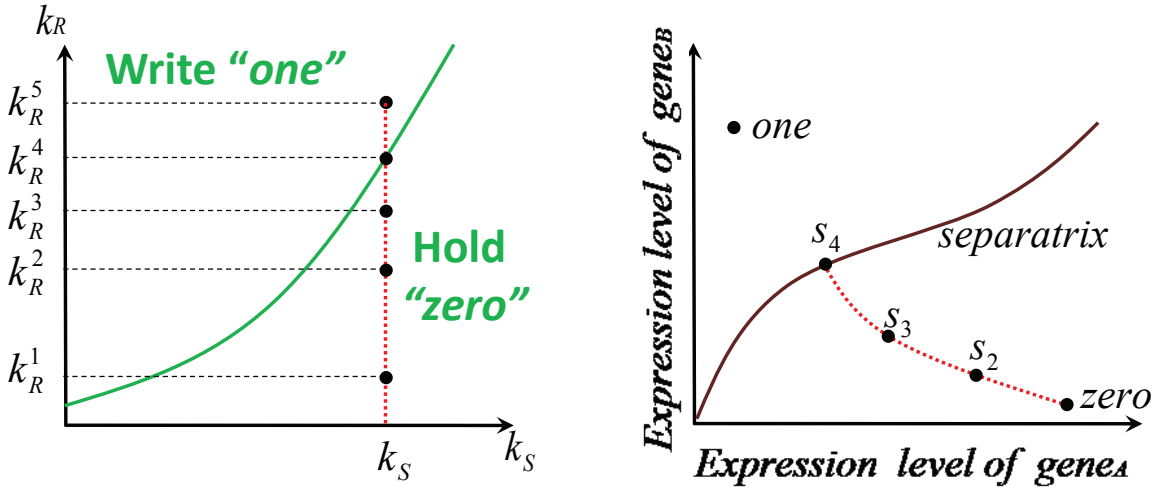

Figure 2.16: Inputs and the steady state of the memory circuit. The values k_R^1,k_R^2,k_R^3,k_R^4 and k_R^5 on the left panel correspond to points *zero*, s_2, s_3, s_4 and *one* on the right panel, respectively. The initial state of the memory circuit is *zero*. In the input space shown in the left figure, if inputs are in the region of *hold "zero"*, the corresponding steady states of the circuit would be in the stable region of *zero* in the right figure. For inputs in the *write "one"* region, the states of the memory circuit would be *one*. And when inputs are on the boundary of *write "one"* and *hold "zero"*, the system states are on the separatrix.

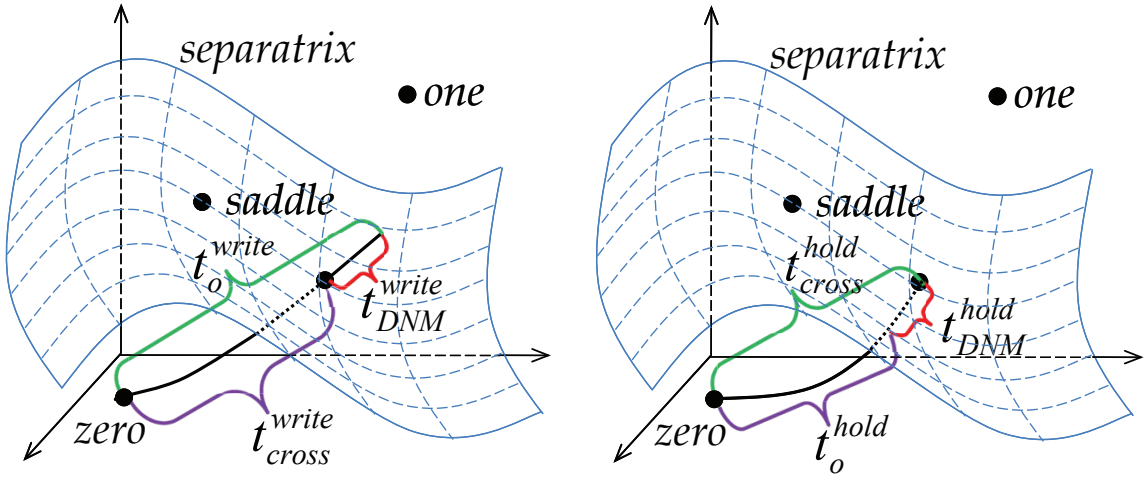

Figure 2.17: The definition of the dynamic noise margins. (Top) The dynamic noise margin is defined as $t_o^{write} - t_{cross}^{write}$ for write. The longer the duration of the inputs, the greater the DNM value. (Bottom) The dynamic noise margin is defined as $t_{cross}^{hold} - t_o^{hold}$ for hold. The shorter the duration of the inputs, the greater the DNM value. As measures to quantify the susceptibility of the memory circuit to injected noises, larger values of DNMs are always preferred.

spaces as shown in Fig. 2.17. In the successful write, the state travels beyond the separatrix before $gene_R$ is deactivated. Therefore, the duration of the $gene_R$ activation must be longer than the separatrix crossing time. For a specific write input, let t_o and t_{cross} denote this duration and the separatrix crossing time, respectively. The dynamic noise margin for the write operation is defined as

$$t_{DNM}^{write} = t_o^{write} - t_{cross}^{write}, \tag{2.36}$$

where t_{DNM}^{write} is positive for a successful write. This quantity characterizes the robustness of the write operation to noises. That is, the further the system state is moved past the separatrix, the less the chance of moving back under stochastic fluctuations on the system. A successful hold is equivalent to a write failure. In Fig. 2.17 (Right), $gene_R$ is deactivated before the state reaches the separatrix. Therefore, in this case, the duration of $gene_R$ activation must be shorter than the separatrix crossing time. If the dynamic noise margin is calculated by (2.36), the value would be negative. Consider the activation of $gene_R$ of which the duration equals to the separatrix crossing time. At the end of this input, the system state would be on the separatrix. This is the case of the metastability. Theoretically, since the dimensionality of the separatrix is lower than that of the entire state space, the probability measure of the metastability is zero. In practice, if the state is close to the separatrix, it also takes long time for the system to reach a stable equilibrium. In addition, random noises may significantly influence the stable equilibrium to which the system would reach. Therefore, the case that the state is close to the separatrix can also be treated as metastability in reality.

Based on all the discussions about the dynamic write processes, the relation

between the predicted results of writes and the DNMs can be summarized as follows

$$\text{result of write}: \begin{cases} success & \text{if } t_{DNM}^{write} > 0 \\ failure & \text{if } t_{DNM}^{write} < 0 \\ metastability & \text{if } t_{DNM}^{write} = 0. \end{cases} \tag{2.37}$$

The hold operation can be analyzed similarly. The DNM for hold is defined as

$$t_{DNM}^{hold} = t_{cross}^{hold} - t_o^{hold}, \tag{2.38}$$

The relation between the predicted results of holds and the DNMs is similar to that of write

$$\text{result of hold}: \begin{cases} success & \text{if } t_{DNM}^{hold} > 0 \\ failure & \text{if } t_{DNM}^{hold} < 0 \\ metastability & \text{if } t_{DNM}^{hold} = 0. \end{cases} \tag{2.39}$$

(2.36) and (2.38) show that the dynamic noise margins are functions of the duration of operations. Because the separatrix crossing time is related to the strength of the input k_R, the DNMs are also functions of k_R. From the analysis in the section of static noise margins, it is clear that the memory inputs k_R and k_S interact with each other. Therefore, the dynamic noise margins should depend on k_R, k_S and the duration of the operation. More general forms of DNM definitions for write and hold are

$$t_{DNM}^{write}(k_R, k_S, t_o^{write}) = t_o^{write} - t_{cross}^{write}(k_R, k_S), \tag{2.40}$$

$$t_{DNM}^{hold}(k_R, k_S, t_o^{hold}) = t_{cross}^{hold}(k_R, k_S) - t_o^{hold}. \tag{2.41}$$

From a memory circuit's perspective, the durations of inputs are controlled by

the external environment. The dynamic noise margin can be easily computed if the separatrix crossing time is known. In the remaining parts of this section, the separatrix crossing time is always a key issue.

2.6 Methodology and Algorithms

2.6.1 Separatrix Tracing for Two-Dimensional Bistable Systems

We review the existing work on separatrix tracing for two-dimensional systems developed for static semiconductor memories [30]. As shown in the left plot of Fig. 2.18, the system S_1 has two state variables V_1 and V_2. Hence, the separatrix is a one-dimensional curve. Another system S_2 is constructed in such a way that $T(t)$ in $[0, t_0]$ is a spontaneous state trajectory of S_2 if and only if $T(t_0 - t)$ in $[0, t_0]$ is a spontaneous state trajectory of S_1. In other words, transient responses of S_2 are inverse processes of those of S_1, as shown in the right plot of Fig. 2.18. Thus, the simulation of S_2 is carried out by conducting the transient simulation of S_1 backwardly in time. Since in S_1 the stability boundary is the stable region of the saddle, in system S_2, that curve becomes the unstable region of the saddle. Therefore, starting from a state close to the saddle, an inverse transient simulation would follow half of the separatrix for the system S_1. This way, the separatrix can be traced by two inverse transient simulations. By conducting transient simulation of the state trajectory starting from an initial state, the separatrix crossing time can be obtained as the time it takes for the state to reach the separatrix.

However, the tracing method becomes inefficient for higher dimensional systems. Take a three-dimensional bistable system as an example. The separatrix is a two-dimensional surface, as illustrated in Fig. 2.19. Since each inverse transient simulation generates a one-dimensional curve, in principle it is impossible to cover the two-dimensional surface by finite number of tracings. Even in practice, potential

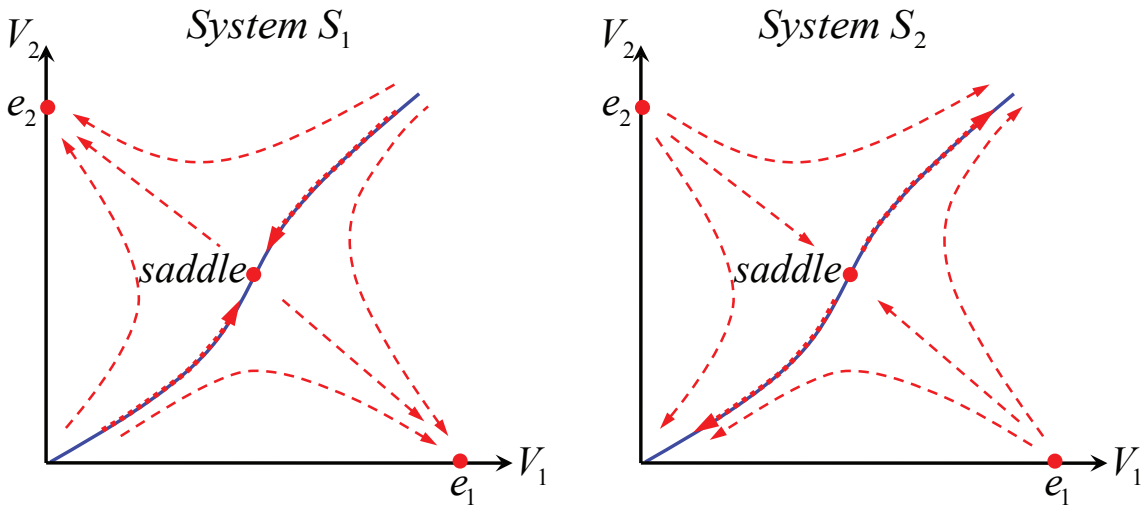

Figure 2.18: Backward tracing of 1-D separatrix. (Left) The state space of a 2-dimensional bistable system S_1 is split into two stable regions of stable equilibria e_1 and e_2. The separatrix is the stable region of the saddle, that is, starting from any point on the separatrix, the spontaneous process of the system would drive the state to the saddle. (Right) The state space of a system S_2, which is constructed in such a way that its transient responses are inverse processes of those of S_1. Therefore, the separtrix for S_1 becomes the unstable region of the saddle, that is, starting from a state in the neighborhood of the saddle, the spontaneous state trajectory for S_2 would follow half of the separatrix.

methods of reconstructing the two-dimensional surface may require a large number of inverse transient simulations. For systems with even higher dimensionality, performance of the inverse tracing deteriorates drastically.

To this end, we propose our exact algorithm to compute the separatrix crossing time of high-dimensional bistable systems directly without obtaining the separatrix. Another tangent-based fast algorithm is also developed. In the following texts, our algorithms are explained in detail starting from related basic system theory.

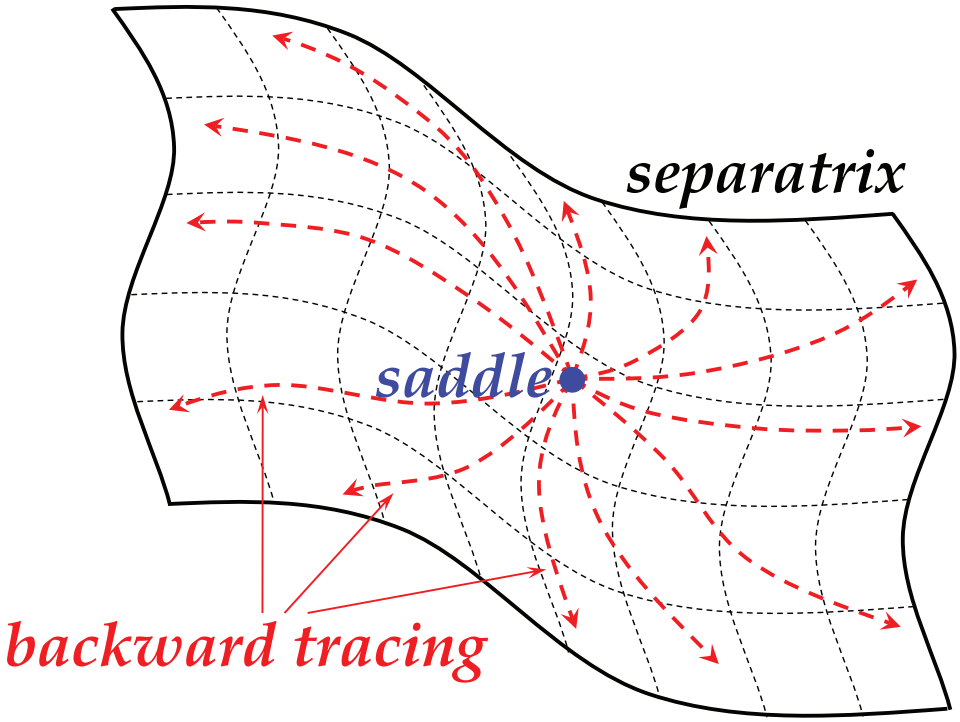

Figure 2.19: Backward transient simulation on a 2-D separatrix. For a two-dimensional separatrix, a backward transient simulation starting from the neighborhood of the saddle generates a state trajectory on the separatrix. However, it is impossible to cover the 2-D surface by finite number of such trajectories.

2.6.2 System Theory of Dynamic Stability

2.6.2.1 Stability of Linear Dynamic Systems

For simplicity, consider a linear dynamic system of which the only equilibrium is at the origin. The dynamic equation is

$$\dot{x} = Ax, \quad x \in R^N, \tag{2.42}$$

where A is the real $N \times N$ system matrix, which has N eigenvalues with the repeated ones counted. We sort them into three groups according to the real part of each

eigenvalue

$$\begin{aligned} &\{\lambda_1, \cdots, \lambda_{ns}\}, \\ &\{\lambda_{ns+1}, \cdots, \lambda_{ns+nu}\}, \\ &\{\lambda_{ns+nu+1}, \cdots, \lambda_{ns+nu+nc}\}, \end{aligned} \tag{2.43}$$

where the first group has negative real parts, the second group has positive real parts and the third group is purely imaginary. The ns, nu, and nc are the numbers of eigenvalues in these three groups and the corresponding eigenvectors are

$$\begin{aligned} &\{u_1, \cdots, u_{ns}\}, \\ &\{u_{ns+1}, \cdots, u_{ns+nu}\}, \\ &\{u_{ns+nu+1}, \cdots, u_{ns+nu+nc}\}, \end{aligned} \tag{2.44}$$

where the ns, nu, and nc satisfy

$$ns + nu + nc = N. \tag{2.45}$$

After a similarity transformation defined by these eigenvectors, the dynamic equation of the system becomes

$$\begin{bmatrix} \dot{\tilde{x}}_s \\ \dot{\tilde{x}}_u \\ \dot{\tilde{x}}_c \end{bmatrix} = \begin{bmatrix} \tilde{A}_s & 0 & 0 \\ 0 & \tilde{A}_u & 0 \\ 0 & 0 & \tilde{A}_c \end{bmatrix} \begin{bmatrix} \tilde{x}_s \\ \tilde{x}_u \\ \tilde{x}_c \end{bmatrix}, \tag{2.46}$$

where the three block rows correspond to the dynamics of the system state within

the subspaces spanned by the corresponding groups of eigenvectors

$$\begin{aligned} S^s &= span\{u_1, \cdots, u_{ns}\}, \\ S^u &= span\{u_{ns+1}, \cdots, u_{ns+nu}\}, \\ S^c &= span\{u_{ns+nu+1}, \cdots, u_{ns+nu+nc}\}. \end{aligned} \tag{2.47}$$

The original N-dimensional space is $S^s \bigoplus S^u \bigoplus S^c$. In (2.46), the $\tilde{x}_s$, $\tilde{x}_u$, and $\tilde{x}_c$ are within S^s, S^u, and S^c, and represent stable, unstable and marginally stable components of the state vector, respectively.

2.6.2.2 Stability of Nonlinear Systems at Fixed Points

For a general nonlinear dynamic system, the system equation is

$$\dot{x} = f(x), \quad x \in R^N, \tag{2.48}$$

where $f(\cdot)$ is a nonlinear equation representing the nonlinear dynamics of the system. The linearization of the system (2.48) by first-order Taylor expansion at an equilibrium point x_e is

$$\frac{d(x - x_e)}{dx} = J^f(x_e)(x - x_e) + o(x - x_e), \tag{2.49}$$

where $J^f(x_e)$ is the Jacobian matrix of the nonlinear system $f(\cdot)$ at point x_e and $o(x - x_e)$ represents higher order terms. Substituting $x^\Delta = x - x_e$ into the above equation leads to

$$\dot{x}^\Delta = J^f(x_e)x^\Delta + o(x^\Delta). \tag{2.50}$$

Because $o(x^{\Delta})$ is higher order terms, the nonlinear system described by (2.50) can be approximated by the linear system

$$\dot{x}^{\Delta} = J^f(x_e)x^{\Delta}, \tag{2.51}$$

where $J^f(x_e)$ is a $N \times N$ system matrix. Applying similarity transformation according to the three groups of eigenvectors of the matrix $J^f(x_e)$, equation (2.51) becomes

$$\begin{bmatrix} \dot{\tilde{x}}_s^{\Delta} \\ \dot{\tilde{x}}_u^{\Delta} \\ \dot{\tilde{x}}_c^{\Delta} \end{bmatrix} = \begin{bmatrix} \tilde{J}_s^f & 0 & 0 \\ 0 & \tilde{J}_u^f & 0 \\ 0 & 0 & \tilde{J}_c^f \end{bmatrix} \begin{bmatrix} \tilde{x}_s^{\Delta} \\ \tilde{x}_u^{\Delta} \\ \tilde{x}_c^{\Delta} \end{bmatrix}, \tag{2.52}$$

where $\tilde{J}_s^f$, $\tilde{J}_u^f$ and $\tilde{J}_c^f$ are the three diagonal blocks in the transformed Jacobian matrix. $\tilde{x}_s^{\Delta}$, $\tilde{x}_u^{\Delta}$ and $\tilde{x}_c^{\Delta}$ are the stable, unstable and marginally stable components of the state vector in the transformed coordinate. Accordingly, the original nonlinear system equation can be expressed as

$$\begin{bmatrix} \dot{\tilde{x}}_s^{\Delta} \\ \dot{\tilde{x}}_u^{\Delta} \\ \dot{\tilde{x}}_c^{\Delta} \end{bmatrix} = \begin{bmatrix} \tilde{J}_s^f & 0 & 0 \\ 0 & \tilde{J}_u^f & 0 \\ 0 & 0 & \tilde{J}_c^f \end{bmatrix} \begin{bmatrix} \tilde{x}_s^{\Delta} \\ \tilde{x}_u^{\Delta} \\ \tilde{x}_c^{\Delta} \end{bmatrix} + \begin{bmatrix} o(\tilde{x}_s^{\Delta}) \\ o(\tilde{x}_u^{\Delta}) \\ o(\tilde{x}_c^{\Delta}) \end{bmatrix}, \tag{2.53}$$

where $o(\tilde{x}_s^{\Delta})$, $o(\tilde{x}_u^{\Delta})$ and $o(\tilde{x}_c^{\Delta})$ are the higher order terms in the stable, unstable and marginally stable subspaces under the transformed coordinates.

The nonlinear system (2.53) and its linearized system (2.52) are linked by the following theorem [78]

Theorem 1 *Suppose the nonlinear dynamic equation (2.53) is* $\boldsymbol{C}^n$, $n \geq 2$. *(2.52) is the linearized system dynamic equation of (2.53). Let* $(\tilde{x}_s^T, \tilde{x}_u^T, \tilde{x}_c^T) = \boldsymbol{0}$ *denote a fixed*

point of (2.53). At $(\tilde{x}_s^T, \tilde{x}_u^T, \tilde{x}_c^T)$*, (2.53) has local invariant stable, unstable and center manifolds with dimensionality of ns, nu, and nc, respectively. These manifolds are tangent to the corresponding invariant subspaces of the linear vector field (2.52) at the fixed point* $(\tilde{x}_s^T, \tilde{x}_u^T, \tilde{x}_c^T)$*.*

The above theorem provides the theoretical basis for computing the tangents of the three manifolds in the state space at a fixed point. In our problem, the fixed point is the saddle, as the separatrix is the stable manifold of the saddle point.

2.6.2.3 Genetic Memory Circuit as a Nonlinear Bistable System

For the conditional memory, if the dimensionality of the state space is N, the separatrix is $(N-1)$-dimensional. Since the separatrix itself is the stable manifold of the saddle point, according to *Theorem* 1, the dimensionality of the tangent hyperplane at the saddle is also $(N-1)$-dimensional.

Since the system under study is a memory circuit, for practical reasons, we do not expect to see oscillations in the network and hence assume that there is no center manifold. According to *Theorem* 1 and (2.45), the summation of the dimensionalities of the three manifolds of (2.53) is N. Therefore, the unstable manifold is 1-dimensional.

For the nonlinear system (2.53), since the tangent of the stable manifold is the stable subspace of the Jacobian matrix $J^f(x_e)$ of the linearized system (2.52), this tangent can be expressed as

$$S^s = span\{v_1, \cdots, v_{N-1}\}, \tag{2.54}$$

where the space is spanned by all the $N-1$ eigenvectors of $J^f(x_e)$ of which the associated eigenvalues have negative real parts. The separatrix and all eigenvectors

at the saddle in the state space are illustrated in Fig. 2.20, where v_N denotes the only eigenvector with positive real part.

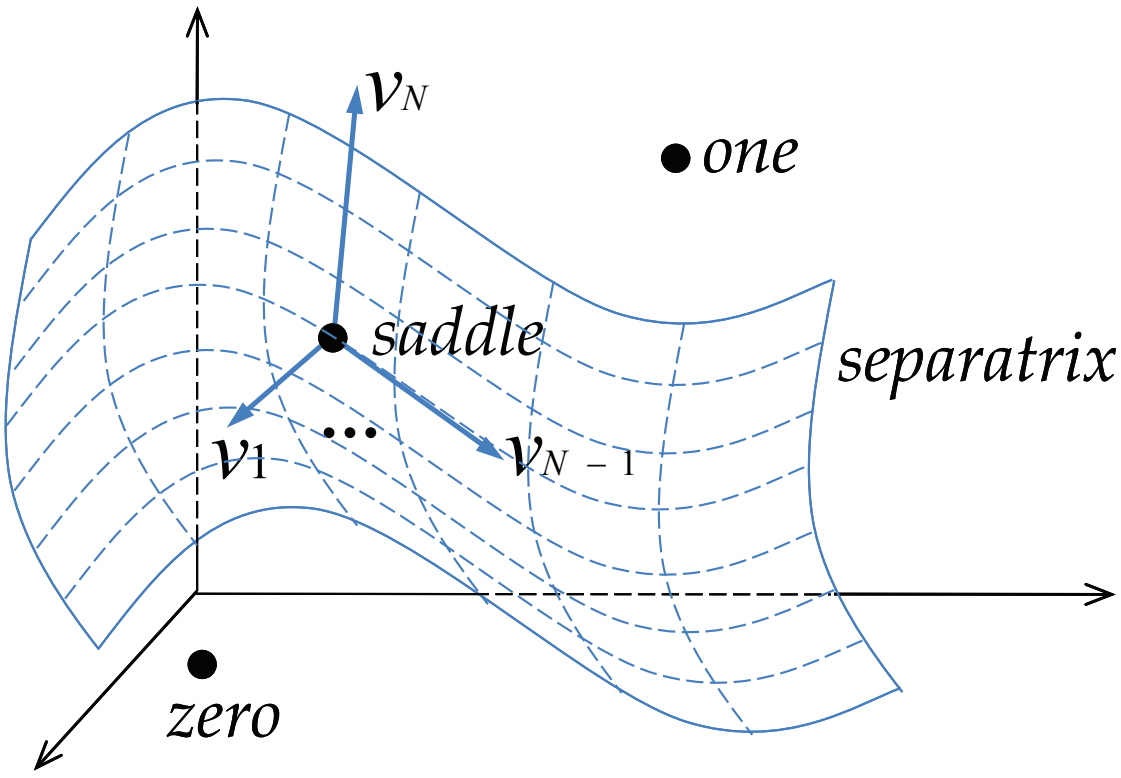

Figure 2.20: Eigenvectors of the system Jacobian at the saddle point. For a N-dimensional bistable memory circuit, the Jacobian matrix has N eigenvectors, among which $N-1$ are stable ones, and the other one is unstable. Stable eigenvectors are tangent to the separatrix while the unstable one is not.

2.6.3 Tangent Crossing Point/Time

Since the shape of the real separatrix is fairly complicated and it is difficult to find a compact expression of the nonlinear high-dimensional manifold, the tangent hyperplane discussed in the previous subsection is used to approximate the real sep-

aratrix. In addition, it is more straightforward to check whether the state trajectory has crossed the tangent or not by

$$v_n \cdot (x - x_e), \tag{2.55}$$

where the normalized vector v_n is perpendicular to the tangent hyperplane, and x_e is the saddle point. As illustrated in Fig. 2.21, if the value of the above expression is 0, the state x is on the tangent, while non-zero values indicate that the state is off the tangent: the absolute value is the distance between x and the tangent, and the sign of the value indicates which side of the tangent a state x is at. During the simulation of the state trajectory, the state can be checked by the above expression on the fly to see at which point the trajectory intersects the tangent. The computation of the normal vector is discussed in the next subsection.

2.6.4 Computing the Tangent and the Normal Vector

In this subsection, we address the problem of computing the normal vector of the tangent (2.54). Staying the same as before, we use $v_1, \cdots, v_{N-1}$ to denote all eigenvectors whose eigenvalues have negative real parts. Let v_N denote the only eigenvector whose eigenvalue has positive real part.

We first consider the simplest case, where all eigenvectors of the Jacobian matrix at the saddle point x_e are real. In this case, the tangent hyperplane is the space spanned by all eigenvectors with negative eigenvalues (2.54). The normal vector of the tangent can be computed in the following way.

Applying the Gram-Schmidt Orthogonalization Process to

$$\{v_1, \cdots, v_N\} \tag{2.56}$$

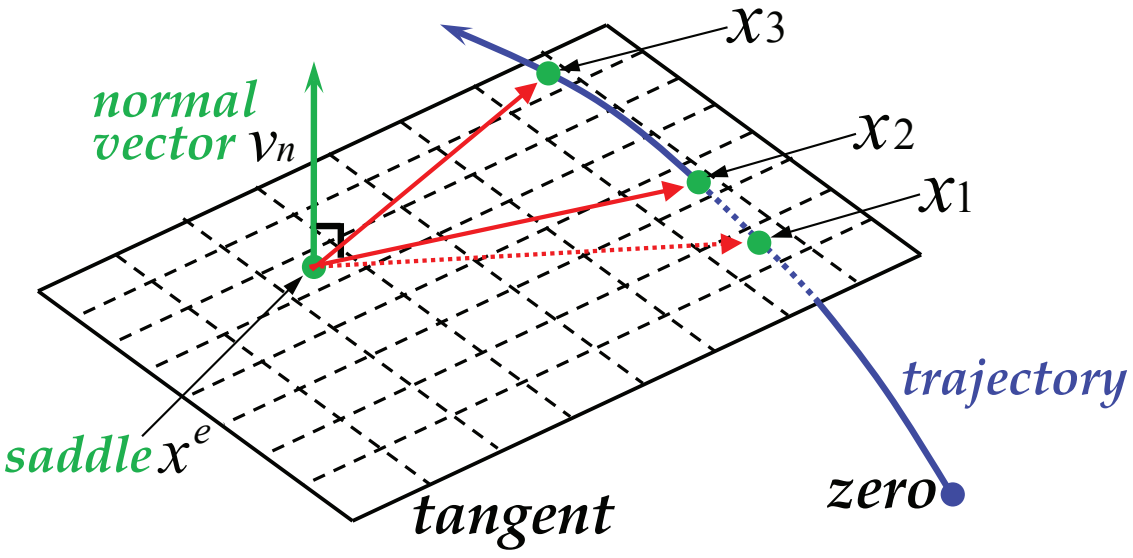

Figure 2.21: Check the tangent crossing point. The trajectory starts from *zero*, and crosses the tangent at x_2. Because both x_2 and the saddle are on the tangent, and the normal vector is perpendicular to the tangent, the inner product $v_n \cdot (x_2 - x^e)$ is 0. For a point x_1 between *zero* and x_2 on the trajectory, the inner product $v_n \cdot (x_1 - x^e)$ is negative because the angle between v_n and $x_1 - x^e$ is obtuse. For point x_3 on the other side of the tangent, the inner product $v_n \cdot (x_3 - x_e)$ is positive because the angle is acute. If the normal vector is normalized, the absolute value of the inner product $|v_n \cdot (x - x_e)|$ is the distance between the tangent and point x. There are two normalized normal vectors of a tangent plane with opposite directions: v_n and $-v_n$. The above analysis also applies to $-v_n$. In this case, the three inner products mentioned before are 0, positive and negative, respectively. In either case, if the inner product is checked on the fly, when the trajectory is crossing the tangent, the inner product would change sign. This can be used to check when the trajectory crosses the tangent.

leads to

$$\{v\prime_1, \cdots, v\prime_N\}, \tag{2.57}$$

where all these N vectors obtained are orthonormalized and $span\{v\prime_1, \cdots, v\prime_i\} = span\{v_1, \cdots, v_i\}$ for $1 \leq i \leq N$. Thus

$$v\prime_N \perp span\{v\prime_1, \cdots, v\prime_{N-1}\}, \tag{2.58}$$

and

$$span\{v\prime_1, \cdots, v\prime_{N-1}\} = span\{v_1, \cdots, v_{N-1}\}, \tag{2.59}$$

where $span\{v_1, \cdots, v_{N-1}\}$ is the tangent hyperplane Therefore, the normal vector of the tangent plane is

$$v_n = v\prime_N. \tag{2.60}$$

However, in most situations the Jacobian matrix J is asymmetric. Therefore, some eigenvalues are likely to be complex numbers. In this scenario, without loss of generality, let $x + iy$ and $u + iv$ denote one of the complex eigenvalues and the associated eigenvector, respectively, where x, y are real numbers and u, v are real vectors. Since J is a real matrix, complex eigenvalues must come in conjugate pairs, that is, $x - iy$ and $u - iv$ are also an eigenvalue and the corresponding eigenvector of J, respectively. Because J has only one unstable eigenvector, the associated eigenvalue must be real. Otherwise the unstable eigenvalue as well as the corresponding eigenvector would come in a pair. Therefore, all complex eigenvalues correspond to stable eigenvectors of J. In this scenario, before applying the Gram-Schmidt Orthogonalization Process to all eigenvectors as shown in (2.56), we add an additional step: $u + iv$ and $u - iv$ are replaced by u and v for each conjugate pair in the eigenvector set. Then, the following steps are carried out as if all eigenvectors are real. In this procedure, the added step is interpreted as follows.

For the conjugate pair of eigenvalues and eigenvectors

$$J \cdot (u + iv) = (x + iy) \cdot (u + iv) \tag{2.61}$$

$$J \cdot (u - iv) = (x - iy) \cdot (u - iv) \tag{2.62}$$

(2.61)+(2.62) and (2.61)-(2.62) lead to

$$J \cdot u = x \cdot u - y \cdot v, \tag{2.63}$$

$$J \cdot v = x \cdot v + y \cdot u. \tag{2.64}$$

For any vector $w \in \{p|p = s \cdot u + t \cdot v, s \in R, t \in R\}$,

$$\begin{aligned} J \cdot w &= (s \cdot x + t \cdot y) \cdot u + (t \cdot x - s \cdot y) \cdot v \\ &\in \{p|p = s \cdot u + t \cdot v, s \in R, t \in R\}. \end{aligned} \tag{2.65}$$

Therefore, for any $w \in span\{u, v\}$, $J \cdot w \in span\{u, v\}$.

For the dynamic equation of the linearized system

$$\dot{x} = J \cdot x, \quad x|_{t=0} = w, \tag{2.66}$$

the state x would stay in $span\{u, v\}$.

As stated above, we can conclude that the tangent hyperplane is spanned by real eigenvectors, and real parts and imaginary parts of complex eigenvectors.

2.6.5 The Exact Algorithm for Computing the Separatrix Crossing Time

We assume that the separatrix crossing time is close to the tangent crossing time. Therefore, a searching based refinement starting from the tangent crossing point is conducted subsequently. The searching based refinement has two steps: looking for a time interval including the separatrix crossing time and binary search within this interval. When looking for the time interval, there are two possibilities. One is that until the tangent crossing time, the trajectory has not crossed the separatrix. The other case is that the trajectory crosses the separatrix before reaching the tangent.

For the first case, the refinement is illustrated in Fig. 2.22 (Left). Let t_1 and $x(t_1)$ denote the tangent crossing time and the corresponding state, respectively. Then we do a *transient test* which is called $check_1$ starting from $x(t_1)$ to check whether t_1 is before the separatrix crossing time or not. For a transient test, the memory circuit is assumed to be free from noise. In addition, the system inputs are set in a way as if the memory is in hold. This way, the simulation follows the rule that the test trajectory would head to certain equilibrium if and only if the starting point is within its stable region.

As shown in Fig. 2.22 (Left), the result of the transient test starting from $x(t_1)$ indicates that the trajectory has not reached the separatrix. Therefore, we continue the simulation of the trajectory. Our method does a transient test every time after the simulation proceeds for Δt. Since a transient test is time consuming, we reduce the number of them by doubling Δt after every use. After finishing $check_3$, which is the first transient test indicating that the starting point is in the stable region of *one*, we know that separatrix crossing time is within the last Δt, so binary search is applied to find it.

For the second case, $check_1$ indicate that t_1 is in the stable region of *one*. In this scenario, the next time point to conduct the transient test is before the tangent crossing time, as illustrated in Fig. 2.22 (Right).

In the process of refinement, it may happen that one transient test converges to the saddle or enters a small neighborhood of the saddle. Under this circumstance, the starting point of the transient test is almost on the stability boundary. Therefore, the searching process as well as the entire algorithm ends at this point.

The searching based exact algorithm for computing the separatrix crossing time is summarized in the flowchart in Fig. 2.23.

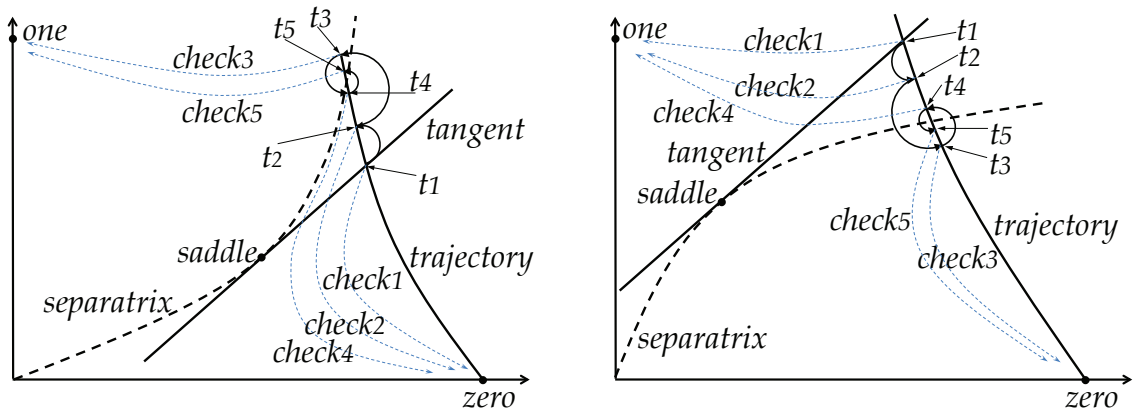

Figure 2.22: (Left) Illustration of search for the separatrix crossing point. While conducting the simulation of the state trajectory starting from *zero*, it is straightforward to find out the tangent crossing time t_1 by checking the sign of $v_n \cdot (x(t) - x_e)$ on the fly, where v_n is the normal vector of the tangent, x_e and $x(t)$ are the saddle and the system state at time point t, respectively. The transient test $check_1$ indicates that $x(t_1)$ is in the stable region of *zero*. Therefore, the simulation of the trajectory continues for Δt and reaches $x(t_2)$, where $t_2 = t_1 + \Delta t$. Because $x(t_2)$ is still in the stable region of *zero*, we double Δt and continue the simulation until $t_3 = t_2 + \Delta t$. Since the $x(t_3)$ is in the stable region of *one*, we do binary search in $[t_2, t_3]$ to find and check t_4 and t_5. (Right) The other case of search for the separatrix crossing point. Here, the transient test $check_1$ indicates that the tangent crossing point $x(t_1)$ is within the stable region of *one*. Therefore, next state to be checked is $x(t_2)$, where $t_2 = t_1 - \Delta t$. Because $x(t_2)$ is still within the stable region of *one*, we double Δt and check the state $x(t_3)$, where $t_3 = t_2 - \Delta t$. Since $x(t_3)$ is in the stable region of *zero*, the separatrix crossing time is within $[t_3, t_2]$. Then, binary search is applied similarly.

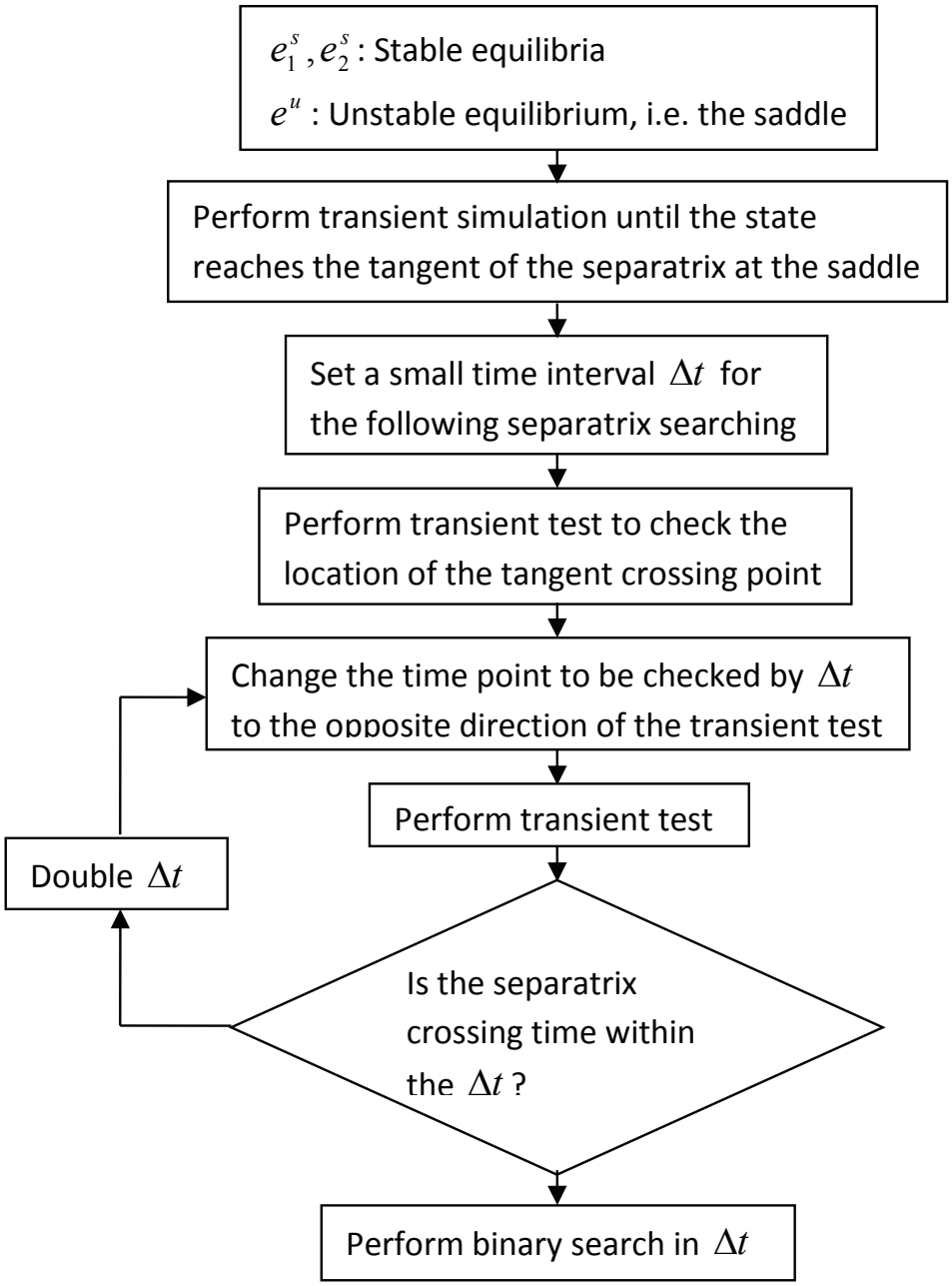

Figure 2.23: Flowchart of the search algorithm. The flow is divided into 3 steps: 1) The state trajectory starts from the stable equilibrium e_1^s and reaches the tangent; 2) As illustrated by the loop, based on the results of transient tests, time points to be checked change monotonically until a time interval containing the separatrix crossing time is found; 3) Binary search is applied to find the exact separatrix crossing time.

2.7 Results

Our biological simulator is implemented in C/C++ by extending an SPICE-like electronic simulation environment. The sequential quadratic programming optimization package [73] is interfaced with our simulation to perform optimization-based parameter identification. All our experiments are conducted on a Linux server.

2.7.1 Simulation of the Conditional Memory

Within our biological simulation environment, we model the conditional genetic memory circuit in [18] using the proposed electrical-equivalent models and perform various dynamic simulations. The reported values in [18] are used to set the model parameters.

In Fig. 2.24, we demonstrate the write operation of the memory circuit using two clock signal (R) durations: 16 minutes and 15 minutes. In the figure, m_R and m_S and the transcription rates for Genes R and S, which are the inputs, and A_{total} and B_{total} are the protein concentrations of species A and B, which are the outputs. As can be seen, a 16-minute long clock signal leads to a success write while the write operation fails when the duration of the clock signal is reduced to 15 minutes.

In Fig. 2.25, we set two clock signal durations, 53 minutes and 52 minutes, to write a logic "one" (on-state) to the memory. As can be seen, the duration of 52 minutes is too short, which leads to a failure. In Fig. 2.26, input "R" is held high for a long period of time. The memory stays in the expected on-state, however, the protein concentration A_{tot} is lower, indicating that the presence of external inputs can alter the bistability of the cell, hence making it possible to overwrite the state. In this case, the presence of high RS signals weakens bistability. This is because that the resulting dimers R_2 and RS reduce the efficiency of promoters A and B, respectively.

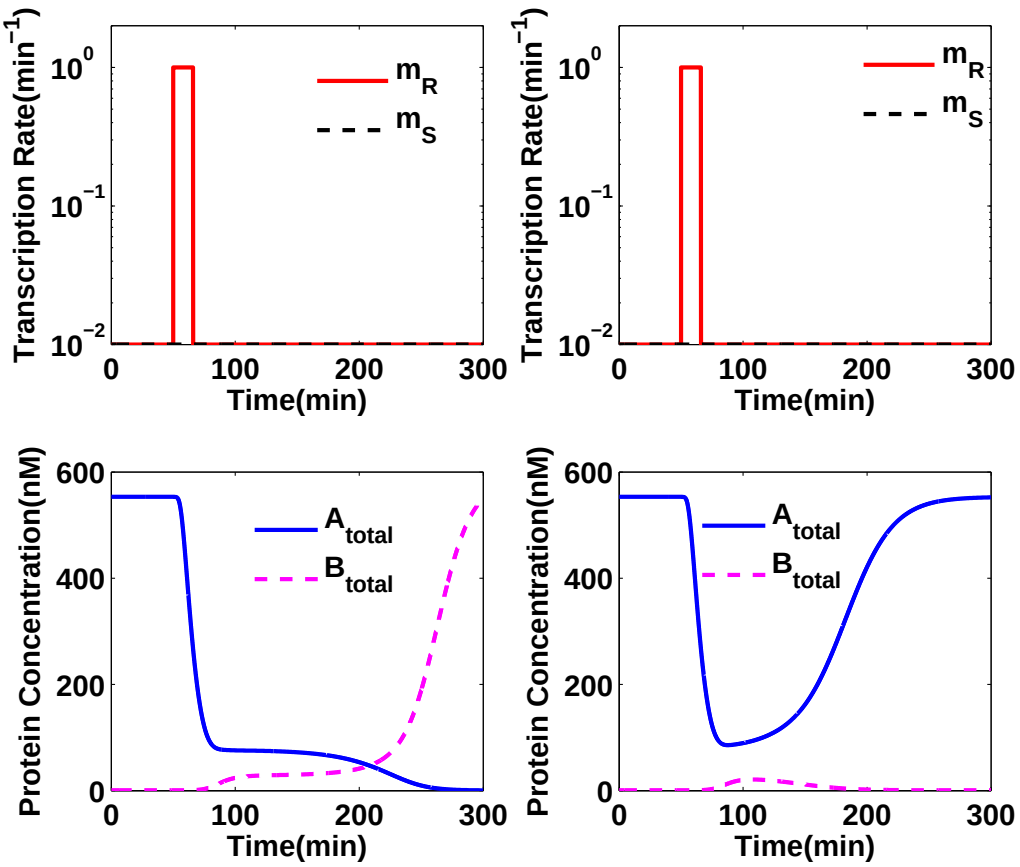

Figure 2.24: Writing a "zero". Left: 16-minute clock signal; Right: 15-minute clock signal.

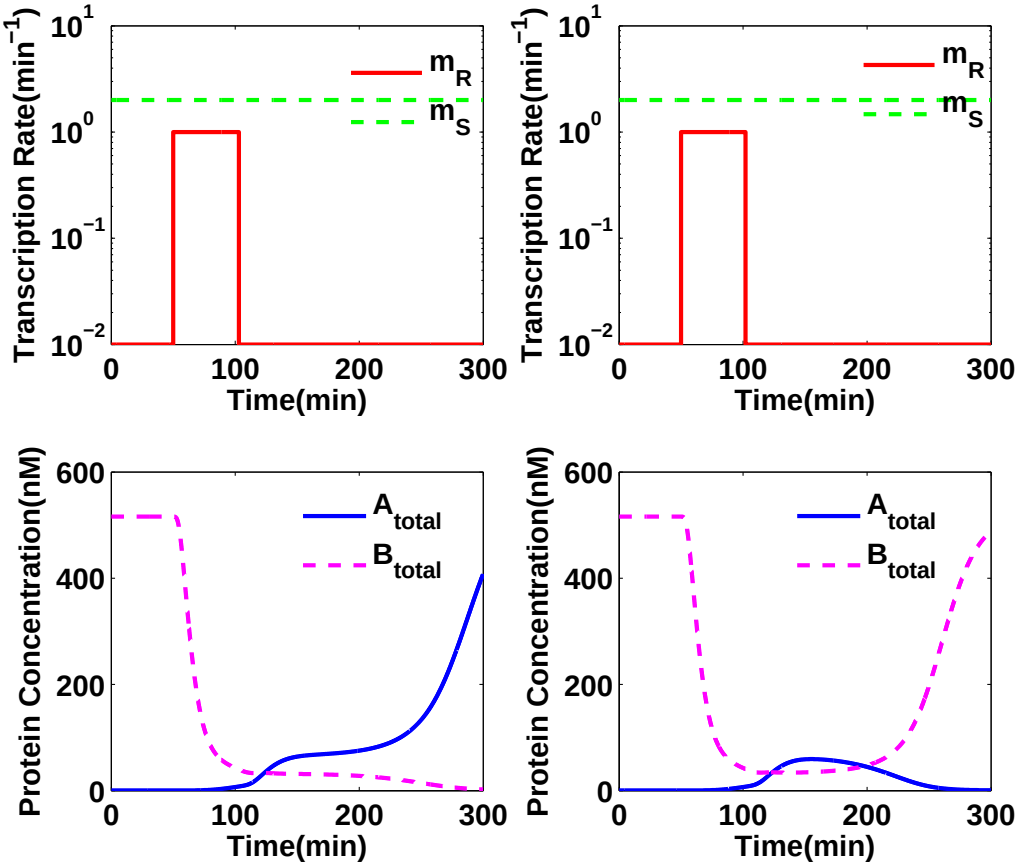

Figure 2.25: Writing a "one". Left: 53-minute clock signal; Right: 52-minute clock signal.

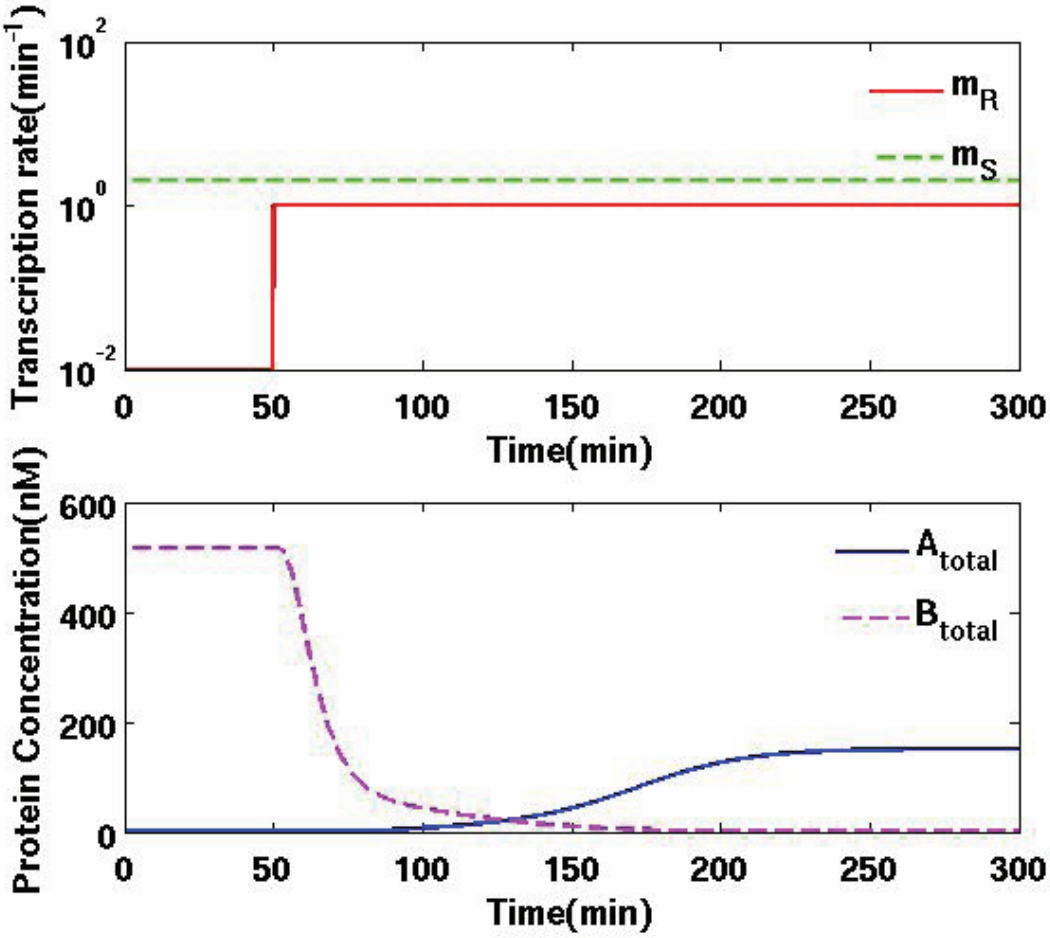

Figure 2.26: Impacts of inputs on bi-stability.

Next, through simulation, we demonstrate how the change of key biological parameters will alter the behaviors of the memory. It is understood biologically that a higher degradation rate of proteins and RNAs leads to faster circuit response [18]. In see this, we double all degradation rates and also double the transcription and translation rates to maintain the same levels of output proteins. In Fig. 2.27, we repeat the simulations done in Fig. 2.24. Now for a successful write, the duration of the clock signal is shortened from 16 minutes to 9 minutes. For the case of writing a logic "one", faster circuit responses are also observed as shown in Fig. 2.28. Variations of biological parameters can also render the circuit loose bistability. We change the degradation rate of mRNA B from 0.231/min to 0.347/min, and those for protein B and the dimer B2 from 0.139/min to 0.277/min. As can be seen in Fig. 2.29, the circuit becomes mono-stable and the off-state cannot be written.

2.7.2 Bayesian Parameter Identification

We demonstrate the application of the proposed parameter identification techniques using two genetic memory circuits. For each of these circuits, lab experiment measurements are mimicked by simulated output responses with added random noises. When generating the "measurements", a set of model parameters are chosen for each circuit while these model parameters are not exposed to the parameter identification process. Since the "exact" models are known, bistability can be checked before hand and used as a known reference. Running on a Linux server, our optimization-based parameter identification takes a few hours to complete for each circuit.

The first genetic memory circuit is created to be bistable. We apply the proposed two-step parameter identification. The first step successfully identifies the bistability of the network and the identification proceeds to the second step. Fig. 2.30 shows

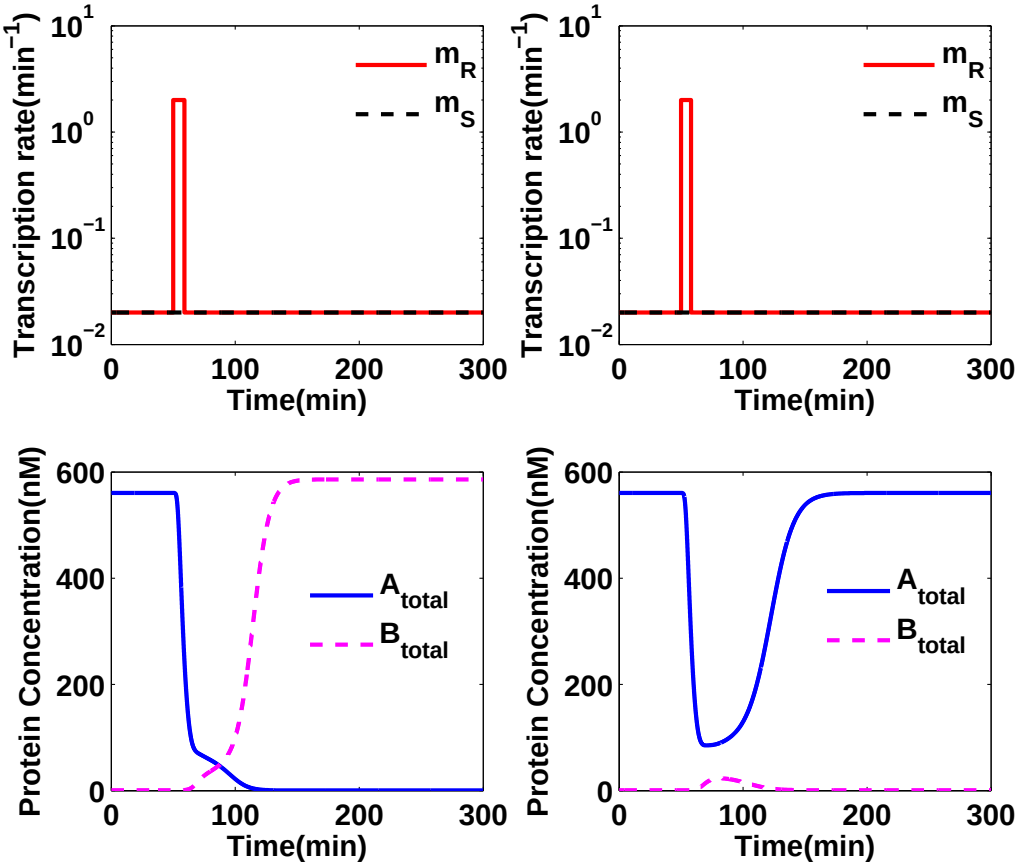

Figure 2.27: Writing a "zero" to the faster circuit. Left: 9-minute clock signal; Right: 8-minute clock signal.

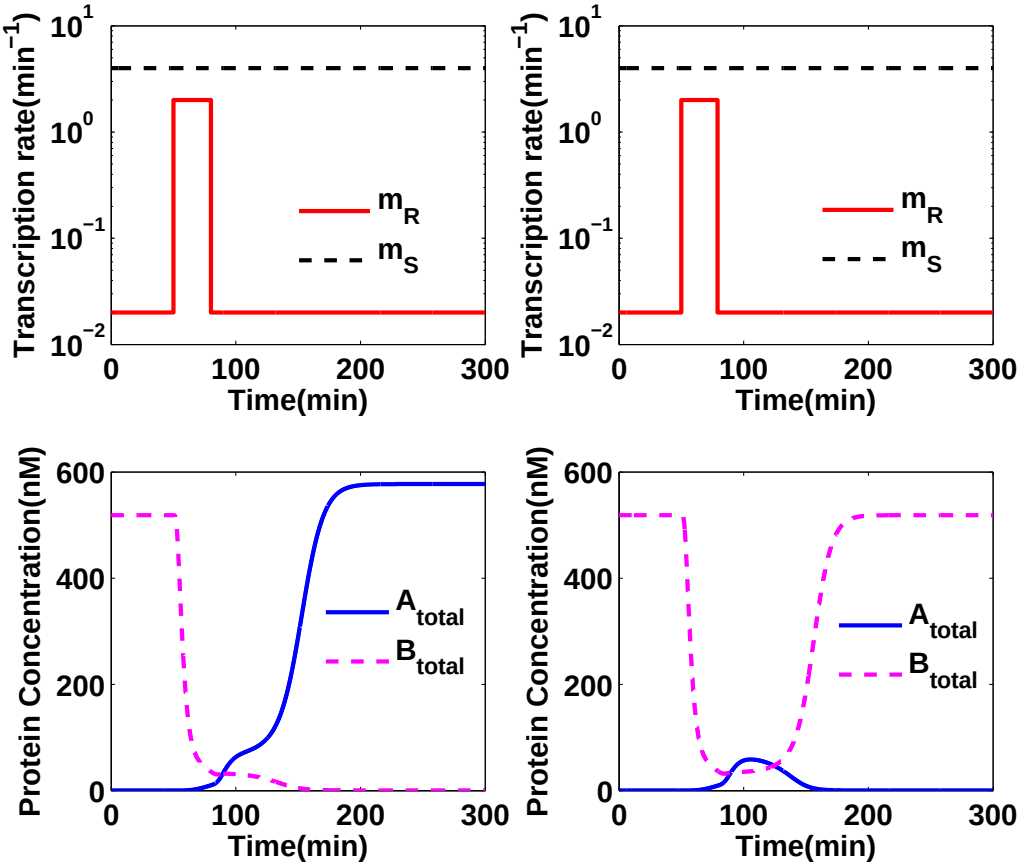

Figure 2.28: Writing a "one" to the faster circuit. Left: 30-minute clock signal; Right: 29-minute clock signal.

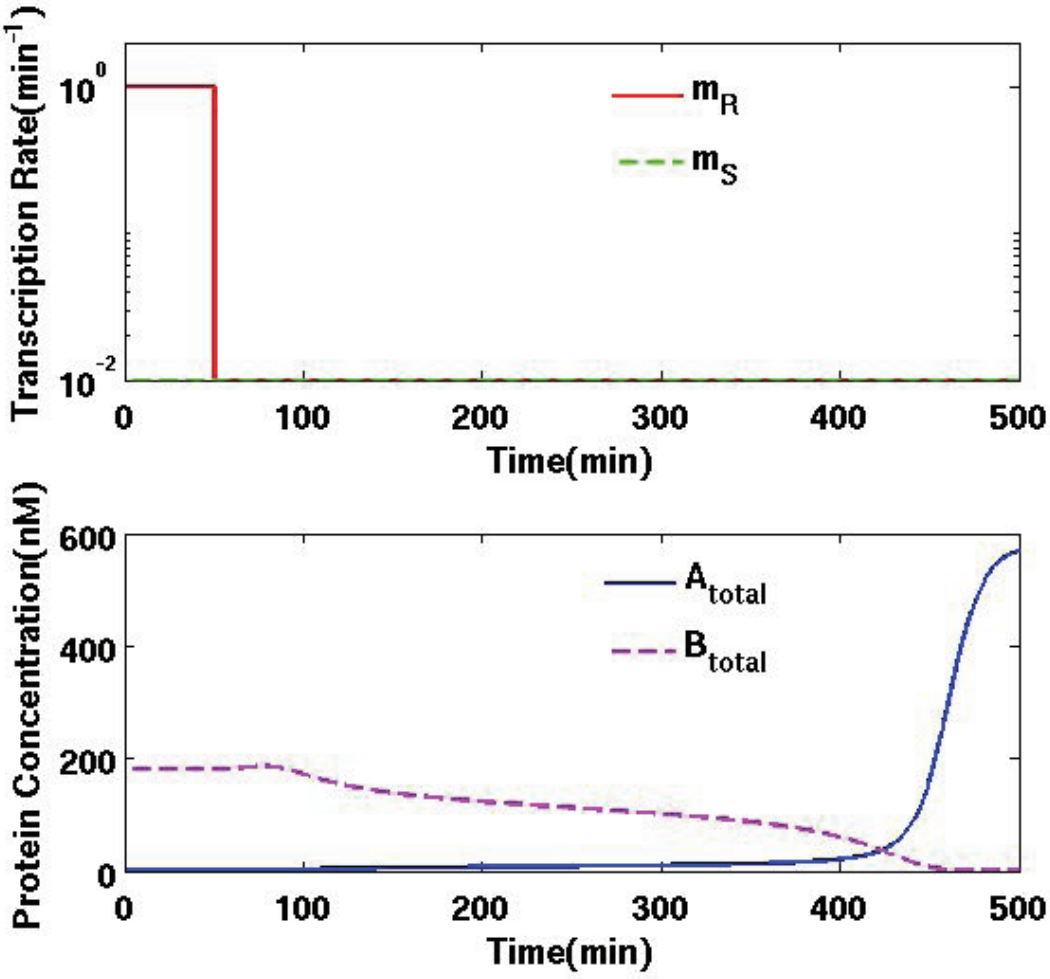

Figure 2.29: Loss of bistability due to parameter change.

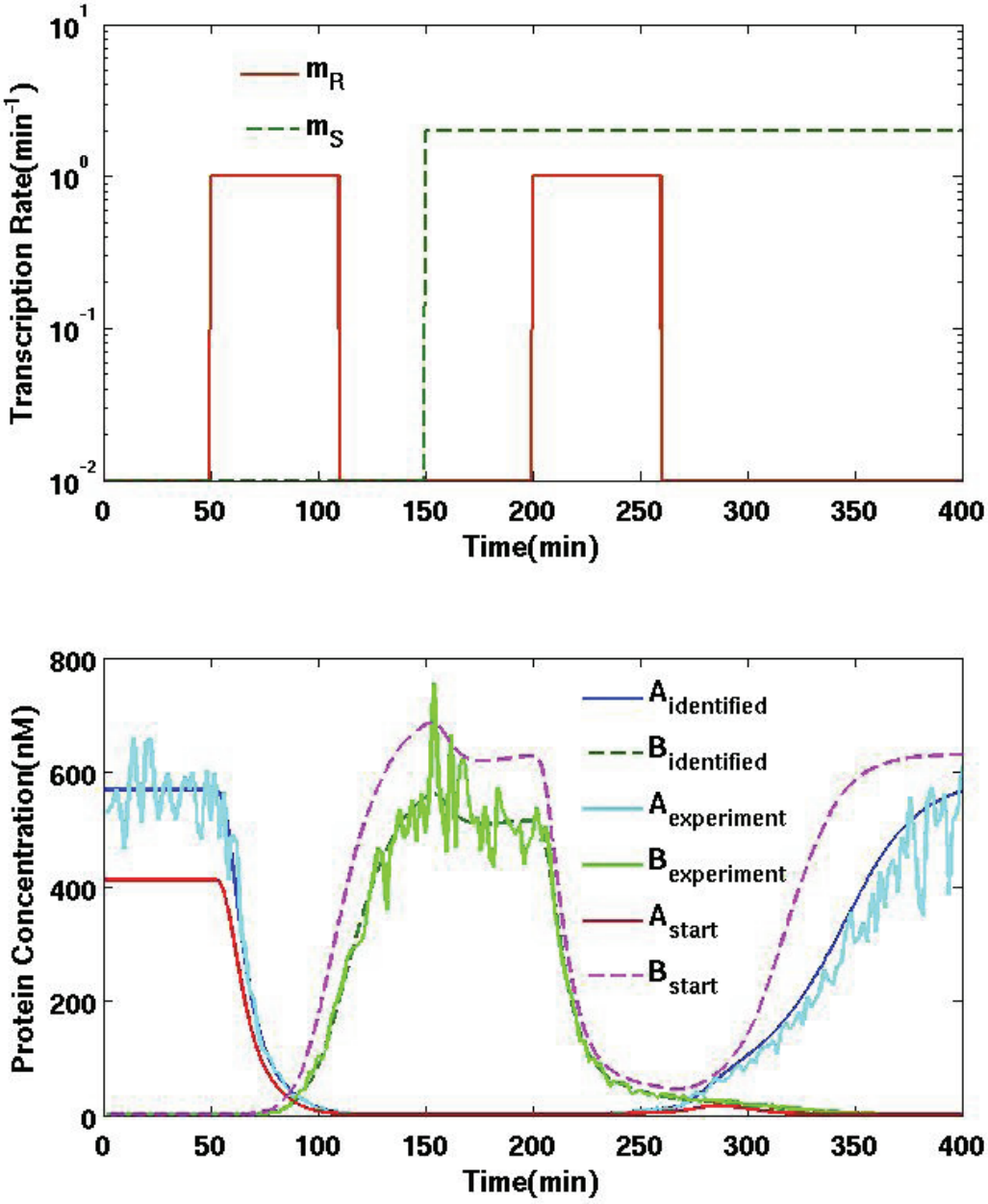

Figure 2.30: Parameter identification of the bistable circuit.

the final parameter identification results in the second step. In the figure, the inputs used to generate the measurements are shown on the top. On the bottom, the "measured" protein output levels (with noise) that are used as the inputs to parameter identification are labeled as "experiment" while the ones based on the identified model are labeled as "identified". A good agreement between the true model and the identified model is achieved. To start the iterative optimization based parameter identification process, an initial guess (model) needs to be provided. The outputs for this initial model are labeled as "start". In Table 2.1, the true model and identified model are compared on the transcription and translation rates of the four DNA or RNA species.

Table 2.1: True vs. Identified Models for the Bistable Circuit.

Parameter	DNA-A	DNA-B	RNA-A	RNA-B
True model	5.0/min	5.0/min	2.3/min	2.3/min
Identified	5.062/min	5.007/min	2.331/min	2.306/min

The second circuit is created to be mono-stable. The first step of parameter identification successfully identifies the mono-stability and the second step is not conducted. Similar to Fig. 2.30, we compare the true model, identified model and initial model but based on the results of the first step, in Fig. 2.31. In this case, the mono-stable system cannot be written to the off-state properly. Even though the measured outputs have large noise, the Bayesian inference still predicts the mono-stability correctly.

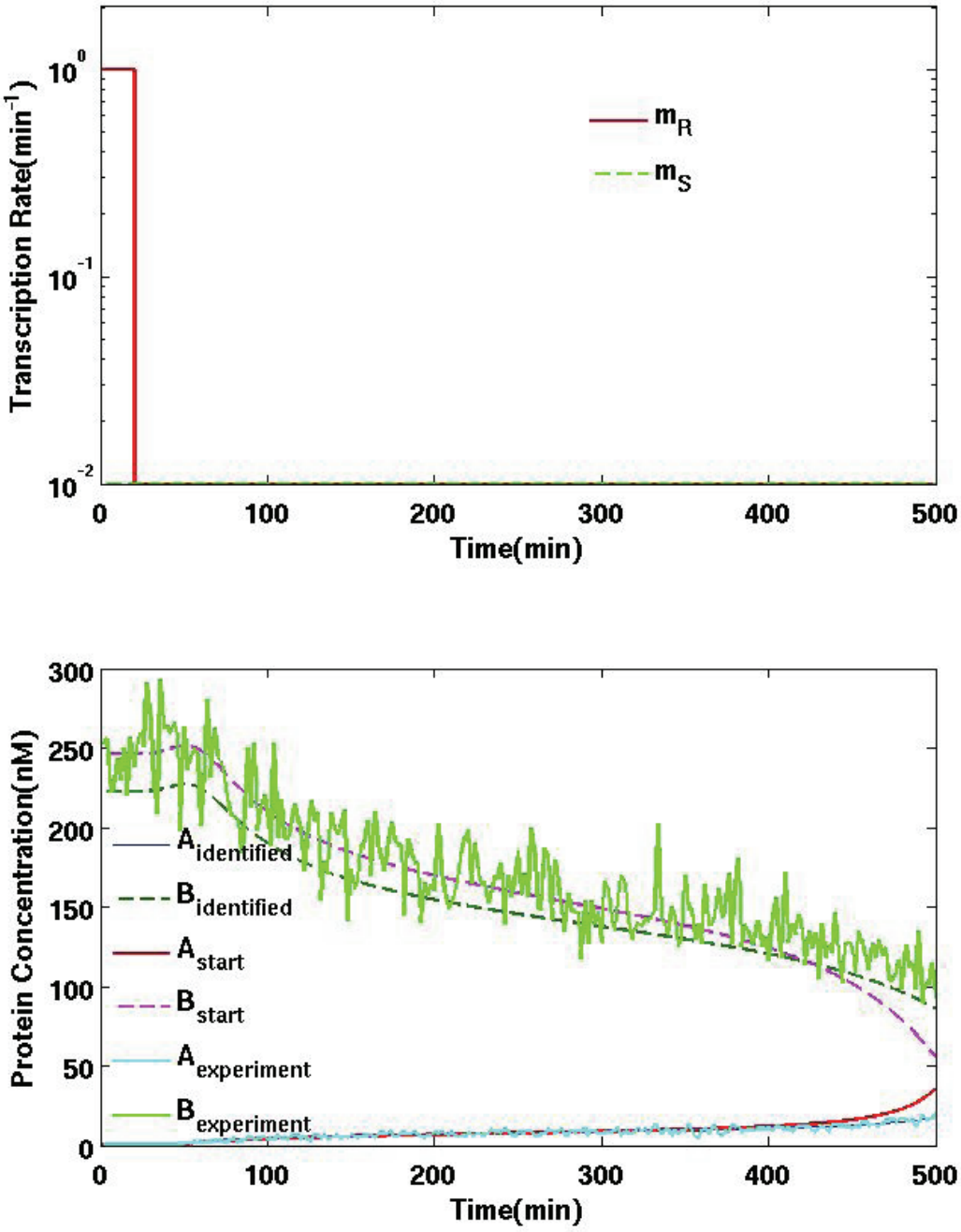

Figure 2.31: Bistability check of the mono-stable circuit.

2.7.3 Dynamic Properties of the Memory Circuit - Crossing the Stability Boundary

In our DNM definitions and dynamic stability analysis, the separatrix crossing time is of key importance. It represents the time used to flip the memory by certain excitation. In this subsection, we show two case studies of the conditional memory shown in Fig. 2.3. Both cases are based upon the same circuit structure, but cell-to-cell variations are reflected by slightly different parameter values. Here we assume that all parameters conform to normal distributions and the standard deviation of each parameter value is 5% of its nominal value.

In the plots for the two case studies, we show tangents instead of the real separatrices. Since the conditional memory circuit has twenty-two state variables, the dimensionalities of the full state space and the separatrix are twenty-two and twenty-one, respectively. Because it is difficult to visualize high-dimensional tangent hyperplanes, we only show several 2-D subspaces to illustrate the real full state spaces. These subspaces are $A_2 - B_2$ planes, $A - B$ planes and $R_2 - RS$ planes which contain the saddle point. On each of these planes, the tangent is the intersection of the 2-D plane mentioned before and the 21-D tangent hyperplane we computed.

For the first case study, all the 34 parameters are at the nominal values. Those of the second one are randomly picked according to the normal distributions mentioned before. The state spaces, tangents and trajectories of the two cases are shown in Fig. 2.32, Fig. 2.33, respectively. In each of these figures, the left, middle and right plots are for $A_2 - B_2$, $A - B$ and $R_2 - RS$ planes, respectively. The trajectory and separatrix crossing point are shown in the top row while the saddle and the tangent are shown in the bottom row.

For plots in the top row, state trajectories start from the stable equilibrium *zero*,

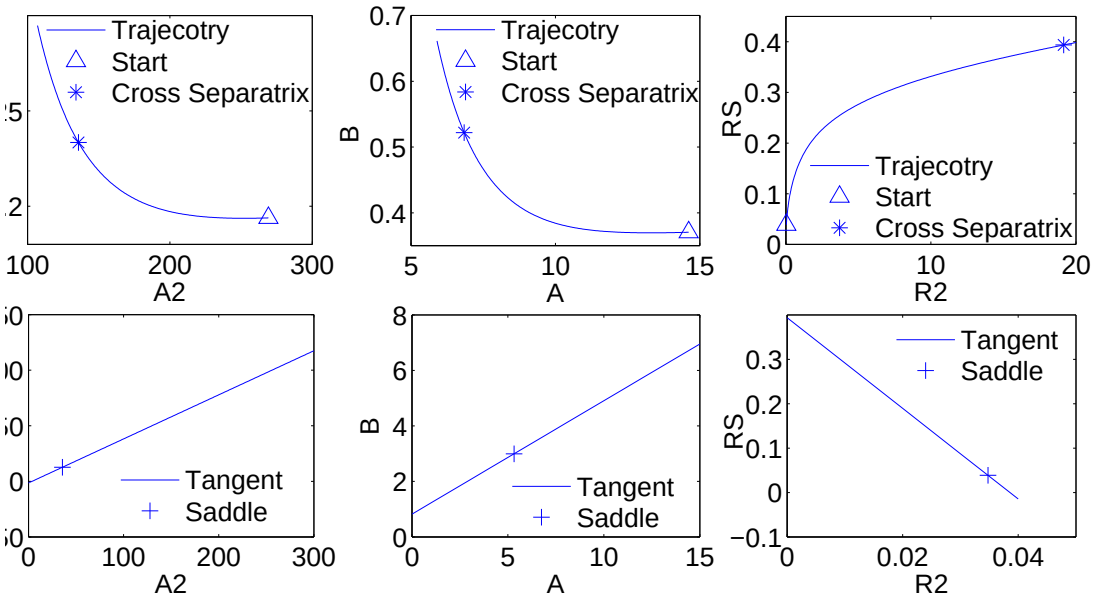

Figure 2.32: State trajectory and the tangent of the separatrix at the saddle for the case study 1. The initial state of the memory circuit is *zero*. After activating $gene_R$, the state is pushed to the stable region of equilibrium *one*. We show subspaces $A_2 - B_2$, $A - B$ and $R_2 - RS$ of the full state space in three columns. In each plot in the top row, the trajectory starts from the stable equilibrium state *zero*, crosses the separatrix and ends at the tangent. Each of the three plots shows the projections of the state trajectory and the separatrix crossing point to the corresponding subspace. In each plot in the bottom row, the straight line is the intersection of the tangent hyperplane and the 2-dimensional space including the saddle.

under the write signal shown in Fig. 2.14 (Left). The state trajectories cross the separatrices and end at the tangent crossing points. The separatrix crossing times and and tangent crossing times are shown in Table 2.2.

Comparing the results of the two cases, it is apparent that small parameter variations can significantly alter the circuit behaviors. This gives rise to the issue of parametric sensitivity.

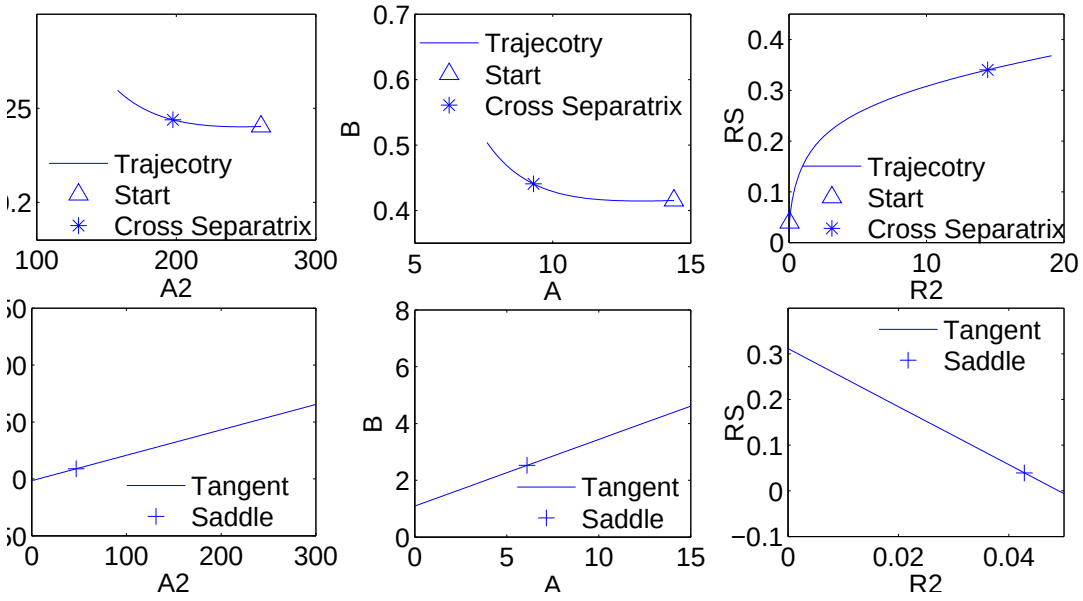

Figure 2.33: State trajectory and the tangent of the separatrix at the saddle for the case study 2. The trajectory of the system state starts from *zero*, and ends at the tangent crossing point. This trajectory as well as the sepratrix crossing point is projected to the subspace $A_2 - B_2$, $A - B$ and $R_2 - RS$ as shown in the three plots in the top row. Each subspace in plots in the bottom row contains the saddle. The straight line going through the saddle is the intersection of the corresponding subspace and the tangent of the separatrix.

Table 2.2: Crossing Times of Separatrices and Tangents.

Cases	Separatrix Crossing Time	Tangent Crossing Time
Circuit 1	16.53 min	19.08 min
Circuit 2	11.13 min	13.93 min

Separatrix/Tangent crossing times of three circuits with different parameters. Parameter values of circuit 1 are nominal values, and circuit 2 is a randomly picked case. Results of these two circuits are very different.

2.7.4 Parametric Sensitivity - Individual Differences on Dynamic Properties and Robustness

In this subsection, we more extensively study the influence of cell-to-cell variations, reflected by varying circuit model parameters. Here, we model each varying

model parameter as a Gaussian random variable with standard deviation equal to 5% of the mean. We first show two case studies of parametric sensitivities where individual differences have a significant impact on the dynamic property of systems. In each of the two case studies, we allow only one parameter to be a Gaussian random variable with all other parameters fixed to their nominal values. The varied parameters are the transcription rate of $gene_A$ and the dimerization rate of protein R for the two cases, respectively.

30 cases are computed for each case study. The results of these two case studies are shown in Fig. 2.34 and Fig. 2.35, respectively. In each figure, the left histogram is for separatrix crossing times and the right one is for tangent crossing times. For each of the four distributions the average value, standard deviation and the coefficient of variation, which is defined as the ratio of the standard deviation and the mean value, of the separatrix crossing times are listed in Table 2.3.

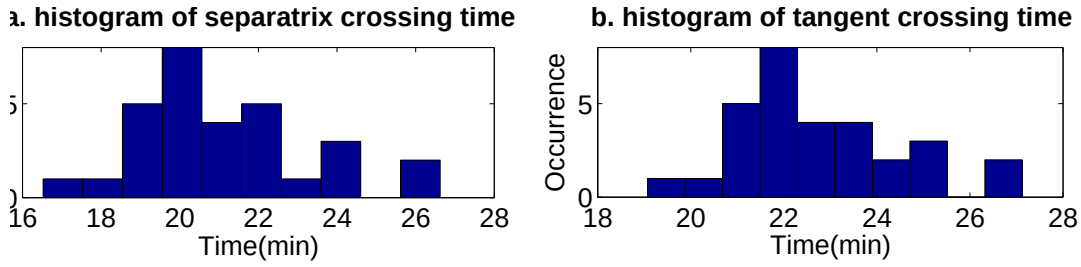

Figure 2.34: Parametric sensitivity to the transcription rate of $gene_A$. The transcription rate of $gene_A$ conforms to Gaussian distribution, of which the standard deviation is 5% of its mean value. 30 random cases are computed. The left plot is the distribution of separatrix crossing times and the right one is that of tangent crossing times.

Here we use the following metric to reflect the sensitivity of the separatrix crossing

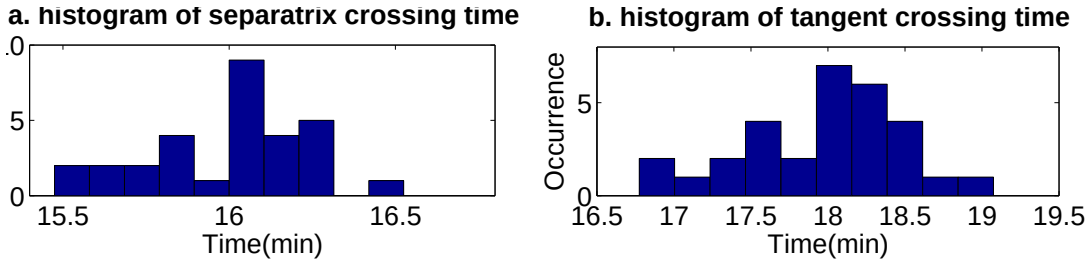

Figure 2.35: Parametric sensitivity to the dimerization rate of protein R. The dimerization rate of protein R conforms to Gaussian distribution, of which the standard deviation equals to 5% of its mean value. We computed 30 random cases. The distribution on the left is for separatrix crossing times. The one on the right is for the distribution of tangent crossing times.

Table 2.3: Mean Values and Standard Deviations of Separatrix/Tangent Crossing Times.

Random Parameter		Transcription Rate of Gene A	Dimerization Rate of Protein R
Separatrix Crossing Time	mean value	21.173 min	16.003 min
	standard deviation	2.725 min	0.2853 min
	CV	0.1287	0.01783
Tangent Crossing Time	mean value	22.813 min	17.962 min
	standard deviation	2.167 min	0.6187 min
	CV	0.09499	0.03444

Distributions of separatrix crossing times and tangent crossing times under the variation of transcription rate of $gene_A$ and dimerization rate of protein R. These two variations have different levels of influence on the dynamic properties of the system. For the same parameter variation, separatrix crossing time and tangent crossing time have different sensitivities. CV = coefficient of variation

time with respect to a parameter p

$$S_p = \frac{CV_t}{CV_p}, \tag{2.67}$$

where CV_t and CV_p stands for the coefficients of variation of the separatrix crossing

time and p, respectively.

For both of the two cases, the coefficient of variation for each parameter is 0.05. In the first case, the coefficient of variation for the separatrix crossing time is 0.1287. The parametric sensitivity is 2.574. For the second case, the coefficient of variation for the separatrix crossing time is 0.01783. The parametric sensitivity is 0.3566. It is apparent that sensitivities of these two parameters have a considerable difference.

In a similar way, we compute more random cases, and obtain the distribution of the separatrix crossing times for each types of parameters in Table 2.4. The resulting histograms are shown in Fig. 2.36. Statistical parameters of these distributions are shown in Table 2.5. For most histograms, we compute 160 random cases with varied RNA lifetimes, protein lifetimes, bonding/unbonding rates, transcription/translation rates and dimerization rates, respectively. (For histogram d and f, we compute 107 and 149 cases, respectively.) Here all random parameters conform to Gaussian distributions, and the standard deviations are 2% of their nominal values.

In view of results for all the subsets, it is clear that the system is more sensitive to reaction rates in the toggle switch, than in the front-end.

2.7.5 Fast Dynamic Noise Margin Analysis Using Separatrix Tangents

For each set of parameters considered in the previous subsection, we also compute the tangent approximations and tangent crossing times. Histograms of tangent crossing times are shown in Fig. 2.37. Shapes of most distributions are similar to those of the separatrix crossing times.

In almost every case, a memory circuit has different separatrix crossing time and tangent crossing time. The relative difference, which is defined as the ratio of the difference and the corresponding separatrix crossing time, also varies greatly for different sets of parameters. The average relative difference computed for all cases is

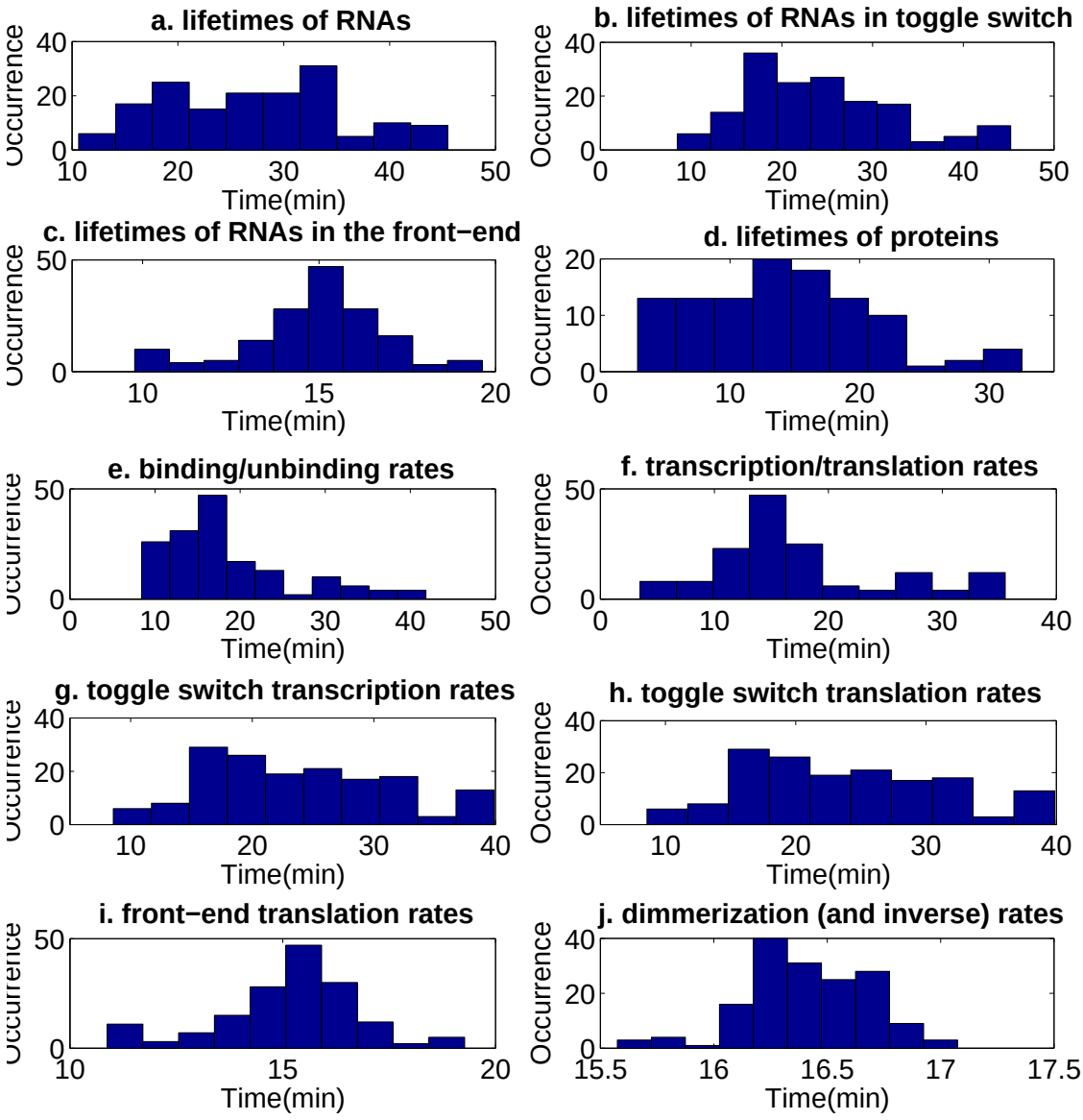

Figure 2.36: Parametric sensitivities of the separatrix crossing time. Each plot is a distribution of separatrix crossing times under parameter variations. The standard deviations of the studied parameters are 2% of their nominal values. We computed 160 cases for most of these plots except that 107 and 149 cases are computed for histograms d and f, respectively. For lifetimes of RNAs, and transcription/translation rates, the variances of the distributions are large. Therefore, we also studied these parameters of the toggle switch and the front-end separately. The comparison between plots b and c, and that between h and i show that the separatrix crossing time is more sensitive to parameters of the toggle switch than those of the front-end. Interestingly, sensitivities of the separatrix crossing time to variations of dimerization rates are much smaller than those of other parameters.

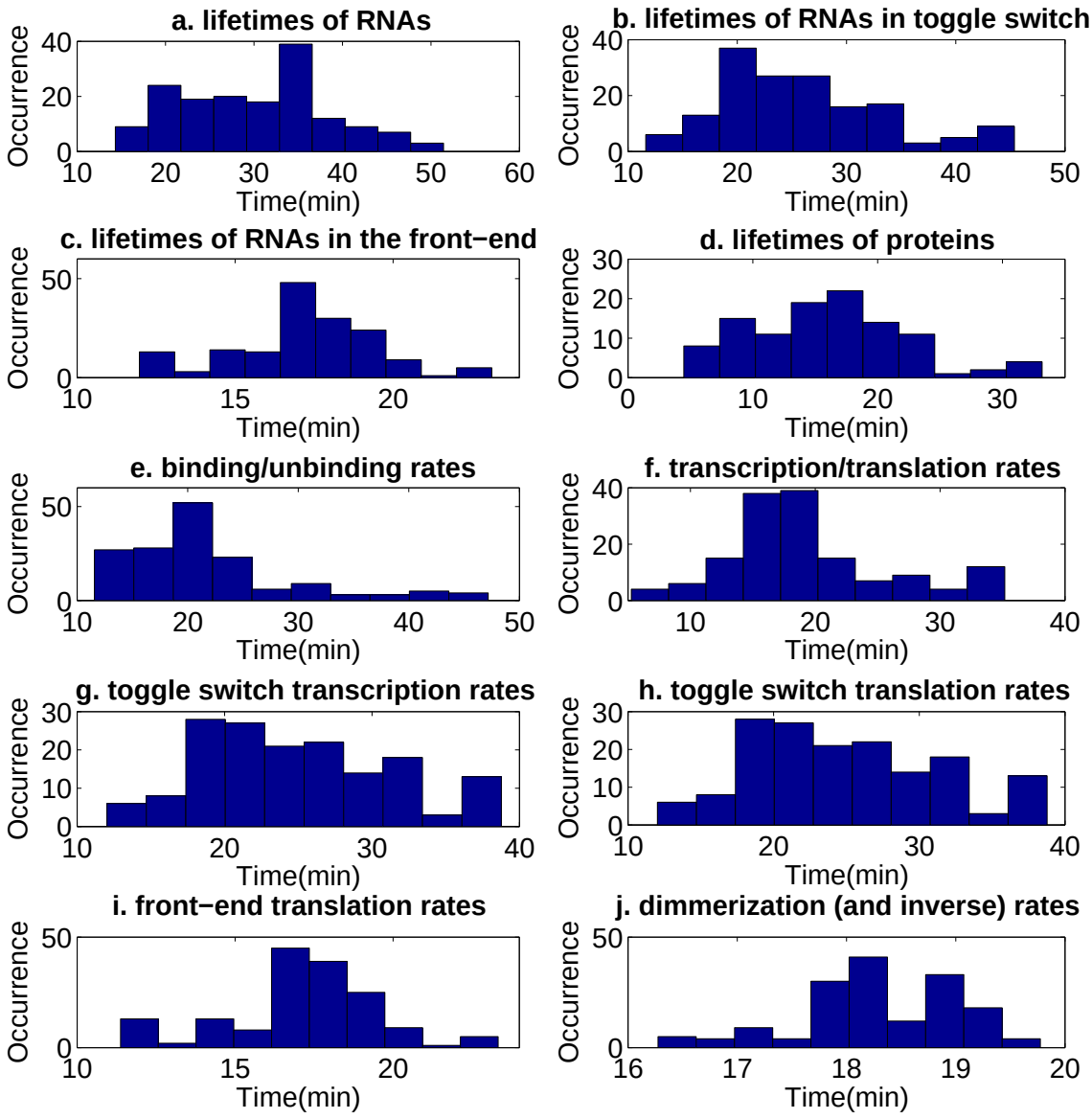

Figure 2.37: Parametric sensitivities of the tangent crossing time. Distributions of tangent crossing times under variations of different parameters. 160 cases are computed for each histogram except for d and f, for which we compute 107 and 149 cases, respectively. Since the variances of tangent crossing time distributions are large for variations of protein/RNA lifetimes and transcription/translation rates, we split these parameters into smaller groups and study them in histogram b, c, g, h and i.

Table 2.4: Different Types of Parameters.

Figures	Number of Parameters	Parameters
a	4	Lifetimes of RNA_A, RNA_B, RNA_R and RNA_S
b	2	Lifetimes of RNA_A and RNA_B
c	2	Lifetimes of RNA_R and RNA_S
d	8	Lifetimes of protein A, B, R, S, A_2, B_2, R_2 and RS
e	8	Binding and unbinding rate constants for A_2, B_2, R_2 and RS
f	6	Translation rate constants of RNA_A, RNA_B, RNA_R and RNA_S Transcription rate constants of $gene_A$ and $gene_B$
g	2	Transcription rate constants of $gene_A$ and $gene_B$
h	2	Translation rate constants of RNA_A and RNA_B
i	2	Translation rate constants of RNA_R and RNA_S
j	8	Dimerization and inverse dimerization rates constant for A_2, B_2, R_2 and RS

Different types of parameters studied in Fig. 2.36 and Fig. 2.37. All 34 parameters are split into 5 groups and studied in panels a, d, e, f and j, respectively. Parameters in the group studied in panel a are further divided into two subgroups and studied in panels b and c. Similarly, the group studied in panel f is split into three subgroups studied in g, h and i.

13.36%. For different sets of parameters, the smallest average relative difference is 7.73%, which is for variations of transcription/translation rates in the toggle switch.

For the bistability analysis of the conditional memory, the tangent approximation is always faster than the exact algorithm. Under variations of transcription rates and translation rates, we conduct experiments on a shared memory Linux server with two quad-core Intel Xeon processors with 2.33GHz clock speed and 8GB memory. On average, computations of the tangent crossing time and separatrix crossing time in the twenty-two-dimensional state space for a single case take 10.2 seconds and 27.0 seconds, respectively. The speedup is 2.65×. In studies of statistical characteristics

Table 2.5: Statistical Parameters of Separatrix Crossing Time.

Figures	Mean Value (min)	Standard Deviation (min)
a	27.324	8.436
b	24.139	8.432
c	14.924	1.990
d	14.105	6.752
e	18.612	7.704
f	17.434	7.591
g	23.723	7.591
h	23.723	7.591
i	15.161	1.689
j	16.428	0.268

Statistical parameters of each distribution in Fig. 2.36. The mean values and standard deviations of each histogram are shown in the second and third column of each row.

such as parametric sensitivities in the previous subsection where thousands of cases have to be computed, the fast DNM analysis saves time considerably.

2.8 Discussion

2.8.1 Dynamic Stability Analysis of General Multi-Stable Systems

For a multistable genetic regulatory network, a stable equilibrium represents a gene expression profile, of which each gene has a stable expression level. Besides the genetic conditional memory we discussed in the previous sections, multistability is also a common attribute for other biological systems and has caught a lot of attention [79–83]. In these previous works, transitions between stable equilibria in multistable systems are topics of interest. In these systems, designed inductions force the expressions of certain genes. While the detailed working mechanisms of many of these systems are not fully understood, the use of partially complete dynamic models provides valuable insights into the dynamic characteristics of the system. With

more biological details becoming available, refined models will provide increasing understanding of these systems. It is worthwhile to note that while these systems may differ significantly in terms of biochemical characteristics and time scales, from a dynamic system perspective, they all share a common attribute: multistability. It is possible to extend the presented work to study the dynamic characteristics of such systems, when appropriate, to aid the system design.

In multistable systems, there might be multiple saddles. Under this circumstance, there is a tangent hyperplane associated with each saddle. By applying our approach, each tangent hyperplane computed approximates one part of the separatrix in the state space of the system. After obtaining all tangent hyperplane of all saddles, stability boundaries of the state space are approximated. In addition, this may provide insights into the stable region of each stable equilibrium.

2.8.2 Parametric Sensitivity

From Fig. 2.36 and Table 2.5, it is apparent that the sensitivities of separatrix crossing time with respect to different sets of parameters vary a lot. Fig. 2.36.b and Fig. 2.36.c show that the sensitivities to the RNA lifetimes in the toggle switch part are greater than those in the front-end. This is also true for the translation rates as revealed in Fig. 2.36.h and Fig. 2.36.i. This phenomenon is probably caused by different ways in which the toggle switch and the front-end influence gene expressions in the network. In both components, the RNA lifetimes and translation rates directly control the quantities of proteins that are produced. However, there exist structural differences between the toggle switch and the front-end. In the front-end, each protein has only one binding site on its target gene, while in the toggle switch part, each gene has two binding sites. Furthermore, the toggle switch itself is a positive feedback loop which may amplify the the effect of binding. Such structural differences may

contribute to the observed parameter sensitivity differences.

In a nonlinear dynamic system, various parametric variations may interact. Therefore, they may alter statistical characteristics of the interested performance other than the standard deviation. For example, Fig. 2.36.a and Fig. 2.36.b show the histograms of the separatrix crossing times with the variations of all RNA lifetimes in the network, and only with those of the toggle switch part, respectively. It is clearly seen that the two cases produce close variances of the separtrix crossing time, although the former includes a larger number of parametric variations. However, a closer inspection reveals that the former case does introduce a larger mean shift, which is likely to see in a nonlinear network.

It is instrumental to understand the parametric sensitivities of key system performances with respect to main variational sources in the network. The knowledge of parametric sensitivities helps to identify dominant sources of variability and provides guidance in robust system design.

2.9 Summary

In this section, we have presented a simulation environment for biological genetic memory networks. The presented electrical-equivalent modeling allows the extension of an electrical circuit simulator for biological applications. The proposed Bayesian based parameter identification is shown to be able to correctly and accurately identify the system model under simulated noisy measurements. The dynamic stability of a genetic conditional memory circuit is quantitatively characterized by the new dynamic noise margins which capture both the amplitudes and durations of noise perturbations. Based on rigorous system theory and the concept of stability boundary (separatrix), we have developed an exact algorithm for the computation of dynamic noise margins. A faster computation based on tangent approximation of the separa-

trix is also presented. Parametric analysis show that due to differences in structure, time scales and nonlinear interactions between multiple reactions, the sensitivities of the dynamic stability of the memory circuit to different biochemical reactions in the network varied significantly. The efficiency of the proposed algorithms is instrumental in analyzing dynamic stability. With proper extensions, presented techniques are broadly applicable to other multi-stable biological systems.

3. MODELING AND BIOPHYSICALLY BASED SIMULATION STUDY OF THE BRAIN[1]

The previous section shows the application of mathematical modeling and computer simulation to the study of relatively small genetic regulatory memory circuit. However, for complicated biological systems, an additional challenge is that large models and large amount of computations may be required to capture their behaviors and functions. In this section, to facilitate the study of brain behaviors, we build large-scale brain models with detailed cellular mechanisms to capture physiological and pathological behaviors of the brain and develop dedicated numerical techniques to tackle the computational challenge.

To understand brain behaviors, it is important to directly associate the network level activities to the underlying biophysical mechanisms, which requires large-scale simulations with biophysically realistic neural models like Hodgkin-Huxley models. However, when simulations are conducted on models with sufficient biophysical details, great challenges arise from limited computer power, thereby restricting most existing computational works with biophysical models only to small-scale networks. On the other hand, with the emergence of powerful computing platforms, many recent works are geared to performing large-scale simulations with simple spiking models. However, the applicability of those works is limited by the nature of the underlying phenomenological model. To bridge the gap, an intermediate step is taken to construct a scalable brain model with sufficient biophysical details. In this work, great efforts are devoted to taking into account not only local cortical microcircuits but

[1]Reprinted from Y. Zhang, B. Yan, M. Wang, J. Hu, and P. Li. Linking brain behavior to underlying cellular mechanisms via large-scale brain modeling and simulation. Neurocomputing, 97:317-331, 2012, Copyright (2012), with permission from Elsevier.

also the global brain architecture, and efficient techniques are proposed and adopted to address the associated computational challenges in simulation of networks of such complexity. With the customized simulator developed, we are able to simulate the brain model to generate not only sleep spindle and delta waves but also the spike-and-wave pattern of absence seizures, and directly link those behaviors to underlying biophysical mechanism. Those initial results are interesting because they show the possibility to determine underlying causes of diseases by simulating the biologically realistic brain model. With further development, the work is geared to assisting the clinicians in selecting the optimal treatment on an individual basis in the future.

3.1 Models

Modeling and simulation are two different aspects. In this section, we present techniques for modeling brain networks. We focus on the construction of the large-scale biophysically realistic brain model. We limit our scope to thalamocortical oscillations to study the mechanisms underlying the transition between physiological and pathological oscillations. The various oscillatory rhythms generated in the thalamocortical system are mediated by intrinsic mechanisms (ion channels), which depend on the interplay between specific intrinsic currents, and network mechanisms (synaptic receptors, axon delays, etc), which require the interaction of excitatory and inhibitory neurons within a population [84].

In order to reproduce typical physiological and pathological oscillations, a large number of mechanisms suggested by studies *in vivo*, *in vitro*, and *in silico*, have to be taken into consideration and integrated together in a consistent way [85]. In the following part, we describe the construction of the model and briefly mention the relevance of the model parameters to the biological phenomena of interest.

3.1.1 Models of Neurons and Underlying Biophysical Mechanisms

Based on morphology and functionality, the neurons in our model are classified into twenty-two basic types as shown in Table 3.1 [41]. As dendrites of some neurons stretch for long distance, a multi-compartment neural model can span several cortical layers. Since the functionality of axons is to transmit signals, we only consider their delays in the model.

Table 3.1: Twenty-Two Basic Neuron Types.

Location	Neuron Type	Excitability	Description
Cortex	p2/3	Excitatory	Pyramidal in L2/3
	ss4(L2/3)	Excitatory	Spiny stellate in L4 (project to L2/3)
	ss4(L4)	Excitatory	Spiny stellate in L4
	p4	Excitatory	Pyramidal in L4 (project to L4)
	p5(L2/3)	Excitatory	Pyramidal in L5 (project to L2/3)
	p5(L5/6)	Excitatory	Pyramidal in L5 (project to L5/6)
	p6(L4)	Excitatory	Pyramidal in L6 (project to L4)
	p6(L5/6)	Excitatory	Pyramidal in L6 (project to L5/6)
	b2/3,b4,b5,b6	Inhibitory	Basket interneurons in L2/3/4/5
	nb1,nb2/3,nb4,nb5,nb6	Inhibitory	Non-basket interneurons in all layers
Thalamus	TC_s/TC_n	Excitatory	Thalamocortical relay cells in specific/nonspecific nucleus
	TI_s/TI_n	Inhibitory	Thalamocortical interneurons in specific/nonspecific nucleus
	RE	Inhibitory	Thalamic reticular cells

3.1.1.1 Compartment Model

For each of the compartments, the Hodgkin-Huxley model is used to describe its dynamics

$$C\frac{dV}{dt} = -g_{leak}(V - E_{leak}) - \sum_i I^i_{int} - \sum_j I^j_{syn} - \sum_k I^k_{cc}, \tag{3.1}$$

where C is the specific capacitance of the neuron membrane, V is the membrane potential, g_{leak} is the conductance for the leakage current, E_{leak} is the reversal potential for the leakage current, $\sum_i I^i_{int}$ is the sum of intrinsic ion currents, $\sum_j I^j_{syn}$ is the sum of synaptic currents, and $\sum_k I^k_{cc}$ is the sum of currents flowing into or out of the neighboring compartments when there are differences in potentials.

3.1.1.2 Models of Ion Channels

Intrinsic mechanisms play important roles in mediating thalamocortical oscillations. Certain physiological oscillation patterns can be generated by intrinsic mechanisms alone. For example, thalamic delta ($1 - 4$ Hz) is a well known example of rhythmic activity generated intrinsically by thalamic relay neurons [86]. When combined with network mechanisms, more physiological oscillation patterns can be generated as discussed in the next section.

Intrinsic currents I_{int} of an ion channels are modeled by kinetic models of the Hodgkin and Huxley type [31] described by the following equations

$$\begin{aligned} I_{int} &= \bar{g}_{int} m^N h^M (V - E_{int}) \\ \dot{m} &= \alpha_m (1 - m) - \beta_m m \\ \dot{h} &= \alpha_h (1 - h) - \beta_h h, \end{aligned} \tag{3.2}$$

where $\bar{g}_{int}$ is the maximal conductance, and E_{int} is the reversal potential. The gating properties of the current are dependent on N activation gates and M inactivation gates, with m and h representing the fraction of gates in open form and with respective rate constants α_m, β_m, α_h, and β_h. Rate constants are dependent on either membrane voltage or intracellular calcium concentration.

Some representative ion channels in our model are introduced as follows. Na^+ and K^+ currents contribute to action potentials and are included in all the cell

models [44]. Interneuron cells contain only Na^+ and K^+ currents and produce "fast-spiking" [43] firing. For the pyramidal cells, one additional slow voltage-dependent K^+ current (I_M) [46] producing "regular-spiking" pattern characterized by adaptation is modeled [43]. There are T-currents in the model of thalamic cells such that bursts of action potentials can be produced. The T-current in reticular cells I_{Ts} are of slow kinetics, which is given in [45, 48]. The T-current in thalamocortical cells I_T is modeled by kinetics similar to the model of [45] with activation considered at steady state and inactivation described by a first order equation [47]. Thalamus plays an important role as it can spontaneously oscillate at low frequencies (<4 Hz) due to the post-inhibitory rebound bursting property of the T-current in thalamocortical cells. In addition to I_T, thalamocortical cells also include leak potassium current I_{KL} and hyperpolarization-activated inward current I_h [47]. We only consider the voltage dependence of I_h and do not include the upregulation of I_h by intracellular Ca^{2+} which leads to wax-and-wanning properties. The electrical parameters of neurons of different types are summarized in Table 3.2.

Table 3.2: Electrical parameters of neurons of different types.

Neuron Type	$C(nF)$	$g_L(S)$	$g_{Na}(S)$	$g_K(S)$	$g_M(S)$	$g_T(S)$	$g_{Ts}(S)$	$g_h(S)$	$g_{KL}(S)$
p2/3,p4,p5,p6,ss4	0.12	$2.43e^{-9}$	$6.63e^{-6}$	$0.71e^{-6}$	$8.88e^{-9}$	0	0	0	0
b2/3,b4,b5,b6	0.1	$3.87e^{-9}$	$5.9e^{-6}$	$0.4e^{-6}$	$8e^{-9}$	0	0	0	0
nb1	0.05	$0.94e^{-9}$	$2.5e^{-6}$	$0.2e^{-6}$	$1.4e^{-9}$	$0.2e^{-7}$	0	0	0
nb2/3,nb4,nb5,nb6	0.25	$4.7e^{-9}$	$12.5e^{-6}$	$1e^{-6}$	$7e^{-9}$	$1e^{-7}$	0	0	0
TC_s/TC_n	0.29	$2.9e^{-9}$	$2.61e^{-5}$	$0.29e^{-5}$	0	$5.8e^{-7}$	0	$5.8e^{-9}$	$3e^{-9}$
TI_s/TI_n	0.029	$2.9e^{-10}$	$2.61e^{-6}$	$0.29e^{-6}$	0	$5.8e^{-8}$	0	$5.8e^{-10}$	$3e^{-10}$
TRN	0.14	$7e^{-9}$	$2.8e^{-5}$	$0.28e^{-5}$	0	0	$4.2e^{-7}$	0	0

A wide variety of ion channels are believed to play important roles in the gen-

eration of seizures, and some of them are important targets for anti-epileptic drug design [85].

3.1.1.3 Models of Synaptic Receptors

In addition to intrinsic mechanism, various oscillatory rhythms generated in the thalamocortical system are also mediated by network mechanisms, which require the interaction of excitatory and inhibitory neurons mediated by various synaptic receptors. For example, spindle oscillations [87–89] depend on the interaction between thalamic relay and reticular neurons as well as on their intrinsic properties. Network mechanisms also have direct influence on the generation of pathological oscillations. For example, when a shift from dominant inhibition to dominant excitation in a neuronal network occurs, the network tends to transit from physiological oscillations to seizure-like oscillations.

Therefore, detailed modeling of those synaptic receptors are extremely important to investigate the role of biophysical and molecular properties of neurons in causing brain disorders and thus provide valuable insight on therapeutic intervention [85].

As shown in Fig. 3.1, in the Hodgkin-Huxley type model of a compartment, the total synaptic currents is the summation of currents flowing through all receptors,

$$\begin{aligned}\sum_j I_{syn}^j = & \sum_{i=1}^{n_{AMPA}} g_{AMPA_i} \cdot (V - E_{AMPA}) \\ & + \sum_{i=1}^{n_{NMDA}} g_{NMDA_i} \cdot \frac{(V+80)^2}{(V+80)^2 + 60^2} \cdot (V - E_{NMDA}) \\ & + \sum_{i=1}^{n_{GABA_A}} g_{GABA_{Ai}} \cdot (V - E_{GABA_A}) \\ & + \sum_{i=1}^{n_{GABA_B}} g_{GABA_{Bi}} \cdot (V - E_{GABA_B}),\end{aligned} \tag{3.3}$$

where n_{AMPA}, n_{NMDA}, n_{GABA_A} and n_{GABA_B} are the number of AMPA, NMDA, GABA_A and GABA_B receptors on the compartment, and E_{AMPA}, E_{NMDA}, E_{GABA_A}

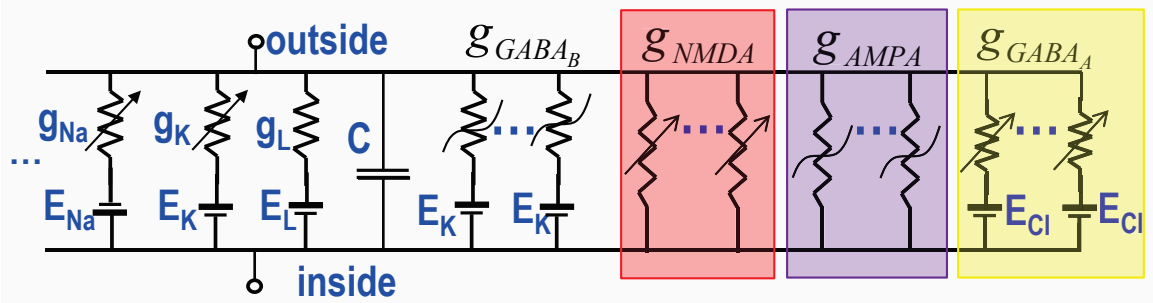

Figure 3.1: Hodgkin-Huxley type model of a compartment. In the model, g_{Na}, g_K are conductances and E_{Na}, E_K are reversal potentials of Na^+ and K^+ ion channels, respectively; g_L and E_L are leakage conductance and reversal potential; C is the membrane capacitance; g_{GABA_A}, g_{GABA_B} and E_{Cl}, E_K are conductances and reversal potentials of inhibitory synaptic receptors GABA_A and GABA_B, respectively; g_{AMPA}, g_{NMDA} are the conductances of excitatory synaptic receptors AMPA and NMDA and the corresponding reversal potentials of both are $0V$.

and E_{GABA_B} are the corresponding reversal potentials, respectively. Typically, $E_{AMPA} = 0mV$, $E_{NMDA} = 0mV$, $E_{GABA_A} = -70mV$ and $E_{GABA_B} = -95mV$.

The dynamic response of g_{AMPA}, g_{NMDA} and g_{GABA_A} to an input arriving at time $t = t_j$ can be obtained by solving the following differential equation [90]

$$\frac{d^2 g_j(t)}{dt^2} + \frac{2}{\tau} \cdot \frac{dg_j(t)}{dt} + \frac{1}{\tau^2} \cdot g_j(t) = 0, \tag{3.4}$$

with initial conditions

$$\begin{aligned} g_j(t)|_{t=0} &= 0 \\ \tfrac{dg_j(t)}{dt}|_{t=0} &= \tfrac{a_j e}{\tau}, \end{aligned} \tag{3.5}$$

where a_j denotes the peak amplitude, τ denotes the time-to-peak value, and e is the base of the natural logarithms. Note that, by exploring linearity, the responses of all the inputs applied to the synaptic receptors of the same type in the same compartment can be merged to significantly reduce the cost of computation [90].

Due to the nonlinear property, the models of $GABA_B$ receptors are described by the following different equations [91]

$$\begin{aligned} I_{GBAB_B} &= \bar{g}_{GBAB_B} \frac{s^n + K_D}{s^n} (V - E_{GABA_B}) \\ \dot{r} &= K_1[T](1-r) - K_2 r \\ \dot{s} &= K_3 r - K_4 s, \end{aligned} \tag{3.6}$$

where $[T]$ is the GABA concentration in the synaptic cleft, r is the fraction of $GABA_B$ receptors in the activated form, s is the normalized G-protein concentration in activated form, $\bar{g}_{GABA_B}$ is the maximal postsynaptic conductance of K^+ channels, K_D is the dissociation constant of G-protein binding on K^+ channels, V is the postsynaptic membrane potential, and E_{GABA_B} is the reversal potential. The parameters are as follows: $K_D = 100$, $K_1 = 9 \times 10^4 M^{-1} s^{-1}$, $K_2 = 1.2 s^{-1}$, $K_3 = 180 s^{-1}$, and $K_4 = 34 s^{-1}$, with $n = 4$ binding sites. As shown in the studies[53], the nonlinearity of $GBAB_B$ receptors are playing an important roles in seizure generation.

Synaptic receptors play complex roles in the generation of epileptic activity. For example, although both $GABA_A$ and $GABA_B$ are inhibitory, while $GABA_A$ receptor is generally believed to inhibit seizure activity, $GABA_B$ receptor has been shown to induce absence seizure like activity [92–95]. As a result, while many anti-absence drug (clobazam, clonazepam, pheobarbital, primidone, etc.) are designed as $GABA_A$ agonists to inhibit seizures, $GABA_B$ antagonists hold the promise as anticonvulsants for absence seizures.

3.1.2 Models of Local Cortical Circuitry and Global Connections

Most existing simulations works based on biophysically realistic models are only limited to local cortical circuitry and do not take into consideration the global structure of the brain, which includes multiple cortical regions connected by the white

matter pathways. However, those ignored elements can play important roles in generating both physiological and pathological behaviors of the brain.

Normal brain function requires the dynamic interaction of functionally specialized but widely distributed cortical regions. Long-range synchronization of oscillatory signals has been suggested to mediate these interactions within large-scale cortical networks, and thus dynamically establish such task-dependent networks of cortical regions [96]. Disturbances of such synchronized networks have been implicated in several brain disorders, such as schizophrenia, autism, epilepsy, Alzheimer's disease, and Parkinson's disease [97]. For example, in terms of absence seizure, recent research has suggested that cortical local and long-range synchronization interplay plays an important role in human absence seizure initiation [98].

To take into consideration all the relevant factors, the global structure of the brain is explicitly modeled in our brain model, which includes multiple cortical regions and thalamic nuclei. As illustrated by Fig. 3.2(Top), a total number of seventy regions are being modeled in this model, and each region includes a cortical part and a thalamic part. The global connections are derived from a macroscopic cortico-cortical connectivity network derived from a diffusion-magnetic resonance imaging (MRI) data set[42]. The relative connectivity between all cortical region pairs is quantified by an structural adjacency matrix (SAM). Based on the distance between regions and the signal speed of 1 m/s for myelinated fibers in whiter matter, we calculate the delays of global connections.

The local cortical circuitry in each region is based on the detailed reconstruction studies of cat area 17 (visual cortex) [40]. To construct the model, it is important to estimate the relative distribution of neuronal types, the relative distribution of synapses, and the typical axonal arborizations of various neuronal types. The relative distribution of neuronal types in the model is based on the published data in [37, 38,

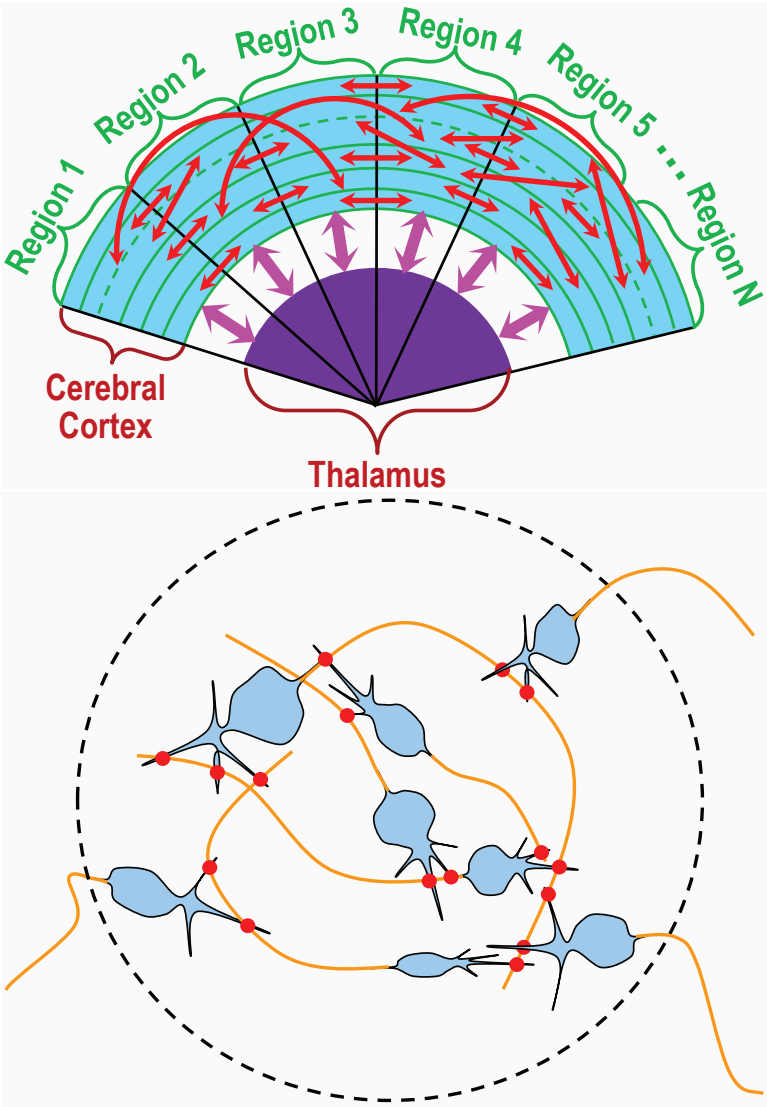

Figure 3.2: (Top) Illustration of the entire brain model. The entire model is divided into many regions. (Bottom) Illustration of a region in the brain model. In a brain region, dots on the contact of axons and dendrites represent local connections. Axons across the region boundary (dashed line) may form global connections with dendrites of neurons in other regions.

40], the relative distribution of synapses in the cortex is based on the data in [40], the relative distribution of synapses in the thalamus is based on the data in [39], and the typical axonal arborizations of various neuronal types are based on the data

in [35, 36]. The above mentioned data were summarized as a distribution matrix, and applied to build phenomenological brain models [41].

3.2 Efficient Network Simulation Techniques

As we mentioned before, modeling and simulation are two different aspects. In this section, we present simulation. To study the physiological and pathological behaviors of the brain, we have constructed a brain model with sufficient biophysical details as described in the previous section. However, significant challenges exist when simulating networks of such complexity. To tackle computational challenges, a customized simulator has been built. In this section, we first give a brief overview of some important aspects of the parallel simulator, and then focus on an advanced numerical integration technique proposed for neural network simulation.

3.2.1 Overview of the Parallel Simulator

In order to address the computational challenge, efficient techniques have been proposed and adopted in our simulator from different angles including locality enhancement, telescopic projective integration, linear receptor merging, and parallel computing with dynamic load balance.

3.2.1.1 Efficient Simulation by Locality Enhancement

The time scales of the model behavior in the network level are quite different. While the time granularity for simulating the neuronal behavior has to be as small as 0.01 ms, the axon delay ranges from milliseconds to dozens of milliseconds. This prompts us to adopt two different time steps for the whole network simulation. A similar ideal has been suggested in [99].

If we use ΔT_{net} and ΔT_{cell} as the macro-step and micro-step, respectively ($\Delta T_{net} =$ 0.01 ms and $\Delta T_{cell} = 1$ ms in our simulation), during each macro-step, one neuron

can be continuously simulated using multiple micro-steps without considering the activities of others. As cache locality is enhanced by continuous simulating one neuron for a longer period of time, accesses to global memory can be significantly reduced and thus the efficiency of simulation can be increased. To do this without sacrificing the accuracy, all the inputs to the neuron being simulated in the period of one macro-step have to be known *a priori* when the simulation of this macro-step starts. As each macro-step is smaller or equal to the axonal delay by definition, the requirement is satisfied by the fact that all the inputs to the neuron being simulated in the period of one macro-step are caused by the firings of other neurons in the previous macro-steps. Note that the locality enhancement does not change the brain model and simulation accuracy. Thus, the simulated brain behaviors are not affected.

3.2.1.2 Telescopic Projective Integration Method

While locality enhancement method is leveraged to speedup simulation in a network-level, in each macrostep, all the neurons have to be simulated with some numerical integration method for a large number of microsteps. To further mitigate the computational cost in the cell-level, a telescopic projective based explicit integration technique is proposed to boost efficiency. The method will be described in detail in Section 3.2.2.

3.2.1.3 Efficient Modeling by Merging Linear Receptor Models

For the biologically realistic model, as the number of synapses per compartment is much larger than the number of ion channels, the majority of these differential equations are for the modeling of synaptic receptors, which accounts for most of the computational effort in simulation. While the number of synaptic receptors is large, as discussed before, most of the synaptic receptors (AMPA, $GABA_A$, NMDA) are typically described by simple linear model. By exploring linearity of synaptic

receptors, the responses of all the inputs applied to the synaptic receptors of the same type on a given compartment are merged mathematically in our implementation, which significantly reduce the cost of computation without sacrificing accuracy [90].

3.2.1.4 Parallel Implementation with Dynamic Load Balancing

The brain simulator is implemented on a 24-core PowerEdge R715 machine with 2 AMD Operton 2.2GHz 12-core processors and 32GB RAM. Multi-thread parallelization is adopted in our simulation, which allows a maximum of 24 threads to work simultaneously. Since the simulation of the network has to be synchronized for each ΔT_{net} time step, we conduct the parallelization for each ΔT_{net} of simulation time. To dynamically adjust the load of each thread based on the load distribution, we adopt the dimension exchange method [100–102] with neighborhood averaging scheme for load balancing [103].

3.2.2 Telescopic Projective Integration Method

In this section, we first briefly compare explicit and implicit integration techniques when applied to neuronal network simulation. To relax the stability constraint of the former, we propose a telescopic projective integration framework, and a stability analysis is given to demonstrate the efficacy. Finally, on top of enhanced stability, we present an on-the-fly error control mechanism to meet the accuracy requirement of network simulation.

3.2.2.1 Explicit and Implicit Integrations

Implicit integration methods like backward Euler and Crank-Nicholson method are adopted by most existing simulation tools [49, 50] due to their good stability properties. On the other hand, from an efficiency point of view, explicit integration methods are very appealing for simulating large-scale brain models. Their application

relaxes the needs for solving any coupled nonlinear system of equations using an iterative method such as the Newton method, and hence the underlying matrix problems as well, whose cost grow superlinearly with the problem size. One of the key limitations of standard explicit methods (e.g. forward Euler) is the key stability property. A small enough time step size must be chosen in order to sufficiently damp the fast error components of the system so as to ensure the stability. In this work, an advanced multi-level explicit integration method with enhanced stability is used.

3.2.2.2 Projective Integration

To boost the step size, we adopt and develop more advanced stable numerical integration methods based on the recently developed telescopic projective integration framework[104]. Often, the long term transient responses of a system are mainly determined by slow components (corresponding to large time constants) in the network. Fast components only exist for a short period of time and dissipate quickly. Since it is often sufficient to only track the slow components in the transient responses, it is desirable to use a time step that is comparable to large time constants. In this regard, the problem with a standard explicit integration method (e.g. forward Euler) is that the step size has to be comparable to the smallest time constant to maintain stability, a significant constraint on efficiency. This problem can be alleviated by adopting the projective integration method shown in Fig. 3.3(Top)

Intuitively, to ensure stability, n+1 explicit integration (e.g. forward Euler) steps are taken at the inner loop to integrate the system from, say from time t_0 to time t_{n+1}. Here, a small step size is used to sufficiently damp fast transient responses. Then, a large projective or extrapolation step with a step size commensurate with the slow time constants is taken to project into the forward direction of time to derive

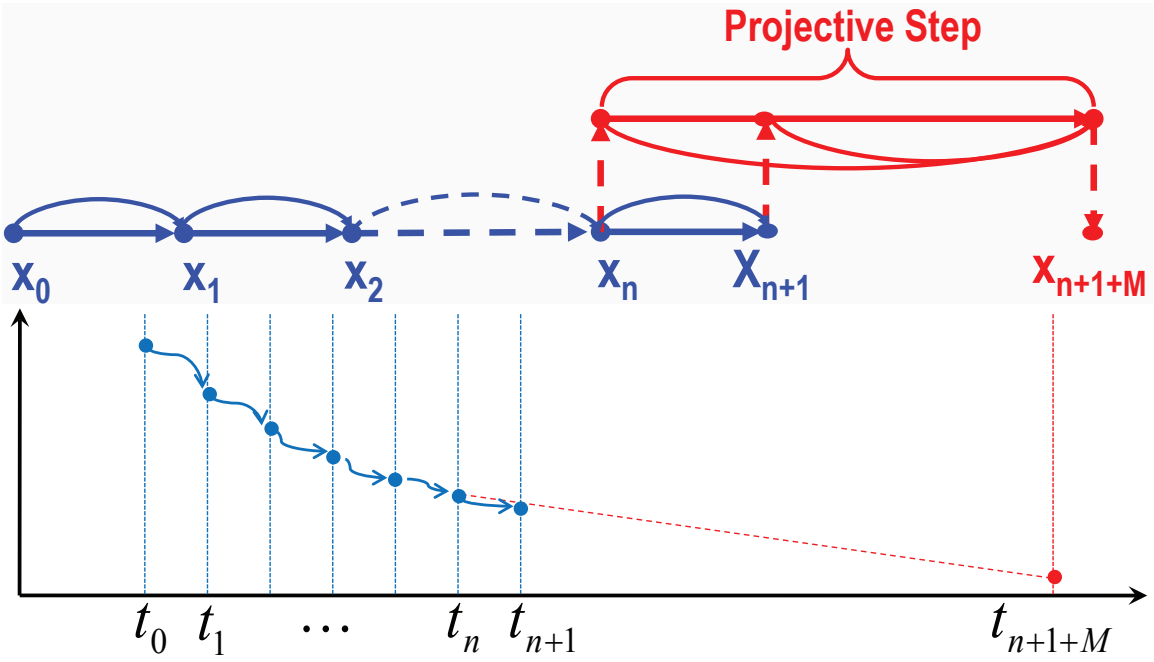

Figure 3.3: (Top) Illustration of projective integration method. There are $n+1$ small steps of forward Euler integration at the beginning followed by a projective integration step. (Bottom) An example of projective integration method in (a), which shows how projective integration method works for the simulation of a curve decreasing exponentially with time.

the response at time t_{n+1+M} (M is desired to be large)

$$x_{n+1+M} = (M+1)x_{n+1} - Mx_n \tag{3.7}$$

where x_{n+1} and x_n are the solutions at the last two time points computed at the inner loop.

From a stability point of view, the sequence of n+1 integration steps with a small step size exponentially (in n) damps the numerical integration error, while the potential error amplification incurred by the large projective step is only linear in step size. This makes it possible to maintain a large M without sacrificing stability.

Hence, the combination of several small integration steps and one large extrapolation step boosts the effective step size of the overall integration scheme.

3.2.2.3 Stability Analysis

In this section, we compare projective integration methods with standard forward Euler based a linear stability analysis.

Consider the standard test problem for stability analysis: $\dot{y} = f(y)$, which represents a linear differential equation. For the system $\dot{y} = f(y)$, the stability of a linear numerical integration technique can be equivalently analyzed by considering a set of scalar equations in the form

$$\dot{y} = \lambda y \tag{3.8}$$

where the λ is an eigenvalue of A. Starting from the initial condition $y(t_0) = y_0$, the application of the inner integrator to equation (3.8) over a single step with step size h gives

$$y_1 = \rho(h\lambda)y_0, \tag{3.9}$$

where ρ is the amplification factor of the method, which is $\rho(h\lambda) = 1+h\lambda$ for forward Euler. Suppose that the error at t_0 corresponding to eigenvalue λ is ϵ_0. After n inner integration steps, the amplified error will be

$$\epsilon_n = \rho(h\lambda)^n \epsilon_0 \tag{3.10}$$

where

$$\sigma(h\lambda) = [(M+1)\rho(h\lambda) - M]\rho(h\lambda)^n. \tag{3.11}$$

Therefore, the region of absolute stability in the $h\lambda$-plane is the set of $h\lambda$ for which $|\sigma(h\lambda)| \leq 1$, which can be found by plotting the locus of all $h\lambda$ for which $|\sigma(h\lambda)| = 1$.

This locus will divide the $h\lambda$-plane into two or more continuous regions. If any point in a region is stable, then all the points in the region are stable by continuity.

Fig. 3.4 shows the comparison of stability regions of different methods. First, we take a look at the the projective integration methods with stepsize h with $n = 2$ and $M = 5, 7, 9$, respectively. As shown in Fig. 3.4, the stability region has split into two parts as M increases. This means the method is suitable to handle very stiff problems if the eigenvalues are separated into two clusters, one containing the stiff, or fast, components, and one containing the slow components. By carefully choosing the parameters k and M, the two parts of the stability regions can cover the two clusters of the eigenvalues of the system.

As shown in Fig. 3.4, the stability region of forward Euler of size h (FE(h)) is the unit circle centered at $(-1, 0)$, which is determined by $|1 + h\lambda| \leq 1$. Note here, to satisfy the stability constraint, the time step h cannot be more than twice as large as the smallest time constant of the system. This significantly limits the efficiency of simulation. In this regard, projective methods are more efficient due to the use of large extrapolation step, which has a stepsize of Mh. For example, compared with projective integration with $n = 2$ and $M = 9$ (P2-9), forward Euler takes $12(= n+1+M)$ steps calculation but projective integration method only takes $4(= n+1+1)$ steps calculation (3 inner integrations and 1 extrapolation). In order to achieve the same cost as method P2-9, forward Euler needs to use step sizes three times larger. In this case, as shown in Fig. 3.4, the stability region of method FE(3h) is much smaller and not sufficient to cover those fast components anymore.

3.2.2.4 Telescopic Projective Integration

It can be shown that with proper choices of n and M the projective integrator maintains the so-called [0, 1] stability [104]. However, in practice, when the eigen-

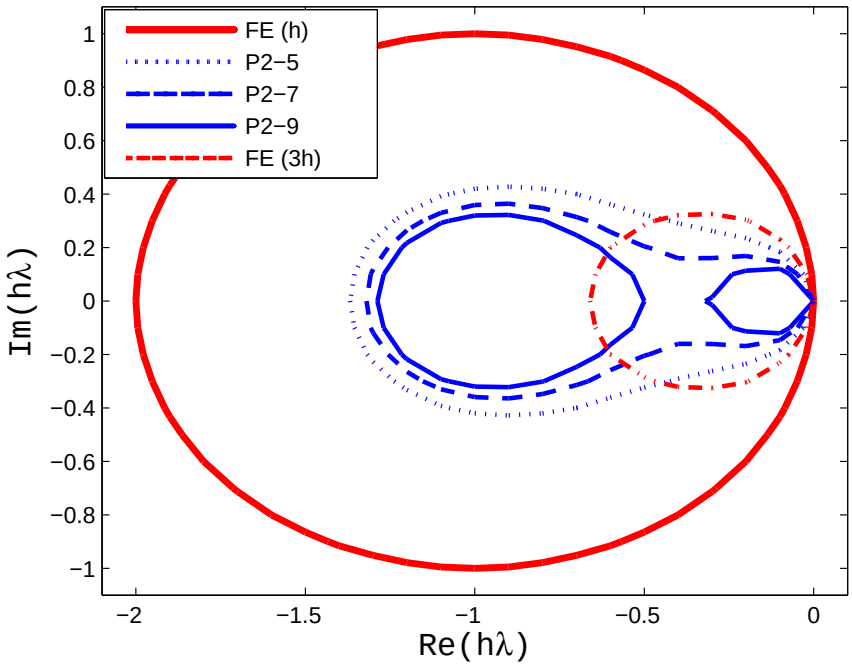

Figure 3.4: The comparison of stability regions of different methods, where FE(h) is forward Euler with stepsize h, P2-5, P2-7, P2-9, are projective integrations of size h with $k = 2$ and $M = 5, 7, 9$, respectively, and FE(3h) is forward Euler with stepsize $3h$, whose computational cost is the same as P2-9.

values of the system are widely distributed with no clear clustering or the eigenvalue distribution is not known a priori, the step size of the outer projective step must be conservatively controlled to ensure stability. In other words, M needs to be chosen conservatively small to ensure the [0, 1] stability, leading to reduced step size amplification. To maintain good efficiency in general practical cases, the concept of projective integration has been generalized to a multi-level telescopic scheme [105]. Fig. 3.5 illustrates a two-level telescopic projective integration scheme. The steps viewed at the top level are similar to those of a projective integrator except that each basic integration step is expanded into a projective integration step at the bot-

tom level. As such, while at each individual projective integration level, limited step size amplification is obtained with a relatively small M, significant overall step size amplification may be obtained in the multi-level telescopic framework.

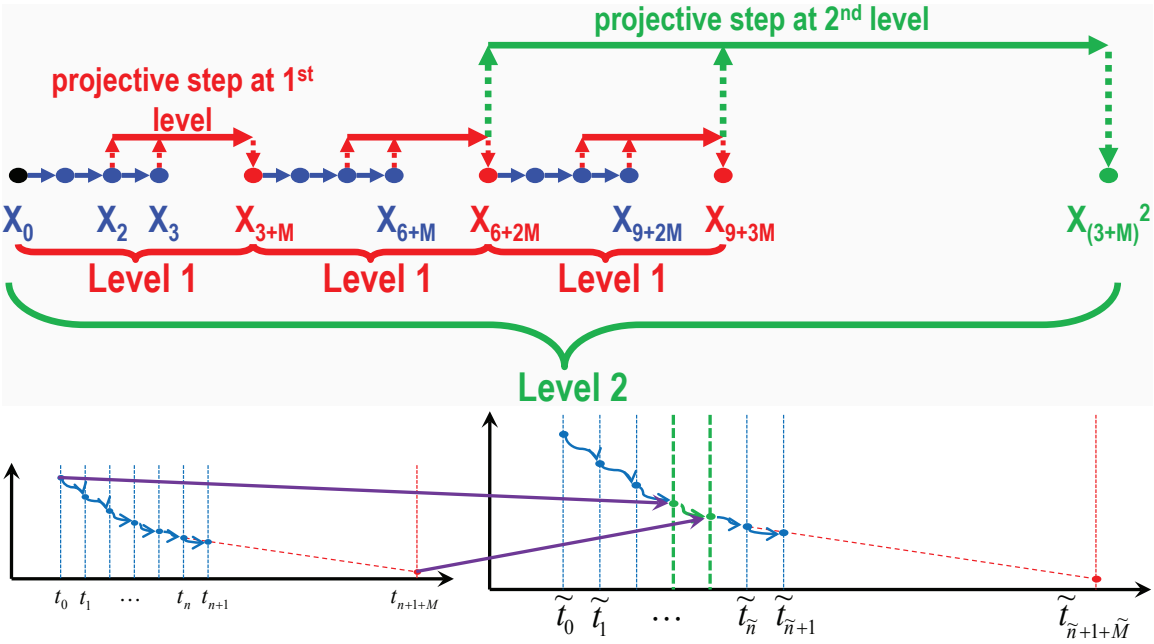

Figure 3.5: (Top) Illustration of two-level telescopic projective method. The scheme of each of the two levels is the same as the projective integration method. In the level 2, each small step of integration at the beginning is the replaced by level 1 projective integration. (Bottom) An example of two-level telescopic projective integration method applied for the simulation of the curve in Fig. 3.3(Bottom).

3.2.2.5 On-the-Fly Error Control

While multi-level telescopic projective integration method addresses the stability issues, accuracy is another important requirement for simulation. During the generation of action potentials, the membrane potential and other state variables evolve very quickly in time so as to produce sharply rising and falling spiking patterns. In

this case, the accuracy requirement is often much more stringent and very small step sizes need to be used to ensure accuracy, which effectively reduces the computational benefit of telescopic projection integration method. On the other hand, around the resting potential, no significant activity takes place in a neuron. In fact, on average a neuron sits around the resting potential, or is inactive, around 80% of time under typical network conditions. This may allow one to use the telescopic projection integration method in such low-activity periods to boost the simulation efficiency.

Therefore, we introduce a on-the-fly mechanism to predict action potentials and disable telescopic projective when needed. Currently, our implementation is based on a two-level telescopic projective integration. As the top-level projective step size is comparable to the duration of the action potential, the top-level projective step needs to be disabled when action potentials are predicted. The prediction is achieved by comparing the membrane potential of each neuron with a predefined threshold voltage. The threshold voltage is chosen conservatively to capture all the firing activities. Whenever the membrane potential goes above the threshold, we disable the top-level projective step.

Fig. 3.6 shows the top-level telescopic projective integration. The simulation starts with top-level inner loop, which is composed of steps of lower-level projective integration. After top-level inner loop, if the membrane potential is lower than the threshold, continue the simulation with top-level projective step to speed up the simulation. Otherwise, disable top-level projective step and continue the simulation with top-level inner loop to be conservative. After top-level projective step, if the membrane potential is lower than the threshold, continue the simulation with top-level inner loop to start the next cycle. Otherwise, the previous top-level projective step is not conservative and might lead to errors. In this case, the simulation done by the previous top-level projective step needs to be redone with top-level inner loop.

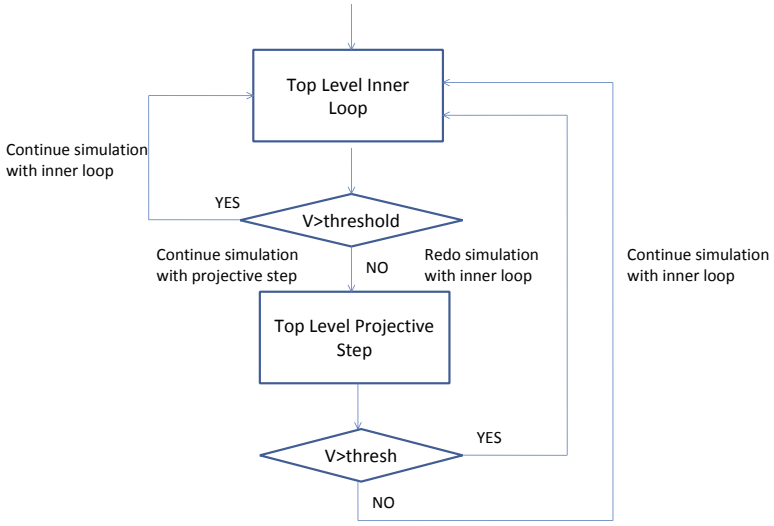

Figure 3.6: Telescopic projective integration method with dynamic disabling top-level projective step.

3.3 Simulation of Brain Activities

While existing works based on mean field or phenomenological models cannot easily link brain behaviors to underlying physiological and pathological mechanisms at the molecular and cellular levels, most computational works based on biophysically realistic models are often limited to local cortical circuitry and ignore the global structure. With biologically realistic brain modeling and advanced simulation techniques, our final goal is to link the brain dynamics to underlying biophysical mechanisms. In this section, we reproduce some physiological and pathological oscillations and demonstrate the transitions between different brainwave states caused by cellular and molecular level mechanisms.

3.3.1 Normal Sleep Spindle and Delta Oscillations

First, we use our brain model to produce normal spindle and delta oscillations based on the mechanisms suggested by studies in vivo, in vitro, and in silico.

Sleep spindle oscillations consist of waxing-and-waning field potentials at 7-14 Hz, which are typically observed during the early stages of sleep in vivo. Studies suggest that the minimal substrate accounting for spindle oscillations consists in the interaction between thalamic reticular and relay cells [87–89]. Burst firing of reticular cells induces inhibitory postsynaptic potentials in thalamocortical cells. This deinactivates low-threshold Ca^{2+} current I_T, inducing burst firing in thalamocortical cells, which, in turn, excites reticular cells allowing the cycle to start again. Spontaneous spindle oscillations are synchronized over large cortical areas during natural sleep and barbiturate anesthesia.

Thalamic delta oscillation (1-4 Hz) is a well known example of rhythmic activity generated intrinsically by thalamic relay cells as a result of the interplay between their low-threshold Ca^{2+} current I_T and hyperpolarization activated cation current I_h. As such, the delta oscillation may be observed during deep sleep when thalamic relay cells are hyperpolarized sufficiently to deinactivate I_T [86]. The mechanism of single cell delta activity is the following: a long-lasting hyperpolarization of thalamic relay cell leads to slow I_h activation that depolarizes the membrane potential and triggers rebound burst, mediated by I_T, which was deinactivated by the hyperpolarization. Both I_h (because of its voltage dependency) and I_T (because of its transient nature) inactivate during burst, so membrane potential becomes hyperpolarized after burst termination. This afterhyperpolarization starts next cycle of oscillations.

The sleep spindle and delta oscillations correspond to the simulation results of the first second in Fig. 3.7 and Fig. 3.8, respectively. In both figures, the two panels on the top show the waveforms of two randomly selected pyramidal cells from layer 6 that project to level 4 and level 5/6, respectively. Since there are 22 types of neurons, we show the spike rasters of 22 neurons with different types in the bottom left panel. Each neuron is randomly selected from neurons of a specific type. There are 22

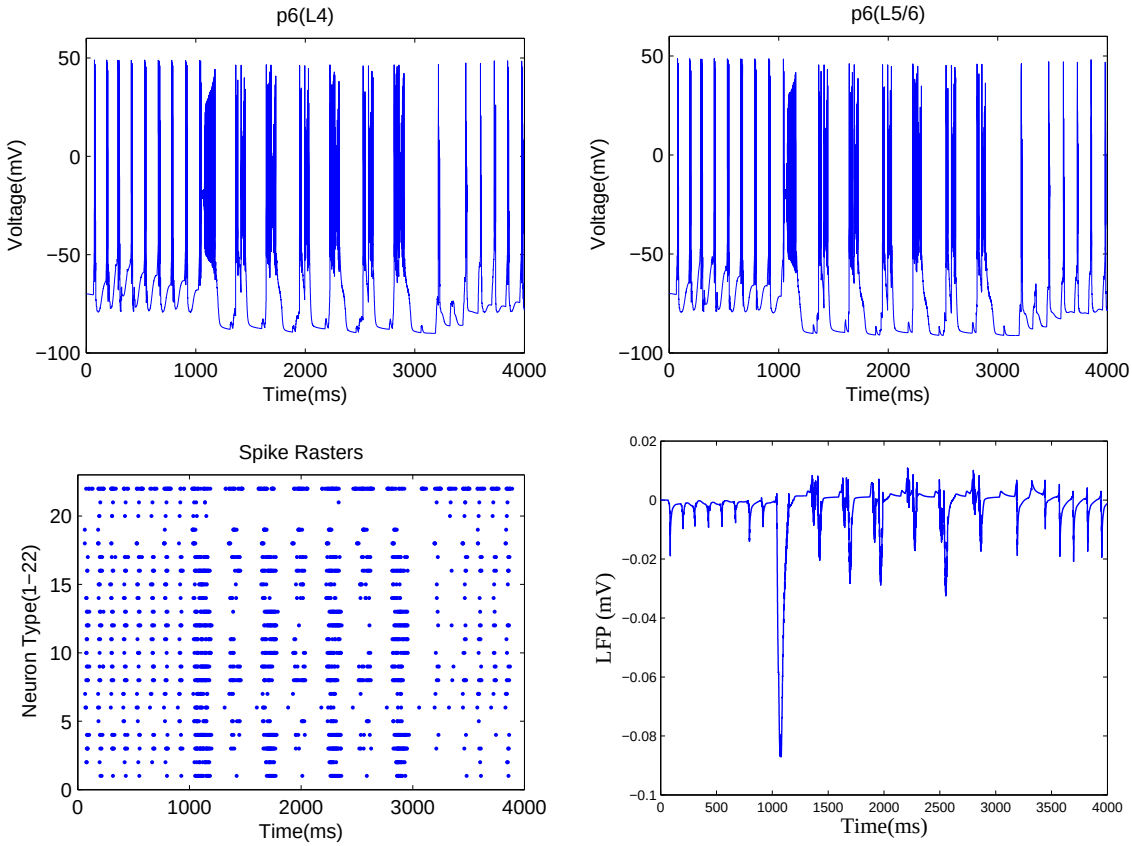

Figure 3.7: Transitions between spindles and the spike-and-wave patterns in epileptic seizure. Each of the two panels on the top shows the firing pattern of a randomly selected pyramidal cell from deep layers of the brain. The bottom left panel shows the spike raster of 22 neurons of different types. The bottom right panel shows the local field potential computed from the postsynaptic currents of neuron p6(L5/6).

rows and each row shows the firing density of the corresponding neuron within the simulated four seconds. In the bottom right panel, the local field potential (LFP) is computed from the postsynaptic currents of one p6(L5/6) neuron. As all the neurons nearby are highly synchronized during the oscillation, the LFP computed based on the average activities of the neurons nearby is very similar.

For spindle oscillation, as shown in Fig. 3.7, the brain begins to produce bursts of rapid, rhythmic brain wave activity at about 9 Hz, and pyramidal cells generate

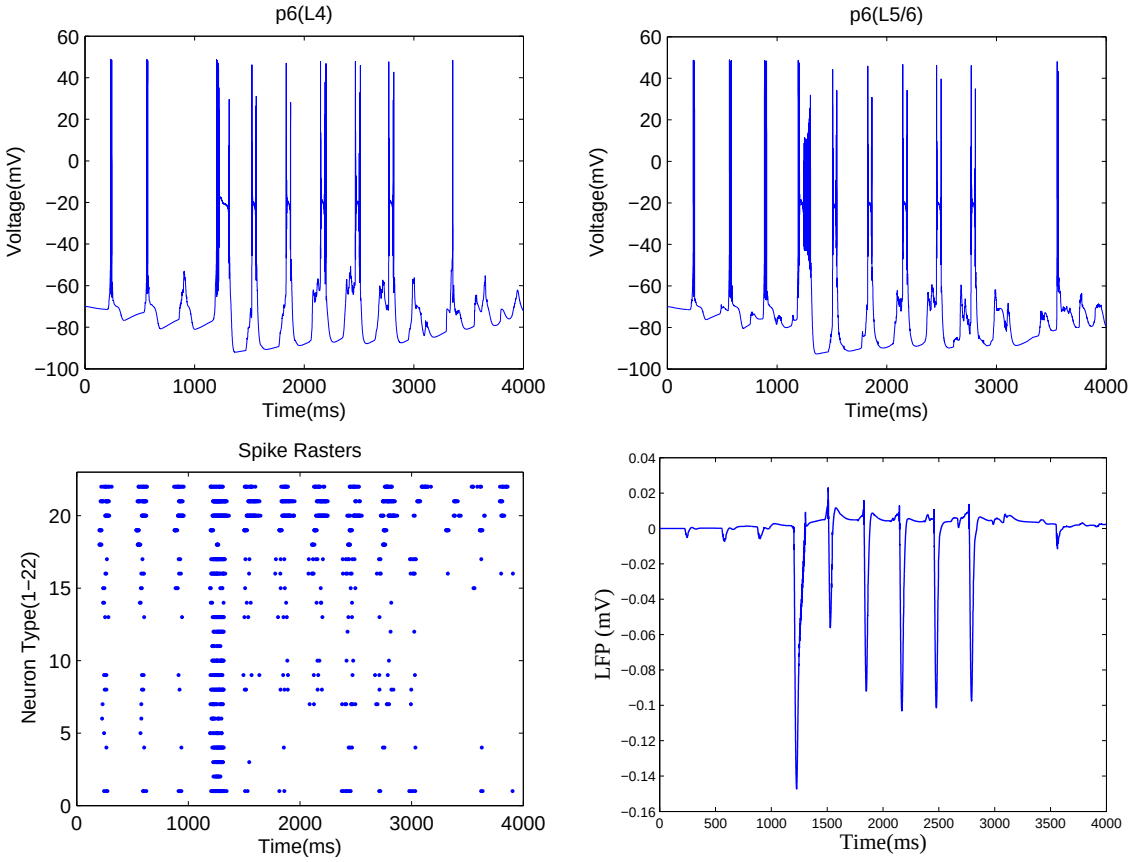

Figure 3.8: Transitions between delta wave and spike-and-wave patterns in epileptic seizure. Each of the two panels on the top shows the firing pattern of a randomly selected pyramidal cell from deep layers of the brain. The bottom left panel shows the spike raster of 22 neurons of different types. The bottom right panel shows the local field potential computed from the postsynaptic currents of neuron p6(L5/6).

a few spikes per cycle (top panels), and the local field potential consists of successive positive and negative deflections (bottom right panel). For delta oscillation, as shown in Fig. 3.8, the behavior of the brain is characterized by 3 Hz wave activity, where pyramidal cells generate a few spikes per cycle, and the field potential consists of successive positive and negative deflections. The two different modes of the brain are realized in this model by modulating the conductance of leakage potassium in thalamocortical cells to mimic the effects of brainstem modulation. Starting from

spindle oscillation, if the conductance is increased, the thalamic relay cells will be more hyperpolarized. In this case, delta oscillations will emerge to replace spindles oscillations, and vice versa.

3.3.2 Spike-and-Wave Epileptic Activities

The term spike-and-wave refers to a pattern of the electroencephalogram (EEG) typically observed during epileptic seizures. In particular, one of the most common types of epileptic manifestations, the absence seizure, displays a clear-cut oscillation consisting of generalized and bilaterally synchronous spike-and-wave (SW) EEG patterns recurring at a frequency of about 3 Hz in humans. In this section, we show epileptic spike-and-wave oscillation can arise from both normal spindle and delta oscillations caused by cellular and molecular level mechanism.

As shown in Fig. 3.7, after the first second, as the synaptic GABA_A receptors are suppressed, the network transits from the spindle oscillations to a slower oscillation at about 4 Hz with field potentials characterized by large-amplitude negative spikes and small-amplitude positive waves. As mentioned before, the spike-and-wave patterns are typically observed during epileptic seizures. In this pathological mode, all the cells fire prolonged high-frequency discharges synchronously during the negative spikes and the positive waves are coincident with the silent periods of all the cells. The spike raster(bottom left panel) demonstrates the synchronization of 22 types of neurons. After two seconds of epileptic seizure, the brain resumes generating spindle oscillation when the GABA_A-mediated inhibition comes back to normal condition. This portrait is typical of experimental recordings of cortical and thalamic cells with the SW oscillation pattern[106–111].

In addition to early sleep stage, we also show the occurrence of seizure at the deep sleep stage. As shown in Fig. 3.8, after the first second, by globally suppressing the

$GABA_A$-mediated inhibition, the network transits to the abnormal delta oscillation characterized by spike-and-wave patterns. After two seconds of epileptic seizure, the brain resumes generating normal delta waves when the $GABA_A$-mediated inhibition comes back to normal condition. Note that, although oscillations in both normal delta wave and spike-and-wave are less than 4 Hz, they are different in terms of the patterns of both action potential firing and field potentials. In terms of action potential firing, while pyramidal cells in normal state generate a few spikes per cycle, they generate prolonged firing per cycle in the pathological state. As a result, while the field potential in the normal state consists of successive positive and negative deflections, the field potential in the pathological state consists of large-amplitude negative spikes and small-amplitude positive waves.

3.3.3 Visualization of the Brain Activities

To illustrate the activities of the brain, we show three-dimensional visualization of postsynaptic currents of pyramidal cells from different regions during the 4 seconds simulation in Fig. 3.8. The visualization is achieved by using the utility functions of Brainstorm [112]. Fig. 3.9 shows the snapshots at 570ms, 1190ms, 2140ms, and 3700ms. The activities at 570ms and 3700ms correspond to the active and inactive phases of delta waves at the deep sleep stage, respectively. The activities at 1190ms and 2140ms correspond to the active and inactive phases of spike-and-waves during seizures, respectively. As neurons generate prolonged firing per cycle during seizure, the magnitudes of the postsynaptic currents are much larger compared with those in normal delta waves.

In addition, during different phases of waves, although coherence exists to a large degree, complex spatial and temporal patterns are exhibited by spontaneous neuronal dynamics within the cerebral cortex. As the same microcircuitry is used in

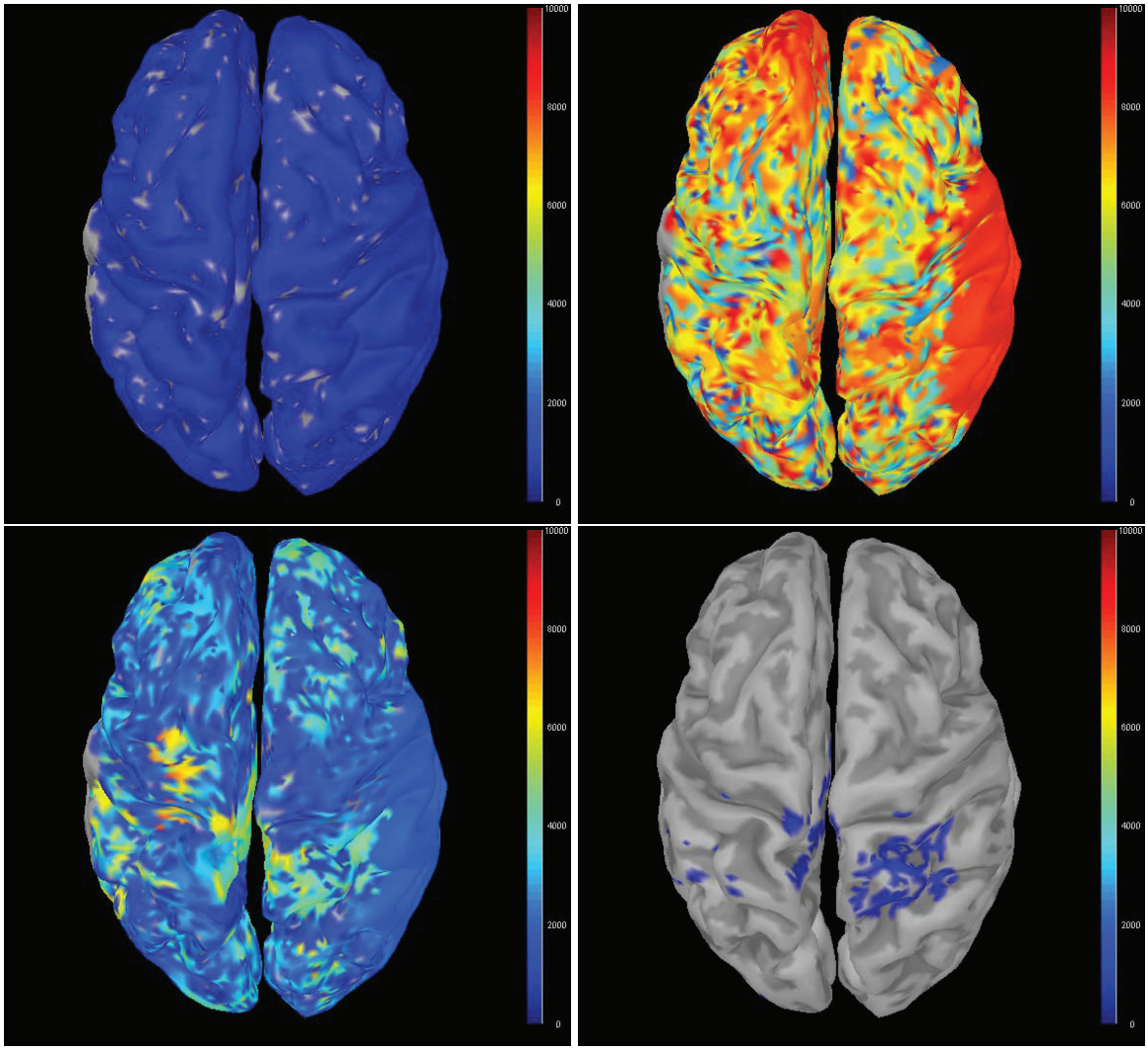

Figure 3.9: Snapshots of postsynaptic currents of deep layer pyramidal cells at different times: 570ms(top left), 1190ms(top right), 2140ms(bottom left), 3700ms(bottom right). The activities at 570ms and 3700ms correspond to the active and inactive phases of delta waves at the deep sleep stage, respectively. The activities at 1190ms and 2140ms correspond to the active and inactive phases of spike-and-waves during seizures, respectively.

different brain areas, such temporal and spatial variations might largely come from the underlying anatomical connections between regions of the cerebral cortex. Recent studies have shown that the structural connections shape functional connectivity

on multiple time scales [113] and form dynamic links mediated by synchrony over multiple frequency bands for large-scale integration [114].

With biologically realistic brain modeling and advanced simulation techniques developed, we have demonstrated some initial results to show the applicability of the modeling and simulation work to link the brain dynamics to underlying biophysical mechanisms. While further development and calibration of the model are still ongoing, those initial results are interesting because they show the possibility to determine underlying cause of diseases by simulating the biologically realistic brain model. At the same time, as the results are consistent with the basic understanding of seizure generation and existing experimental results[47, 51–53], the effectiveness of the modeling and simulation works has been initially validated. With further development, the work is geared to assisting the clinicians in determining underlying causes of brain disorders and selecting the optimal treatment on an individual basis in the future.

3.4 The Efficiency of the Simulator

So far, we have proposed advanced techniques to develop a highly efficient parallel simulator to cope with the computational challenges associated with the biologically realistic brain model with millions of neurons and hundreds of millions of synapses. In this section, we show the efficiency of the simulator with networks of different scales.

To show the efficiency of the proposed techniques, brain models with four different sizes are generated by scaling the total number of neurons (30K, 60K, 105K and 1.05M). The detailed characteristics of these models are listed in Table 3.3.

First, we demonstrate the performance improvement achieved via parallelization with dynamic load balancing. We simulate the 30K network with linear receptor

Table 3.3: Characteristics of Networks.

Network	30K	60K	105K	1.05M
Number of regions	10	20	70	70
Total number of neurons	30,090	60,180	105,770	1,050,630
Number of excitatory neurons	23,550	47,100	82,670	822,640
Number of inhibitory neurons	6,540	13,080	23,100	227,990
Total number of compartments	223,940	447,880	786,590	7,819,700
Total number of synapses	8,760,347	17,523,723	30,784,998	305,959,213
Number of excitatory synapses	7,352,717	14,788,923	26,081,068	259,235,823
Number of inhibitory synapses	1,407,630	2,734,800	4,703,930	46,723,390
Total number of state variables	5,295,550	10,430,180	18,120,550	180,054,490

merging for 10 ms of simulation time using different number of threads and the results are listed in Table 3.4. The performance is measured by parallel efficiency, which is defined as the ratio between the achieved parallel speedup and the number of threads. As shown in Table 3.4, the parallel efficiency is high (still close to 1 even for the maximum number of threads).

Table 3.4: Outcome of Parallelization with Dynamic Load Balancing.

Number of Threads	Runtime (s)	Speedup	Parallel Efficiency
1	177.721	1	1
12	14.9522	11.886	0.9905
18	9.98306	17.802	0.9890
24	7.5014	23.692	0.9872

On top of the parallelization, we show the performance improvement with linear receptor merging, locality enhancement, and telescopic projective integration. Similarly, the 30K network in deep sleep mode is used for performance evaluation. The setups and results of all three simulations are shown in Table 3.5, where the macro-

step is the step size for continuous simulation of a neuron without being interrupted by the computing work of other neurons, and the micro-step is the step size for numerical integration. Note that, with two-level telescopic projective integration, the micro-step is the smallest step being used (the steps in the inner loop of the bottom level telescopic projective integration).

Table 3.5: Speedup by Receptor Merging (RM), Locality Enhancement (LE), and 2-Level Telescopic Projective Integration (2LTPI).

Setup	Basic	RM	RM+LE	RM+LE+2LTPI
Macro-step Size (ms)	0.01	0.01	1	1
Micro-step Size (ms)	0.01	0.01	0.01	≥ 0.01
Simulation Time (s)	0.3	0.3	0.3	0.3
Runtime(s)	7127.04	2103.26	1335.74	323.95

The basic simulation setup without merging is shown in the second column of Table 3.5, where parallelization with dynamic load balancing is applied. On top of the basic setup, we merge linear synaptic receptors and this provides about 3.39× speedup as shown in the third column of Table 3.5. In the previous two simulations, the macro-step is the same as the micro-step. On top of the basic setup, we introduce the locality enhancement method. By continuously simulating each neuron during one macro-step of $1ms$, the locality of cache is enhanced and accesses to global memory can be significantly reduced. As shown in the forth column of Table 3.5, this brings 1.57× speedup in terms of runtime. On top of the setup with 1 ms macro-step, the 2-level telescopic projective integration is applied within the 1 ms

macro-step. For the bottom level projective integration, the parameters are

$$n = 1, M = 2, \tag{3.12}$$

and for the top level projective integration, the parameters are

$$n = 7, M = 17. \tag{3.13}$$

As shown in the fifth column of Table 3.5, the 2-level telescopic projective integration method further brings 4.12× speedup.

We also demonstrate the capability of the simulator for simulating the networks of different sizes listed in Table 3.3. The simulation time and corresponding runtime for different networks are shown in Table 3.6. The results demonstrate that the runtime scales linearly with the size of networks.

Table 3.6: Runtime Scaling with Network Size.

Network Size	Runtime (s)	Simulation Time (s)
30K	1083 (18.05 min)	1
60K	2162.33 (36.04 min)	1
105K	3899.59 (1.08 h)	1
1.05M	16035.7 (4.45 h)	0.5

To summarize, parallelization with dynamic load balancing provides linear speedup in the number of processors. With our 24-core processor, the maximum speedup is about 24× for the 30K network. On top of parallelization, linear receptor merging, locality enhancement, and telescopic projective integration further produce 3.39×, 1.57× and 4.12× speedup, respectively, and the accumulated speedup brought by

the three techniques is about 21.93×. Therefore, with all the techniques proposed, the accumulated speedup is about 526×.

3.5 Discussion

Given the complexity and inaccessibility of the brain, computational modeling is an essential tool for bridging the gap between the understanding of neural circuits at cellular level and that of full-scale behavior and function of nervous system. To achieve this goal, computational models should be biophysically detailed, constrained, and validated by available experimental data at various levels.

In this section, we discuss specifically the granularity and applicability of the proposed model.

3.5.1 The Morphology of Pyramidal Cells

The morphology of pyramidal neurons is believed to play important roles in synaptic integration. For example, the existence of dendritic domains with distinct synaptic inputs, excitability, modulation and plasticity appears to be a common feature that allows synapses throughout the dendritic tree to contribute to action-potential generation [115]. These properties support a variety of coincidence-detection mechanisms, which are likely to be crucial for synaptic integration and plasticity.

In terms of epilepsy, while studies often focus on the changes in inhibitory and excitatory synaptic function, the role of changes in intrinsic excitability and abnormal dendritic properties has recently received some attention. For example, cortical dysplasia is often associated with intractable seizures. Evidence in the rat freeze-lesion model of cortical dysplasia has suggested that alterations in HCN channels and dendritic morphology may play a role [116]. At current stage, our brain model is still generic and the multi-compartments models are crude. Modeling and simulating

brain behavior at a large scale is very challenging and it is limited by the availability of data. However, given the existing evidence of the importance of dendritic morphology, we believe inclusion of multi-compartmental models into the analysis, even in a limited manner, as what has been done in our model, can be beneficial.

3.5.2 The Brain-Wide Connectivity

Due to the limitations of existing experimental techniques, a complete picture of all the pathways in the brain is still largely unknown. Nevertheless, it is still necessary to incorporate existing brain-wide connectivity information into the model. While the knowledge is still incomplete, a wide range of typical cortical dynamics and functions have been demonstrated by incorporating the brain-wide connectivity information into computational models [113, 117–119]. For example, in [113], network structure of cerebral cortex has been shown to shape functional connectivity on multiple time scales by simulating nonlinear neuronal dynamics on a network the captures the large-scale interregional connections of macaque neocortex. Recently, significant efforts [120, 121] have been devoted to identifying signal pathways to make the picture clearer.

In terms of epilepsy, it is necessary to include the brain-wide connectivity into the model to study the pathways of propagation as seizures (secondary generalized types) start from one region of the brain and spread to both sides of the brain. Recently, the interplay of cortical local and long-range synchronization has been found to play an crucial role in human absence seizure initiation by M/EEG recordings [98]. The observations reveal a multifocal fronto-central network and contradict the classical view of sudden generalized synchronous activities in absence epilepsy. Currently, we are using computational models to investigate the dependencies of epilepsy on global connectivity. Hopefully, we will be able to report more results along this line in the

near future.

3.5.3 The Size of the Network

Due to limited computer power, most exiting computational works with biophysical models are often restricted to small-scale networks. In this section, large-scale brain models are constructed to link network level activities to the underlying biophysical mechanisms. Due to a large number of brain regions, finite-size effects, and the spatial resolution for practical applications, it is necessary to perform simulations with large-size networks.

The influence of finite-size effects can be in different aspects. First, finite-size effects influence the convergence time of a system. The time needed for convergence to a stable state starting from an initial state is studied numerically for networks up to $N = 10^5$, and the variance of the distribution in convergence times as a function of N is demonstrated in [122]. It suggests that the average convergence time is increasing as a function of N. Additionally, the spread of the distribution of convergence times decreases for large N. Therefore, for a large-size network, the transition time between different states tends to be longer and more deterministic while for a small-size network, the transition time tends to be shorter and more randomized. The transition time is important to accurately characterize the dynamic behavior of the brain. For example, a transition to a synchronization state of the brain can result in a seizure, and the transition time is critical for seizure prediction. As a result, large-size networks are needed to obtain simulation results close to reality.

In addition to convergence time, finite-size effects can also influence the mean field in a population of interacting oscillators [123]. If the coupling strength exceeds a threshold value, a nontrivial state (mutual synchronization) with a finite macroscopic mean field (its order parameter) appears. The order parameter is a complex-valued

number used to characterize the synchronization of the network. The amplitude of the order parameter measures the coherence of the oscillator population and the phase is the average phase. It has been shown that both the variance of the amplitude and the diffusion constant of the phase scales with the system size as $1/N$. Therefore, given the same set of parameters, while a small size network gets synchronized, a large size network might not be. As the size increases, the simulation results will become more deterministic and closer to reality. While these observations were made when oscillator models were used, it is expected that they may hold true in realistic models of the brain because under some conditions realistic models can be simplified to oscillator models by phase reduction method [124].

The large size network is necessary not only because multiple brain regions are included and finite size effects are considered but also because it is important to maintain a high spatial resolution in the presence of heterogeneity for practical applications. For example, this is the case if the effects of local structure changes (axon sprouting, cell death, brain lesions, etc) of some parts of a brain region are to be studied. Another example is the potential usages of computational models to simulate the effects of a surgery, which is to remove the brain tissue producing seizures, or interrupt the nerve pathways through which seizure impulses spread. To maintain critical brain functions such as language, sensation and movement, it is important to determine if a surgery is appropriate, where to cut, and how to cut. If the brain tissue cannot be safely removed, multiple subpial transection is needed to control seizure, which makes a series of transection in the brain tissue. These cuts interrupt the movement of seizure impulses but do not disturb normal brain activity, leaving the person's abilities intact. To simulate the effects of a surgery, the spatial resolution of the model has to be high and a large size network has to be employed to fulfill this mission.

3.5.4 Comparison with Phenomenological Models

Compared with phenomenological models, the advantage of HH models is that it is convenient to link the network-level activities to underlying physiological and pathological mechanisms at the cellular and molecular levels, which are especially useful to model therapeutics targeting molecular pathways. For example, cellular mechanisms (ion channels) in pyramidal cells have been shown to play important roles in seizure generation [85, 125, 126]. Those ion channels, which are potential therapeutic targets, include persist sodium current, high-threshold calcium current, M-current, afterhyperpolarization current, hyperpolarization-activated inward current, etc. While the cell-level responses are complex and highly dependent on the interplay of those ion channels, only the effects of spike frequency adaptation (M-current) are taken into consideration and all other effects are omitted in the corresponding phenomenological model for pyramidal cells (regular spiking types [127]). Moreover, as the cell-level responses are dependent on the numbers and characteristics of ion channels, once the ion channels are changed, the effects can be immediately evaluated with HH models. In contrast, the goal cannot be easily achieved with phenomenological models because the coefficients in those models have no direct link to the underlying biophysics.

3.5.5 Comparison with Macroscopic Models

For macroscopic models, there is a key distinction between models that summarize the activity of a neuronal population with a single state (e.g., its mean activity) and those that model the distribution of states in terms of probability densities (i.e., density dynamics). Following the terminology established in [128], if the model can be cast as a set of ordinary differential equations describing the evolution of neuronal states, it is called a neural-mass model [129–131]. This is motivated by treating the

current state as a point'mass' (i.e., delta function) approximation to the underlying density on the population's states. In addition, models that are formulated as partial differential equations of space and time are referred to as neural field models [132–134], because they model the spatiotemporal dynamics on spatially extensive (e.g., cortical) fields. Conversely, models based on stochastic differential equations that include random fluctuations are referred to as mean-field models [4, 135–140]. This nomenclature appeals to the use of the term 'mean-field' in statistical physics and machine learning.

Both neural field and mass models are parsimonious models of mean activity (e.g., firing rate or membrane potential) and have been used to emulate a wide range of brain rhythms and dynamics [141, 142]. Neural-mass models are particularly suitable for data that reflect the average behavior of neuronal populations such as the EEG and MEG [143, 144]. Recently, by unifying data from different imaging modalities, neural field models have provided a direct connection from neural activity to EEG and fMRI data [145–147]. Moreover, dynamic causal modeling (DCM), which is frequently invoked for the interpretation of fMRI data, has been extended from a data-driven perspective to incorporate activity models based upon neural field equations [148].

As the the switches of brain activity between qualitatively different states can be represented by dynamical behavior of many neural population models, those models have been widely applied to understand the genesis and evolution of epileptic activity [149–160]. While the simplest neural mass models only include two subpopulation models (excitatory and inhibitory) with both inhibitory and excitatory connections among them, more complex models with more than two subpopulations are needed to animate various complex EEG patterns seen in experiments. Wendling et al. elaborated those models to describe the hippocampus activity in temporal lobe

epilepsy(TLE) [149, 151, 153]. Suffczynski et al. investigated the mechanisms of transition between normal EEG activity and epileptiform paroxysmal activity using a computational macroscopic model of thalamocortical circuits [154]. Robinson et al. used this type of modeling approach to study epileptic seizures and different normal EEG rhythms such as slow-wave sleep, alpha and low-gamma waves [150, 158]. Liley et al. showed that seizure-inducing properties of some general anaesthetic agents could be reproduced with neural mass model [157]. Molaee-Ardekani et al. proposed a neural mass model to analyze mechanism underlying the generation of fast oscillation (80Hz and beyond) at the onset of seizures [160].

Those macroscopic models have certain advantages over the more detailed models. First of all, they are easier to analyze numerically because relatively few variables and parameters are involved. Their computational efficiency is an important advantage. Second, since the models describe mean activity of neuronal populations, they are particularly appropriate for data that reflect the average behavior of neuronal populations like EEG and MEG.

The main disadvantage is that important biological details (ionic currents, dendritic structure, etc) tend to get lost in those macroscopic models or are captured using high-level characteristic parameters. As such, the models are less suitable for suggesting physiological and pathological mechanisms at the molecular and cellular levels and modeling therapeutics targeting molecular pathways.

3.5.6 Limitation Due to Data Availability

Finally, we would like to point out that, while it is very promising to build biophysically realistic models, there is not enough information to fill the very detailed parameter space of the model currently. For example, while the local cortical circuitry might be different from region to region, due to lack of data availability, the

local cortical circuitry in all brain regions in our model is based on the detailed reconstruction studies of cat area 17 (visual cortex). Under current conditions, the potential benefits of such a detailed model with global brain connectivity as well as local biophysics cannot be fully materialized. Therefore, the level of details in the modeling of the network shall be made consistent with the amount of available data. As such, the power of detailed large-scale computer simulation may only be fully leveraged when sufficient biological data becomes available in the future.

3.5.7 Implementation on Other Computing Platforms

The simulation experiments adopting multi-thread parallelization in this section are conducted on the shared-memory machine PowerEdge R715 [161]. To further parallelize the simulation by utilizing the supercomputers with more CPUs and computing power, this work is extended to leverage hybrid MPI/OpenMP simulation [162] (Not included in this dissertation). Through message passing interface, the communications between neurons would be much slower. In this work, since the simulation is conducted on a share-memory machine, the exchange of data between neurons is more efficient. We simulate the network with $105K$ neurons for half second. The time spend on exchanging data is below 0.12%. The brain is most active during epilepsy, where lots of neurons fire simultaneously and exchange data much more frequently. Under this circumstance, less than 5.5% of simulation time is spend on exchanging data between neurons.

Another popular hardware platform for large scale parallel simulation is GPU. Decent simulation speeds are obtained by utilizing the large number of cores in GPU [33][34]. But disadvantages of GPU are also obvious. This single-instruction multiple-data platform is efficient only for the situation in which all cores execute the same instruction and no branches are in the sequence of instructions. However,

this is rarely the case due to the dynamical behavior of the network and the diversity of neuron activities across the entire network. Managing thread divergence is a key factor in achieving good simulation efficiency on GPUs.

3.6 Summary

The simulation-based studies of physiological and pathological behaviors like brain disorders are typically limited to small scale networks due to the lack of computing power. On the other hand, most recent large-scale simulation works are based on simple spiking models and thus cannot be used to directly associate network level behavior to underlying physiological and pathological mechanisms at the molecular and cellular levels. To bridge the gap, in this work, we have constructed large-scale biophysically realistic brain models with sufficient biophysical details, which consist of six-layered cerebral cortex with seventy regions as well as multiple thalamic nuclei. To address the associated computational challenges in simulation of networks of such complexity, we have proposed and adopted efficient techniques from different angles. With all of the techniques applied, the simulator is able to simulate the detailed model to generate biologically meaningful results. We have demonstrated the applicability of the modeling and simulation work by showing the role of $GABA_A$-mediated inhibition in the generation of absence seizure from early and deep stages of sleep. With further development, the work is geared to assisting the clinicians in determining underlying causes of brain disorders and selecting the optimal treatment on an individual basis in the future.

4. A HARDWARE-ORIENTED LIQUID STATE MACHINE WITH BIOLOGICALLY INSPIRED LEARNING

In the previous two sections, engineering methods, i.e. mathematical modeling and computer simulation, are applied to facilitate the study of biological systems. In this section, we take an opposite direction in which we leverage the inspirations drawn from biology to build better engineering systems. With the mainstream Von Neumann computer architecture increasingly challenged by concerns in device reliability and process variations, we propose a hardware-friendly bio-inspired learning system, consisting of a digital liquid state machine (LSM) and biologically inspired learning rule. With optimized design parameters, we also demonstrate its good performance on isolated utterance recognition.

This section is organized as follows: Section 4.1 briefly introduces the concept of liquid state machine, and describes its network structure, and application to speech recognition. Section 4.2 explains the motivation for our proposed learning rule, followed by the design and spiking neural network implementation of the new rule in Section 4.3. In Section 4.4, we study the influence of the synaptic model on the fading memory and LSM performance. To be hardware-friendly, Section 4.5 conducts efficiency optimization of the network model. Section 4.6 evaluates the application of our spiking neural networks to speech recognition and studies the influence of the synaptic precision and reservoir size on the LSM performance. We summarize our key findings in Section 4.7.

4.1 Liquid State Machine for Speech Recognition

4.1.1 General Network Structure

The network structure of a liquid state machine is illustrated in Fig. 4.1. The reservoir in the middle is comprised of a group of neurons connected either purely randomly [163] or in a way following certain distributions of spatial locations of neurons [55]. As multiple recurrent loops are created by these synaptic connections, the reservoir features transient behavior which memorizes information of its inputs in the past. Reservoir neurons and readout neurons are connected by plastic synapses of which the weights are to be adjusted according to training data and adopted learning rules. Through its plastic synapses, each readout neuron receives weighted sum of signals from reservoir neurons.

From the LSM structure, it is clear that input signals are processed in two steps. The first step involves input neurons, reservoir neurons and synapses connecting these neurons. Since the number of neurons in the reservoir is generally larger than that of the neurons providing inputs, in this step the reservoir maps the input signal to liquid response, a higher dimensional transient state. Note that this mapping is nonlinear in nature and that after being nonlinearly casted to higher dimensional space, complex patterns are more likely to be linearly separable [164]. In the second step, the liquid response is projected to readout neurons through plastic synapses

$$I_o(t) = \sum_i w_{i,o} \cdot r_i(t), \tag{4.1}$$

where t is time, $I_o(t)$ is the net input to a readout neuron, $r_i(t)$ is the output of the i^{th} reservoir neuron, and $w_{i,o}$ is the weight of the synapse connecting the i^{th} reservoir neuron and the readout neuron. Over the duration $[0, T]$ of a input temporal signal,

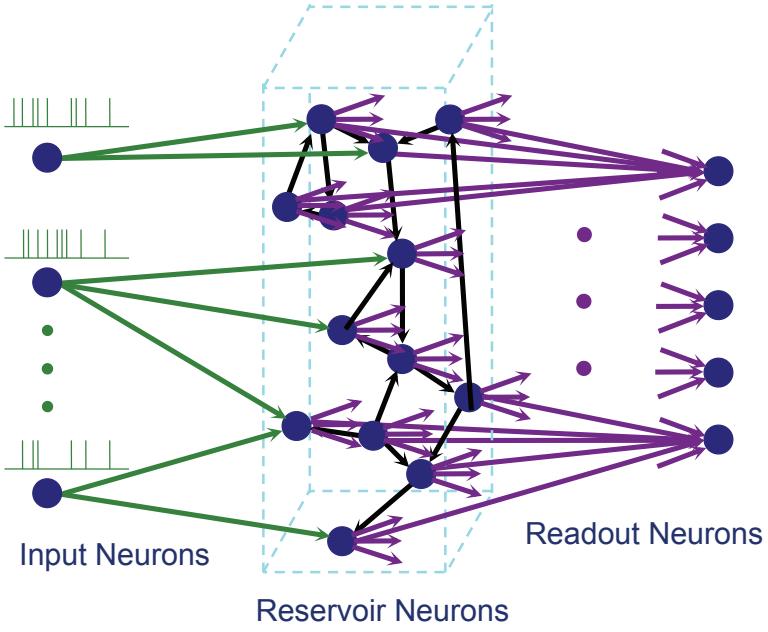

Figure 4.1: Illustration of the network structure of a liquid state machine. Dots and arrows represents neurons and synapses, respectively. Neurons on the left provide input spike trains to reservoir neurons. The reservoir in the middle consists of a group of neurons randomly connected to each other, receives input spike trains and project outputs through plastic synapses to readout neurons on the right.

the net input to the readout neuron is

$$\int_0^T I_o(t) = \sum_i w_{i,o} \cdot \int_0^T r_i(t). \tag{4.2}$$

Since the net input to each readout neuron is a linear combination of outputs of all reservoir neurons, each readout neuron can be viewed as a linear classifier of liquid responses. In this linear classification, only liquid responses produced by inputs of certain class are expected to activate a specific readout neuron. In the feature space, a hyperplane defined by all $w_{i,o}$ separates these inputs from others. This linear classification problem is solved by calculating weights of plastic synapses

mathematically using Ridge regression [65]. In [66], a hidden layer is added between the reservoir and readout neurons. Thus, backpropagation-based training algorithms for multi-layer perceptrons are used such that liquid responses do not have to be linearly separable.

4.1.2 Preprocessing of Speech Signals

To apply the LSM to speech recognition, the speech signals have to be preprocessed and encoded by spike trains. A number of methods exist for this step, such as Hopfield coding [165], MFCC (Mel-Frequency Cepstral Coefficients)[166][167], passive ear model [168][169], BSA (Ben's Coding Algorithm) [170] as summarized in [65] and temporal based linear predictive coding [66]. For good performance and to be biologically plausible, we use data processed by Lyon passive ear model [168][171] and BSA.

The preprocessing stage combining Lyon passive ear model and BSA is illustrated in Fig. 4.2. After the filter modeling the outer ear and middle ear, the speech signal is processed by the cochlea modeled by 77 cascaded band pass filters with each extracting certain frequency band of the voice spectrum. Filter 1 extracts the highest frequency band and the filter 77 the lowest. The signal extracted from each filter is further modified by a half wave rectifier and compressed by automatic gain control (AGC) modules. The signal compression by AGC is inspired by the fact that human ear can hear sound levels in a dynamic range of about 120 dB while the firing frequency of neurons in response to sounds only varies within about two orders of magnitude. After the compression, the time domain signal is converted to spike trains by BSA [170], where a stronger signal is converted to a spike train with a higher instantaneously spiking rate.

To summarize the preprocessing stage, this module converts the input speech

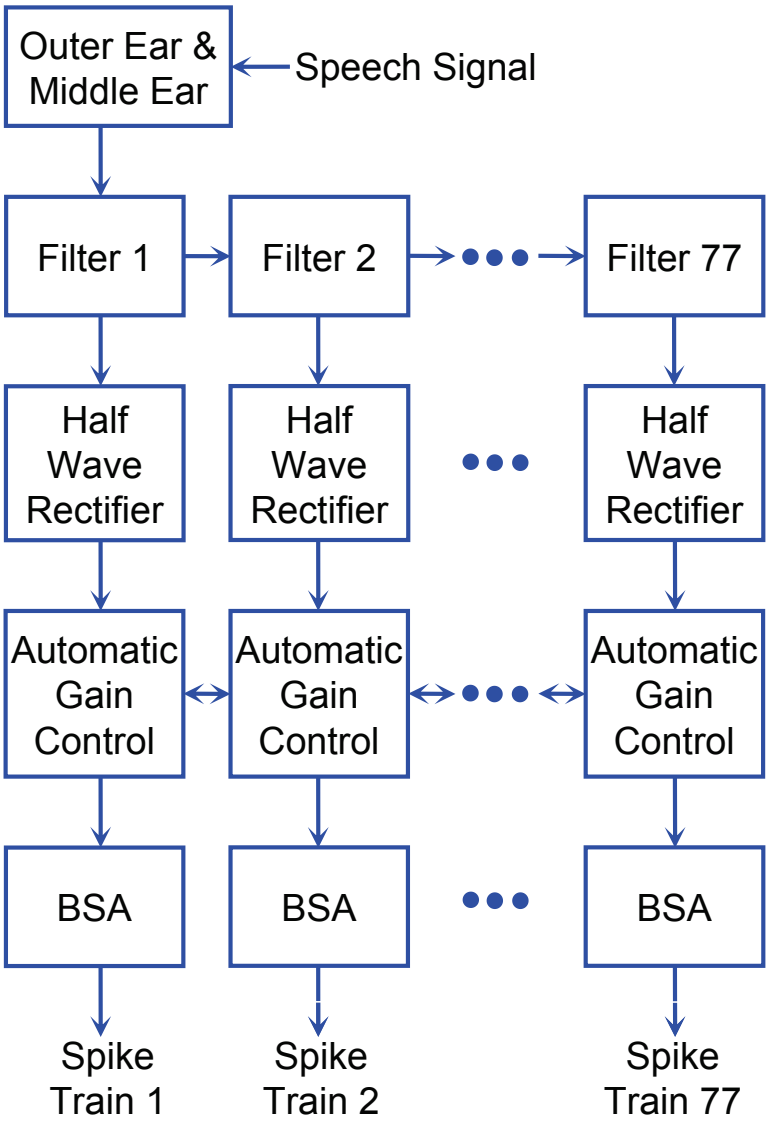

Figure 4.2: Preprocessing of speech signal. The speech signal is processed by a filter representing the outer ear and middle ear followed by 77 cascaded bandpass filters modeling the cochlea. After each half wave rectifier, the magnitude of time domain signal in each frequency band is compressed by an automatic control module. The resulting signal is converted to spike trains by BSA.

signal into 77 parallel spike trains representing different frequency channels covering the entire voice spectrum.

4.1.3 The Entire System for Speech Recognition

For the liquid state machine used as a speech recognizer in this section, the reservoir has a regular grid structure [62]. It is an $k \times l \times m$ grid with an neuron in each grid point, as illustrated in Fig. 4.3. Synaptic connections are allocated randomly such that neurons closer to each other have a higher probability to be connected. The probability of a connection [55] is

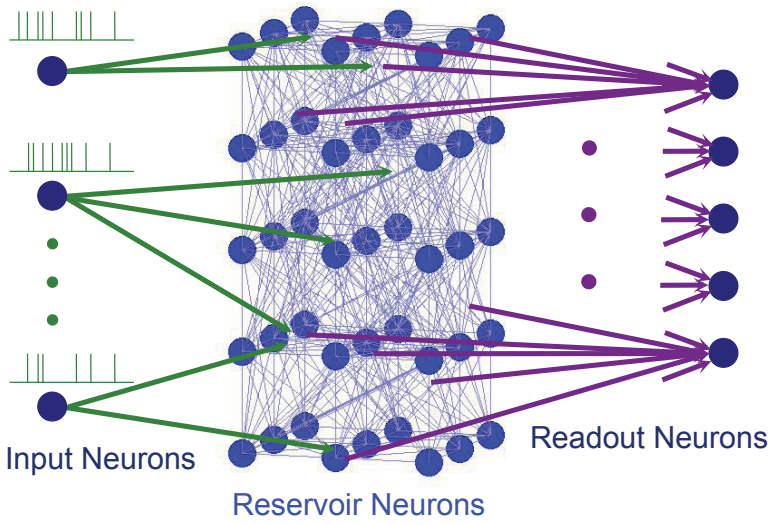

Figure 4.3: Illustration of a liquid state machine with reservoir of $3 \times 3 \times 5$ grid structure. Dots and lines represent neurons and synapses, respectively. The connectivity of neurons are determined by the distance between neurons. This grid structure more closely resembles a neuron column in the cortex.

$$P_{connection}(N_1, N_2) = k \cdot e^{-\frac{D^2(N_1,N_2)}{r^2}} \tag{4.3}$$

where N_1, N_2 represent two neurons, $D(N_1, N_2)$ is the Euclidean distance between the two neurons, k and r are two appropriately chosen constants.

The 77 spike trains produced by the preprocessing stage are fed to the liquid state machine as the inputs. Conceptually, there is a input layer of 77 neurons of the LSM whose outputs are generated by the BSA algorithm. The entire speech recognition system combining the preprocessing stage and the liquid state machine is illustrated in Fig. 4.4.

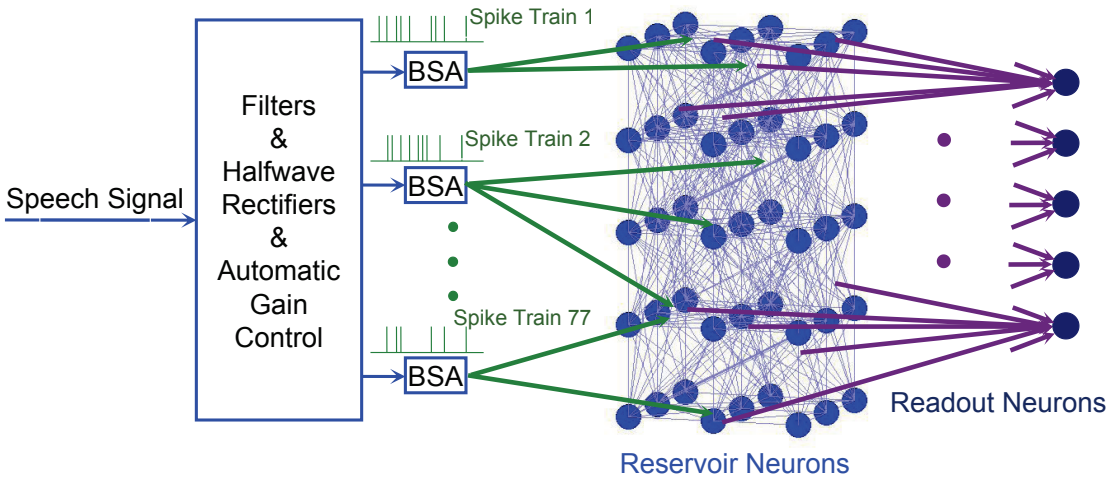

Figure 4.4: The entire system for speech recognition using liquid state machine. 77 spike trains from the the preprocessing stage are used as input to the reservoir of the LSM. In our implementation, the BSA algorithm is implemented in each input neuron of the LSM.

4.2 Motivation for the Proposed Bio-Inspired Learning Rule

4.2.1 Hebbian Learning

As stated in the previous section, each readout neuron is viewed as a linear classifier. The online training of a linear classifier is conducted as input data streams in. This process is illustrated in Fig. 4.5(Left). The 3 lines represent the training process of the hyperplane to separate two classes of data in the feature space. Our

goal is to find a spike-based learning rule for this training process.

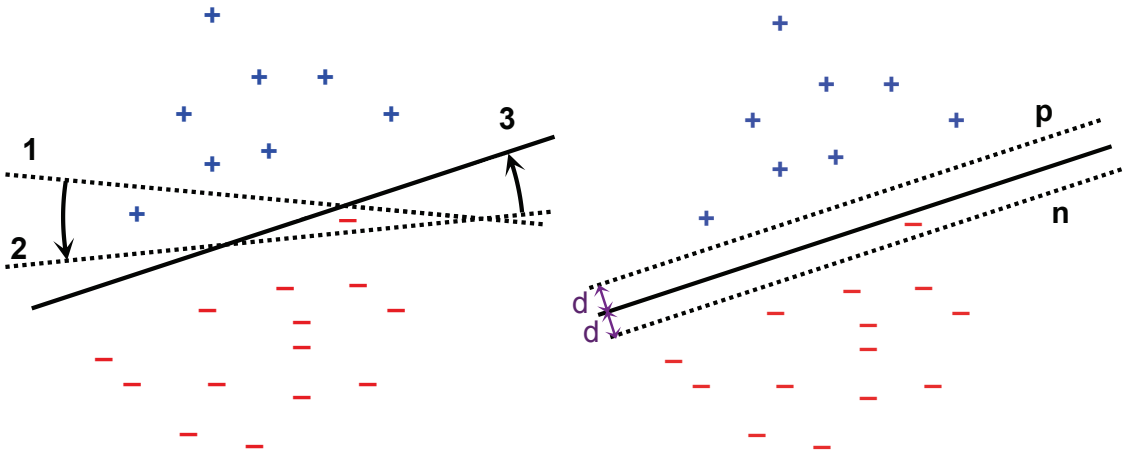

Figure 4.5: (Left) Online training of a linear classifier. The orientation of a hyperplane is adjusted gradually to separate two classes of data. (Right) Linear classifier with margins to guarantee better generalization performance.

In the neuroscience community, activity-based synaptic plasticity is believed to be the basic phenomenon in the learning process [172]. Based on this observation, Hebb's postulate is proposed and widely accepted. The postulate claims that neurons that fire together wire together [173]. More specifically, if the firing activity of a neuron A tends to induce/inhibit spikes of another neuron B, the synapse sending signals from A to B tends to be potentiated/depressed.

The basic plasticity rule that follows Hebb's postulate is

$$\tau_l \frac{dw}{dt} = u_{pre} u_{post}, \tag{4.4}$$

where w is the synaptic weight, u_{pre} and u_{post} represent the activities of the presynaptic neuron and postsynaptic neuron, respectively. τ_l is a time constant associated

with the learning speed. To include synaptic depression into the learning rule, different forms of covariance rules are proposed as summarized in [172].

4.2.2 Instability and Saturation

The Hebbian learning rules in the previous subsection capture the most fundamental behavior of synapse and are of great theoretical importance. However, there are practical issues mainly because of instability, that is, synaptic weight may exhibit uncontrolled growth according to these rules. The solution to this problem is imposing upper/lower limits to the synaptic weight. However, with these limits these rules suffer from a new problem of synaptic saturation, in which all synaptic weights can be driven to the upper/lower limit such that the synapses stop learning and the network loses its capability to discriminate different input patterns. In the previous example for illustration, this case corresponds to the situation that all data points, regardless of its class label, move the hyperplane to the same direction such that the hyperplane loses its classification capability.

To solve the synaptic saturation problem, quite a few research works have been conducted. The BCM rule [174] uses a sliding threshold to modulate synaptic plasticity. Other rules are also proposed by modifying Hebb rules to include various forms of synaptic normalization or synaptic competition, e.g. Oja rule [172]. Theoretically, they all successfully solve the saturation problem, thus show the potential to be used in real applications. On the other hand, these rules also have obvious drawbacks as candidates for hardware implementation. First of all, the complicated computation involved in synaptic dynamics requires a great deal of hardware resource. In addition, synaptic normalization/competition based rules are not local, that is, the learning dynamics of a synapse does not only depend on the firing activity of the presynaptic and postsynaptic neurons. The additional communications between different parts

of the neural network and the associated computations are not hardware-friendly and limit the network scalability.

As stated above, the best online learning rule for LSM is expected to be free from the saturation problem and preferably simpler than existing rules such as BCM rule, Oja rule, and so on.

4.3 The Proposed Learning Rule

Previous sections show that a well designed learning rule for the LSM is expected to be local and free from synaptic saturation. In addition, as a common issue in machine learning [175], overfitting problem shall also be considered in the design of the learning algorithm. In this section, we first introduce an abstract learning rule that meet these requirements. A similar abstract rule for binary synapses is used for two-layer feed-forward networks in [54]. Then we implement the abstract learning rule for the proposed LSM. Comparing with [54], our way of implementation is greatly simplified, thereby more hardware-friendly.

4.3.1 Abstract Learning Rule

In this subsection, we introduce the abstract local learning rule that is free from saturation issue. Then this rule is further improved on its generalization performance.

As illustrated in Fig. 4.5(Left), the learning process is driven by incorrectly classified data points. In a neural network, if an inactive (active) neuron is desired to be active (inactive), synapses providing inputs tend to be potentiated (depressed)

$$\begin{cases} \text{potentiation of } w_i & \text{if } u_i = A \ \& \ u_r = I \ \& \ u_d = A \\ \text{depression of } w_i & \text{if } u_i = A \ \& \ u_r = A \ \& \ u_d = I, \end{cases} \tag{4.5}$$

where u_r and u_d represents the real and desired activity of the readout neuron,

respectively, u_i is the activity of the i^{th} presynaptic neuron, and w_i is the corresponding synaptic weight, A and I represent that the neuron is active and inactive, respectively.

Since learning requires presynaptic neural activity, presynaptic spikes are used to induce learning in the rule. Quantitatively, the value of w_i after a presynaptic spike may become

$$\begin{cases} w_i + \Delta w & \text{if } u_r < u_\theta \ \& \ u_d > u_\theta \\ w_i - \Delta w & \text{if } u_r > u_\theta \ \& \ u_d < u_\theta, \end{cases} \tag{4.6}$$

where u_r and u_d are quantitative values of neuron activity, u_θ is a threshold value to determine whether the postsynaptic neuron is active or not, i.e., whether the data represented by all presynaptic neurons is classified to one class or the other. It is clear that for any data point on the hyperplane of a linear classifier, $u_\theta = u_d = \sum_i w_i \cdot u_i$.

For better generalization performance, a margin is introduced around the hyperplane for the linear classifier, as illustrated in Fig. 4.5(Right), This way, the learning is not only driven by incorrectly classified data, but also driven by data that are correctly classified but fall into the margins, i.e. between the two hyperplanes p and n. For any data on p (or n), $u_\theta + \Delta u = u_d = \sum_i w_i \cdot u_i$ (or $u_\theta - \Delta u = u_d = \sum_i w_i \cdot u_i$), where Δu corresponds to d in Fig. 4.5(Right).

Since slow learning rate is usually used in neural networks for good performance [176][54], to some extent smaller learning steps are preferred in the training. However, the resolution of discrete synaptic weights may be limited since we target at hardware implementation. So we modify synaptic weights stochastically to further reduce the learning rate.

With the above improvements, the further modified learning rule states that the

value of w_i following a spike from the i^{th} presynaptic neuron may become

$$\begin{cases} w_i + \Delta w \text{ with } p_+ \text{ if } u_r < u_\theta + \Delta u \ \& \ u_d > u_\theta \\ w_i - \Delta w \text{ with } p_- \text{ if } u_r > u_\theta - \Delta u \ \& \ u_d < u_\theta, \end{cases} \tag{4.7}$$

where Δw is the potentiation/depression granularity, p_+ and p_- are the potentiation and depression probability, Δu represents the margin described above.

In a biological neuron, the internal calcium concentration is a good indicator of its instantaneous activity in a time window as suggested in [54]. Therefore, by replacing neuron activity u by calcium concentration c, the equation (4.7) becomes

$$\begin{cases} w_i + \Delta w \text{ with } p_+ & \text{if } c_r < c_\theta + \Delta c \ \& \ c_d > c_\theta \\ w_i - \Delta w \text{ with } p_- & \text{if } c_r > c_\theta - \Delta c \ \& \ c_d < c_\theta, \end{cases} \tag{4.8}$$

Note that this rule is mathematically similar to a more abstract rule for binary synapses in [54], which is proved to work for linearly separable patterns with a finite number of iterations of training [177][178].

4.3.2 Learning in Spiking Neural Networks

The previous subsection introduces the abstract learning rule used in the LSM. In this subsection, an implementation of the abstract rule in spiking neural networks is described. Our implementation is much simpler than that in [54], thereby more hardware-friendly.

To implement the learning rule to mimic the behavior of a biological synapse, the plasticity shall follow the Hebb's postulate that synaptic potentiation/depression is possible only if the presynaptic neuron tends to induce/inhibit firing of the post-synaptic neuron. From this perspective, the learning rule is expected to take the

following form: The learning of the synapse following a presynaptic spike is

$$\begin{cases} w_i \to w_i + \Delta w \text{ with prob. } p_+ & \text{if } c_r > c'_\theta \\ w_i \to w_i - \Delta w \text{ with prob. } p_- & \text{if } c_r < c'_\theta, \end{cases} \tag{4.9}$$

where c'_θ is another threshold of calcium concentration to determine if synaptic potentiation or depression is possible.

To take advantage of the merits of both equations (4.8) and (4.9), we combine these two principles. To be intuitive, we first illustrate these two rules in Fig. 4.6, where each subfigure is divided into several regions that show the learning activity of a synapse under different combinations of c_r and c_d.

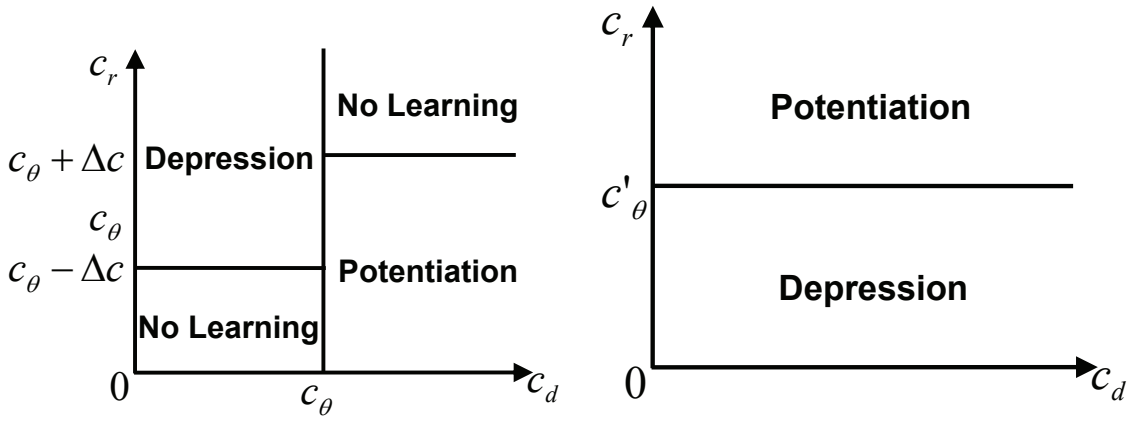

Figure 4.6: (Left) This subfigure illustrates the abstract learning rule of (4.8), where the difference between the real neural activity and expected activity determines whether the learning is potentiation or depression. (Right) This subfigure illustrates Hebbian learning of (4.9), where a presynaptic spike together with an active (inactive) postsynaptic neuron leads to synaptic potentiation (depression).

The subfigure on the left illustrates the abstract learning rule in (4.8), where the

direction of learning is determined by the difference between the real activity of the postsynaptic neuron and its desired activity. The subfigure on the right illustrates equation (4.9), where the direction of the learning is determined by the activity of the postsynaptic neuron.

We choose c'_θ to be the same as c_θ when combining these two principles. Since in a biological neuron, synaptic learning is determined only by the real activity of presynaptic and postsynaptic neurons but not the desired activity, learning is expected to be independent of c_d, thereby only depends on c_r. To avoid synaptic saturation, we keep the "stop learning" regions for $c_r > c_\theta + \Delta c$ and $c_r < c_\theta - \Delta c$. In the region $c_\theta - \Delta c < c_r < c_\theta + \Delta c$, Hebbian learning is employed such that potentiation happens for $c_\theta < c_r < c_\theta + \Delta c$ and depression happens for $c_\theta - \Delta c < c_r < c_\theta$. Thus the combined learning rule states that the learning of a synapse following a presynaptic spike is

$$\begin{cases} w_i \to w_i + \Delta w \text{ with prob. } p_+ \text{ if } c_\theta < c_r < c_\theta + \Delta c \\ w_i \to w_i - \Delta w \text{ with prob. } p_- \text{ if } c_\theta > c_r > c_\theta - \Delta c. \end{cases} \tag{4.10}$$

Comparing the combined learning (4.10) with (4.8), it is clear that these two descriptions disagree in the region $\{(c_d, c_r)|c_d < c_\theta, c_r > c_\theta\}$ and the region $\{(c_d, c_r)|c_d > c_\theta, c_r < c_\theta\}$, where depression and potentiation occur, respectively, according to (4.8). To this end, we employ a teacher signal to alter the real activity of the postsynaptic neuron to induce the desired learning of the synapse. Take the region $\{(c_d, c_r)|c_d > c_\theta, c_r < c_\theta\}$ as an example. Since the desired learning according to (4.8) is potentiation, while depression occurs according to (4.10), the teacher signal moves a point in $\{(c_d, c_r)|c_d > c_\theta, c_r < c_\theta\}$ to $\{(c_d, c_r)|c_d > c_\theta, c_\theta < c_r < c_\theta + \Delta c\}$, where potentiation occurs according to (4.10). The teacher signal under various scenarios

is illustrated by arrows in Fig. 4.7. To summarize, the application of the teacher

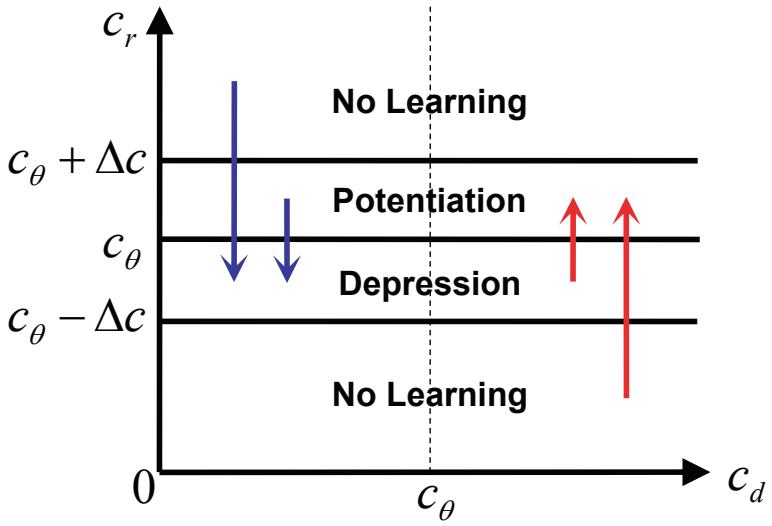

Figure 4.7: Illustration of our proposed learning rule. The eight regions in the diagram shows how different combinations of c_d and c_r of the postsynaptic neuron determine the plasticity of a synapse. The four arrows show how the teacher signal may drive the output neuron from one region to another region.

signal modulates the activity of the postsynaptic neuron such that the its calcium concentration is driven to

$$\begin{cases} [c_\theta, c_\theta + \Delta c] & \text{if } c'_r < c_\theta \ \& \ c_d > c_\theta \\ [c_\theta - \Delta c, c_\theta] & \text{if } c'_r > c_\theta \ \& \ c_d < c_\theta, \end{cases} \tag{4.11}$$

where c'_r is the internal calcium concentration of the postsynaptic neuron without applying the teacher signal.

The teacher signal of a postsynaptic neuron is implemented as a large constant current. In the case of a positive teacher signal, which increases the activity of

the corresponding postsynaptic neuron, a large constant current is injected to the target neuron if $c_d > c_\theta$ and $c_r < c_\theta + \delta$ ($c_d < c_\theta$ and $c_r > c_\theta - \delta$), where δ is a value in $[0, \Delta c]$ such that the teacher signal moves the (c_d, c_r) deep into the region $\{(c_d, c_r) | c_d > c_\theta, c_\theta < c_r < c_\theta + \Delta c\}$, where potentiation is induced according to the equation (4.10). Note that the teacher signal is stopped when $c_r > c_\theta + \delta$. The quantitative model of the teacher signal is introduced in the next subsection.

The existence of the teacher signal in our proposed learning rule requires a supervised learning scheme in the liquid state machine. We label the readout neurons by their corresponding class label R_i ($i = 1, 2, 3, \dots$). The goal of training is that when an input signal in class i is applied to the system shown in Fig. 4.4, R_i is the most active readout neuron, i.e. R_i emits more spikes than all other readout neurons. Thus, during training, a positive teacher signal is applied to R_i while negative teacher signals are applied to other readout neurons.

To summarize, the rule described by equations (4.10) and (4.11) is free from saturation issue, Hebbian and local, i.e. the synaptic plasticity only depends on the activities of it presynaptic neuron and postsynaptic neuron. In addition, the generalization performance is considered by using the margin Δc.

4.3.3 Models for Implementing the Proposed Learning Rule

In the liquid state machine, we adopt the widely used leaky-integrate-and-fire (LIF) model for each reservoir and readout neuron. The dynamics of the membrane potential of a neuron is described by the following differential equation

$$\frac{dv_m}{dt} = -\frac{v_m}{\tau_m} + \sum_i \sum_j w_{mi} \cdot s(t - t_{ij} - d_i), \tag{4.12}$$

where v_m and τ are the membrane potential of the neuron and the time constant of its first-order dynamics, respectively. i and j are indices of presynaptic neurons and spikes from them, respectively. More specifically, w_{mi} is the weight of the synapse that transmits the spiking signal from the i^{th} presynaptic neuron. t_{ij} is the time of the j^{th} spike emitted from the i^{th} presynaptic neuron. And d_{ij} is the corresponding synaptic delay. $s(\cdot)$ is the dynamic response of a synapse to an input spike.

Since the teacher signal is modeled as a strong current to the neuron to increase or reduce the activity of a specific neuron, the neuron model is modified to

$$\frac{dv_m}{dt} = -\frac{v_m}{\tau_m} + \sum_i \sum_j w_{mi} \cdot s(t - t_{ij} - d_i) + i_t(c), \tag{4.13}$$

where $i_t(\cdot)$ is the current modeling the teacher signal, which is a function of calcium concentration c.

In the digitized neural network, the above equation is rewritten as

$$\begin{aligned} V_m^n =& V_m^{n-1} - \frac{V_m^{n-1}}{\boldsymbol{\tau}_m} \\ &+ \sum_i \sum_j W_{mi} \cdot S(T^n, T_{i,j} + D_i) + I_t^{n-1}, \end{aligned} \tag{4.14}$$

where capitalized letters represent the digitized variables. And superscripts of V, T and I_t are the indices of time steps. The neuron fires and resets the membrane voltage to rest potential V_{rest} if the membrane voltage reaches or exceeds V_{th}. Following each spike, there is an absolute refractory period τ_{refrac}, within which the neuron cannot fire.

The dynamics of the calcium concentration is

$$\frac{dc}{dt} = -\frac{c}{\tau_c} + \sum_i \delta(t - t_i - d_i) \tag{4.15}$$

where τ_c is the time constant for the first-order dynamics of calcium concentration, respectively. i is the index of spikes emitted from the neuron itself. From this differential equation, it is clear that the internal calcium concentration level of the neuron increases with its firing frequency. The digitized dynamics is

$$C^n = C^{n-1} - \frac{C^{n-1}}{\tau_c} + \sum_i \delta_{T^n, T_i + D_i} \tag{4.16}$$

where the same conventions for capitalization and superscripts as mentioned above are used, $\delta_{i,j}$ is the Kronecker delta, of which the value is 1 if $i = j$, and 0 if $i \neq j$.

To digitize the synaptic weight, let W_{max} and W_{min} denote the upper and lower limits of weight values. We also denote the quantization step size $\frac{W_{max} - W_{min}}{2^b - 1}$ by ΔW, where b is the number of bits used to represent the weight.

In the digitized learning rule, the update of synaptic weight is initialized by a spike emitted by the presynaptic neuron. We assume a presynaptic spike at time T_n, then synaptic potentiation happens, i.e. $W^n = W^{n-1} + \Delta W$, with a probability p_+ if the following conditions are satisfied

$$\begin{cases} C_\theta < C^{n-1} < C_\theta + \Delta C \\ W^{n-1} < W_{max} \end{cases} \tag{4.17}$$

where C_θ is the threshold of the calcium concentration that allows synaptic potentiation. Similarly, synaptic depression happens, i.e. $W^n = W^{n-1} - \Delta W$, with probability p_- when the following requirements are met

$$\begin{cases} C_\theta - \Delta C < C^{n-1} < C_\theta \\ W^{n-1} > W_{min}. \end{cases} \tag{4.18}$$

4.4 Influence of Synaptic Model on LSM Performance and Implication on Hardware Implementation

Theoretical studies show that the computational performance of a recurrent neural network depends critically on its dynamics [179][180][181]. For superior computational performance, the reservoir in a liquid state machine shall operate at the edge of chaos, where the system has long transients (fading memory). This is in sharp contrast to an ordered system or chaotic system, where the state evolves to a simple steady state or stable limit cycle (a state cycle relatively short in time) in the former, while exhibits chaotic behavior (long state cycle for an discrete system with finite number of possible states) in the latter [180]. This implies that longer fading memory is of great importance in increasing the computational performance of the LSM in terms of boosting its separation property [55][180]. Since the fading memory arises from recurrent loops formed by synapses. The model of synapse may have great impact on the length of fading memory, and consequently the LSM performance. However, this problem has not been studied. In addition, for the hardware-oriented LSM, evaluating the complexity of different synaptic models and their impacts on the LSM performance at the same time provides design guidance to optimize the hardware cost. Therefore, in this section, we empirically study the influence of synaptic models with different complexity on the length of the fading memory and the performance of the LSM.

4.4.1 Impact of Synaptic Models on LSM Performance

Since our liquid state machine is hardware oriented, simple synaptic models are preferred over complicated ones. Therefore, we first test the performance of the LSM with the simplest static synaptic model, where the synaptic response to an incoming spike is also static, i.e. the temporal transfer function is Dirac delta function $\delta(\cdot)$, as

used in many research works [11][10][61]. By using static response, the LIF model becomes

$$\frac{dv_m}{dt} = -\frac{v_m}{\tau_m} + \sum_i \sum_j w_{mi} \cdot \delta(t - t_{ij} - d_{ij}). \tag{4.19}$$

We test this model by training the liquid state machine with settings similar to [65] except that we use the spike-based learning rule. More detailed information of the LSM including settings and parameter values are introduced in Section. 4.6. Here we only study the influence of synaptic models on the network performance. 500 utterances of digits $0-9$ are used in 5-fold cross validation to test the LSM. We train the LSM for 500 iterations and test its classification rate on the fly. The results are plotted in the top left panel of Fig. 4.8. We observe that the recognition rate reaches about 88.85%, i.e. 88.85% of the training samples are correctly recognized by the liquid state machine. Note that the best performance is an averaged value over a window of 20 epoches to reduce the influence of performance fluctuation. The same method is used hereinafter.

For comparison, we also test the LSM performance with dynamical behavior involved in the synaptic model, where the synapse has first order dynamical response to an presynaptic spike. The corresponding LIF model is described as follows

$$\begin{aligned} \frac{dv_m}{dt} = & -\frac{v_m}{\tau_m} \\ & + \sum_{i,j} w_{mi} \cdot \frac{1}{\tau^s} e^{-\frac{t-t_{ij}-d_{ij}}{\tau^s}} \cdot H(t - t_{ij} - d_{ij}), \end{aligned} \tag{4.20}$$

where $\tau^s = 4\ ms$ for all synapses is the time constant of the first order response. $H(\cdot)$ is the Heaviside step function. And $\frac{1}{\tau^s}$ normalizes the first order transfer function. Without modifying any other models or parameters, solely adding dynamical response in the synaptic model improves the recognition performance to 90.49%, as

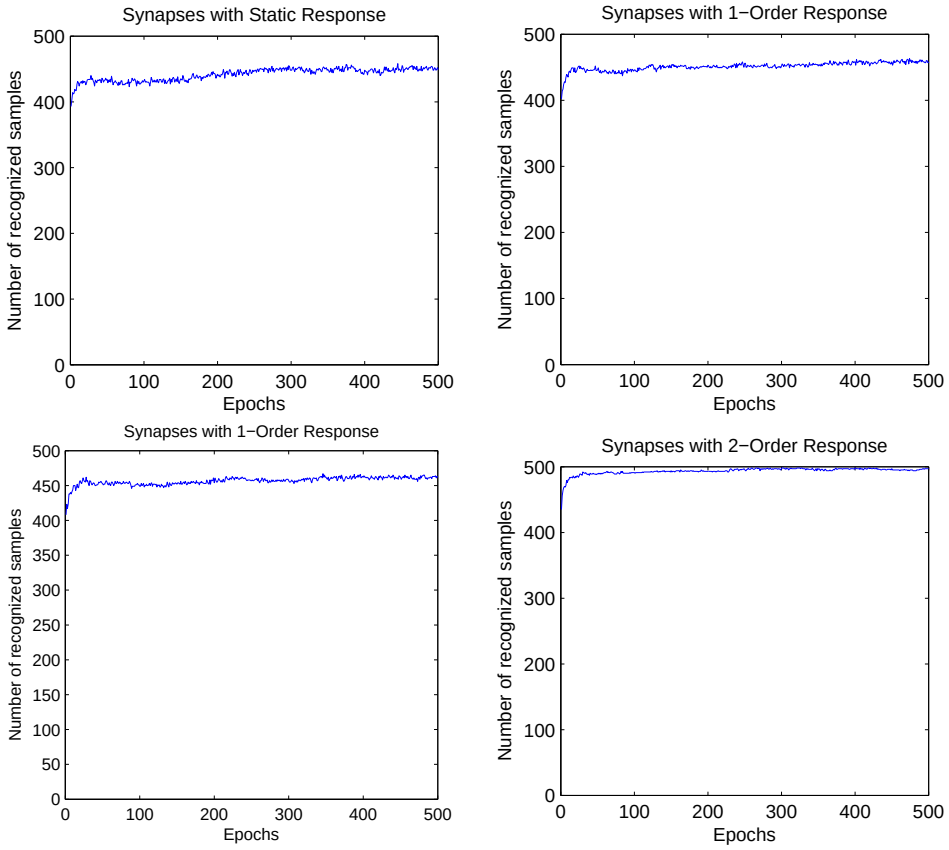

Figure 4.8: (Top Left) Performance of the liquid state machine using synapses with static response. (Top Right) Performance of the liquid state machine using synaptic model using first order response with time constant 4 ms. (Bottom Left) Performance of the liquid state machine using synaptic model with first order response with time constant 8 ms. (Bottom Right) Performance of the liquid state machine using synaptic model with second-order response.

shown in the top right panel of Fig. 4.8. We also test the performance using time constant 8 ms. As shown in the bottom left panel of Fig. 4.8, the recognition rate is 90.73%, accordingly.

We further test the network performance with second order transfer function

$$\begin{aligned}\frac{dv_m}{dt} = & -\frac{v_m}{\tau_m} \\ & + \sum_i \sum_j \frac{w_{mi} \cdot e^{-\frac{t-t_{ij}-d_{ij}}{\tau_1^s}} \cdot H(t-t_{ij}-d_{ij})}{\tau_1^s - \tau_2^s} \\ & - \sum_i \sum_j \frac{w_{mi} \cdot e^{-\frac{t-t_{ij}-d_{ij}}{\tau_2^s}} \cdot H(t-t_{ij}-d_{ij})}{\tau_1^s - \tau_2^s}\end{aligned} \tag{4.21}$$

where τ_1^s and τ_2^s are the time constants of the second order response. And $\frac{1}{\tau_1^s - \tau_2^s}$ normalizes the second order dynamical response function. For excitatory synapses, τ_1^s and τ_2^s are 4 ms and 8 ms, respectively. For inhibitory synapses, τ_1^s and τ_2^s are 4 ms and 2 ms, respectively. Results are plotted in the bottom right panel of Fig. 4.8. The recognition rate is further improved to 99.09%.

The above results show that by using synaptic models with higher order dynamics, the LSM performance is improved significantly. A more complicated model is used in [65], where short term plasticity (STP) is captured, that is, synaptic weight has dynamical behavior in response to presynaptic and postsynaptic neural activities. The comparison of the LSM performance in [65] and that of our LSM using synapses with only second order response is in the Section 4.6.4. This comparison shows that the performance of our LSM using synapses without STP does not degrade. This is probably because the reservoirs used in these LSMs are not large enough to take advantage of the STP in the computation [182][183]. By avoiding the STP, the associated state variables and computations are eliminated from the synaptic model. Therefore, the hardware complexity and cost are also reduced. For sake of simplicity, we stop trying synaptic models with higher order responses. And we adopt the synaptic model using second order dynamical response without STP in

our liquid state machine to obtain good performance with simple hardware-friendly implementation.

4.4.2 Fading Memory - Linking Synaptic Model to LSM Performance

The results in the previous subsection show the performance improvement of neural networks by using dynamic synaptic responses instead of static responses, especially for the second-order dynamic response. The reason for this big difference lies in the reservoir, where recurrent network is used to memorize temporal patterns by fading memory.

The impacts of synaptic models on the fading memory are shown in Fig. 4.9-4.12. In each figure, the 3 plots show the activity of the reservoir as a response to input spike trains ended at 23 ms. Each plot shows the total number of spikes in the reservoir with each 1 ms bin. The plot on the top of each figure uses the same input spike trains. The frequency of spike trains are 3 times and 10 times higher for plots in the middle and bottom of all figures. As shown in Fig. 4.9, responses of the reservoir vanish immediately after the end of spike trains. This indicates little temporal memory in the reservoir. Only the strength of the reservoir activity increases with input frequencies. Using the first order synaptic model, Fig. 4.10 shows that the reservoir exhibits temporal memory, that is, the reservoir activity persists for a period of time after the end of input spike trains. It is straightforward to see that by increasing the time constant of synaptic response, the temporal memory in Fig. 4.11 lasts longer. And the network performance also increases accordingly. However, larger time constants only increase the memory of strong input signals significantly, as shown in the bottom plots. By using the second order synaptic response, Fig. 4.12 shows that the memory of weaker input signal in the middle of the plot also increases significantly, as shown in middle plots.

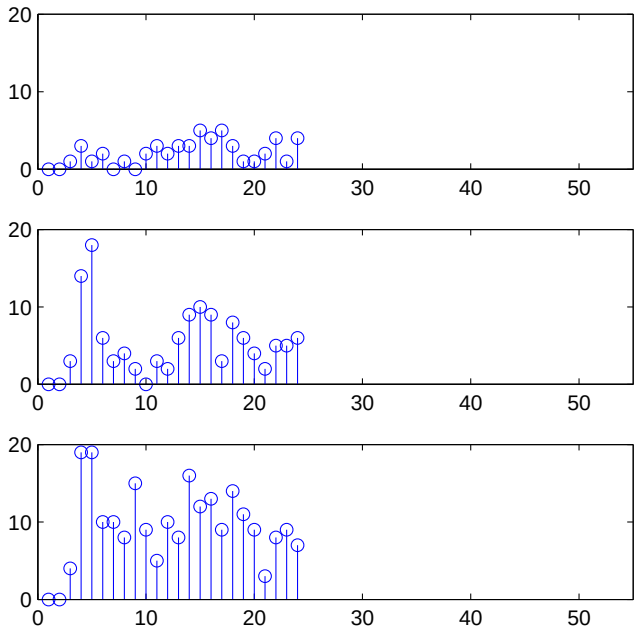

Figure 4.9: Responses of the reservoir using static synaptic response. With input spike trains end at 23 ms, the reservoir shows little temporal memory. The top, middle, bottom plots use input spike trains with ascending frequency. These plots show that only the magnitude of reservoir response increases with input frequencies.

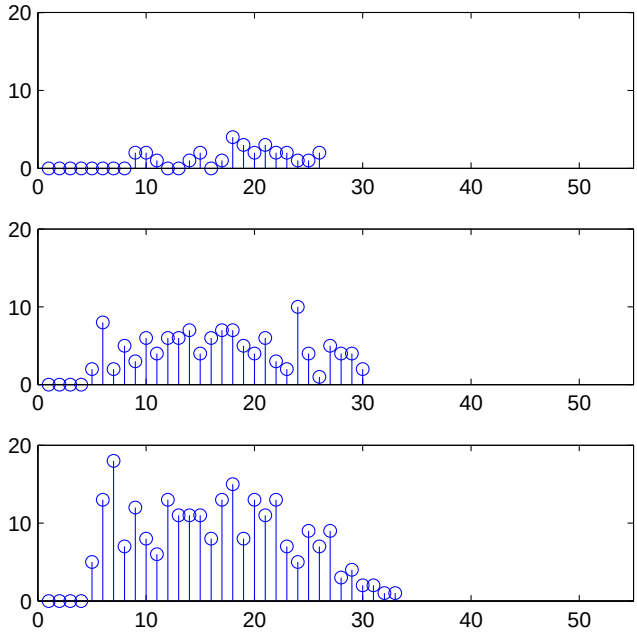

Figure 4.10: Responses of the reservoir using first-order synaptic response with time constant 4 ms. Comparing to the results for static synaptic response, this reservoir exhibits temporal memory. The duration of the responses also increases with the frequencies of input spike trains.

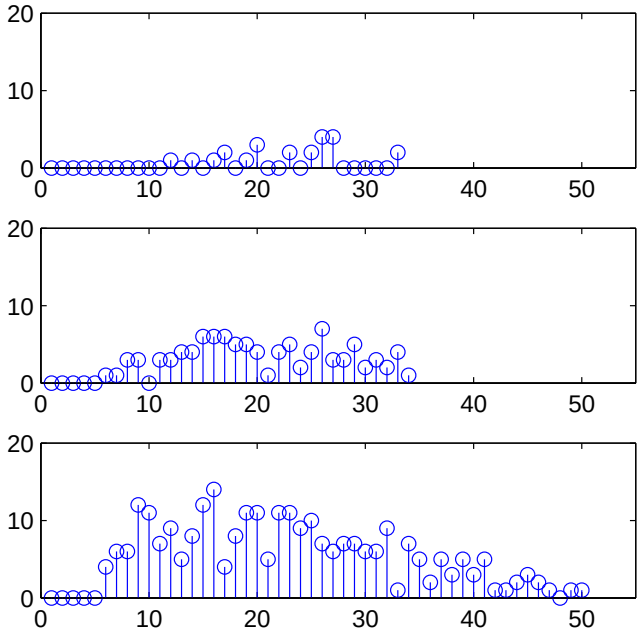

Figure 4.11: Responses of the reservoir using first-order synaptic response with time constant 8 ms. Comparing to the results for first-order synaptic response with time constant 4 ms, the temporal memory is longer. Specially, the bottom plot shows much longer memory than the middle plot.

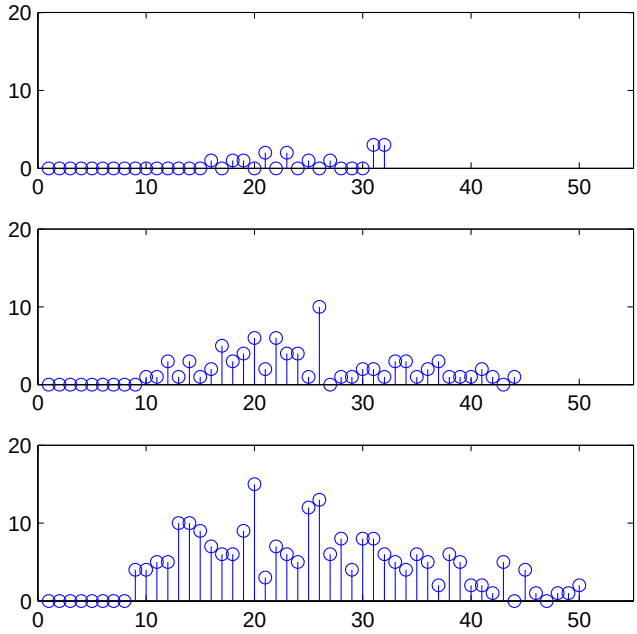

Figure 4.12: Responses of the reservoir using second-order synaptic response. The temporal memory is longer than that for first-order synaptic response. Specially, the middle plot also shows longer temporal memory comparing to the response of the LSMs with first-order synaptic models.

The phenomena in these figures show that higher order synaptic dynamics helps the reservoir to produce longer fading memory, which further helps produce richer dynamics and boosts the recognition performance. By using dynamic responses, the influence of a spike to the postsynaptic neuron is spread out in time. The long lasting effect helps asynchronously emitted spikes to interact with each other in the reservoir. Together with the recurrent structure of the network, these interactions create rich dynamics that forms short-term memory and enhances the readout neurons' ability to recognize the given pattern. Comparing with the first-order dynamics of a exponential form

$$\frac{1}{\tau^s} e^{-\frac{t-t_{ij}-d_{ij}}{\tau^s}} \cdot H(t - t_{ij} - d_{ij}),$$

where the peak value of the response is located at the arrival of the spike and the response decreases exponentially subsequently, the second-order dynamics

$$\frac{\left(e^{-\frac{t-t_{ij}-d_{ij}}{\tau_1^s}} - e^{-\frac{t-t_{ij}-d_{ij}}{\tau_2^s}}\right) \cdot H(t - t_{ij} - d_{ij})}{\tau_1^s - \tau_2^s}$$

ramps up to the peak value then decreases over time. Thus, the second-order response spreads out even more, leading to longer interactions between neurons. This may be the part of the reason that the second-order response produces better results. For the second-order model, the time constants for excitatory responses and inhibitory responses are different. This can also produce richer dynamics which helps to boost the network performance.

4.5 Hardware-Friendly Efficiency Optimization

4.5.1 Division Simplification

The above text shows that computations that have to be implemented include the neuron model and synaptic model described in (4.16) and (4.21), and learning rules described in (4.17) and (4.18). While there are well developed VLSI modules for efficient addition and comparison operations [75] and random number generation [184], division and evaluation of exponential terms are expensive to implement in hardware. In this subsection, we simplify the division operations in (4.16) and (4.21).

In (4.16) and (4.21), division operations are used to calculate the spontaneous exponential decrease of calcium concentration and membrane potential. While there is previous work using look-up tables (LUT) to implement exponential decrease of digitized variables [61], the precision of computations is limited by the affordable LUT area.

In the neuron models, since both the time constants of calcium concentrations and membrane voltages are constant numbers throughout the entire neural network, a general purpose division module is not necessary. Therefore, we use customized division modules for specific denominator values to perform division operations. To divide a binary number n by another number m, a computationally rather simple case is when $m = 2^k$ for some non-negative integer k. In this case, division can be done by simply right shifting the binary number n by k bits. Since bit shift is much cheaper to implement on VLSI [75], we choose time constant values of calcium concentration and membrane voltage to be binary (i.e. 2^k for some integer value of k) such that bit shift modules can be used to perform customized division operations.

4.5.2 Synaptic Model Simplification

As a neural network, the liquid state machine is comprised of neurons and synapses. A closer look at the LSM reveals that the number of synapses is usually much larger than that of neurons. Take the LSM for speech recognition as an examples, to fully connect hundreds of reservoir neurons and 10 output neurons requires thousands of plastic synapses already, not to mention the synapses needed for connecting the reservoir neurons. Therefore, reducing the complexity of synaptic models is of great importance for reducing the hardware cost of network implementation.

To reduce the complexity, we first take a closer look at the models used in the network. Since (4.21) describes the dynamics of both the neuron and incident synapses, the model can be rewritten as

$$\frac{dv_m}{dt} = -\frac{v_m}{\tau_m} + \sum_i I_i \tag{4.22}$$

$$\begin{aligned} I_i = &\sum_j w_{mi} \cdot \frac{1}{\tau_1^s - \tau_2^s} e^{-\frac{t-t_{ij}-d_{ij}}{\tau_1^s}} \cdot H(t - t_{ij} - d_{ij}) \\ &- \sum_j w_{mi} \cdot \frac{1}{\tau_1^s - \tau_2^s} e^{-\frac{t-t_{ij}-d_{ij}}{\tau_2^s}} \cdot H(t - t_{ij} - d_{ij}), \end{aligned} \tag{4.23}$$

where (4.22) and (4.23) are the models of the membrane voltage and the i^{th} incident synapse, respectively. I_i is the current injected to the neuron from the i^{th} incident synapse. Let the values of all t_{ij} to be positive, $e^{-\frac{t-t_{ij}-d_{ij}}{\tau_1^s}} \cdot H(t - t_{ij} - d_{ij})$ and $e^{-\frac{t-t_{ij}-d_{ij}}{\tau_2^s}} \cdot H(t - t_{ij} - d_{ij})$ are respectively the solutions of the following differential equations

$$\begin{cases} \frac{dx}{dt} = -\frac{1}{\tau_1} x + \delta(t - t_{ij} - d_{ij}) \\ x(t)|_{t=0} = 0 \end{cases} \tag{4.24}$$

and

$$\begin{cases} \frac{dx}{dt} = -\frac{1}{\tau_2}x + \delta(t - t_{ij} - d_{ij}) \\ x(t)|_{t=0} = 0. \end{cases} \tag{4.25}$$

For these linear ordinary differential equations, it is clear that $w_{mi}\sum_j \cdot e^{-\frac{t-t_{ij}-d_{ij}}{\tau_1^s}} \cdot H(t - t_{ij} - d_{ij})$ and $w_{mi}\sum_j \cdot e^{-\frac{t-t_{ij}-d_{ij}}{\tau_2^s}} \cdot H(t - t_{ij} - d_{ij})$ are respectively the solutions of

$$\begin{cases} \frac{dx}{dt} = -\frac{1}{\tau_1}x + w_{mi}\sum_j \cdot \delta(t - t_{ij} - d_{ij}) \\ x(t)|_{t=0} = 0 \end{cases} \tag{4.26}$$

and

$$\begin{cases} \frac{dx}{dt} = -\frac{1}{\tau_2}x + w_{mi}\sum_j \cdot \delta(t - t_{ij} - d_{ij}) \\ x(t)|_{t=0} = 0. \end{cases} \tag{4.27}$$

This way, each synapse is modeled by two state variables that can be calculated by the method introduced in the previous subsection. However, the cost of implementing these second-order synapse is still considerable since the number of synapse is significantly larger than that of neurons.

To this end, we further explore the linearity of synaptic models to merge the responses of all synapses incident to a neuron [90][185], as illustrated in Fig. 4.13. Considering that all synaptic responses are described by the same linear ordinary differential equations with different inputs, by superposition the terms $\sum_i w_{mi}\sum_j \cdot e^{-\frac{t-t_{ij}-d_{ij}}{\tau_1^s}} \cdot H(t - t_{ij} - d_{ij}) = \sum_{i,j} w_{mi} \cdot e^{-\frac{t-t_{ij}-d_{ij}}{\tau_1^s}} \cdot H(t - t_{ij} - d_{ij})$ and $\sum_i w_{mi}\sum_j \cdot e^{-\frac{t-t_{ij}-d_{ij}}{\tau_2^s}} \cdot H(t - t_{ij} - d_{ij}) = \sum_{i,j} w_{mi} \cdot e^{-\frac{t-t_{ij}-d_{ij}}{\tau_2^s}} \cdot H(t - t_{ij} - d_{ij})$ are respectively the solutions of

$$\begin{cases} \frac{dx}{dt} = -\frac{1}{\tau_1}x + \sum_i w_{mi}\sum_j \cdot \delta(t - t_{ij} - d_{ij}) \\ x(t)|_{t=0} = 0 \end{cases} \tag{4.28}$$

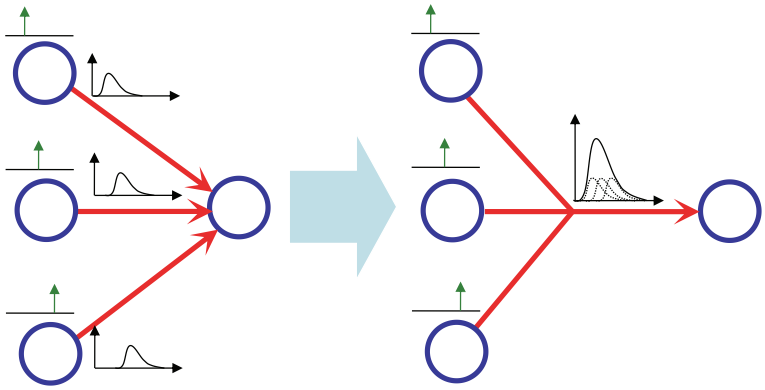

Figure 4.13: Illustration of linear merging of synaptic responses. Responses of three synapses incident to a neuron is linearly merged to form the response of one synapse receiving all spikes from the three presynaptic neurons.

and

$$\begin{cases} \frac{dx}{dt} = -\frac{1}{\tau_2}x + \sum_i w_{mi} \sum_j \cdot \delta(t - t_{ij} - d_{ij}) \\ x(t)|_{t=0} = 0. \end{cases} \tag{4.29}$$

From (4.23), we have

$$\begin{aligned} & \sum_i I_i \\ = & \sum_{i,j} w_{mi} \frac{e^{-\frac{t-t_{ij}-d_{ij}}{\tau_1^s}} - e^{-\frac{t-t_{ij}-d_{ij}}{\tau_2^s}}}{\tau_1^s - \tau_2^s} H(t - t_{ij} - d_{ij}) \\ = & \sum_{i,j} \frac{w_{mi}}{\tau_1^s - \tau_2^s} \cdot e^{-\frac{t-t_{ij}-d_{ij}}{\tau_1^s}} \cdot H(t - t_{ij} - d_{ij}) \\ & - \sum_{i,j} \frac{w_{mi}}{\tau_1^s - \tau_2^s} \cdot e^{-\frac{t-t_{ij}-d_{ij}}{\tau_2^s}} \cdot H(t - t_{ij} - d_{ij}), \end{aligned} \tag{4.30}$$

which states that $\sum_i I_i$ is the summation of the solutions of (4.28) and (4.29). Thus, $\sum_i I_i$ can be computed by solving the two state variables in (4.28) and (4.29).

It is clear that instead of having two state variables for each synapse, the state

variables of all synapses incident to the same postsynaptic neuron are merged into two. In other words, the number of state variables describing the dynamic behaviors of synapses is reduced from "two per synapse" to "two per postsynaptic neuron". As estimated at the beginning of this subsection, an LSM is comprised of hundreds of neurons and at least thousands of synapses. Thus, to merge these synaptic responses, the number of state variables describing all synaptic dynamics is reduced by at least one order of magnitude.

(4.30) shows that the two state variables have to be divided by $\tau_1^s - \tau_2^s$. To simplify the division operation, we use the approach introduced in the previous subsection and choose parameters such that τ_1^s, τ_2^s and $\tau_1^s - \tau_2^s$ can all be expressed in the form of 2^k.

4.5.3 Precision of Synaptic Weight

Due to the large number of synapses, the number of bits used to represent the synaptic weight also has a considerable influence on the total hardware resources needed to implement the LSM. Using less bits reduces the hardware cost but limits the precision of synaptic weights. Intuitively, this limits the precision of adjusting the location and orientation of the hyperplane in the linear classifier, thereby may potentially degrade the LSM performance. The trade-off between the number of bits and network performance is studied in the next section.

4.6 Experiments

We first introduce the settings of our experiments. To reduce hardware resources of potential implementations, we study the impact of bitwidth of synaptic weights and reservoir size on the network performance. Finally, our results are compared against existing works.

4.6.1 Experimental Settings

Before presenting experimental results, this subsection introduces experimental settings, including parameter values in our models, the benchmark used to evaluate the LSM performance, and settings for the LSM training and validation.

First of all, the liquid state machine is set up using parameters summarized in Tables 4.1, 4.2 and 4.3 for parameter values used in the LIF neuron model, synaptic connectivity and learning rule, respectively. In Table 4.2, E and I denote excitatory and inhibitory neurons, respectively. $E \rightarrow I$ denotes the type of synapses with excitatory presynaptic neurons and inhibitory postsynaptic neurons. Note that time constants of membrane voltage and calcium variable in Table 4.1 are set to be 2^i with integer i for simplified division as discussed in Section. 4.5.1. In each neuron, 16-bit binary numbers are used for membrane voltage and calcium concentration. Within the reservoir, 20% neurons are inhibitory and 80% are excitatory. Weights of synapses connecting these neurons are fixed. In the LSM, reservoir neurons are fully connected to readout neurons by plastic synapses. Note that in the network, we set all synaptic delays d_{ij} to be 1 ms. This is an simplified model. Using different delay values may potentially help neurons firing asynchronously to interact better and improve the memory capacity [186] for further boosting the network performance.

Table 4.1: Parameter Values in LIF Neuron Model.

Parameters in LIF model	Value
Threshold voltage V_{th}	20 mv
Resting potential V_{th}	0 mv
Time constant τ_m	32 ms
Time constant τ_c	64 ms
Refractory period τ_{refrac}	2 ms

Table 4.2: Parameter Values in Synaptic Model.

Parameters in synaptic model	Value	
r (in Eqn. (4.3))	2	
k (in Eqn. (4.3))	type	value
	$E \to E$	0.45
	$E \to I$	0.3
	$I \to E$	0.6
	$I \to I$	0.15
fixed synaptic weight in the reservoir	type	value
	$E \to E$	3
	$E \to I$	6
	$I \to E$	-2
	$I \to I$	-2
W_{max}	type	value
	$E \to E/I$	$8(1-2^{n_{bit}-1})$
	$I \to E/I$	8
W_{min}	type	value
	$E \to E/I$	-8
	$I \to E/I$	$-8(1-2^{n_{bit}-1})$
ΔW	$0.0002 \times 2^{n_{bit}-4}$	
d_{ij}	1 ms	

Table 4.3: Parameter Values in Learning Rule.

Parameters in the learning rule	Value
C_θ	5
ΔC	3
δ	1

To evaluate the performance of the proposed liquid state machine, we use a subset of TI46 speech corpus as benchmark. The benchmark contains isolated word utterances of 5 different speakers. 10 different utterances of each word in 'zero' to 'nine' are recorded for each speaker. Thus, the benchmark contains 500 speech

samples. We also adopt 5-fold cross validation to test the LSM performance. In this setup, all samples in the benchmark are divided into 5 groups: G_1, G_2, G_3, G_4, and G_5. And 5 different LSMs are trained and tested. For the k^{th} ($k = 1, 2, 3, 4, 5$) LSM, the testing dataset is the group G_k and training dataset is the union of all other groups. During the training of an LSM, the temporal signal (speech) is applied to the input of the system shown in Fig. 4.4. In the mean time, the teacher signal is applied to each readout neuron. After the preprocessing, the input signal is transformed into 77 spike trains that feed into the reservoir. The combined influence from activities of reservoir neurons and readout neurons determines the learning of synapses connecting them. After applying each training sample to the LSM once, the LSM is trained for one *epoch*. To optimize the LSM performance, the LSM has to be trained for multiple epoches. During the testing, temporal signal of a testing speech sample is applied to the input while the teacher signal is not applied to any of the readout neuron. The recognition is based on the activities of readout neurons. The neuron that fires the most spikes is the winner, which indicates the classification result.

4.6.2 Precision of Synaptic Weights

As discussed in Section. 4.5.2, the number of synapses in the LSM is much larger than that of neurons. Therefore, reducing the hardware cost of synapse can potentially reduce the cost of the entire system significantly. In this subsection, we study the influence of the bitwidth (precision) of synaptic weight on the performance of liquid state machines. The reservoir size used in LSMs is $3 \times 3 \times 15$ [55].

Fig. 4.14 shows the performance of the LSM with different precisions of synaptic weights. The influence of the bitwidth of synaptic weights is summarized in Table 4.4, where initial results and best results are shown for synapses using different bitwidths. The initial results are obtained by training the LSM for only one epoch. The decent

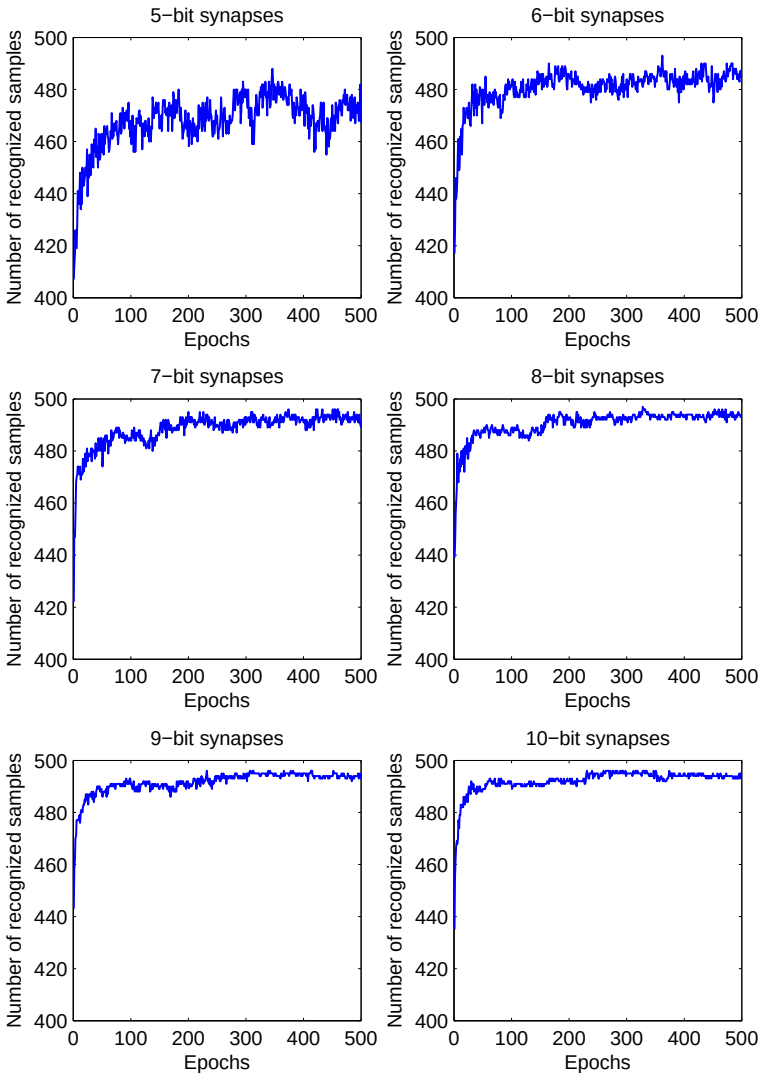

Figure 4.14: Influence of the precision of synaptic weights on the LSM performance. The six panels show performance of liquid state machines with synaptic weights with a resolution of five to ten bits. Results are for 5-fold cross validations with 500 speech samples. Plots show the recognition performance for the first 500 epochs of training. As shown in each plot, the performance almost saturate after less than 300 epochs of training. Comparison between these plots shows that the network performance is generally improved as the number of bits increases. And the performance almost saturates for synaptic weights using more than eight bits.

recognition rates show that the LSMs learn quite fast at the beginning of the training process. The best results represent the highest performance achieved in 500 epoches of training. To reduce the performance fluctuations over different epoches as shown in Fig. 4.14, the best performance is an average value over a 20-epoch interval.From these results, it is clear that the performance of the LSM generally increases with the bitwidth of synaptic weights. However, the performance almost saturates when the bitwidth of synaptic weights exceeds eight. This observation indicates that by using 8-bit synaptic weights, the precision of synaptic weights is not a performance bottleneck for the LSM anymore. Therefore, synapses with bitwidth larger than 8 is not necessary on hardware since it increases the storage as well as computational cost. Note that the summation of the recognition rate and the error rate is less than 1. This is because the network cannot recognize some speech samples due to fact that the number of readout neurons firing the most spikes can be more than one. Since in this case the input can be potentially recognized by using another classifier, this situation is better than recognizing the input erroneously.

Table 4.4: Performance of LSM vs. Bitwidth of Synaptic Weight.

Bitwidth of synaptic weights	No. of correctly recognized inputs		Recognition rate		Error rate	
	Initial	Best	Initial	Best	Initial	Best
5	407	480.5	81.4%	96.09%	17.0%	3.24%
6	417	487.5	83.4%	97.49%	12.4%	2.00%
7	422	493.9	84.4%	98.77%	13.0%	0.96%
8	439	494.8	87.8%	98.96%	7.8%	0.75%
9	443	495.2	88.6%	99.03%	5.4%	0.86%
10	435	495.5	87.0%	99.09%	5.4%	0.80%

Training and testing of 5 recognizers (in 5-fold cross validation) for 500 epoches

takes about 25.8 hours wallclock time. Or about 5.16 hours for each recognizer. On average, the training and testing of each recognizer for one epoch take only about 37 seconds in total. While the average duration of each word utterance is 0.66 second, the simulation of the recognition takes 0.073 second. That is, the temporal signal processing (recognition) speed of the LSM simulation is about 9 times faster than read time. Because of the parallel nature, we expect that the hardware implementation can be more efficient, thereby can process input signals in real time with a low clock speed to save power. To more closely resemble the hardware behavior, simulation on FPGA can also be used to prototype the proposed LSM.

4.6.3 Size of the Reservoir

For more cost effective designs of the LSM, we test the recognition rates and error rates of liquid state machines with reservoirs of different sizes. We use the same settings and parameter values introduced at the beginning of this section and 10-bit binary numbers for synaptic weights. Results are shown in Table 4.5. The first and second columns show the shape and the size of the reservoir. The third and fourth columns show the recognition rates and error rates, respectively. In all these tested LSMs with different size of the reservoir, the best performance reaches 99.79% recognition rate and 0.08% error rate. Clearly, the performance of LSMs varies with the reservoir size. The relation between them are visualized in Fig. 4.15. In the diagram on the top, each diamond represents an LSM. The two axes show the recognition rates and the number of neurons in the reservoir. Similarly, the bottom diagram shows that the error rate decreases with the size of the reservoir. From the figure, it is clear that with reservoirs of sizes smaller than 100 neurons, performance of LSMs are significantly lower than those with larger reservoirs. For reservoir sizes larger than 100 neurons, the recognition and error rates are around 99% and 1%,

respectively. And further increasing the reservoir size does not significant increase the LSM performance. Therefore, taking into account both performance and cost effectiveness, a reservoir of size slightly larger than 100 neurons is a good choice for an LSM for this application.

Table 4.5: Performance of LSM with Various Reservoir Sizes.

Reservoir shape	Reservoir size	Recognition rate	Error rate
$2 \times 2 \times 20$	80	96.25%	3.24%
$2 \times 2 \times 30$	120	99.31%	0.41%
$2 \times 2 \times 40$	160	98.74%	1.16%
$2 \times 2 \times 50$	200	99.21%	0.58%
$3 \times 3 \times 10$	90	97.16%	2.18%
$3 \times 3 \times 15$	135	99.09%	0.80%
$3 \times 3 \times 20$	180	98.78%	1.01%
$3 \times 3 \times 25$	225	98.87%	1.01%
$3 \times 3 \times 30$	270	98.55%	1.37%
$3 \times 3 \times 40$	360	98.90%	0.88%
$3 \times 3 \times 50$	450	98.91%	0.92%
$4 \times 4 \times 5$	80	96.90%	2.97%
$4 \times 4 \times 10$	160	98.43%	1.44%
$4 \times 4 \times 15$	240	98.69%	1.29%
$4 \times 4 \times 20$	320	99.08%	0.74%
$4 \times 4 \times 25$	400	98.15%	1.78%
$5 \times 5 \times 5$	125	99.10%	0.81%
$5 \times 5 \times 10$	250	99.00%	1.00%
$5 \times 5 \times 15$	375	98.54%	1.27%
$6 \times 6 \times 6$	216	99.24%	0.61%
$6 \times 6 \times 8$	288	98.79%	1.11%
$6 \times 6 \times 10$	360	99.64%	0.28%
$6 \times 6 \times 12$	432	98.86%	0.92%
$7 \times 7 \times 7$	343	99.79%	0.08%

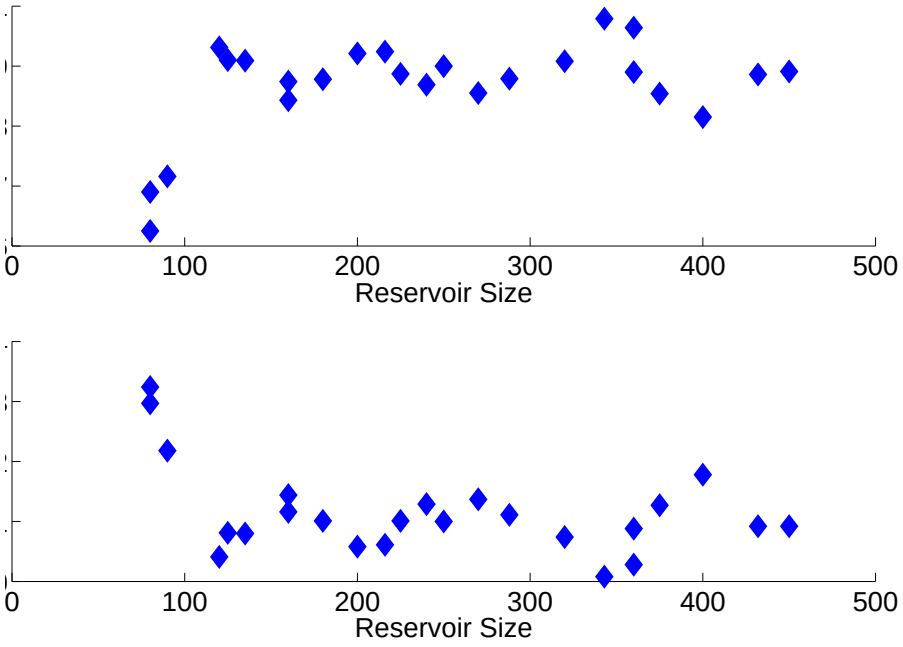

Figure 4.15: The diagram on the top shows the recognition rate of LSM vs. the size of the reservoir. The bottom one shows the error rate of LSM vs. the size of the reservoir.

4.6.4 Comparison with Other Methods and Discussion

The liquid state machine in [65] uses the same TI46 benchmark and the same preprocessing module introduced in Section. 4.1.2. The main difference lies in the training method. [65] adopts ridge regression for training readout neurons. With the reservoir size ranging from 300 to 2000 neurons, the word error rate (WER) on testing dataset is between 10% and 3%, which is significantly higher than that of the proposed LSM. The advantage of the proposed method may be due to the following factors. In [65], the temporal signal is sampled every 20 ms for the training. This may lead to loss of information in higher-frequency components of the temporal signal. In the proposed method, learning is driven by neuron activities at any time.

Thus, the loss of information is less. Therefore, the proposed LSM may overperform the method used in [65] in terms of both error rate and storage (Off-line learning requires large storage for all data). To reduce storage, [187] discusses an online sequential ELM (extreme learning machine) algorithm, which can be interpreted as an online version of the ridge regression. Even when this algorithm is adopted by the LSM system in [65], sampling of the temporal signals would be still required. In comparison, the proposed approach may be still more advantageous for hardware implementation because simpler computation is involved and the adopted learning rule is biologically motivated, local and hence highly parallel.

In [66], the liquid state machine uses different settings. The speech signals from TI46 are preprocessed by temporal based linear predictive coding. With only 8 to 27 neurons, the reservoirs are much smaller. But more complicated multi-layer feedforward networks consisting of 62 to 168 neurons are used for backpropagation learning algorithms. This work reports recognition rates between 80% and 100%. It should be pointed out that [66] uses less speech samples (200 in total, divided into two datasets for training and testing, respectively), which potentially leads to greater variations on the recognition rate when compared to results on a larger dataset. And [66] only reports the best result in different trials for each network configuration.

[69] builds long short-term memory recurrent neural networks for speech recognition. The network is constructed by connecting 121 units of different types. Using MFCC method for signal preprocessing, the WER on TI46 benchmark reaches 2%.

The state-of-the-art HMM based recognizer Sphinx-4 is introduced in [13] (A previous version Sphinx-2 is used in commercial products). Tested on the TI46 data set, a WER of 0.168% is reported. Comparison between this WER and our best results of 0.08% error rate and 99.79% recognition rate shows the top-notch performance of the proposed LSM. Note that Sphinx-4 uses dataset-specific language models and

acoustic models and is heavily tuned for the specific dataset. In contrast, the proposed LSM is not designed specifically for any language or dataset. Thus, without changing the LSM, we expect similar performance if it is applied to another dataset even in another language. The resulted LSM trained by another dataset is only different in terms of the readout neurons and the weights of their incident synapses. In other words, other parts of the liquid state machine, including the reservoir and the preprocessing stages, are generally applicable to different datasets even in different languages for speech recognition. To take advantage of this generality, a hardware-base LSM may be built as an accelerator for speech recognition and a number of other applications.

In addition to the behaviors of LSMs, the speech aerodynamics is also chaotic. In [188], this is characterized by multiscale fractal dimension (MFD). Due to friction and aspiration, dynamics of airflow often result in certain degree of turbulence, which contributes to geometrical complexity and fragmentation of speech waveforms. This geometrical complexity is modeled by their fractal dimensions at multiple scales and used as an additional feature in speech recognition. Utilizing the extracted information from the speech waveform by MFD, [188] shows improved performance of speech recognition. Therefore, it is possible to take into account chaotic behaviors in both speech aerodynamics and liquid state machines to further improve the LSM performance.

4.7 Summary

We propose a fully bio-inspired digital liquid state machine for the purpose of low power hardware implementation. The spike-based online learning rule reduces the algorithm complexity and facilitated VLSI implementation by removing communications between non-neighboring elements in the neural network. Using a subset of

TI46 speech corpus as benchmark, we study the influence of synaptic dynamics on the LSM performance and merge the dynamics for simplified hardware implementation. To reduce the potential hardware cost, we also study the trade-off between the number of bits used for synaptic weights and the LSM performance. To better understand the LSM, we study the influence of reservoir sizes on the recognition performance. Tested on TI46 dataset, our proposed LSM shows top-notch performance among all speech recognizers, including the state-of-art HMM based Sphinx-4.

5. CONCLUSION

This dissertation develops techniques for modeling and analyzing dynamic behaviors of biologically realistic genetic circuits and brain models and design of brain-inspired computing systems.

As a promising candidate for information storage in synthetic biological systems, genetic memory circuits exhibit dynamic behaviors, which have significant impact on their biological functions. As such, the dynamic stability of genetic memory circuits is studied to understand their functionality for their potential applications in synthetic biology. Based on the electrical-equivalent models of biochemical reactions, simulation techniques widely used for electronic systems are applied to provide quantitative analysis capabilities. In particular, system-theoretical techniques are used to study the dynamic behaviors of genetic memory circuits, where the notion of stability boundary is employed to characterize the bistability of such circuits. While the contribution is focused on the bistable genetic memory circuits, with proper extension, our proposed techniques are broadly applicable to other multi-stable biological systems [189].

To apply large-scale computation to quantitative studies of complicated biological systems, we construct large-scale brain models with detailed cellular mechanisms to facilitate the simulation-based studies of physiological and pathological behaviors in brain disorders. By developing dedicated numerical techniques for brain simulation, the simulation speed is greatly improved such that dynamic simulation of large thalamocortical models with more than one million multi-compartment neurons and hundreds of synapses on commodity computer servers becomes feasible. Simulation of such large model produces biologically meaningful results demonstrating the

emergence of sigma and delta waves in the early and deep stages of sleep, and suggesting the underlying cellular mechanisms that may be responsible for generation of absence seizure. With rapid progress in experimental brain research, increasingly available data in the future would help to build more complete and accurate brain models capturing more biologically meaningful brain behaviors for broader applications [190][191]. On the side of computation, the numerical techniques may be extended to larger computing platforms for faster simulation of the brain [162].

Brain-inspired computing paradigms may offer promising solutions to many challenges facing the main stream Von Neumann computer architecture. To this end, we develop a biologically inspired learning system amenable to VLSI implementation. The proposed solution consists of a digitized liquid state machine (LSM) and a spike-based learning rule, providing a fully biologically inspired learning paradigm. The key design parameters of this liquid state machine are optimized to maximize learning performance while considering hardware implementation cost. When applied to speech recognition of isolated words using TI46 speech corpus, the performance of the proposed LSM rivals several existing state-of-art techniques including the Hidden Markov Model based recognizer Sphinx-4 [13]. These initial results on the brain-inspired learning system provide a new starting point for further exploration of both machine learning and the brain. On the side of machine learning, the specific application of the LSM on speech recognition could be further extended to larger vocabulary and continuous speech. This could lead to more powerful speech recognizers for practical applications [192]. For the general study of the LSM, introducing learning into the reservoir might lead to more memory capacity and close mimicking of the brain behavior [186]. Taking advantage of the low power nature of spiking neural networks, in the future reservoirs with a 3-D structure could be manufactured on 3-D chips [193]. On the side of brain study, the critical dependency of

LSM performance on the chaotic behavior of the reservoir [179][181] provides a new perspective on the study of learning in the brain, which may be further explored.

REFERENCES

[1] M. Pospischil, M. Toledo-Rodriguez, C. Monier, Z. Piwkowska, T. Bal, Y. Frégnac, H. Markram, and A. Destexhe. Minimal Hodgkin-Huxley type models for different classes of cortical and thalamic neurons. *Biological Cybernetics*, 99(4-5):427–441, 2008.

[2] E. May and R. Schiek. *Xyce parallel electronic simulator: Biological pathway modeling and simulation*. Sandia National Laboratories, Albuquerque, NM, 2006.

[3] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha. The cat is out of the bag: Cortical simulations with 10^9 neurons, 10^{13} synapses. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12, 2009.

[4] A. Omurtag, B. W. Knight, and L. Sirovich. On the simulation of large populations of neurons. *Journal of Computational Neuroscience*, 8(1):51–63, 2000.

[5] H. Kitano. Computational systems biology. *Nature*, 420(6912):206–210, 2002.

[6] C. A. Ouzounis. Rise and demise of bioinformatics? Promise and progress. *PLOS Computational Biology*, 8(4):e1002487, 2012.

[7] O. G. Clark, R. Kok, and R. Lacroix. Mind and autonomy in engineered biosystems. *Engineering Applications of Artificial Intelligence*, 12(3):389–399, 1999.

[8] J. J. Videler, E. J. Stamhuis, and G. D. Povel. Leading-edge vortex lifts swifts. *Science*, 306(5703):1960–1962, 2004.

[9] J. F. V. Vincent, O. A. Bogatyreva, N. R. Bogatyrev, A. Bowyer, and A. K. Pahl. Biomimeticsits practice and theory. *Journal of the Royal Society Inter-*

face, 3(9):471–482, 2006.

[10] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha. A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm. In *Proceedings of the Custom Integrated Circuits Conference*, pages 1–4, 2011.

[11] B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, D. S. Modha, and D. J. Friedman. A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. In *Proceedings of the Custom Integrated Circuits Conference*, pages 1–4, 2011.

[12] W. Mass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[13] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel. *Sphinx-4: A Flexible Open Source Framework for Speech Recognition*. Sun Microsystems Inc., San Jose, CA, 2004.

[14] D. Endy. Foundations for engineering biology. *Nature*, 438(7067):449–453, 2005.

[15] J. Hasty, D. McMillen, and J. J. Collins. Engineered gene circuits. *Nature*, 420(6912):424–230, 2002.

[16] T. Gardner, C. Cantor, and J. Collins. Construction of a genetic toggle switch in *Escherichia coli*. *Nature*, 403(6767):339–342, 2000.

[17] M. R. Atkinson, M. A. Savageau, J. T. Myers, and A. J. Ninfa. Development of genetic circuitry exhibiting toggle switch or oscillatory behavior in *Escherichia coli*. *Cell*, 113(5):597–607, 2003.

[18] G. Fritz, N. E. Buchler, T. Hwa, and U. Gerland. Designing sequential transcription logic: A simple genetic circuit for conditional memory. *System and*

Synthetic Biology, 1(2):89–98, 2007.

[19] L. Pollack and W. W. Webb. Complex molecular dynamics in the spotlight. *Nature Biotechnology*, 28(6):564–565, 2010.

[20] C. Schöfer and K. Weipoltshammer. Gene dynamics and nuclear architecture during differentiation. *Differentiation*, 76(1):41–56, 2008.

[21] R. K. Chesser. Heteroplasmy and organelle gene dynamics. *Genetics*, 150(3):1309–1327, 1998.

[22] S. Faisal, G. Lichtenberg, and H. Werner. Polynomial models of gene dynamics. *Neurocomputing*, 71(13):2711–2719, 2008.

[23] Y. Zhang and P. Li. Gene-regulatory memories: Electrical-equivalent modeling, simulation and parameter identification. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 491–496, 2009.

[24] S. Widder, J. Macía, and R. Solé. Monomeric bistability and the role of autoloops in gene regulation. *PLoS ONE*, 4(4):e5399, 2009.

[25] N. Nguyen, C. Myers, H. Kuwahara, C. Winstead, and J. Keener. Design and analysis of a robust genetic Muller C-element. *Journal of Theoretical Biology*, 264(2):174–187, 2010.

[26] P. S. Swain, M. B. Elowitz, and E. D. Siggia. Intrinsic and extrinsic contributions to stochasticity in gene expression. *Proceedings of the National Academy of Science*, 99(20):12795–12800, 2002.

[27] J. M. Raser and E. K. O'Shea. Noise in gene expression: Origins, consequence, and control. *Science*, 309(5743):2010–2013, 2005.

[28] J. M. Pedraza and A. van Oudenaarden. Noise propagation in gene networks. *Science*, 307(5717):1965–1969, 2005.

[29] A. Colman-Lerner, A. Gordon, E. Serra, T. Chin, O. Resnekov, D. Endy, C. G. Pesce, and R. Brent. Regulated cell-to-cell variation in a cell-fate decision

system. *Nature*, 437(7059):699–706, 2005.

[30] W. Dong, P. Li, and G. M. Huang. SRAM dynamic stability: Theory, variability and analysis. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 378–385, 2008.

[31] A. Hodgkin and A. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117(4):500–544, 1952.

[32] E. M. Izhikevich, J. A. Gally, and G. M. Edelman. Spike-timing dynamics of neuronal groups. *Cerebral Cortex*, 14(8):933–944, 2004.

[33] A. K. Fidjeland and M. P. Shanahan. Accelerated simulation of spiking neural networks using GPUs. In *Proceedings of the World Congress on Computational Intelligence*, pages 536–543, 2010.

[34] M. Wang, B. Yan, J. Hu, and P. Li. Simulation of large neuronal networks with biophysically accurate models on graphics processors. In *Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN)*, pages 3184–3193, 2011.

[35] G. G. Blasdel, J. S. Lund, and D. Fitzpatrick. Intrinsic connections of macaque striate cortex: Axonal projections of cells outside lamina 4C. *Journal of Neuroscience*, 5(12):3350–3369, 1985.

[36] D. Fitzpatrick, J. S. Lund, and G. G. Blasdel. Intrinsic connections of macaque striate cortex: Afferent and efficient connenctions of lamina 4C. *Journal of Neuroscience*, 5(12):3329–3349, 1985.

[37] A. Peters and B. R. Payne. Numerical relationship between geniculocortical afferents and pyramidal cell modules in cat primary visual cortex. *Cerebral Cortex*, 3(1):69–78, 1993.

[38] J. A. Winer and D. T. Larue. Evolution of GABAergic circuitry in the mam-

malian medial geniculate body. *Proceedings of the National Academy of Science*, 93(7):3083–3087, 1996.

[39] S. C. van Horn, A. Erisir, and S. M. Sherman. Relative distribution of synapses in the A-laminae of the lateral geniculate nucleus of the cat. *Journal of Comparative Neurology*, 416(4):509–520, 2000.

[40] T. Binzegger, R. J. Douglas, and K. A. C. Martin. A quantitative map of the circuit of cat primary visual cortex. *Journal of Neuroscience*, 24(39):8441–8453, 2004.

[41] E. M. Izhikevich and G. M. Edelman. Large-scale model of mammalian thalamocoritcal systems. *Proceedings of the National Academy of Science*, 105(9):3593–3598, 2008.

[42] A. Zalesky and A. Fornito. A DTI-derived measure of cortico-cortical connectivity. *IEEE Transactions on Medical Imaging*, 28(7):1023–1036, 2009.

[43] B. W. Connors and M. J. Gutnick. Intrinsic firing patterns of diverse neocortical neurons. *Trends Neuroscience*, 13(3):99–104, 1990.

[44] R. D. Traub and R. Miles. *Neuronal Networks of the Hippocampus*. Cambridge University Press, Cambridge, UK, 1991.

[45] J. R. Huguenard and D. A. Prince. A novel T-type current underlies prolonged Ca^{2+}-dependent bursts firing in GABAergic neurons of rat thalamic reticular nucleus. *The Journal of Neuroscience*, 12(10):3804–3817, 1992.

[46] J. Huguenard and D. A. Prince. Neurotransmitter control of neocortical neuronal activity and excitability. *Cerebral Cortex*, 3(5):387–398, 1993.

[47] A. Destexhe, T. Bal, D. A. McCormick, and T. J. Sejnowski. Ionic mechanisms underlying synchronized oscillations and propagating waves in a model of ferret thalamic slices. *Journal of Neurophysiology*, 76(3):2049–2070, 1996.

[48] A. Destexhe, D. Contreras, M. Steriade, T. J. Sejnowski, and J. R. Huguenard.

In vivo, in vitro, and computational analysis of dendritic calcium currents in thalamic reticular neurons. *The Journal of Neuroscience*, 16(1):169–185, 1996.

[49] J. M. Bower and D. Beeman. *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* Springer-Verlag, New York, NY, 1998.

[50] M. L. Hines and N. T. Carnevale. NEURON: A tool for neuroscientists. *The Neuroscientist*, 7(2):123–135, 2001.

[51] P. Gloor, L. F. Quesney, and H. Zumstein. Pathophysiology of generalized penicillin epilepsy in the cat: The role of cortial and subcortical structures. II. Topical appliation of penicillin to the cerebral cortex and subcortical structures. *Electroencephalography and Clinical Neurophysiology*, 43(1):79–94, 1977.

[52] A. Destexhe, D. Contreras, and M. Steriade. Mechanisms underlying the synchronizing action of corticothalamic feedback through inhibition of thalamic relay cells. *Journal of Neurophysiology*, 79(2):999–1016, 1998.

[53] A. Destexhe. Spike-and-wave oscillations based on the properties of $GABA_B$ receptors. *Journal of Neuroscience*, 18(21):9099–9111, 1998.

[54] J. M. Brader, W. Senn, and S. Fusi. Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Computing*, 19(11):2881–2912, 2007.

[55] W. Maass, T. Natschlager, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[56] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers. SWAT: A spiking neural network training algorithm for classification problems. *IEEE Transactions on Neural Networks*, 21(11):1817–1830, 2010.

[57] S. M. Bohte, J. N. Kok, and J. A. La Poutre. SpikeProp: Backpropagation

for networks of spiking neurons. In *Proceedings of the European Symposium on Artificial Neural Networks*, 2000.

[58] T. Natschlager and B. Ruf. Spatial and temporal pattern analysis via spiking neurons. *Network: Computation in Neural Systems*, 9(3):319–332, 1998.

[59] J. L. McKinstry and G. M. Edelman. Temporal sequence learning in winner-take-all networks of spiking neurons demonstrated in a brain-based device. *Frontiers in Neurorobotics*, 7(10):1–10, 2013.

[60] J. Misra and I. Saha. Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing*, 74(1):239–255, 2010.

[61] Y. Kim, Y. Zhang, and P. Li. A digital neuromorphic VLSI architecture with memristor crossbar synaptic array for machine learning. In *Proceedings of the International SOC Conference*, pages 328–333, 2012.

[62] W. Maass, T. Natschlager, and H. Markram. Computational models for generic cortical microcircuits. In *Computational Neuroscience: A Comprehensive Approach*, pages 575–605. CRC Press, London, UK, 2004.

[63] M. J. Embrechts, L. A. Alex, and J. D. Linton. Reservoir computing for static pattern recognition. In *Proceedings of the European Symposium on Artificial Neural Networks*, 2009.

[64] Y. Zhang and K. Wang. The application of liquid state machines in robot path planning. *Journal of Computers*, 4(11):1182–1186, 2009.

[65] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout. Isolated word recognition with the liquid state machine: A case study. *Information Processing Letters*, 95(6):521–528, 2005.

[66] A. Ghani, T. M. McGinnity, L. Maguire, L. McDaid, and A. Belatreche. Neuro-inspired speech recognition based on reservoir computing. In *Advances in Speech Recognition*, pages 7–36. Springer, New York, NY, 2010.

[67] R. K. Moore. Twenty things we still don't know about speech. In *Proceedings of the Workshop on Progress and Prospects of Speech Research and Technology*, 1994.

[68] M. A. Anusuya and S. K. Katti. Speech recognition by machine: A review. *Internaltional Journal of Computer Science and Information Security*, 6(3):181–205, 2009.

[69] A. Graves, D. Eck, N. Beringer, and J. Schmidhuber. Biologically plausible speech recognition with LSTM neural nets. In *Biologically Inspired Approaches to Advanced Information Technology*, volume 3141, pages 127–136. Springer, New York, NY, 2004.

[70] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25), 1977.

[71] J. Ting, M. Mistry, J. Peters, S. Schaal, and J. Nakanishi. A bayesian approach to nonlinear parameter identification for rigid body dynamics. In *Robotics: Science and Systems II*, pages 247–254. The MIT Press, Cambridge, MA, 2007.

[72] K. J. Friston. Bayesian estimation of dynamical systems: An application to fMRI. *Neuroimage*, 16(2):513–530, 2002.

[73] P. Spellucci. An SQP method for general nonlinear programs using only equality constrained subproblems. *Mathematical Programming*, 82(3):413–448, 1993.

[74] H. K. Khalil. *Nonlinear Dynamics*. Prentice Hall, Upper Saddle River, NJ, 2 edition, 1996.

[75] N. H. E. Weste and D. Harris. *CMOS VLSI Design: A Circuit and Sysmtems Perspective*. Addison-Wesley, Boston, MA, 3 edition, 2005.

[76] Y. Zhang, P. Li, and G. M. Huang. Separatrix in high-dimensional state space: System-theoretical computation and application to SRAM dynamic stability analysis. In *Proceedings of the IEEE/ACM Design Automation Conference*,

pages 567–572, 2010.

[77] J. Zaborszky, G. Huang, B. Zheng, and T. C. Leung. On the phase portrait of a class of large nonlinear dynamic systems such as the power system. *IEEE Transaction on Automatic Control*, 33(1):4–15, 1988.

[78] S. Wiggins. *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. Springer, New York, NY, 2 edition, 2003.

[79] M. Laurent and N. Kellershohn. Multistability: A major means of differentiation and evolution in biological systems. *Trends in Biochemical Science*, 24(11):418–422, 1999.

[80] A. Becskei, B. Séraphin, and L. Serrano. Positive feedback in eukaryotic gene networks: Cell differentiation by graded to binary response conversion. *The EMBO Journal*, 20(10):2528–2535, 2001.

[81] J. E. Ferrell. Self-perpetuating states in signal transduction: Positive feedback, double-negative feedback and bistability. *Curren Opinion in Cell Biology*, 14(2):140–148, 2002.

[82] A. D. Hernday, B. A. Braaten, and D. A. Low. The mechanism by which DNA adenine methylase and PapI activate the pap epigenetic switch. *Molecular Cell*, 12(4):947–957, 2003.

[83] G. Huang, H. Wang, S. Chou, X. Nie, J. Chen, and H. Liu. Bistable expression of *WOR1*, a master regulator of whiteopaque switching in *Candida albicans*. *Proceedings of the National Academy of Sciences*, 103(34):12813–12818, 2006.

[84] M. Bazhenov and I. Timofeev. Thalamocortical oscillations. *Scholarpedia*, 1(6):1319, 2006.

[85] B. Yan and P. Li. An integrative view of mechanisms underlying generalized spike-and-wave epileptic seizures and its implication on optimal therapeutic treatments. *PLoS ONE*, 6(7):e22440, 2011.

[86] D. A. McCormick and H. C. Pape. Noradrenergic and serotonergic modulation of a hyperpolarization-activated cation current in thalamic relay neurones. *The Journal of Physiology*, 431(1):319–342, 1990.

[87] M. Steriade and M. Deschenes. The thalamus as a neuronal oscillator. *Brain Research Reviews*, 8(1):1–63, 1984.

[88] M. Steriade, M. Deschenes, L. Domich, and C. Mulle. Abolition of spindle oscillations in thalamic neurons disconnected from nucleus reticularis thalami. *Journal of Neurophysiology*, 54(6):1473–1497, 1985.

[89] M. von Krosigk, T. Bal, and D. A. McCormick. Cellular mechanisms of a synchronized oscillation in the thalamus. *Science*, 261(5119):361–364, 1993.

[90] C. Bernard, Y. C. Ge, E. Stockley, J. B. Willis, and H. V. Wheal. Synaptic integration of NMDA and non-NMDA receptors in large neuronal network models solved by means of differential equations. *Biological Cybernetics*, 70(3):267–273, 1994.

[91] A. Destexhe and T. J. Sejnowski. G-protein activation kinetics and spillover of GABA may account for differences between inhibitory responses in the hippocampus and thalamus. *Proceedings of the National Academy of Sciences*, 92(21):9515–9519, 1995.

[92] D. A. Hosford, S. Clark, Z. Cao, Jr. W. A. Wilson, F. Lin, R. A. Morrisett, and A. Huin. The role of $GABA_B$ receptor activation in absence seizures of lethargic (lh/lh) mice. *Science*, 257(5068):398–401, 1992.

[93] O. C. Snead. Evidence for $GABA_B$-mediated mechanisms in experimental generalized absence seizures. *European Journal of Pharmacology*, 213(3):343–349, 1992.

[94] Z. Liu, M. Vergnes, A. Depaulis, and C. Marescaux. Involvement of intrathalamic $GABA_B$ neurotransmission in the control of absence seizures in rat. *Neu-*

roscience, 48(1):87–93, 1992.

[95] K. A. Smith and R. S. Fisher. The selective $GABA_B$ antagonist CGP-35348 blocks spike-wave bursts in the cholesterol synthesis rat absence epilepsy model. *Brain Research*, 729(2):147–150, 1996.

[96] F. Varela, J. Lachaux, E. Rodriguez, and J. Martinerie. The brainweb: Phase synchronization and large-scale integration. *Nature Reviews Neuroscience*, 2(4):229–239, 2001.

[97] P. J. Uhlhaas and W. Singer. Neural synchrony in brain disorders: Relevance for cognitive dysfunctions and pathophysiology. *Neuron*, 52(1):155–168, 2006.

[98] F. Amor, S. Baillet, V. Navarro, C. Adam, J. Martinerie, and M. L. V. Quyen. Cortical local and long-range synchronization interplay in human absence seizure initiation. *Neuroimage*, 45(3):950–862, 2009.

[99] H. E. Plesser, J. M. Eppler, A. Morrison, M. Diesmann, and M. Gewaltig. Efficient parallel simulation of large-scale neuronal networks on clusters of multiprocessor computers. In *Euro-Par 2007 Parallel Processing*, volume 4641, pages 672–681. Springer Berlin Heidelberg, Berlin, Germany, 2007.

[100] S. Ranka, Y. Won, and S. Sahni. Programming a hypercube multicomputer. *IEEE Software*, 5(5):69–77, 1988.

[101] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7(2):279–301, 1989.

[102] S. H. Hosseini, B. Litow, M. Malkawi, J. Mcpherson, and K. Vairavan. Analysis of a graph coloring based distributed local balancing algorithm. *Journal of Parallel and Distributed Computing*, 10(2):160–166, 1990.

[103] V. A. Saletore. A distributed and adaptive dynamic load balancing scheme for parallel processing of medium-grain tasks. In *Preceedings of the Fifth Distributed Memory Computing Conference*, pages 994–999, 1990.

[104] C. W. Gear and G. Kevrekidis. Projective methods for stiff differential equations: Problems with gaps in their eigenvalue spectrum. *SIAM Journal on Scientific Computing*, 24(4):1091–1106, 2003.

[105] C. W. Gear and G. Kevrekidis. Telescopic projective methods for parabolic differential equations. *Journal of Computational Physics*, 187(1):95–109, 2003.

[106] D. A. Pollen. Intracellular studies of cortical neurons during thalamic induced wave and spike. *Electroencephalography and Clinical Neurophysiology*, 17(4):398–404, 1964.

[107] M. Steriade. Interneuronal epileptic discharges related to spike-and-wave cortical seizures in behaving monkeys. *Electroencephalography and Clinical Neurophysiology*, 37(3):247–263, 1974.

[108] M. Avoli, P. Gloor, G. Kostopoulos, and J. Gotman. An analysis of penicillin-induced generalized spike and wave discharges using simultaenous recordings of cortical and thalamic single neurons. *Journal of Physiology*, 50(4):819–837, 1983.

[109] R. S. McLachlan, M. Avoli, and P. Gloor. Transition from spindles to generalized spike and wae discharges in the cat: Simultaneous single cell recordings in the cortex and thalamus. *Experimental Neurology*, 85(2):413–425, 1984.

[110] G. Buzsaki, A. Smith, S. Berger, L. J. Fisher, and F. H. Gage. Petit mal epilepsy and parkinsonian tremor: Hypothesis of a common pacemaker. *Neuroscience*, 36(1):1–14, 1990.

[111] M. Inoue, J. Duysens, J. M. L. Vossen, and A. M. L. Coenen. Thalamic multiple-unit activity underlying spike-wave discharges in anesthetized rats. *Brain Research*, 612(1):35–40, 1993.

[112] F. Tadel, S. Baillet, J. C. Morrison, D. Pantazis, and R. M. Leahy. Brainstorm: A user-friendly application for MEG/EEG analysis. *Computational Intelligence*

and Neuroscience, 2011(8):1–13, 2011.

[113] C J. Honey, R Kotter, M. Breakspear, and O. Sporns. Network structure of cerebral cortex shapes functional connectivity on multiple time scales. *Proceedings of the National Academy of Science*, 104(24):10240–10245, 2007.

[114] F. Varela, J. Lachaux, E. Rodriguez, and J. Martinerie. The brainweb: Phase synchronization and large-scale integration. *Nature Reviews Neuroscience*, 2(4):229–239, 2001.

[115] N. Spruston. Pyramidal neurons: Dendritic structure and synaptic integration. *Nature Reviews Neuroscience*, 9(3):206–221, 2008.

[116] J. J. Hablitz and J. Yang. Morphological and electrophysiological characterization of abnormal cell types in pediatric cortical dysplasia. *Epilepsia*, 72(4):472–486, 2003.

[117] C J. Honey, O. Sporns, L. Cammoun, X. Gigandet, J. P. Thiran, R. Meuli, and P. Hagmann. Predicting human resting-state funcational connectivity from structural connectivity. *Proceedings of the National Academy of Science*, 106(6):2035–2040, 2009.

[118] G. Deco, V. K. Jirsa, and A. R. Mclntosh. Emerging concepts for the dynamical commentoutorganization of resting-state activity in the brain. *Nature Reviews Neuroscience*, 12(1):43–56, 2011.

[119] A. Ghosh, Y. Rho, A. R. Mclntosh, R. Kotter, and V. K. Jirsa. Noise during rest enables the exploration of the brain's dynamic repertoire. *PLoS Computational Biology*, 4(10):e1000196, 2008.

[120] B B. Biswal and et al. Toward discovery science of human brain function. *Proceedings of the National Academy of Science*, 107(10):4734–4739, 2010.

[121] D. S. Modha and R. Singh. Network architecture of the long-distance pathways in the macaque brain. *Proceedings of the National Academy of Science*,

107(30):13485–13490, 2010.

[122] G. A. Kohring. Convergence tiem and finte size effects in neural networks. *Journal of Physics A: Mathematical and General*, 23(11):2237, 1990.

[123] A. Pikovsky and S. Ruffo. Finite-size effects in a population of interacting oscillators. *Physical Review E*, 59(2):1633–1636, 1999.

[124] D. Hansel, G. Mato, and C. Meunier. Synchrony in excitatory neural networks. *Neural Computation.*, 7(2):307–337, 1995.

[125] M. A. Dichter and G. F. Ayala. Cellular mechanisms of epilepsy: A status report. *Science*, 237(4811):157–164, 2004.

[126] I. Timofeev, F. Grenier, and M. Steriade. Contribution of intrinsic neuronal factors in the generation of cortically driven electrographic seizures. *Journal of Neurophysiology*, 92(2):1138–1143, 2004.

[127] E. M. Izhikevich. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 15(5):1063–1070, 2004.

[128] G. Deco, V. K. Jirsa, P. A. Robinson, M. Breakspear, and K. J. Friston. The dynamic brain: From spiking neurons to neural-masses and cortical fields. *PLoS Computational Biology*, 4(8):e1000092, 2008.

[129] W. J. Freeman. *Mass Action in the Nervous System.* Academic Press, New York, NY, 1975.

[130] F. H. Lopes da Silva, A. van Rotterdam, P. Barts, E. van Heusden, and W. Burr. Models of neuronal populations: The basic mechanisms of rhythmicity. *Progress in Brain Research*, 45:281–308, 1975.

[131] S. Rodrigues, J. R. Terry, and M. Breakspear. On the genesis of spikewave oscillations in a mean-field model of human thalamic and corticothalamic dynamics. *Physics Letters A*, 355(4):352–357, 2006.

[132] H. R. Wilson and J. D. Cowan. Excitatory and inhibitory interactions in

localized populations of model neurons. *Biophysical Journal*, 12(1):1–24, 1972.

[133] S. Amari. Characteristics of random nets of analog neuron-like elements. *IEEE Transactions on Systems, Man and Cybernetics*, 2(5):643–657, 1975.

[134] S. Amari. Homogeneous nets of neuron-like elements. *Biological Cybernetics*, 17(4):211–220, 1975.

[135] B. W. Knight. The relationship between the firing rate of a single neuron and the level of activity in a population of neurons: Experimental evidence for resonant enhancement in the population response. *The Journal of General Physiology*, 59(6):767–778, 1972.

[136] B. W. Knight. Dynamics of encoding in a population of neurons. *The Journal of General Physiology*, 59(6):734–766, 1972.

[137] H. Sampolinsky and A. Zippelius. Relaxational dynamics of the Edwards-Anderson model of the mean-field theory of spin-glasses. *Physical Review B*, 25(11):6860–6875, 1982.

[138] D. Q. Nykamp and D. Tranchina. A population density approach that facilitates large-scale modeling of neural networks: Analysis and an application to orientation tuning. *Journal of Computational Neuroscience*, 8(1):19–50, 2000.

[139] T. D. Frank, A. Daffertshofer, and P. J. Beek. Multivariate Ornstein-Uhlenbeck processes with mean-field dependent coefficients: Application to postural sway. *Physical Review E*, 63(1):1–16, 2001.

[140] E. Haskell, D. Q. Nykamp, and D. Tranchina. Population density methods for large-scale modelling of neuronal networks with realistic synaptic kinetics: Cutting the dimension down to size. *Network: Computation in Neural Systems*, 12(2):141–174, 2001.

[141] S. Coombes. Large-scale neural dynamics: Simple and complex. *Neuroimage*, 52(3):731–739, 2010.

[142] P. C. Bressloff. Spatiotemporal dynamics of continuum neural fields. *Journal of Physics A: Mathematical and Theoretical*, 45(3):033001, 2012.

[143] O. David, S. J. Kiebel, L. M. Harrison, J. Mattout, J. M. Kilner, and K. J. Friston. Dynamic causal modeling of evoked responses in EEG and MEG. *Neuroimage*, 30(4):1255–1272, 2006.

[144] A. C. Marreiros, S. J. Kiebel, S. J. Kiebel, J. Daunizeau, and K. J. Friston. Population dynamics under the Laplace assumption. *Neuroimage.*, 44(3):701–714, 2009.

[145] I. Bojak, T. F. Oostendorp, A. T. Reid, and R. Kotter. Realistic mean field forward predictions for the integration of co-registered EEG/fMRI. *BMC Neurosci*, 10(Suppl 1):L2, 2009.

[146] I. Bojak, T. F. Oostendorp, A. T. Reid, and R. Kotter. Connecting mean field models of neural activity to EEG and fMRI data. *Brain Topogr*, 23(2):139–149, 2010.

[147] I. Bojak, T. F. Oostendorp, A. T. Reid, and R. Kotter. Towards a model-based integration of co-registered electroencephalography/functional magnetic resonance imaging data with realistic neural population meshes. *Philos Transact A Math Phys Eng Sci*, 369(1952):3785–3801, 2011.

[148] J. Daunizeau, S. J. Kiebel, and K. J. Friston. Dynamic causal modelling of distributed electromagnetic responses. *Neuroimage*, 47(2):590–601, 2009.

[149] F. Wendling, J. J. Bellanger, F. Bartolomei, and P. Chauvel. Relevance of nonlinear lumped-parameter models in the analysis of depth-EEG epileptic signals. *Biological Cybernetics*, 83(4):367–378, 2000.

[150] P. A. Robinson, C. J. Rennie, and D. L. Rowe. Dynamics of large-scale brain activity in normal arousal states and epileptic seizures. *Physical Review E*, 65(4):1–9, 2002.

[151] F. Wendling, F. Bartolomei, J. J. Bellanger, and P. Chauvel. Epileptic fast activity can be explained by a model of impaired GABAergic dendritic inhibition. *European Journal of Neuroscience*, 15(9):1499–1508, 2002.

[152] F. H. L. da Silva, W. Blanes, S. N. Kalitzin, J. Parra, P. Suffczynski, and D. N. Velis. Dynamical diseases of brain systems: Different routes to epileptic seizures. *IEEE Transactions on Biomedical Engineering*, 50(5):540–548, 2003.

[153] F. Wendling, A. Hernandez, J. J. Bellanger, P. Chauvel, and F. Bartolomei. Interictal to ictal transition in human temporal lobe epilepsy: Insights from a computational model of intracerebral EEG. *Journal of Clinical Neurophysiology*, 22(5):343–356, 2005.

[154] P. Suffczynski, S. Kalitzin, and F. H. L. da Silva. Dynamics of non-convulsive epileptic phenomena modeled by a bistable neuronal network. *Neuroscience*, 126(2):467–484, 2004.

[155] P. Suffczynski, F. H. L. da Silva, J. Parra, D. N. Velis, and S. N. Kalitzin. Epileptic transitions: Model predictions and experimental validation. *Journal of Clinical Neurophysiology*, 22(5):288–299, 2005.

[156] M. A. Kramer, H. E. Kirsch, and A. J. Szeri. Pathological pattern formation and cortical propagation of epileptic seizures. *Journal of the Royal Society Interface*, 2(2):113–127, 2005.

[157] D. T. J. Liley and I. Bojak. Understanding the transition to seizure by modeling the epileptiform activity of general anesthetic agents. *Journal of Clinical Neurophysiology*, 22(5):300–313, 2005.

[158] M. Breakspear, J. A. G. Roberts, J. R. Terry, S. Rodrigues, N. Mahant, and P. A. Robinson. An unifying explanation of primary generalized seizures through nonlinear brain modeling and bifurcation analysis. *Cerebral Cortex*, 16(9):1296–1313, 2006.

[159] F. Marten, S. Rodrigues, O. Benjamin, M. P. Richardson, and J. R. Terry. Onset of polyspike complexes in a mean-field model of human electroencephalography and its application to absence epilepsy. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1891):1145–1161, 2009.

[160] B. Molaee-Ardekani, P. Benquet, F. Bartolomei, and F. Wendling. Computational modeling of high-frequency oscilaltions at the onset of neocortical partial seizures: From 'altered structure' to 'dysfunction'. *Neuroimage*, 52(3):1109–1122, 2010.

[161] *PowerEdge R715 Technical Guide*. Dell Inc., Round Rock, TX, 2013.

[162] J. Hu. *Biophysically Accurate Brain Modeling and Simulation Using Hybrid MPI/OpenMP Parallel Processing*. Texas A&M University, College Station, TX, 2012.

[163] T. Netschlager, W. Maass, and H. Markram. The "liquid computer": A novel strategy for real-time computing on time series. *Special Issue on Foundations of Information Processing of TELEMATIK*, 8(1):39–43, 2002.

[164] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14(3):326–334, 1965.

[165] J. J. Hopfield and C. D. Brody. What is a moment? "Cortical" sensory integration over a brief interval. *Proceedings of the National Academy of Science*, 97(25):13919–13924, 2000.

[166] M. Xu, L. Duan, J. Cai, L. Chia, C. Xu, and Q. Tian. HMM-based audio keyword generation. In *Advances in Multimedia Information Processing - PCM 2004: 5th Pacific Rim Conference on Multimedia*, pages 566–574. Springer, New York, NY, 2004.

[167] M. Sahidullah and G. Soha. Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition. *Speech Communication*, 54(4):543–565, 2012.

[168] Richard F. Lyon. A computational model of filtering, detection, and compression in the cochlea. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1282–1285, 1982.

[169] L. Van Immerseel and J. P. Martens. Pitch and voice/unvoiced determination with an auditory model. *The Journal of the Acoustical Society of America*, 91(6):3511–3526, 1993.

[170] B. Schrauwen and J. Van Campenhout. BSA, a fast and accurate spike train encoding scheme. In *Proceedings of the International Joint Conference on Neural Networks*, pages 2825–2830, 2003.

[171] M. Slaney. *Lyon's Cochlea Model*. Apple Computer, Inc, Cupertino, CA, 1988.

[172] P. Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, Cambridge, MA, 2001.

[173] D. O. Hebb. *The Commentoutorganization of Behavior*. Wiley & Sons, New York, NY, 1949.

[174] E. L. Bienenstock, L. N. Cooper, and P. W. Munro. Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2(1):32–48, 1982.

[175] P. I. Good and J. W. Hardin. *Common Errors in Statistics (and How to Avoid Them)*. Wiley, Hoboken, NJ, 4 edition, 2012.

[176] N. Brunel, F. Carusi, and S. Fusi. Slow stochastic Hebbian learning of classes of stimuli in a recurrent neural network. *Network: Computation in Neural Systems*, 9(1):123–152, 1998.

[177] W. Senn and S. Fusi. Learning only when necessary: Better memories of correlated patterns in networks with bounded synapses. *Neural Computation*, 17(10):2106–2138, 2005.

[178] W. Senn and S. Fusi. Convergence of stochastic learning in perceptrons with binary synapses. *Physical Review E*, 71(6):061907, 2005.

[179] N. Bertschinger and T. Natschlager. Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7):1413–1436, 2004.

[180] R. Legenstein and W. Maass. What makes a dynamical system computationally powerful? In *New Directions in Statistical Signal Processing: From Systems to Brains*, pages 127–154. The MIT Press, Cambridge, MA, 2007.

[181] R. Legenstein and W. Maass. Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, 20(3):323–334, 2007.

[182] D. V. Buonomano and W. Maass. State-dependent computations: Spatiotemporal processing in cortical networks. *Nature Reviews Neuroscience*, 10(2):113–125, 2009.

[183] M. Tsodyks and S. Wu. Short-term synaptic plasticity. *Scholarpedia*, 8(10):3153, 2013.

[184] W. Cui, H. Chen, and Y. Han. VLSI implementation of universal random number generator. In *Proceedings of the Asia-Pacific Conference on Circuits and Systems*, 2002.

[185] Y. Zhang, B. Yan, M. Wang, J. Hu, and P. Li. Linking brain behavior to underlying cellular mechanisms via large-scale brain modeling and simulation. *Neurocomputing*, 97:317–331, 2012.

[186] E. M. Izhikevich. Polychronization: Computation with spikes. *Neural Computation*, 18(2):245–282, 2006.

[187] G. Huang, D. Wang, and Y. Lan. Extreme learning machines: A survey. *International Journal of Machine Learning and Cybernetics*, 2(2):107–122, 2011.

[188] P. Maragos and A. Potamianos. Fractal dimensions of speech sounds: Computation and application to automatic speech recognition. *The Journal of the Acoustical Society of America*, 105(3):1925–1932, 1999.

[189] W. Pan, Z. Zhang, and H. Liu. Multistability of genetic regulatory networks. *International Journal of Control*, 41(1):107–118, 2010.

[190] P. Wellstead and M. Cloutier. Modelling and simulation of brain energy metabolism: Energy and Parkinson's disease. In *Systems Biology of Parkinson's Disease*, pages 19–38. Springer, 2012.

[191] J. Kim and B. Horwitz. How well does structural equation modeling reveal abnormal brain anatomical connections? An fMRI simulation study. *Neuroimage*, 45(4):1190–1198, 2009.

[192] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Large vocabulary continuous speech recognition with context-dependent DBN-HMMs. In *Proceedings of 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4688–4691, 2011.

[193] S. Mysore, B. Agrawal, N. Srivastava, S. Lin, K. Banerjee, and T. Sherwood. Introspective 3D chips. *ACM SIGOPS Operating Systems Review*, 40(5):264–273, 2006.