Oscillatory Neural Systems

by

Connor Bybee

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computation Biology

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

,
Professor Friedrich T. Sommer, Co-chair
Professor Haiyan Huang, Co-chair
Professor Bruno Olshausen
Assistant Professor Peter H Sudmant

Fall 2022

Oscillatory Neural Systems

Abstract

Oscillatory Neural Systems

by

Connor Bybee

Doctor of Philosophy in Computation Biology

University of California, Berkeley

,

Professor Friedrich T. Sommer, Co-chair
Professor Haiyan Huang, Co-chair

The brain, while being small, low-power, and robust, performs complex computations that we cannot yet replicate or fully understand. Oscillatory signals are ubiquitously observed in the brain across multiple scales, e.g., from individual neural membranes to large-scale averages measured in electroencephalograms. Explaining the computational function and generation of brain oscillations is an active area in neuroscience. Computation with oscillatory signals is also interesting from an engineering standpoint in the field of analog computing. Digital computing represents objects with discrete, Boolean variables. In contrast, analog computing investigates how to use the continuous dynamics of physical systems to perform fast, energy-efficient computing. The potential advantage of analog computing has been hard to realize due to the challenges of working with analog systems and competing with rapid advances in digital computing. Recently, neuromorphic computers and coupled oscillator networks have shown potential as efficient analog computers for certain applications. Thus, motivated by neuroscience as well as engineering, here we explore computations in oscillatory systems that efficiently perform specific functions. Our results demonstrate that models of computation using oscillator neural networks can be used as tools for neuroscience and as the basis of efficient analog computers. Specifically, we investigate inference in feedforward deep neural networks, the analog implementation of associative memories, and optimization performed through the dynamics of coupled oscillator networks.

Presumably, the function of a brain critically relies on a combination of continuous and discrete signals, e.g., membrane voltages in neurons and their averages, local field potentials, and action potentials or spikes. Neuromorphic computers that use this combination of signaling are emerging as alternatives to traditional computers for certain tasks. How information is encoded by spiking neural activity can impact key efficiency metrics, such as the number of

transmitted spikes required to perform a calculation. Chapter 2 proposes an efficient coding method for implementing deep artificial neural networks in which the times of spikes encode the phase in an ongoing rhythm. The proposed phase code is advantageous because it uses significantly fewer spikes per neuron for each calculation than a rate code, the most common encoding method in neuromorphic computing. In addition, we present results obtained from an implementation of phase-coded deep neural networks on neuromorphic hardware.

Networks of coupled oscillators are being investigated for efficient implementation of machine learning and artificial intelligence algorithms, such as associative memory. Chapter 3 presents new models of associative memories implemented in networks of coupled oscillators using discretized Q-state phase codes. We show that the memory capacity for 3-state phase codes is significantly higher than for traditional binary 2-state codes. Further, we present a new oscillator model that is capable of implementing Q-state and continuous associative memories with sparse activity patterns.

The use of Ising machines, i.e., large networks of interacting 2-state elements, have been proposed as a way for finding near-optimal solutions to combinatorial optimization problems. We argue that current Ising machines are limited due to a focus on second-order, or pairwise, interactions. Chapter 4 explores new methods for finding solutions to combinatorial optimization problems through the use of oscillator Ising machines with higher-order interactions, referred to as higher-order oscillator Ising machines. We present results comparing second-order oscillator Ising machines to higher-order oscillator Ising machines from solving benchmark optimization problems. We show that for benchmark satisfiability problems, higher-order Ising machines require fewer optimization variables and network connections. In addition, we show that higher-order Ising machines find solutions that satisfy a greater fraction of problem constraints compared to existing methods.

I dedicate this work to

my friends and family

Contents

List of Figures

List of Tables

Acknowledgments

We become like those we interact with; therefore, surround yourself with those that make you better. I'm lucky to have people in my life who are lovely and exceptional in many ways. I've taken from them those parts that make up the good in me.

I'd first like to thank my parents. My wonderful mother Julia Molony, who has always encouraged and supported me, and continues to be a loving role-model in my life. I am also lucky to have as a mentor my father Jerald Bybee, who is an extraordinary person and has shown me how to be curious, seek adventure, and make the most out of every moment.

I could not be luckier to have my siblings and their spouses in my life as well. My brother Keevin continues to challenge me, encourage me, and show me what it is like to always be bettering one's self. My sister Meghann has shown me how to have endless motivation, ambition, and kindness. My brother Patrick continues to demonstrate how to learn and seek perfection. I look up to my younger brother Phelan whose wisdom and support have been essential to my successes. My sister Erin is a loving person and has taught me to always be optimistic even in times of trouble.

I am also grateful to have been influenced by and had the support of many remarkable friends and colleagues. They've encouraged and enabled me to pursue my dreams. William Nicholson has inspired me to stay sharp, go big, and take risks. I would not have pursued a PhD without the support and encouragement of Barbara Simpson.

My work was influenced greatly by the faculty that advised me during my time as a graduate student. First is my mentor Friedrich Sommer, Fritz, who in guiding me through the PhD, provided creative and moral support, opportunities within the research community, and many hours of fantastic conversation and brainstorming. I'm incredibly grateful to have been his student. I'm also privileged to have the mentorship of Bruno Olshausen, who invited me to collaborate in his lab. These collaborations have become a part of my dissertation.

My work and personal life were supported by fellow students, postdocs, and other members of the Redwood center: Vasha Dutell, Mike Fang, Chris Warner, Sophia Sanborn, Chris Kymn, Christian Shewmake, Alex Belsten, Galen Chuang, Steven Lee, Jamie Simon, Yubei Chen, Spencer Kent, Dylan Paiton, Ryan Zarcone, Neha Wadia, Shariq Mobin, Brian Cheung, Mayur Mudigonda, Jesse Livezey, Paxon Frady, Zengyi Li, Alex Anderson, Pratik Sachdeva, Guy Isley, Charles Frye, Chris Hillar, Eric Dodds, Eric Weiss, Gautam Agarwal, James Arnemann, Jasmine Collins, Louis Kang, Saeed Saremi, Steven Shepard, Sylvia Madhow, Adrianne Zhong, Denis Kleyko, Ping-chen Huang, Amir Khosrowshahi, Pentti Kanerva, and Jeff Teeters.

I am also grateful for the support of my collaborators and friends from the Center for Computational Biology community: Diana Aguilar-Gomez, Chenling Xu, Henry Pinkard, Gonzalo Benegas, Ryan Chung, Shihab Dider, Sandra Hui, Maya Lemmon-Kishi, Stacy Li, Amanda Mok, Graham Northrup, Helen Sakharova, Isabel Serrano, Jordi Silvestre-Ryan, Tal Ashuach, Nicholas Everetts, Carlos Buen Abad Najar, Brooke Rhead, Robert Tunney, Fiona Callahan, and Galen Xing.

Finally, I am thankful for the sources of funding that supported me: The United States taxpayers through the National Science Foundation (NSF) Graduate Research Fellowship Program (GFRP) fellowship, a grant from the Intel Labs Neuromorphic Computing Lab, a grant from Intel Components Research, and a grant from the UC Berkeley Student Technology Fund.

Chapter 1

Introduction

This thesis is motivated by an interest in neuroscience, machine learning, and engineering. Therefore, it focuses on the use of analog oscillatory neural networks for efficient computation. This topic is interesting from a neuroscience perspective because oscillations are ubiquitously observed at multiple scales in the brain. Not all oscillatory behavior has been explained and characterized, and the understanding of many oscillations is lacking. By examining the computational properties of oscillatory systems, insight may be gained into the functional role of oscillations in the brain. Oscillatory systems are also interesting from an engineering perspective. Coupled oscillator networks are being investigated for the efficient implementation of certain machine learning algorithms. Also, at the heart of almost every radio and digital computer, you will find an oscillator. Oscillations can act to synchronize circuits, coordinate computation, and serve as a carrier for information transfer. Motivated by these facts, this thesis aims to show that networks of coupled oscillators can perform computations that are interesting to both neuroscientists and engineers. We show that a variety of functions can be implemented through the dynamics of coupled oscillator networks including feedforward inference in artificial deep neural networks, error correction in attractor neural networks, and optimization of hard combinatorial problems. By proposing new models and providing new empirical and theoretical results, this dissertation suggests potential advantages of analog computation based on coupled oscillator networks compared to existing systems.

1.1 Spiking neural networks

Information representation is a key aspect of efficient computing systems. For real-world applications and biological systems, the way signals are represented and manipulated can significantly impact energy consumption and the time required to perform a calculation. The brain appears to represent some information in the form of action potentials, also known as spikes, which are fast changes in the electric potential of the neural membrane. There is also interest in the construction of low-power, low-latency brain-inspired or neuromorphic computers based on the same computational principles of the brain. Potentially, there exists efficient phase codes, inspired by the brain, that could improve neuromophic computers. Therefore, chapter 2 examines feedforward inference in deep artificial neural networks that have been efficiently implemented in networks of spiking oscillator neurons.

Neuromorphic computers, such as spiking neural networks, are being considered for the low-power and low-latency implementation of deep artificial neural networks. The particular way information is encoded as spikes has an important impact on power consumption and the time required to perform a calculation. Additionally, in neuroscience, characterizing the way information is represented as spikes is an active area of research. A common strategy for understanding the relationship between a stimulus or input and the observed neural activity is through the idea of neural codes. Well-studied neural codes include rate codes, timing codes, phase codes, and population codes. In rate codes, information is coded as the number or instantaneous rate of spikes. In timing codes, information is coded by the timing of individual spikes. In phase codes, information is coded as the timing of spikes relative to an ongoing oscillation. Lastly, in population codes, information is coded as the coordinated activity of multiple neurons.

This study examines efficient neural codes representing the phase of an oscillation with the time of a spike relative to a background oscillation. In contrast to rate codes where a signal is encoded by multiple spikes, oscillatory neural codes or phase codes encode a signal that can be represented by a single or few spikes. To illustrate this concept Figs. 1.1 and 1.2 present examples of a rate-coded neuron. Figs. 1.1 shows that a real-valued variable can be encoded into the spike rate, R, of an abstract neuron measured in spikes per second. In this example, there is a linear relationship between the encoded variable and the spike rate. In rate codes, there is a trade-off between the precision or accuracy with which a variable can be represented and the number of spikes or integration time. Fig. 1.2 shows an example of this trade-off. In order to increase the precision or accuracy, either the maximum spike rate, $R_{\max}$, can be increased or the integration period can be increased. The plot on the left shows a membrane potential encoded into spikes accumulated over a 1sec interval. The figure on the right shows the same range of values accumulated over 100msec. There is a loss of information compared to the longer integration period. Larger ranges of membrane potentials map to the same spike rate. Phase coded neurons are able to present the phase of an oscillation with the time of spikes, see Fig. 1.5. Motivated by these findings, we present results demonstrating the efficient implementation of deep artificial neural networks in phased coded spiking neural networks where neural activity can be robustly represented in a few spikes.

1.2 Oscillator associative memories

Associative memories are data structures used for the storage and robust retrieval of high-dimensional patterns. Chapter 3 examines autoassociative memories based on complex-valued representations, phasors, and their implementation in coupled oscillator networks.

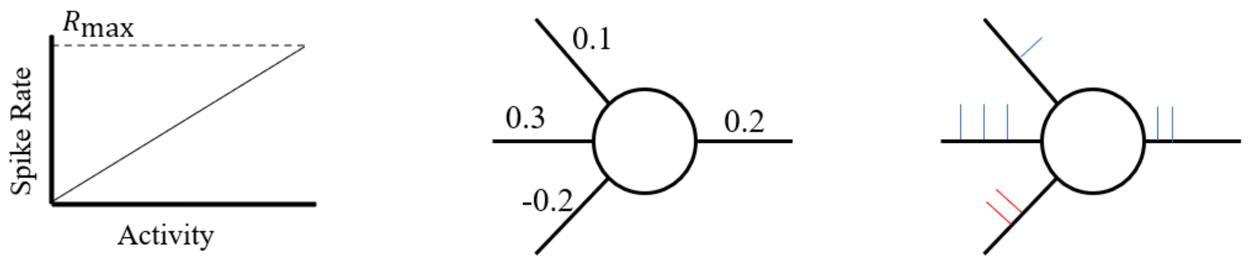


Figure 1.1: **Rate-coded neuron.** Left) A plot of a real-valued scale variable, activity, encoded linearly into a spike rate. Middle) An abstract point neuron with three real-valued inputs. The output is a linear sum of the inputs. Right) An abstract point neuron receives inputs as a series of spikes.

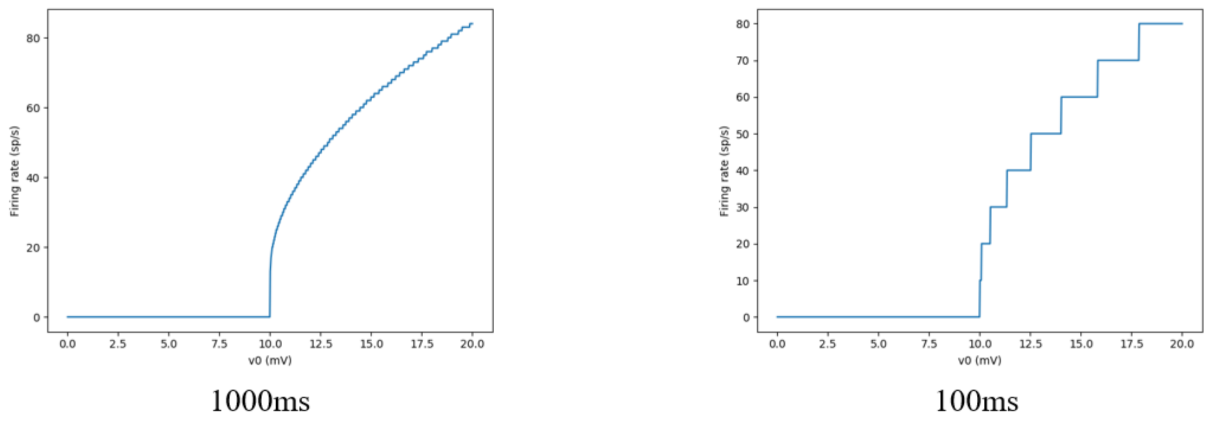
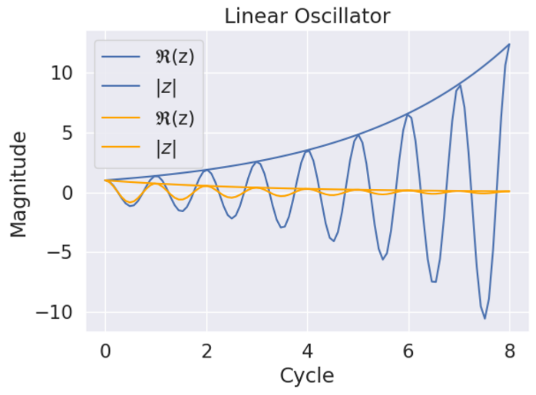


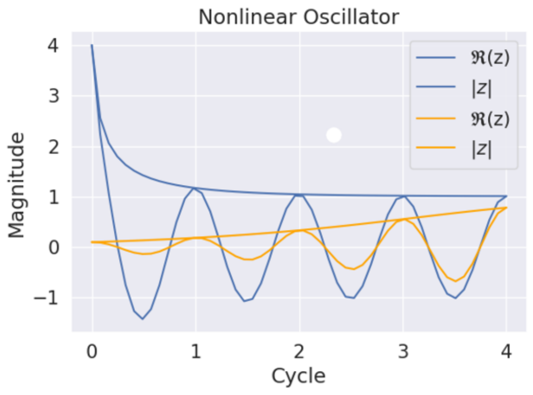
Figure 1.2: **Tuning curve.** Left) A tuning curve for a leaky integrate-and-fire neuron measured over 1000ms time window. Right) A tuning curve for a leaky integrate-and-fire neuron measured over 100ms time window.

Studies of associative memories have largely focused on units with binary activity even though it has been shown that units with multi-state activity have useful properties including increased pattern capacity and information content. Multi-state units can be efficiently represented by phasors. We present empirical results validating predictions made by mean-field theory on the capacity and information content of Q-state phasor associative memories. In addition, we report new results on the capacity and information content for larger values of Q. Finally, we propose a novel model for Q-state phasor associative memories with sparse pattern activity.

The efficient implementation of associative memories in hardware is an active area of research. Most methods for implementing autoassociative memories with phasor neural networks have used discrete-time iterative update dynamics in digital computers. Implementing associative memories in analog systems, e.g., networks of biological neurons or electrical components with continuous-valued states, may provide advantages over digital computers, but



(a)



(b)

Figure 1.3: **Linear and nonlinear oscillators.** a) Real part of the complex-valued variable and magnitude of the complex-valued variable for a linear oscillator with two different gain coefficients. b) Real part of the complex-valued variable and magnitude of the complex-valued variable for a nonlinear limit-cycle oscillator with two different initial conditions.

also introduces new challenges. Specifically, the functional properties of analog systems must be shaped in order to make the stored patterns fixed points of the continuous-time dynamical system. Oscillators are a promising choice for the analog implementation of associative memories because phasors map naturally to the oscillatory signals. In addition, oscillators are a promising candidate for use in analog computers because they can be manufactured at a large scale with current technologies, operate at high frequency, and have low power consumption. Therefore, we present a coupled oscillator network model capable of implementing Q-state associative memories that successfully recalls stored patterns in the presence of noise. We present new empirical results on the basins of attraction around stored patterns for different values of Q.

There are several oscillator models to choose from. We choose an oscillator model that has stable limit-cycle behavior and can synchronize to input. Figs. 1.3 and 1.4 demonstrate the concepts of limit-cycles oscillations and synchronization, respectively. Fig.1.3a presents a linear oscillator with damped behavior and unstable growth. Fig.1.3b presents a nonlinear oscillator that gives rise to stable limit-cycle oscillators. The introduction of nonlinearities to the oscillator dynamics allows for a greater richness of behavior.

An oscillator network can synchronize to phase-locked states, i.e., the relative phase of each oscillator remains constant and there is a fixed phase shift between oscillators. Of particular interest are the cases when the phase shift between oscillators takes specific values, e.g., zero for all oscillators. Fig. 1.4 presents an oscillator network where all the oscillators synchronize to phase-locked states.

Spiking neurons can also be used to construct limit-cycle oscillators that phase-lock to inputs. Fig. 1.5 presents results from a simulation of a neuron model that produces oscillatory

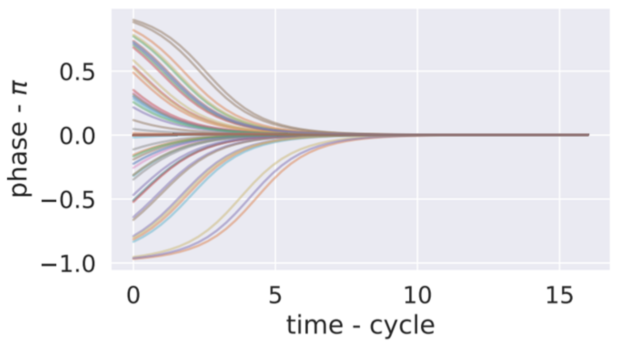


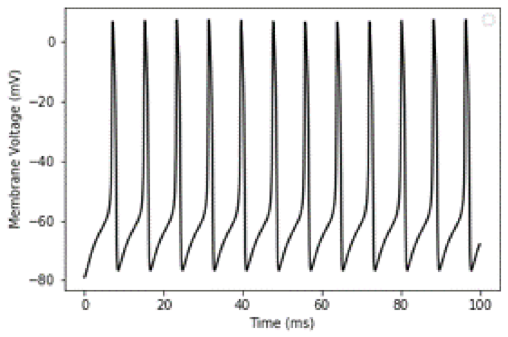
Figure 1.4: **Synchronization in a simple kuramoto model.**

spiking behavior. Fig 1.5a shows the membrane potential over time. Fig.1.5b shows the phase space of the two dynamic parameters. The state of the two parameters traces out a stable orbit in phase space. Fig. 1.5c shows the phase response curve (PRC) for the neuron model. This particular type of PRC allows the neuron to phase-lock to input.

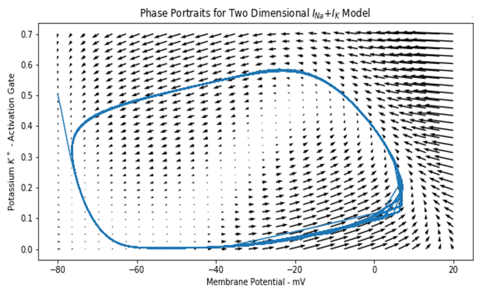
Finally, an important class of associative memories is one that stores patterns with sparse activity, i.e., the individual units in the network may either be active or inactive. To date, it is unknown how to implement sparse associative memories in networks of coupled oscillators with continuous coupling. We present a novel oscillator model that successfully implements a sparse associative memory.

1.3 Optimization with oscillator networks

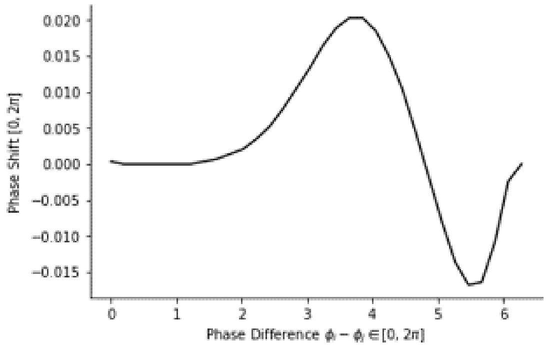
Finding optimal or near-optimal solutions to combinatorial optimization problems has important applications in many disciplines. Much of the existing work solving combinatorial optimization problems has focused on search-based or heuristic methods. Recently, there has been growing interest in Ising machines, i.e., methods which use the inherent parallel dynamics of physical systems to find solutions to combinatorial optimization problems extremely fast and with low power consumption. Chapter 4 examines solving combinatorial optimization problems with coupled oscillator networks. Among these methods, many of the systems being investigated make use of pair-wise or second-order interactions. Many combinatorial optimization problems naturally contain higher-order interactions between the optimization variables. Therefore, mapping combinatorial optimization problems to hardware requires converting the higher-order objective function to second-order. This conversion introduces auxiliary variables. In effect, the required resources, in terms of the number of variables and connections between variables, increases. This study examines solving combinatorial optimization problems with higher-order objective functions directly with coupled oscillator networks. We find that not only do higher-order formulations of combinatorial optimization



(a)



(b)



(c)

Figure 1.5: **Spiking neuron oscillator.** a) Oscillations emerge from a neuron with spikes occurring at a constant frequency. b) The phase space for the two parameter dynamical system describing the neuron. The horizontal axis represents the membrane voltage. The vertical axis represents the potassium channel activation. c) The phase response curve for the neuron oscillator. The horizontal axis represents the phase difference between the input phase and the current phase. The vertical axis represents the phase shift resulting from the input.

problems result in fewer network resources (in terms of the number of variables and their connections), but also the optimization with the higher-order Ising machine results in solutions to the combinatorial optimization problem that satisfy a greater fraction of problem constraints.

Chapter 2

Spiking Phasor Deep Neural Networks

Spiking Neural Networks (SNNs) have attracted the attention of the deep learning community for use in low-latency, low-power neuromorphic hardware, as well as models for understanding neuroscience. Of particular interest are strategies that efficiently encode information into the timing of action potentials, also known as spikes. In this paper, we introduce Spiking Phasor Neural Networks (SPNNs). SPNNs are based on complex-valued Deep Neural Networks (DNNs), representing phases by spike times. In our model, a complex-valued neural network is trained and then mapped to a SNN. The phases of the complex-valued neurons are represented using spike times in the SNN. We train SPNNs on CIFAR-10, and demonstrate that the performance exceeds that of other timing-coded SNNs, approaching results with comparable real-valued DNNs.

2.1 Introduction

The highest classification accuracy on image recognition tasks has been achieved by real-valued deep neural networks with floating point precision. Recent efforts have attempted to reformulate deep learning into formats more suitable for hardware acceleration and low-energy edge computing. This work aims to demonstrate the benefits of using spike timing codes in a way that can advantage deep learning acceleration according to specific metrics, especially energy efficiency and latency.

Implementation of DNN algorithms on next-generation computing platforms, such as neuromorphic computer architectures and networks of coupled oscillators, is promising and may prove to be more energy-efficient and faster than current hardware based on the Von Neumann architecture [83, 68, 21]. SNNs are typically challenged by a non-differentiable objective function, which prohibits direct training with backpropagation. Successful methods include training artificial neural networks with the purpose of converting them to SNNs, using spiking versions of backpropagation compatible with spike timing-dependent-plasticity, and the other learning rules [56, 64]. In addition, when mapping DNN neurons to SNNs, the machine learning and neuroscience community have largely focused on the use of rate codes [42, 56, 79, 74]. Rate codes represent a single real-valued scale with multiple spikes. Some timing codes, e.g., latency codes, improve upon rate codes by representing real-valued variables with a single spike but are less robust to noise. This paper presents a method for using spike events to represent complex-valued neural states, which can easily be differentiated in the complex domain. In addition, the phase code presented in this work robustly represents real-valued phases with a few spikes, achieving a balance between rate codes and latency codes.

In a rate code, a stimulus feature is represented by the firing rate of a neuron measured by the count of spikes in an extended time window. Conversely, in a spike timing code, a feature is represented by the precise timing of a single spike. Potential benefits of spike timing codes include (i) efficiency in terms of the number of spikes per computation, and (ii) computation speed, since the duration of a step of neural update with a rate code cannot be shorter than the integration window required for estimating the lowest rate. On the other hand, spike timing codes are often regarded as brittle and not suited for implementing robust computations [58].

Recently, Thresholding Phasor Associative Memories (TPAM) were introduced [25] that illustrate how a complex-valued neural state can be directly mapped to a spike timing code. Essentially, relative to an ongoing oscillation, the timing of the spike can be used to indicate the phase of a complex-valued number. This was shown to provide robust spike timing codes in the context of attractor neural networks. Here, we extend this timing code to feedforward deep networks and show how backpropagation-based learning can be applied in the complex domain and robustly mapped to SNNs. Motivated by that aim, this paper demonstrates deep learning in spiking phasor networks on benchmarks MNIST [55] and CIFAR-10 [52].

2.1.1 Spiking backpropagation

Supervised learning of spike timing codes was explored in [64]. In [64], neurons are trained to integrate input and fire at a precise time upon crossing a threshold. Training is performed by first determining the set of input spikes leading to an output spike, called the causal set. Parameter updates are then calculated by backpropagating through the causal set. Our work presents a method to train a spike timing coded SNN without the need to calculate a casual set.

A method for converting real-valued ANNs to timing coded SNNs was introduced in [80]. It demonstrates on a simple machine learning task (MNIST) that spike timing codes are more efficient than rate codes and real-valued DNNs when considering the number of operations needed to perform a computation. Our paper extends spike timing coded SNNs to CIFAR-10.

2.1.2 Neuroscience

In theoretical neuroscience, it is an open question how a network of spiking biological neurons can precisely coordinate spikes over time and space. Izhikevich proposed a spiking network

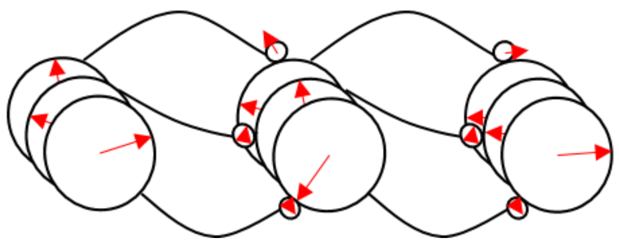


Figure 2.1: **An illustration of a feed forward PNN with normalizing activation function or Spiking PNN (SPNN).**

that creates "polychronous" spiking patterns, as a model for describing neural dynamics in brain circuits [45]. Polychronization means that neurons exhibit time-locked spiking patterns, but with arbitrary spike timing and not perfectly synchronous spiking. The spiking patterns we propose are also polychronous patterns, but these are generated by different mechanisms.

Precise spike timing patterns are observed in multiple brain regions and cell types. The electrical properties of the neuron membrane give rise to sustained intrinsic spike oscillations, subthreshold oscillations, and membrane resonance [44]. Neuron morphology and localized genetic expression patterns of membrane ion-channels can create compartments with specific electrical properties [7, 61]. Rate and timing codes have been explored as a method of mapping ANNs to models of biological plausible neural networks [82, 81, 64, 74]. Inspired by these observations, this study develops SNNs with spike timing codes based on oscillator neural networks.

2.2 Methods

2.2.1 Spiking Phasor Neural Networks

Complex-valued neural networks have been explored in several contexts. Noest [72] introduced phasor neural networks (PNN) with applications as an associative memory and suggested use with backpropagation. Complex domain backpropagation was introduced in [26] and [34]. Recent developments have been made with deep neural networks [89]. We provide a concrete example of a feed forward PNN trained with backpropagation to implement spike timing codes. This paper refers to PNNs with the thresholding phasor associative memory activation function [25] as Spiking PNNs (SPNNs).

In PNNs, neurons are represented by complex-valued phasors $s_i \in \mathbb{C}$. Where $s_i = r_i e^{\mathrm{i}\theta_i} = r_i \cos\theta_i + \mathrm{i} r_i \sin\theta_i = u_i + \mathrm{i}v_i$. Here, $\mathrm{i} = \sqrt{-1}$. Phasors are restricted to the unit circle with the use of a non-linear activation function that preserves phase information. $|s_i| = 1, \forall i \in \{1, .., N_s\}$

The network units typically consist of a set of input, output, and hidden variables, $\mathbf{s} = \{\mathbf{x}, \mathbf{y}, \mathbf{h}\} \in \mathbb{C}^{N_s}$. The synaptic weights and neuron biases, $W_{ij} \in \mathbb{C}$ and $b_i \in \mathbb{C}$, act as attenuating or amplifying phase shifters.

We present a two-layer network with layer units $\mathbf{h}^{(l)} \in \mathbb{C}^{N_h}, l \in \{1, 2\}$. Input units can be real-valued but are transformed to unit length complex-valued vectors whose phase is a function of the real-valued input. $x_i = e^{\mathrm{i}\theta_i}$, where $\theta_i = g_x(\tilde{x}_i)$, $\tilde{x}_i \in \mathbb{R}$. Where $g_x : \mathbb{R} \to [0, \pi]$. The output of the network is mapped to phases similar to binary phase shift keying. $y_i = e^{\mathrm{i}\theta_i}$, where $\theta_i = g_y(\tilde{y}_i)$, $\tilde{y}_i \in \{0, 1\}$ and $g_y : \{0, 1\} \to \{0, \pi\}$.

Inference proceeds by propagating activity from the input layer to the output layer:

$$\mathbf{h}^{(l)} = f(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}, \Theta) \tag{2.1}$$

The vector $\mathbf{h}^{(l)}$ is the activity of hidden layer l, and for computing the activity in the first hidden layer, equation (2.1) is applied to the input vector, i.e., $\mathbf{h}^{(0)} = \mathbf{x}$. The element-wise thresholding and normalizing function is given by:

$$f(z_i, \Theta) = \begin{cases} \dfrac{z_i}{|z_i|} & \text{if } |z_i| - \Theta > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

where Θ is a trainable thresholding parameter.

The objective is to minimize the phase difference between the network output and targets. In supervised learning tasks, it is common for the softmax with cross-entropy loss to be used with the target variables encoded as one-hot binary vectors. Here, we encode class labels onto the unit circle. Positive class labels are encoded to lead negative class labels in phase. A classification is successful when the phase of the positive class leads the phase of all other outputs. The class targets are binary variables encoded onto the unit circle, similar to binary phase shift keying. Positive and negative classes are out of phase by 180° or π radians. The loss function is given by:

$$L = \frac{1}{2}\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 = N_y - \sum_{i=1}^{N_y} \cos(\theta_i - \hat{\theta}_i) \tag{2.3}$$

Where θ_i are the target phase angles and $\hat{\theta}_i$ are the estimated phase angles. Minimizing the objective corresponds to phase aligning the output and targets:

$$\frac{\partial L}{\partial \hat{\theta}_i} = \sin(\theta_i - \hat{\theta}_i) \tag{2.4}$$

As we will show in the results, PNNs with the TPAM activation function can be trained with the backpropagation algorithm to minimize equation 2.3.

2.2.2 Complex domain backpropagation

In [26], complex domain backpropagation was derived for the activation function below.

$$\begin{aligned} y &= f(z) \\ &= \frac{z}{c + \frac{1}{r}|z|} \\ &= u + \mathrm{i}v \end{aligned}$$

Where, $z = a + \mathrm{i}b \in \mathbb{C}$, $\{a, b\} \in \mathbb{R}$ and $|z| = \sqrt{a^2 + b^2}$. The partial derivatives are

$$\frac{\partial u}{\partial a} = \begin{cases} \frac{r(b^2+cr|z|)}{|z|(cr+|z|)^2} & \text{if } |z| > 0 \\ \frac{1}{c} & \text{if } |z| = 0 \end{cases} \qquad \frac{\partial u}{\partial b} = \begin{cases} -\frac{rab}{|z|(cr+|z|)^2} & \text{if } |z| > 0 \\ 0 & \text{if } |z| = 0 \end{cases}$$

$$\frac{\partial v}{\partial a} = \begin{cases} -\frac{rab}{|z|(cr+|z|)^2} & \text{if } |z| > 0 \\ 0 & \text{if } |z| = 0 \end{cases} \qquad \frac{\partial v}{\partial b} = \begin{cases} \frac{r(a^2+cr|z|)}{|z|(cr+|z|)^2} & \text{if } |z| > 0 \\ \frac{1}{c} & \text{if } |z| = 0 \end{cases}$$

Considering the case where $\Theta = 0$ and $|z| > 0$, the SPNN activation function is found by setting $r = 1$ and in $\lim_{c \to 0}$. The partial derivative can be written in the complex domain by adding together the individual terms.

$$(\frac{\partial u}{\partial a} + \frac{\partial v}{\partial a}) + \mathrm{i}(\frac{\partial u}{\partial b} + \frac{\partial v}{\partial b}) = \frac{(b^2 - ab) + i(a^2 - ab)}{(a^2 + b^2)^{3/2}}$$

2.2.3 Spiking neuron model

The continuous-valued phasor network can be mapped to discrete spiking events in time. For a given choice T for the length of a cycle, the each phase angle, $\theta_i \in [0, 2\pi]$ can be mapped to a time $t_i = \theta_i * T/2\pi$.

We first consider a network where all layers are phase-locked to the input layer. The real-valued inputs are mapped to unique phases on the unit circle. This creates an ambiguity between the largest and smallest values since 2π is equivalent to 0. Therefore, we limited the values to $[0, \pi]$. We chose this option since it is simple and is sufficient for demonstration purposes. Each input phasor can also be assigned a fixed, random phase shift.

In the first cycle, the input units spike and each synapse is activated with magnitude $|W_{ij}|$ after a delay of $\theta_{ij} * T/2\pi$. Each neuron integrates the information from the current cycle and spikes after the appropriate delay during the subsequent cycle. Therefore, each cycle propagates signals to adjacent layers, see figure 2.4.

2.2.4 Circuit Model

We simulate a circuit model to demonstrate that SPNNs can be implemented by simple electrical components and are robust to noise induced by numerical simulation and changing

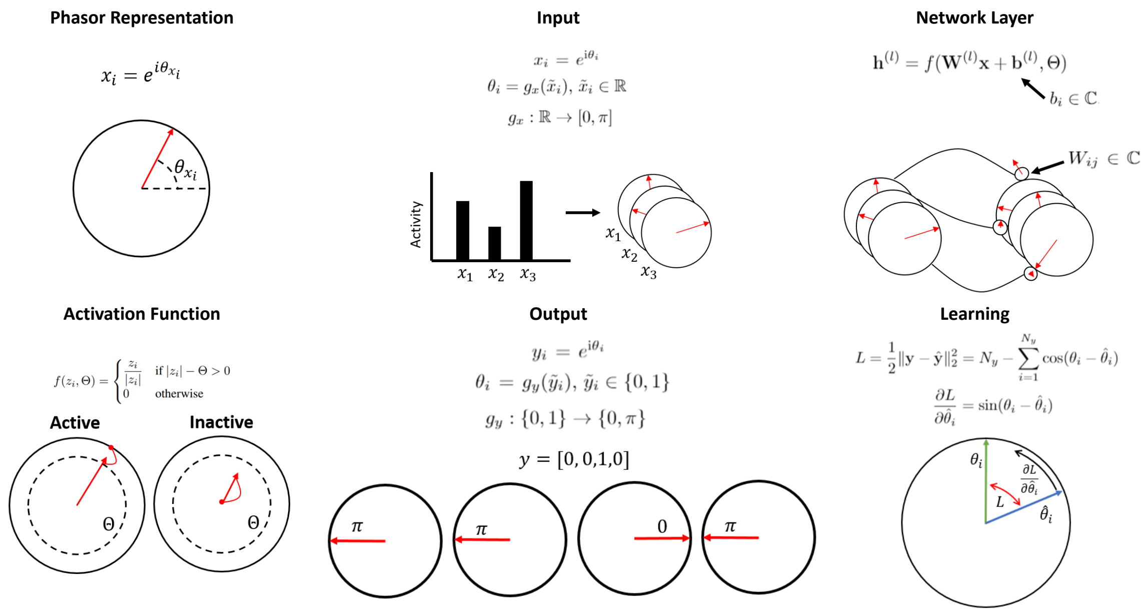

Phasor Representation
$x_i = e^{i\theta_{x_i}}$
θ_{x_i}
Input
$x_i = e^{i\theta_i}$
$\theta_i = g_x(\tilde{x}_i),\ \tilde{x}_i \in \mathbb{R}$
$g_x : \mathbb{R} \to [0, \pi]$
Activity
x_1 x_2 x_3
Network Layer
$\mathbf{h}^{(l)} = f(\mathbf{W}^{(l)}\mathbf{x} + \mathbf{b}^{(l)}, \Theta)$
$b_i \in \mathbb{C}$
$W_{ij} \in \mathbb{C}$
Activation Function
$f(z_i, \Theta) = \begin{cases} \frac{z_i}{|z_i|} & \text{if } |z_i| - \Theta > 0 \\ 0 & \text{otherwise} \end{cases}$
Active
Inactive
Θ
Output
$y_i = e^{i\theta_i}$
$\theta_i = g_y(\tilde{y}_i),\ \tilde{y}_i \in \{0, 1\}$
$g_y : \{0, 1\} \to \{0, \pi\}$
$y = [0,0,1,0]$
π
0
Learning
$L = \frac{1}{2}\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 = N_y - \sum_{i=1}^{N_y} \cos(\theta_i - \hat{\theta}_i)$
$\frac{\partial L}{\partial \hat{\theta}_i} = \sin(\theta_i - \hat{\theta}_i)$
θ_i
$\frac{\partial L}{\partial \hat{\theta}_i}$
L
$\hat{\theta}_i$

Figure 2.2: **Parts of a feed-forward PNN with normalizing activation function.**

input. A change in soma membrane potential is driven by a leakage current and combined input from dendrites:

$$\frac{\mathrm{d}V_m}{\mathrm{d}t} = \frac{g_l(V_l - V_m) + g_c(V_d - V_m - \overline{V}_d)}{C_m}$$
$$\frac{\mathrm{d}\overline{V}_d}{\mathrm{d}t} = \frac{(V_d - \overline{V}_d)}{\tau_d}$$

V_m is the membrane potential, g_l is the conductance of the leak channel, V_l is the leak reversal potential, g_c is the conductance of between the soma and dendrite, V_d is the dendrite potential, $\overline{V}_d$ is the average dendrite potential, τ_d is average dendrite potential time constant, and C_m is the membrane capacitance. To correct for bias around the threshold voltage, the average dendrite potential is subtracted from the instantaneous dendrite membrane potential. Dendrites integrate current from synapses which generate membrane potential oscillations after a spike arrives:

$$V_d = \sum_i w_i V_{s,i}$$
$$\frac{\mathrm{d}V_{s,i}}{\mathrm{d}t} = \frac{-W_{s,i}}{C_m}$$
$$\frac{\mathrm{d}W_{s,i}}{\mathrm{d}t} = \frac{V_{s,i}}{L} - \frac{W_{s,i}}{\tau_s}$$

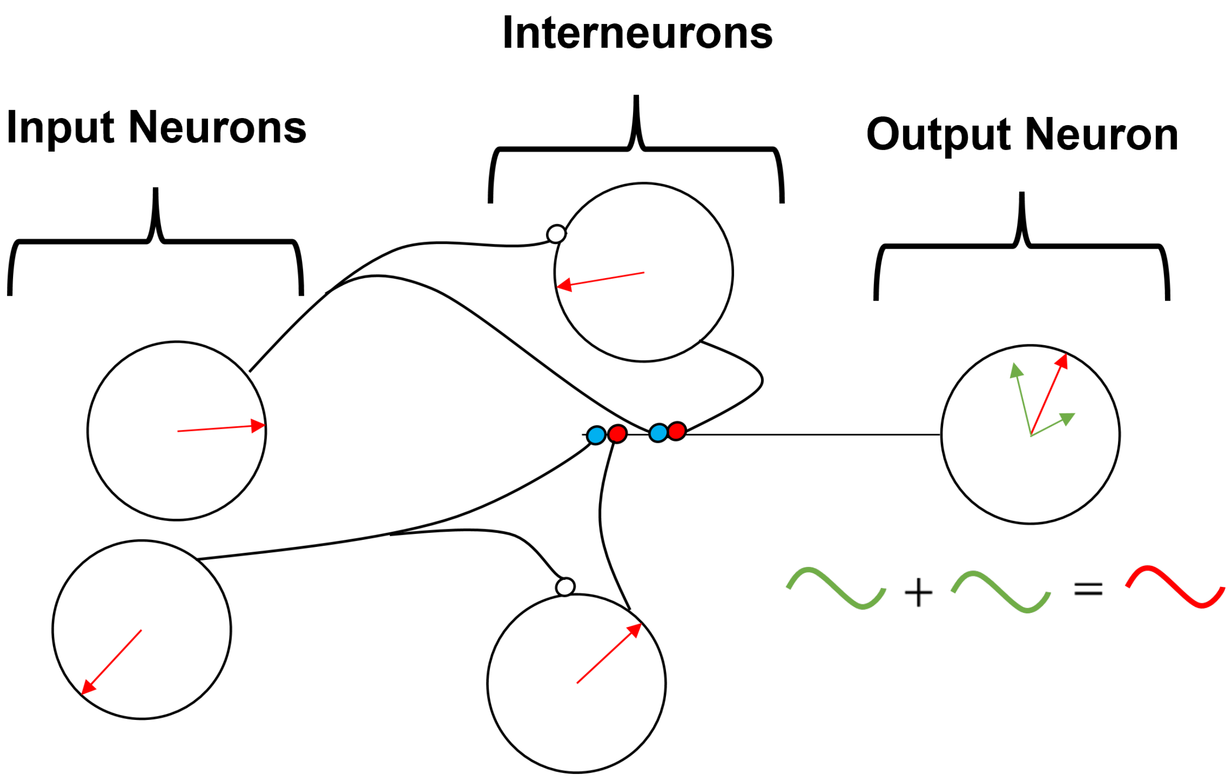


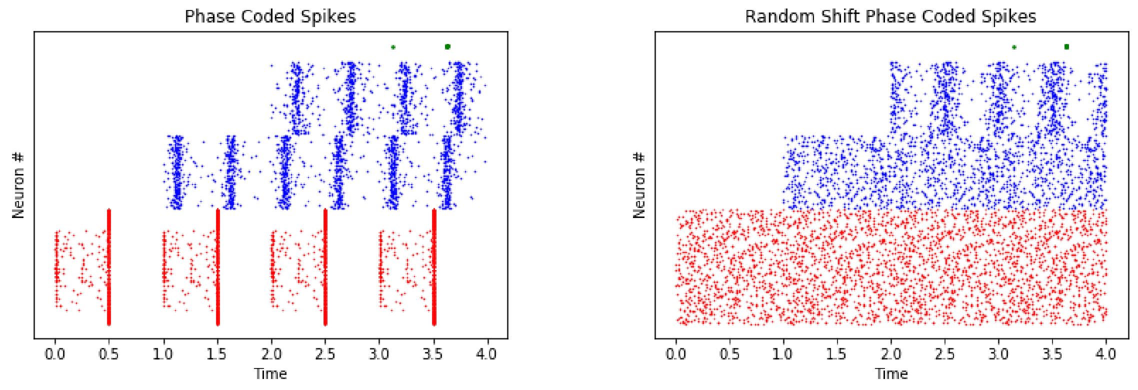
Figure 2.3: **Illustration of a spiking phasor neural network implemented by leaky integrate-and-fire neurons.**

$V_{s,i}$ is the potential at synapse i, w_i is the synaptic weight. $W_{s,i}$ are activation parameters which act to generate membrane potential oscillations at a specific frequency, L is a constant used to tune the membrane potential frequency, and τ_s is the synaptic time constant which controls oscillation damping. A spike resets the synapse parameters such that $V_{s,i} = 0$ and $W_{s,i} = W_{spike}$. The neuron spikes when the soma membrane potential cross a threshold. The refractory period is maintained until the soma membrane potential becomes negative.

2.3 Results

For visualization purposes, we train a fully connected SPNN on MNIST, 784-512-512-10. We unroll the network activity in time to demonstrate how signals propagate (Fig. 2.4 - left panel). The spike times represent a stable limit cycle. For demonstration purposes, we add a fixed, random phase shift to each component of the input while maintaining correct classification (Fig. 2.4 - right panel). A system that has yet to reach the limit cycle will have a settling time (transient settling dynamics visible later in Fig. 2.8).

Convolutional SPNNs are trained on MNIST and CIFAR-10. The convolutional networks consist of two layers of convolution without padding followed by three fully connected layers,



(a) (b)

Figure 2.4: **Raster plot of SPDNN.**The first three cycles of inference for a feedforward SPNN trained on MNIST. The network is phased-locked to the input signal. Each cycle allows information to propagate to the next layer. Each line in the vertical axis is a neuron. The input, hidden, and output layers are colored red, blue, and green, respectively. The output neurons are accurately separated in phase. The input pixel amplitudes are mapped to phase angles inversely proportional to the magnitude. 2.4a: Similar input amplitudes are phase locked. 2.4b: For demonstration, each input can have a fixed phase shift while still maintaining performance.

conv(6,3x3)→conv(16,3x3)→FC128→FC128→FC10. No normalization or pooling is used. We train using Pytorch [76]. The Adam optimizer [50] is used with default parameters and a learning rate of 0.001. We train small networks compared to state-of-the-art for simulation purposes. Future work will investigate the performance of SPNNs on more difficult tasks and with larger networks.

Training and test error rates with the convolutional SPNNs were measured for classifying MNIST data (Fig. 2.6). Our results were obtained with 32-bit floating point precision, and therefore should be regarded as an upper-bound of the performance SPNN hardware implementations are expected to achieve using parameters from a trained model. The classification precision our model achieved surpasses the best results previously reported for SNNs with spike timing codes [80]. Further, SPNNs approach the performance level of comparable DNNs, our model comes within 0.02% compared to the error rate of the DNN used for conversion to a SNN. Last, our network outperforms previous methods for directly training SNNs with spike timing codes [64].

When trained on CIFAR-10. To our knowledge, there is no previous reports of performing supervised learning and inference with spike timing codes on CIFAR-10. However, due to the rather small network size we used, our results on training and test error rates do not reach the state-of-the-art (Fig. 2.6). It is promising that SPNN approach the performance of real-valued DNNs on CIFAR despite minimizing the mean-squared error opposed to the softmax with cross-entropy loss.

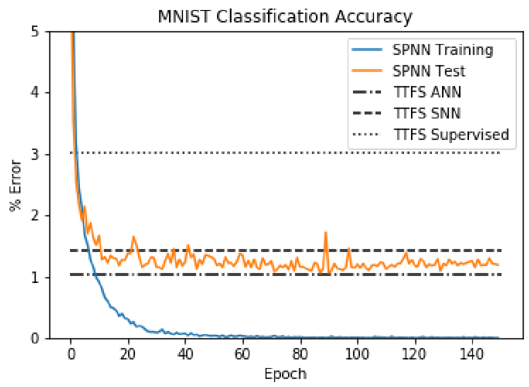
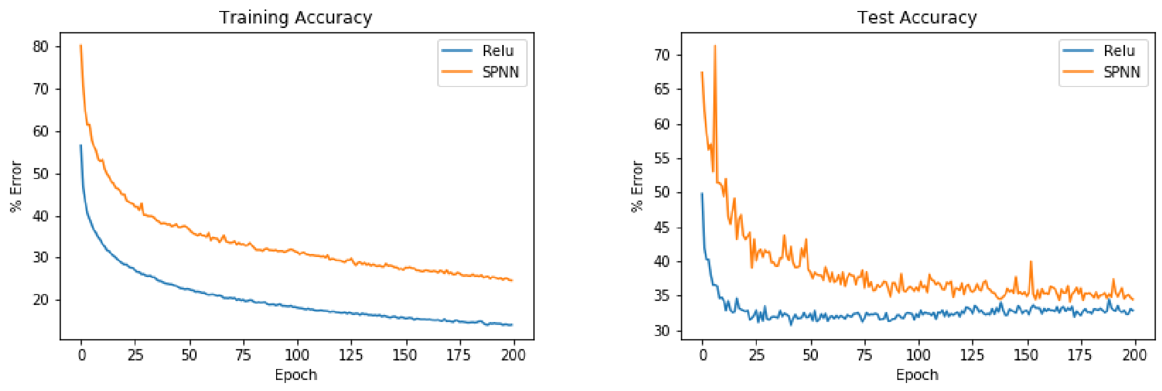


Figure 2.5: **Results for a SPNN on MNIST.** The lowest test error rate previously reported for SNNs with spike timing codes and the ANN used for conversion to the SNN is 1.43% and 1.04%, respectively [80]. The lowest test error rate previously reported for supervised training of spike timing codes is 3.03% [64]. SPNNs achieve a test error rate of 1.06%.



(a) (b)

Figure 2.6: **Training and test accuracy on CIFAR-10.** The minimum test error rates for the relu network and the SPNN network are 30.73% and 34.09%, respectively.

We performed simulations of convolutional SPNNs implemented with Brian2 [28], an open-source simulator for spiking neural networks. The simulations of SPNNs in the circuit are modeled at time scales and firing rates relevant to spiking biological neurons, but the simulation time scale is arbitrary. For the parameter values, $g_l = \pi C_m/T$, $g_c = 60\pi C_m/T$, $V_l = 0\text{mV}$, $C_m = 10\text{pF}$, $L = \frac{1}{(2\pi/T)^2 C_m}$, $W_{\text{spike}} = .3\text{pA}$, $\tau_d = 0.8T$, and $\tau_s = 0.0T$. Simulations are for 300ms. The cycle period, T, is 10ms with a frequency of 100Hz. Numerical integration is performed with forward euler. The time step is 0.025ms. The period to time step ratio is approximately 400 to 1. The first example is presented for 15 cycles or 150ms (Fig. 2.8). The second example is then presented for 15 cycles. The output prediction is calculated as the unit which is furthest out of phase with respect to the other output units within a 3 cycle window, $\text{argmax}_i \ \mathbb{E}_{t_{spike,i}} \frac{1}{2}(|t_{\text{next spike},j\neq i} - t_{spike,i}| + |t_{\text{last spike},j\neq i} - t_{spike,i}|)$. Here, the expectation is taken with respect to the spike times of each component i, within a three-cycle window.

At this resolution, we notice a decrease in performance compared to the original SPNN. Currently, simulating all MNIST and CIFAR test samples is prohibitively expensive. We report qualitative results, leaving a quantitative analysis of the SPNN performance versus numerical precision for future work. The performance gap decreases as the time step decreases relative to the cycle period and as the confidence in the output prediction increases. It is promising that even at this resolution SPNN are robust to noise in the spike times.

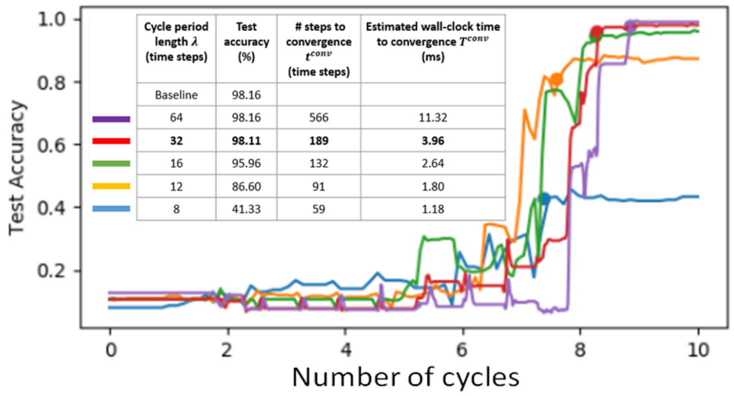
The phenomenon of pipelining emerges in SPNNs as an effect of a changing input stimulus (Fig. 2.8). Pipelining is a technique used for reducing the latency and increasing the resource utilization of computing hardware by connecting processing elements in series and passing the output of one unit to the input of the next unit in the pipeline. The signals propagate similarly to traveling waves in one direction.

2.3.1 Spiking phasor neural networks on neuromorphic hardware

Loihi [21] is a neuromorphic computer designed for use with low-power, low-latency applications. We implemented SPDNNs on Loihi and demonstrate high-performance test accuracy on the MNIST dataset. SPDNNs are implemented on Loihi using populations of excitatory and inhibitory neurons. The complex-valued synaptic weights in SPDNNs are mapped to positive-valued synaptic strengths and positive-valued spiking timing delays, $R_{ij} = |W_{ij}|$ and $\Delta T_{ij} = \arg(W_{ij}) + \pi$. Loihi implements neural dynamics using finite-precision values. Specifically, the weights, delays, and membrane potentials are represented by fixed-point and floating-point values. Up to 6-bit fixed-point values can be used to represent synaptic delays. To test the sensitivity of test accuracy to mapping the high-precision synaptic weights to lower-precision fixed-point values, we varied the precision of synaptic delays from 6 to 3 bits. Fig. 2.7 presents test accuracy versus the number of cycles. 5 bits of synaptic delay are required to maintain a high test accuracy.

2.4 Discussion

With the advent of powerful hardware solutions (TrueNorth [2], Loihi [21]) for spiking neurons, the question arises how deep learning can be implemented with spikes. The current approach is to implement real-valued activation values by spike rate [42, 56]. Some of these approaches yield near state-of-the-art results, but require a large number of spikes and exhibit rather long response latency (time to solution) due to the fact that the estimation of small rates requires large integration windows. A potential remedy to this problem is to compute with spike times [80]; however, current solutions are brittle and cannot be scaled up to address challenging machine learning problems.



	Cycle period length λ (time steps)	Test accuracy (%)	# steps to convergence t^{conv} (time steps)	Estimated wall-clock time to convergence T^{conv} (ms)
	Baseline	98.16		
🟪	64	98.16	566	11.32
🟥	**32**	**98.11**	**189**	**3.96**
🟩	16	95.96	132	2.64
🟨	12	86.60	91	1.80
🟦	8	41.33	59	1.18

Figure 2.7: **Spiking phasor neural networks on Loihi.**

Here, we present SPNNs employing spike timing codes and show them to be successful at solving challenging machine learning tasks. In our SPNN model, spike times represent phase angles of a complex number. Upon receiving input, the network computes robustly by relaxing to a stable limit cycle of periodic spike trains, which is a fixed-point in the complex domain. Our work is seemingly the first demonstration how a SNN employing a spike timing code can learn the CIFAR-10 dataset.

Another problem that our model can address is how to implement gradient learning in spiking neural networks using local signals. If deep learning is implemented by a rate code, pre- and post-synaptic spikes need to be integrated to estimate the activity values required in the gradient-based learning rule. In contrast, in the SPNN, the timing of spikes exactly encode the continuous-valued phase variables, which are required in the learning rule.

Our model is based on a phasor network with a complex-valued normalizing activation function, first proposed in [25], constraining the phasor variables to binary magnitude values of 0 or 1. It may be possible to extend our model to combine rate and timing codes by representing continuous-valued magnitudes of phasor variables in spike rates. In addition, there is other recent work on deep learning in complex networks [89], proposing activation functions different from the one used here. Future work should explore these other activation functions in the context of SPNNs.

Our work has also interesting implications for neuroscience. As demonstrated in the results, SPNNs exhibit time-locked, but not necessarily synchronous spiking. This phenomenon is similar to polychronous spike patterns [45]. Therefore, SPNNs may serve as a model for how to compute with such spike patterns, which also have been observed in neural activity [88].

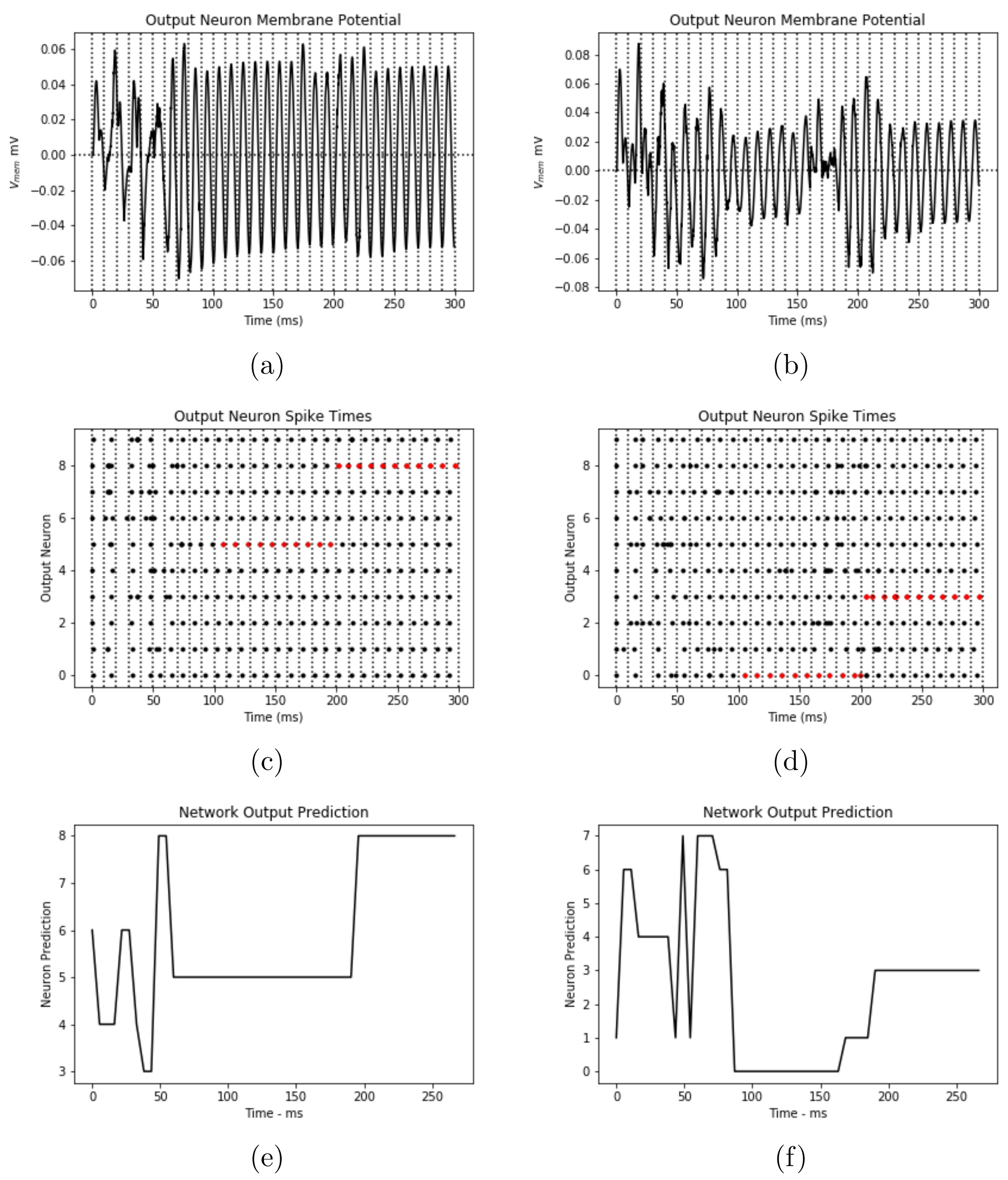


Figure 2.8: **Simulation of the circuit model SPNN.** MNIST (left) and CIFAR (right). Left: An example with label 5 is presented for 15 cycles or 150ms. Immediately following, an example with label 8 is presented for 15 cycles. 2.8a: The membrane potential for a unit in the output layer. The unit is unstable until cycle 7 when it beings to lock to the input signal. 2.8b: Spike times for the output layer. 2.8e: Predicted class label over time. Right: An example with label 0 is presented for 15 cycles or 150ms. Immediately following, an example with label 3 is presented for 15 cycles. 2.8b: The membrane potential for a unit in the output layer. The unit is unstable until cycle 9 when it beings to lock to the input signal. 2.8d: Spike times for the output layer. 2.8f: Predicted class label over time.

Chapter 3

Associative Memory in Recurrent Oscillatory Neural Networks

Associative memories are important models for robust pattern storage and retrieval that are relevant to many fields, such as neuroscience and artificial intelligence. Understanding their efficient implementation in biological systems or electrical hardware, such as coupled oscillator networks, is an area of active research. We present two novel coupled oscillator models which implement associative memories. The first model implements a Q-state phasor associative memory where each of the units in the associative memory can take one of Q possible values. The second model implements a thresholding phasor associative memory where each unit can have zero amplitude or take a continuous value on the complex-valued unit circle. We validate predictions made by mean-field theory and present new results for the pattern capacity and synaptic information of Q-state phasor associative memories. Additionally, we present the first coupled oscillator model capable of implementing associative memories with sparse pattern activity.

3.1 Introduction

Understanding computation in coupled oscillator networks is important to neuroscience, computer engineering, optimization, and artificial intelligence. In Neuroscience, oscillations are observed at multiple scales, from oscillations in the membranes of individual neurons to global oscillations between brain regions. For example, models of coupled oscillations explain auditory processing and the perception of music, rhythms, and language [54].

Recurrent neural networks (RNNs) are a class of neural networks with recurrent connections, i.e., the output of the network is fed back as input to the network. RNNs have been used to implement content-addressable memory (CAM). Specifically, RNNs can be used to implement an autoassociative memory. An autoassociative memory is a type of CAM where the input is the same as the output; therefore, this maps nicely to RNNs. A key feature of many autoassociative memories is that the retrieval of the output is robust to corruption or noise in the input. If the input pattern contains noise, the output will be a noise-free pattern. In essence, the network performs error correction. We begin by introducing the types of associative memories, then proceed to present how associative memories can be implemented efficiently by networks of coupled oscillators.

3.1.1 Associative memories

Associative memories have been studied extensively as neural networks for robust pattern storage and retrieval [86, 98, 75, 36]. Beyond real-valued and binary neural networks [36], coupled phasor networks [73, 71] have been proposed for building associative memories. Phasor neural networks are powerful units that allow for representing binary-valued, multi-valued, or continuous-valued states. Another benefit of phasor neural networks is the capability to be implemented efficiently by special purpose hardware consisting of networks of coupled oscillators [4, 40, 68, 53] or, as recently proposed, with networks of spiking, neuromorphic neurons [24].

The key metrics for the efficiency of associative networks used in this study are storage capacity ($\frac{\text{M patterns}}{\text{N dimensions}}$) and synaptic information ($\frac{\text{bits}}{\text{synapse}}$). The critical storage capacity measures how many patterns a network can retain for a given network size. Here, M is the number of stored patterns and N is the number of units in the network or the dimensional of the pattern vectors. The synaptic information measures the information content of the network with respect to the number of parameters (synapses). As explained below, there is a tradeoff between representation complexity and capacity.

3.1.2 Oscillator associative memories

Implementation of the binary-valued Hopfield associative memory in an analog system was first proposed in [37]. The method maps bipolar $\{-1, 1\}$ neuron activity to graded neuron responses $(-1, 1)$ that take a continuum of real values. Oscillators are interesting in both engineering and neuroscience since they can be used as efficient electrical hardware capable of representing a continuous value with phase and oscillations are found in many neural systems, respectively. Therefore, in [41, 39], it was proposed to unit ideas in neuroscience and engineering by implementing an analog associative memory in a network of coupled oscillators. Instead of converging to fixed, binary values, the network converges to a limit cycle. The binary values in the Hopfield network are represented by binary phases in the oscillators $(0, \pi)$. By synchronizing to phased locked states (in-phase, 180° out-of-phase), the network recalls stored patterns.

In [70], it was shown that in most cases oscillator associative memories are unstable. I.e., the stored patterns do not correspond to fixed points or limit cycles of the dynamical system. This is true for binary-valued, discrete-valued, and continuous-valued network states with dense activity. There are a few special cases. When $Q = 2$, i.e., the standard Hopfield network, the oscillator associative memory is unstable when the number of stored patterns is greater than 2. When $Q > 2$, the oscillator network is stable when the pattern loading is less than a critical value of 0.038, though the recall is not perfect and there is a small amount of error.

For Q-state associative memories, the oscillator network can be stabilized with the use of higher-order harmonics. Specifically, two types of higher-order harmonics. The first type involves higher-order harmonics between oscillators in the memory network. The second type is sub-harmonic injection locking [29, 95] where an independent oscillator that is not part of the memory network operates at a harmonic frequency of the memory network and injects into the memory network. In [70], it was shown that a 2nd-order harmonic between oscillators stabilized patterns for the case of $Q = 2$. Here, we generalize this method to Q-state oscillator networks to include networks with $Q > 2$. Additionally, we propose the use of sub-harmonic injection locking.

3.1.3 Hopfield associative memory

The classical Hopfield type associative memory [36] consists of a network of computational nodes which store a set of binary patterns and perform a robust recall of the stored patterns in the presence of noise through collective state computation. The Hopfield network stores a set of patterns that have binary or spin states, $\boldsymbol{\xi}^k \in \{-1, 1\}^N$. Here, $\boldsymbol{\xi}^k$ is the kth pattern of dimensionality N that is to be stored in the network. The Hopfield network with bipolar spin variables originates from a model of ferromagnetic material, called an Ising model [43] or spin-glass model.

The patterns are stored in the weights or synaptic connections of the network using a Hebbian [32] type associative learning rule,

$$\boldsymbol{W} = \frac{1}{M}\sum_{k=1}^{M} \boldsymbol{\xi}^k \boldsymbol{\xi}^{kT}. \tag{3.1}$$

The learning rule is also referred to as the outer product learning rule since the weights are a weighted sum of the outer product of the stored patterns. Therefore, the network weights are a superposition of the stored patterns and the stored patterns are the eigenvectors of the symmetric weight matrix.

The network dynamics act to minimize the energy function,

$$E(\boldsymbol{x}) = -\boldsymbol{x}^T \boldsymbol{W} \boldsymbol{x}. \tag{3.2}$$

Here, $\boldsymbol{x} \in \{-1, 1\}$ is the vector of network variables. The system dynamics proceed according to a discrete-time iterative update, $\boldsymbol{x}(t+1) = f(\boldsymbol{W}\boldsymbol{x}(t))$. Here, $\boldsymbol{x}(t+1)$ is the state of the network after the update at time $t+1$, $\boldsymbol{x}(t)$ is the state of the network at time t, and $f(\boldsymbol{u}) = \text{sign}(\boldsymbol{u})$ is activation function applied element-wise and return the sign of each element in . Here, $\boldsymbol{u} = \boldsymbol{W}\boldsymbol{x} \in \mathbb{R}$ is the real-valued postsynpatic sum or preactivation.

A Hopfield associative memory is also a type of attractor neural network. In attractor neural networks, the state of the networks moves towards stable attractor states. In Hopfield networks, the stored patterns act as stable fixed-point attractors. Attractors can be

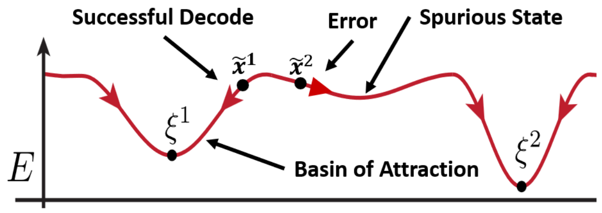


Figure 3.1: **Conceptual energy landscape for a Hopfield associative memory.** ξ^1 and ξ^2 represent patterns stored in the network. $\tilde{\mathbf{x}}^1$ and $\tilde{\mathbf{x}}^2$ are noisy inputs. The network dynamics act to minimize the energy. Depending on the initial state of the input, the network variables will converge to a basin of attraction corresponding to a stored pattern or to a spurious state, corresponding to an recall error.

characterized by their basins of attractions. Fig. 3.1 presents a conceptual energy landscape for a Hopfield network. Key features of the energy landscape are deep, wide basins of attraction located at stored patterns and shallow, local minimum at spurious states. A basins of attraction for stored patterns should be wide and well separated in order to be robust to noise.

3.1.4 Crosstalk interference

The capacity of the Hopfield network is limited by crosstalk interference. Crosstalk interference arises from the fact that the stored patterns are not orthogonal. This can be understood by separating the preactivation into a signal and noise component. For this analysis, we redefine the learning rule so that weight matrix is scaled by the dimensionality instead of the number of patterns, $\mathbf{W} = \frac{1}{N}\sum_{k=1}^{M}\xi^k\xi^{kT}$. In practice, the weight matrix can be scaled by any value.

$$\mathbf{W}\mathbf{x} = \frac{1}{N}\sum_{k=1}^{M}\xi^k\xi^{kT}\mathbf{x} \tag{3.3}$$

$$= \alpha\xi^\mu + \sum_{k\neq\mu}\beta\xi^k \tag{3.4}$$

Here, ξ^μ corresponds to the pattern to be recalled, $\alpha \approx 1$ is the signal coefficient, and $\beta \approx 1/\sqrt{N}$ is the magnitude of crosstalk interference component for the other patterns stored in the network. Eq. (3.3) shows that the current state of the network, $\mathbf{x}$, is compared to every stored pattern by the inner product. The inner product is a measure of the similarity between two vectors. In this case, it measures the similarity between the current state of the network and each of the stored patterns. When the current state is similar to one of the stored patterns, the inner product will be close to N. This is the signal component of the update. Any two random patterns are pseudo-orthogonal, i.e., close to orthogonal. Therefore, there similarity is relatively small. For random patterns, the expected value of the inner product is zero with variance N. After scaling and summing, the variance of the crosstalk interference becomes M/N. By the central limit theorem, the noise will be normal distributed with zero mean and variance equal to M/N. Fig. 3.2 provides an illustration of the crosstalk interference and when an error is made.

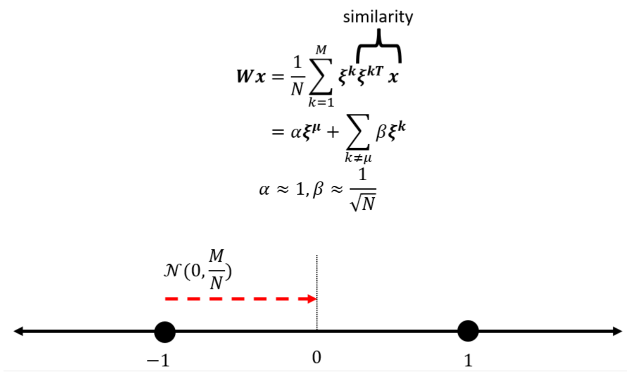

similarity
$\boldsymbol{Wx} = \frac{1}{N}\sum_{k=1}^{M} \boldsymbol{\xi^k} \overbrace{\boldsymbol{\xi^{kT}} \boldsymbol{x}}$
$= \alpha\boldsymbol{\xi^\mu} + \sum_{k\neq\mu} \beta\boldsymbol{\xi^k}$
$\alpha \approx 1, \beta \approx \frac{1}{\sqrt{N}}$
$\mathcal{N}(0, \frac{M}{N})$
-1
0
1

Figure 3.2: **Illustration of crosstalk interference in a Hopfield network.** Top) The upper part of the figure shows that the preactivation can be separated into a signal and a noise component. Bottom) The lower part of the figure shows that the an error occurs when the noise has magnitude greater than 1 and has sign opposite to that of the correct state.

3.1.5 Error correction

The Hopfield associative memory is able to perform recall of stored patterns in the presence of noise in the input and crosstalk interference. Therefore, the network is able to perform error correction. There are two types of error correction in the network dynamics. The first is collective state computation, i.e., all of the network acts collectively to remove noise and recall stored patterns. Even if several units have errors, there is redundancy programmed into the network due to the outer product learning rule. The patterns are distributed between the synaptic connections of the network. The state of each unit in a pattern is redundantly stored in $N - 1$ connections and then placed in superposition with the other patterns. Therefore, the outer product learning rule creates redundancy and distributes information across the entire network. This allows the network to act collectively to remove errors.

The second type of error correction comes from the activation function that quantizes the state of each unit at each iteration. The preactivation, $\mathbf{Wx}$, will have analog values. The preactivation can be thought of as a prediction or probability of a unit being -1 or 1. By then quantizing the analog values, the network corrects noise or error when the prediction is correct. If allowed to propagate through the network, the error would grow to cause catastrophic failure.

3.2 Results

3.2.1 Q-state phasor associative memories

As discussed in Section 3.1.3, spin-glass models have been used to construct associative memories. E.g., the Ising model [43] was the basis for the Hopfield associative memory [36]. In addition, the Potts model [78] is the basis for associative memories where the individual units can take one of Q possible states [48, 71]. Specifically, the units in a Q-state phasor associative memory [71] take one of Q possible values distributed equally on the unit circle. This representation maps naturally to complex-valued phasors. Here, we explore Q-state phasor associative memories with varying levels of activation sparsity and present novel empirical results on their capacity and information content.

A Q-state phasor associative memory [71] is a complex-valued phasor neural network that stores a set of M N-dimensional phasor patterns with discrete phases. The state of unit i for pattern k is $\xi_i^k = a_i^k e^{i(2\pi q_i^k/Q+\psi_i)} \in \mathcal{C}^N$, $\forall k \in \{1, .., M\}$ where the phases can have one of Q discrete states $q_i^k \in \{0, .., Q-1\}$. Here, ψ_i is a fixed phase shift for unit i and $a_i^k \in \{0, 1\}$ is a binary activity variable. The patterns are stored in the synaptic weights using the complex outer product,

$$\mathbf{W} = \frac{1}{N} \sum_{k=1}^{m} \boldsymbol{\xi}^k \boldsymbol{\xi}^{k*T}, \tag{3.5}$$

generalizing Hebbian learning in real-valued associative memories. Here, $\boldsymbol{\xi}^{k*T}$ is the complex conjugate transpose of pattern k. It is assumed the diagonal of the weight matrix is zero, $\text{diag}(\boldsymbol{W}) = \mathbf{0}$ The energy,

$$E = -\frac{1}{2}\mathbf{z}^{*T}\mathbf{W}\mathbf{z}, \tag{3.6}$$

governs the system dynamics that move the network state to stable, fixed-points in complex space, corresponding to limit-cycle attractors in the time domain. Here, $\mathbf{z} \in \mathbb{C}^N$ is the complex-valued vector representing the network state. The energy function is similar to that of the Hopfield network except for the use of the complex conjugate transpose. Again, the learning rule is similar but the complex conjugate transpose is used. Association is performed through discrete-time iterative dynamics using the update rule $z_i(t+1) = f(\sum_j W_{ij} z_j(t))$. The activation function,

$$f(u_i) = \exp(i\frac{2\pi}{Q}\ \text{argmin}_q|\phi_i^u - \frac{2\pi q}{Q} - \psi_i|)H(|u_i| - \Theta), \tag{3.7}$$

quantizes the phase, ϕ_i, of the ith unit to the nearest value of $\frac{2\pi q}{Q} + \psi_i$ and either sets the amplitude to one if $|u_i| > \Theta$, or zero, otherwise. The Heaviside function, H, acts to threshold activity below the threshold value, Θ. Here, $u_i = \sum_j W_{ij} z_j$ is the preactivation for unit i.

Q-state phasor neural networks generalize Hopfield networks to units which have Q states. As a consequence, the Hopfield network is a special case of the Q-state phasor neural network when $Q = 2$. For larger values of Q, the states are distributed equally on the unit circle. Fig. 3.3 presents an example of a Q-state phasor for $Q = 4$. There are four states the phasor can take. Here, the states are aligned with the real and imaginary axes, though, the four states could be rotated such that the phase difference between states is $\pi/4$ and $-\pi/4$.

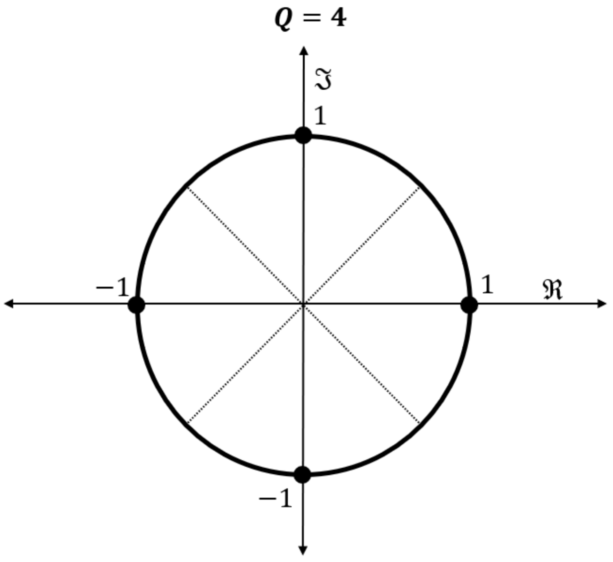


Figure 3.3: **Example Q-state phasor.** A Q-state phasor is presented for $Q = 4$. The unit can take one of four possible states equally distributed on the unit circle. Here, the four states are on the real and imaginary axes.

In [18], it was predicted that for Q-state phasor associative memories specific values of Q exhibit greater pattern capacity and information content than binary associative memories or phasor neural networks with continuous phase values for dense patterns, i.e. when every unit is active in a pattern.

We compare the pattern capacity and synaptic information for several values of Q. The pattern capacity is defined as the number of stored patterns divided by the dimensionality of the network, $C = M/N$. In order to compute the capacity, we compare the similarity between stored patterns and patterns after recall. The similarity is the coherence of the recalled pattern to the stored pattern,

$$\rho_{\boldsymbol{z},\boldsymbol{\xi}^k} = \frac{|\boldsymbol{z}^{*T}\boldsymbol{\xi}^k|^2}{|\boldsymbol{z}^{*T}\boldsymbol{z}||\boldsymbol{\xi}^{k*T}\boldsymbol{\xi}^k|} \tag{3.8}$$

Here, $\boldsymbol{z}$ is the state of the network after recall, $\boldsymbol{\xi}^k$ is a stored pattern. The synaptic information, I_s, is defined as the number of bits stored in the network divided by the number of synaptic connections,

$$I_s = \mathbb{E}_k[\rho_{\boldsymbol{z},\boldsymbol{\xi}^k}]MI_p/N_s. \tag{3.9}$$

Here, I_p is the information in each pattern, N_s is the number of nonzero synaptic connections, and $\mathbb{E}_k[\rho_{\boldsymbol{z},\boldsymbol{\xi}^k}]$ is the expected similarity taken over stored patterns. For Q-state phasor neural networks, the information in each pattern, $I_p = \log_2(Q)N$, can be computed based on the number of bits per state for a specific value of Q, $\log_2(Q)$, and the number of units in the network, N.

The capacity and synaptic information for dense patterns are presented in figure 3.4. The results from simulations are consistent with previous findings [18]. Additionally, we present new empirical results on the capacity and synaptic information for values of Q greater than 4. As predicted by mean-field theory, the greatest pattern capacity is found for $Q = 3$. $Q = 2$ and $Q = 4$ have the same pattern capacity. As Q increases the pattern capacity decreases. In the limit as Q goes to infinity the pattern capacity approaches that of the continuous phasor associative memory. Again, when $Q = 2$, the Q-state phasor associative memory is equivalent to the binary Hopfield network [36]. The same pattern capacity and information content are recovered, approximately 0.14.

The information is also greatest for $Q = 3$. Unlike the pattern capacity, the information remains greater than the binary Hopfield network for $Q < 7$. Though, $Q = 2$ and $Q = 4$ have the same pattern capacity, $Q = 4$ has greater information since each unit has 2 bits of entropy.

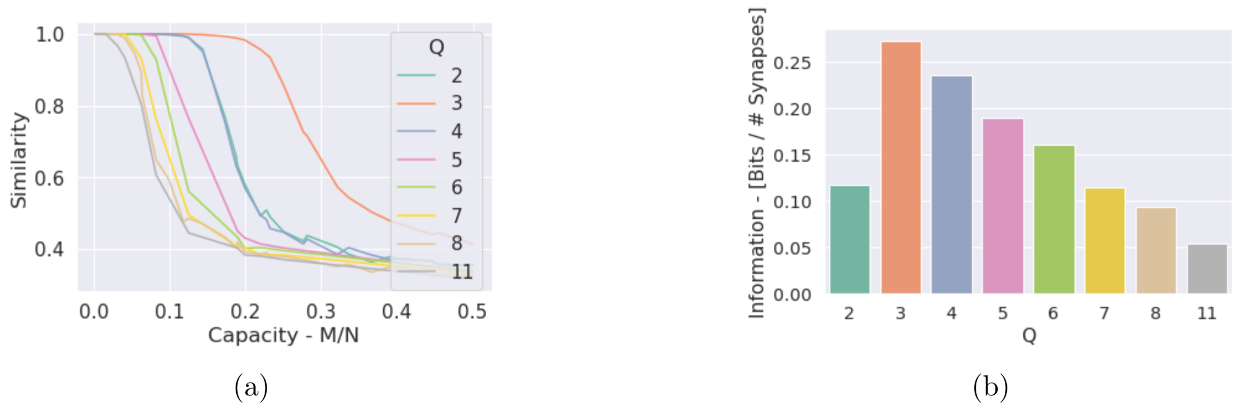


Figure 3.4: **Pattern capacity and synaptic information for Q-state phasor associative memory networks** a) The mean similarity is plotted as a function of the pattern capacity. b) Information is plotted for varying values of Q.

The capacity and information results can be intuitively understood by examining the trade-off between complexity and robustness in Q-state phasor representations and the complex crosstalk interference. Q-state phasor representations trade-off complexity for robustness. Fig. 3.5 provides an illustration of this effect. Two Q-state phasors are presented for $Q = 4$ and $Q = 8$. The phasors have 2 bits and 3 bits of entropy for $Q = 4$ and $Q = 8$, respectively. The boundaries between states are shown with dashed lines. Noise vectors are displayed as red arrows with dashed red circles. For $Q = 4$, it can be seen that the noise can have a greater magnitude before crossing the boundary between states. For $Q = 8$, the same noise would cause an error. The distance between nearest points, or minimum distance, in Q-state phasors is given by the chord of a unit radius circle with angle equal to $2\pi/Q$, $d_{\min} = 2\sin(\pi/Q)$. Therefore, the closest point on the boundary between states is $\sin(\pi/Q)$. The noise vector must be greater than this to cause an error.

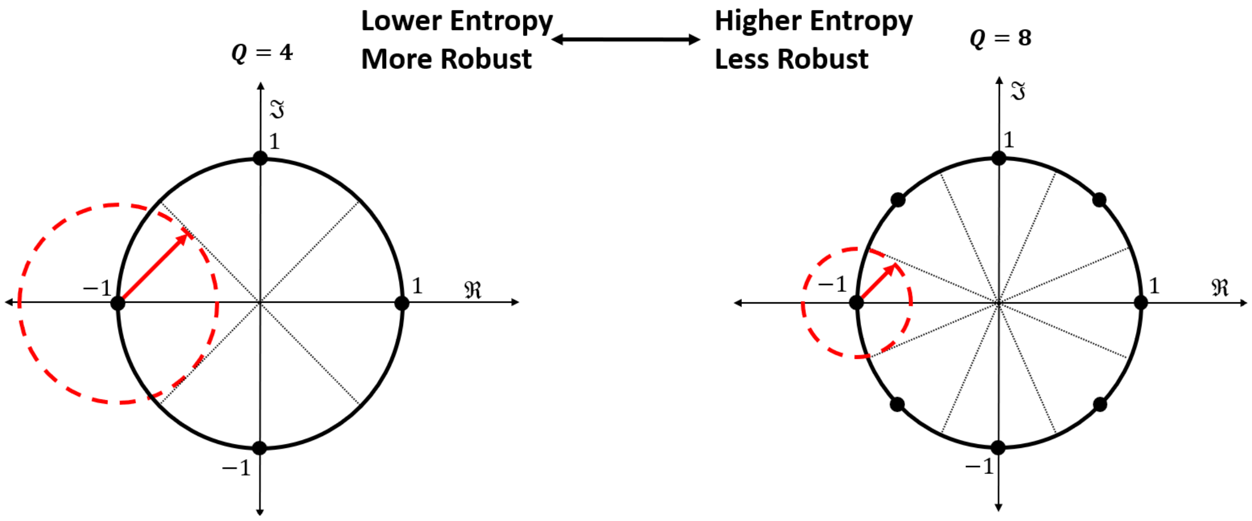


Figure 3.5: **Q-state phasor trade-off pattern complexity for robustness.**

3.2.2 Basins of attraction

A key feature of Q-state phasor associative memories is the ability to perform recall in the presence of noise. In essence, the network performs error correction. Here we characterize the basins of attraction for Q-state phasor neural networks. Fig. 3.6 presents results from simulations of Q-state phasor neural networks for different values of Q and pattern loading, $C = M/N$. Each subfigure plots the initial similarity versus the final similarity for a different value of C. The lines in each plot represent a network with a different value of Q. The black line with a slope equal to one are the points where the initial similarity is equal to the final similarity. At low pattern loadings, all networks can correct a large number of errors. As the pattern loading increases beyond the critical pattern capacity for particular values of Q, the networks fail to recall stored patterns. In agreement with the results in Fig. 3.4a, $Q = 3$ has the greatest capacity. $Q = 2$ and $Q = 4$ have the second greatest capacity.

3.2.3 Complex crosstalk interference

The increase in capacity by changing from $Q = 2$ to $Q = 3$ may seem counterintuitive. The distance between states is less for $Q = 3$ than for $Q = 2$. Therefore, it may be expected that 3-state phasors are more sensitive, or less robust, to noise. Intuition may be gained by

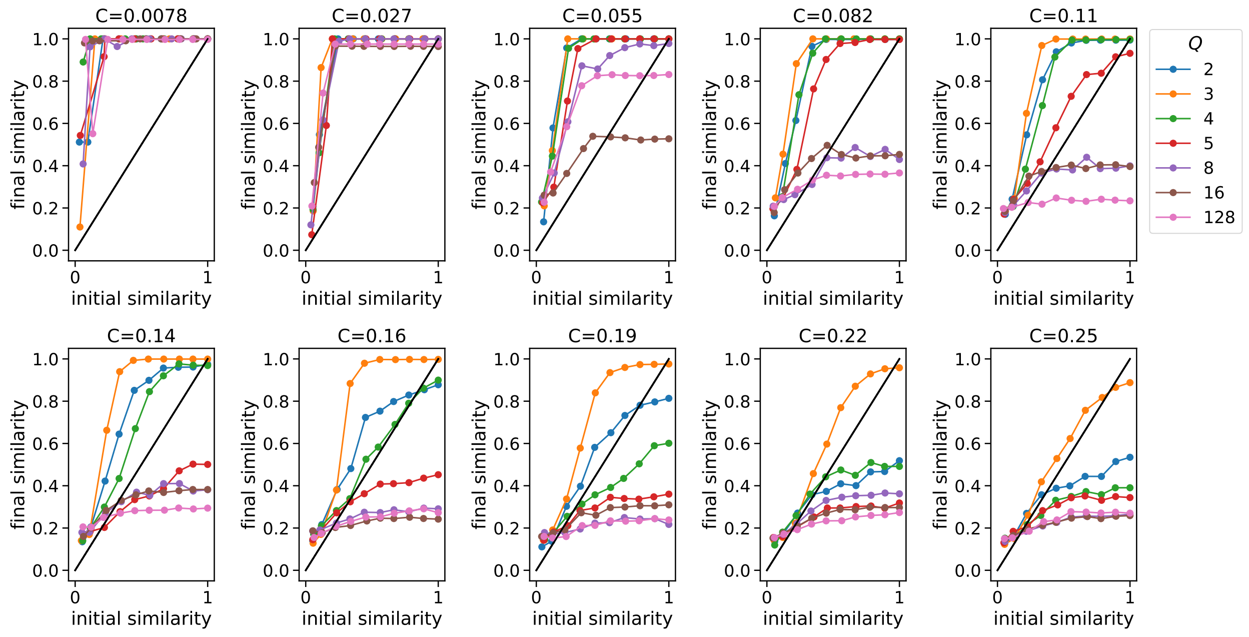


Figure 3.6: **Basins of attraction for Q-state phasor neural networks.**

examining the crosstalk interference in Q-state phasor neural networks. For Q-state phasor neural networks, the crosstalk interference is distributed in the complex plane.

$$\boldsymbol{W}\boldsymbol{z} = \frac{1}{N}\sum_{k=1}^{M} \boldsymbol{\xi}^k \boldsymbol{\xi}^{k*T} \boldsymbol{z} \tag{3.10}$$

$$= \frac{1}{N}\boldsymbol{\xi}^\mu \sum_{j=1}^{N} \bar{\xi}_j^\mu z_j + \frac{1}{N}\sum_{k\neq\mu} \boldsymbol{\xi}^k \sum_{j=1}^{N} \bar{\xi}_j^k z_j \tag{3.11}$$

$$= \alpha\boldsymbol{\xi}^\mu + \sum_{k\neq\mu} \beta\boldsymbol{\xi}^k. \tag{3.12}$$

Here, α and β are complex-valued coefficients corresponding to the signal and noise components, respectively. For the signal component, the complex dot product will be more coherent when compared to the noise component, having a greater magnitude. The noise component is complex normal distributed with an expected value equal to zero and an isotropic covariance. For $Q = 2$, the noise resides on the real line. The magnitude of the noise is half-normal distributed. For $Q > 2$, the magnitude of the noise is Rayleigh distributed. Compared to the half-normal distribution, the Rayleigh distribution is more concentrated around the mode and has a lower kurtosis. Therefore, the Rayleigh distribution has a lower probability of extreme magnitude values.

Fig. 3.7 presents a histogram of noise magnitudes and a scatter plot of preactivation values for individual units obtained from simulations of Q-state phasor neural networks with pattern loading of $M/N = .14$ for Q equal to 2, 3, and 4. The noise, $n = u_i - 1$, is computed based on the difference between the preactivation and the target value of $z_i = 1$. The histogram shows that for $Q > 2$ the noise distribution has a larger mode but has a lower probability of extremely large values. $Q = 2$, the scatter plot shows that the noise resides on the real line. It can be seen that some of the preactivations are closer to -1 than to 1, indicating an error. This is expected since the pattern loading (0.14) is larger than the capacity for $Q = 2$ (0.0.138). For $Q > 2$, the noise is distributed in the complex plane around $z_i = 1$. As shown in the plot on the left, there are fewer extreme values. As expected, for $Q = 3$, almost all of the values lie within the radius of the circle since the pattern loading (0.14) is less than the capacity (0.22). For $Q = 4$, values extend past the radius of the circle indicating an error, Again, this is expected since the pattern loading (0.14) is greater than the capacity (0.138).

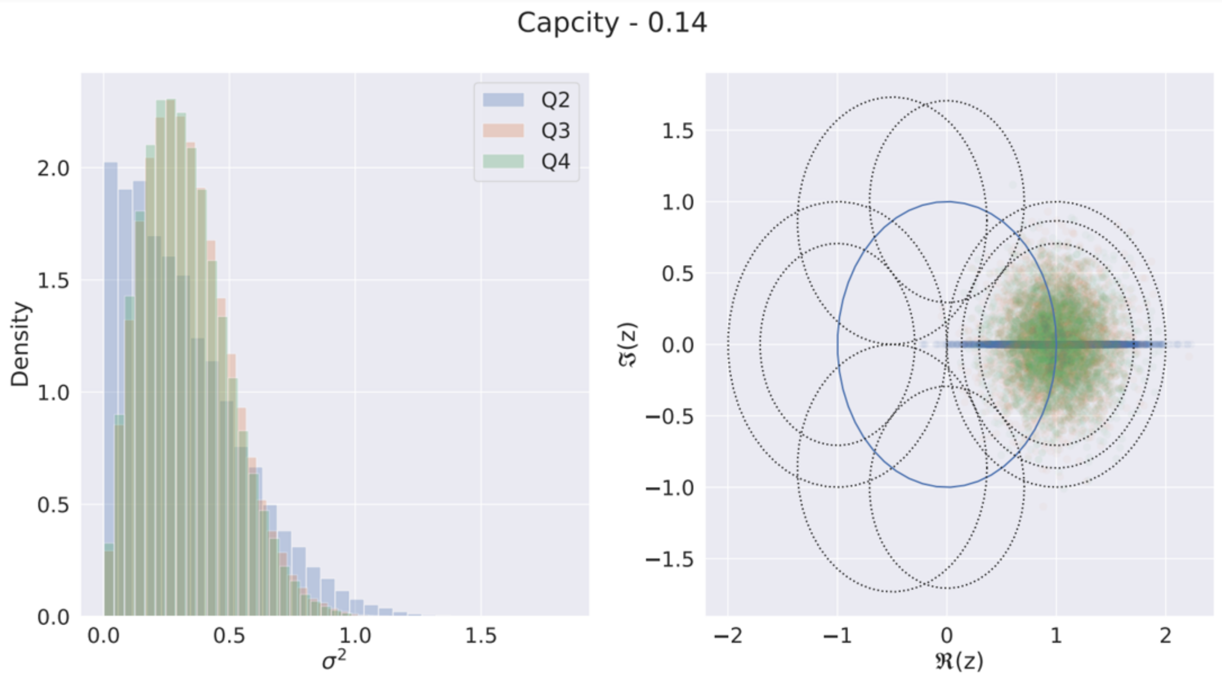


Figure 3.7: **Crosstalk interference in Q-state phasor associative memories.** Left) Histogram of noise magnitudes for several values of Q. Right) Scatter plot of noise values for several values of Q. The solid circle is a unit circle center at zero. Dotted circles are plotted for $Q = 2$, $Q = 3$, and $Q = 4$. The dotted circles have radius equal to the half the distance between nearest states. the In both plots the networks have capacity equal to 0.14.

3.2.4 Continuous phasor neural networks

So far, we've introduced associative memories where each unit takes discrete states. For Q-state phasor neural networks, as Q approaches infinity, the state of each unit approaches a continuous value. A model for continuous phasor neural networks was originally proposed in [73]. The associative memory is capable of storing continuous values compared to discrete values in Q-state phasor associative memories. The ability to store continuous values is an advantage. Though, this comes at a cost. The pattern capacity is less for continuous phasor networks, $M/N = 0.038$.

Recently, it was found that the capacity of continuous phasor associative memories can be increased if the networks store patterns with sparse activity [25]. Thresholding phasor associative memories store sparse, continuous phasor patterns, $\xi_i^k = a_i e^{i\phi_i} \in \mathbb{C}$. Here, ξ_i^k is the ith element of the pattern vector, $a_i \in \{0, 1\}$, and $\phi_i \in [-\pi, \pi]$. The network stores patterns using the same complex outer product learning rule as the for continuous phasor neural networks [73] and Q-state phasor neural networks [71], $\mathbf{W} = \frac{1}{M}\sum_{k=1}^{M} \boldsymbol{\xi}^k \boldsymbol{\xi}^{k*T}$. At each time step, t, each unit receives input and forms the preactivation $\boldsymbol{u}(t) = \boldsymbol{W}\boldsymbol{z}(t)$. The activation function creates sparse patterns by thresholding units when the preactivation magnitude is less than a specific value $\Theta(t)$,

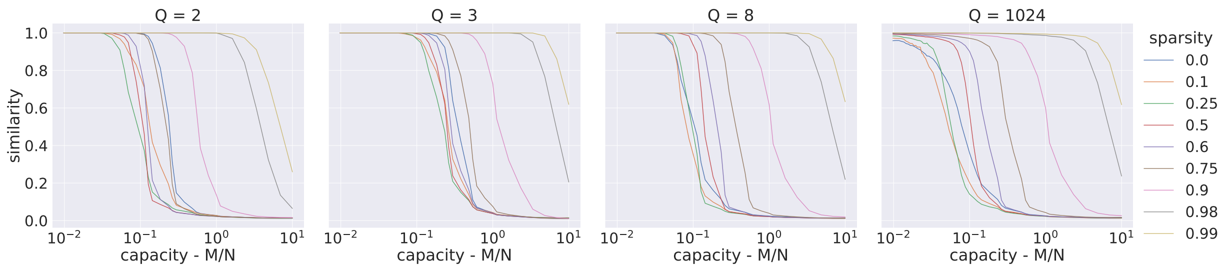
$$z_i(t+1) = f(u_i(t), \Theta(t)) := \frac{u_i(t)}{|u_i(t)|} H(|u_i(t)| - \Theta(t)). \tag{3.13}$$

Here, $H(x)$ is the Heaviside function. If $|u_i(t)|$ is less than $\Theta(t)$ the unit amplitude is set to zero.

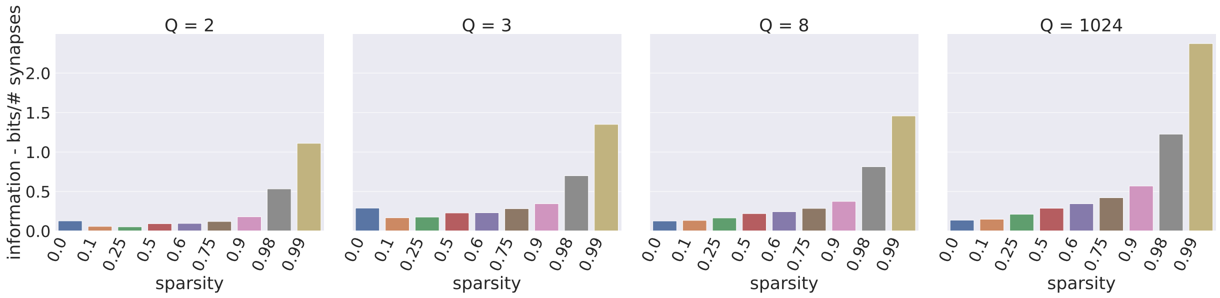
3.2.5 Sparse Q-state phasor neural networks

In this section, we present new results for sparse Q-state phasor neural networks. We show the tradeoffs between sparsity and the number of states for each unit, Q. Fig. 3.8a displays the similarity versus capacity for varying sparsity and Q. Each subfigure is computed with a different value of Q and each line is computed for a different sparsity. Sparsity is the fraction of inactive units in a pattern. E.g., sparsity equal to 0.99 corresponds to patterns where 0.01% of units are active. A sparsity of zero represents dense patterns, i.e., patterns where every unit is active. For all values of Q, the highest capacity is achieved for the sparsest patterns. For intermediate sparsity values, an interesting trend appears. For $Q < 8$, as the sparsity increases from zero the capacity decreases until a point at which the capacity starts to increase. For $Q > 8$, increase sparsity only increases the capacity.

Fig. 3.8b plot information versus sparsity for different values of Q. The results are like those for similarity versus capacity. The greatest information is achieved for the sparsest patterns. For $Q < 8$, there is a bimodal distribution. Information first decreases with increasing sparsity until a critical values at which information increases. For $Q >= 8$, increasing sparsity only increases the information.



(a)



(b)

Figure 3.8: **Effects of sparsity on capacity and information for Q-state phasor associative memories.** Top) Similarity versus capacity with varying levels of Q and sparsity. Bottom) Information versus sparsity for different values of Q.

3.2.6 Oscillatory associative memoires

The dynamics of Q-state phasor neural network are implemented in discrete time. Therefore, analog implementations of Q-state associative memories require mapping phasor neural networks to models of physical systems, such as oscillatory neural networks. Here, we consider a continuous-time dynamical system consisting of a network of coupled oscillator that is capable of implementing a Q-state phasor associative memory.

Original efforts [40] to map phasor associative memories to oscillator neural networks resulted in low capacity and poor pattern retrieval. This results from the fixed points of the dynamical system not corresponding to stored patterns. The issue can be resolved in several ways.

The first method discussed uses sub-harmonic injection locking, where the frequency of the harmonic injection signal is Q times the frequency of memory oscillators, $Q = \omega_{\text{inj}}/\omega_{\text{mem}}$. This biases the phases of the oscillators in the associative memory towards discrete values.

The second method uses sparse pattern activity. Implementation of sparse phasor neural networks in oscillator neural networks requires a thresholding mechanism to inactivate oscillators corresponding to phasors with zero magnitudes. We present a novel thresholding oscillator model which takes advantage of the Hopf bifurcation in order to create two modes of behavior. One mode has zero magnitudes. The other mode produces limit cycle oscillations with fixed, nonzero magnitudes.

3.2.7 Q-state oscillatory neural networks

We present two models that implement arbitrary Q-state phasor associative memories in oscillator neural networks. A similar method was proposed in [69] but only implemented the special case of binary patterns, or $Q = 2$. Our method can be seen as a generalization that allows for pairwise higher-order harmonic coupling and sub-harmonic injection locking.

In the following section, we discuss the case of dense patterns, i.e., patterns where all units are active. Then, we present models that implement sparse Q-state associative memories. For the first dense oscillator neural network model, the system of ordinary differential equations (ODE) is

$$\frac{d\phi_i}{dt} = -\epsilon \sum_{j \neq i} R_{ij} \sin(\phi_i - \phi_j - \Phi_{ij}) - h_i \sum_{j} \sin(Q(\phi_i + \psi_i)). \tag{3.14}$$

Here, ϕ_i is the phase of the ith oscillator, ψ_i is the fixed phase shift for the ith oscillator, $R_{ij} = |W_{ij}|$, $\Phi_{ij} = \arg(W_{ij})$, ϵ is a global coupling parameter, and h_i is the strength of harmonic infection for the ith oscillator. R_{ij} represents the strength of coupling between oscillators z_i and z_j. The coupling imposes a phase shift of Φ_{ij}. The constants ϵ and h control the relative strength of network coupling and sub-harmonic injection locking, respectively. The system energy is composed of two parts determined by network interactions and sub-harmonic injection locking,

$$E_{\text{system}}(\phi) = E_{\text{network}}(\phi) + E_{\text{SHIL}}(\phi). \tag{3.15}$$

The network energy,

$$E_{\text{network}}(\phi) = -\frac{1}{2} \sum_{ij} R_{ij} \cos(\phi_i - \phi_j - \Phi_{ij}), \tag{3.16}$$

measures the similarity of network activity compared to stored patterns. The energy function sub-harmonic injection locking dynamics,

$$E_{\text{SHIL}}(\phi) = -\frac{1}{Q\omega} \sum_{i} h_i \cos(Q(\phi_i + \psi_i)), \tag{3.17}$$

measures the deviation of the oscillator phase from one of the Qth roots of unity.

To demonstrate its effects on the coupled oscillator network we run simulations with and without sub-harmonic injection locking. Fig. 3.9a plots phase versus time for oscillators in a 3-state oscillator neural network without sub-harmonic injection locking. The states of the oscillators are initialized to a stored pattern. As the network evolves, the oscillators drift from their initial condition. Therefore, the stored patterns are not fixed points of the

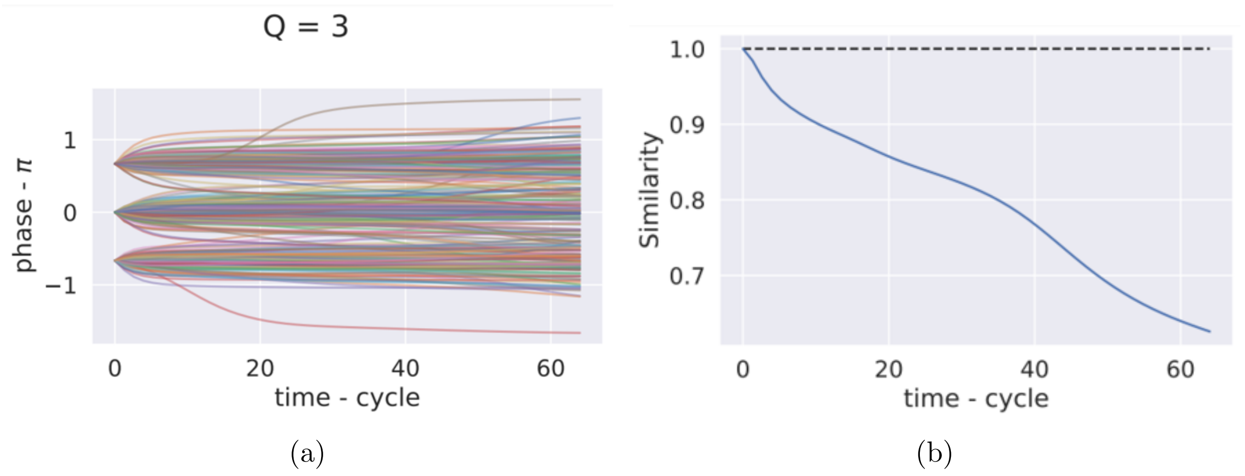


Figure 3.9: **Q-state oscillator neural network without sub-harmonic injection locking.** a) Phase versus time for a 3-state oscillator associative memory. b) similarity versus time.

dynamical system. Fig. 3.9b shows the similarity of the network state to the initialized pattern versus time. The similarity starts at one since the network was initialized to the stored pattern. As time increases, the similarity decreases.

Sub-harmonic injection locking uses signals from an oscillator or a set of oscillators that operate at a harmonic of the memory oscillators. Alternatively, the network can be stabilized by higher-order coupling between oscillators. For the second model, the system of ODEs is

$$\frac{d\phi_i}{dt} = -\epsilon \sum_{j \neq i} R_{ij} \sin(\phi_i - \phi_j - \Phi_{ij}) - h_i \frac{1}{N} \sum_{j} \sin(Q(\phi_i + \psi_i - \phi_j - \psi_j)). \tag{3.18}$$

Here, ψ_i is the fixed phase shift for the ith oscillator. Eq. (3.18) model generalizes the model proposed in [70] by allowing a fixed phase shift for each oscillator.

3.2.8 Stability of oscillator neural networks

As previously mentioned, in most cases without higher-order harmonics, the stored patterns are not stable states of the oscillator network. We show how higher-order harmonics can act to make stored patterns fixed points of the dynamical system. Fig. 3.10 is an illustration of sub-harmonic injection locking. An oscillator, θ, with angular frequency ω received input from another oscillator, γ, with an angular frequency that is a harmonic of ω. In this case, the harmonic is 3ω. The phase of the γ oscillator is biased towards three angles (0, $2\pi/3$, and $4\pi/3$).

A method for analyzing the stability of dynamical systems is by examining the maximum eigenvalue of the Jacobian matrix. Fig 3.11 displays histograms of the maximum eigenvalue, $\lambda_{\max}(J)$ of the Jacobian matrix for Eq. (3.14). Let $f_i(\boldsymbol{z})$ be equal to Eq. (3.14). Then, an entry in the Jacobian matrix is

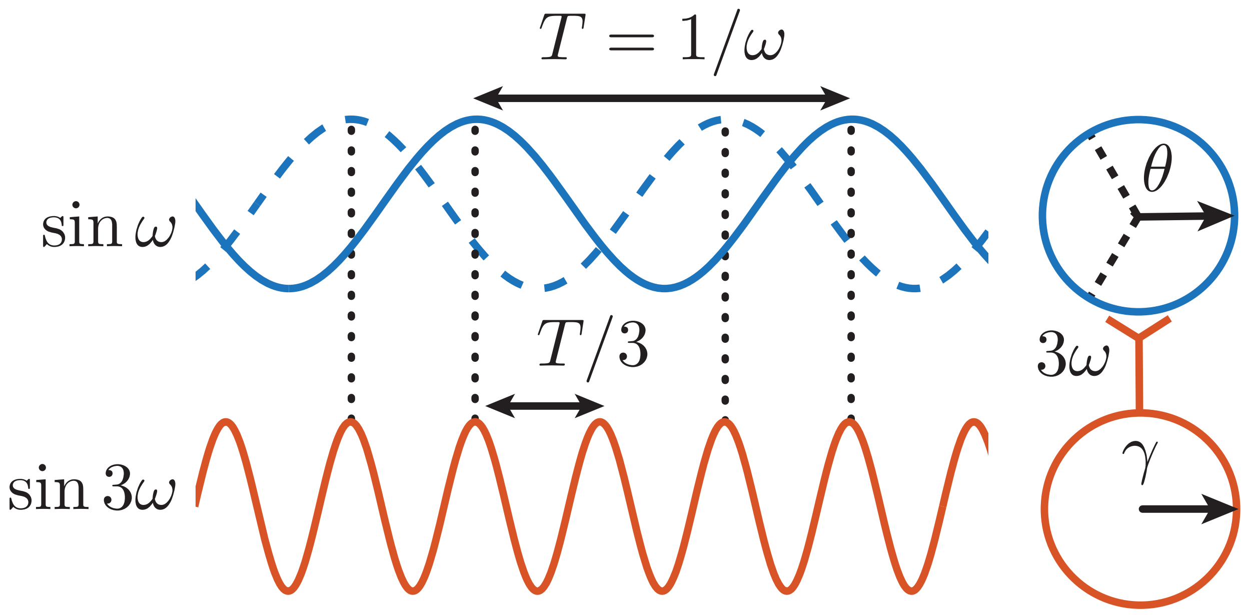
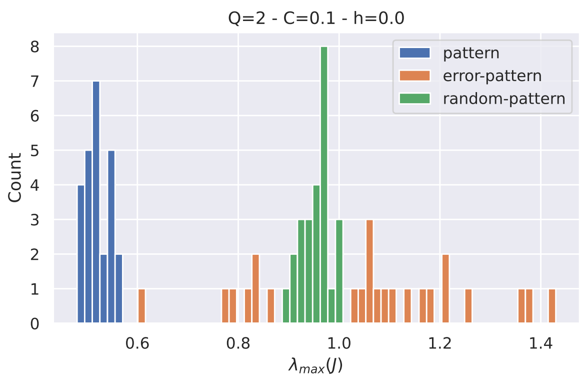


Figure 3.10: **Example of sub-harmonic injection locking.**

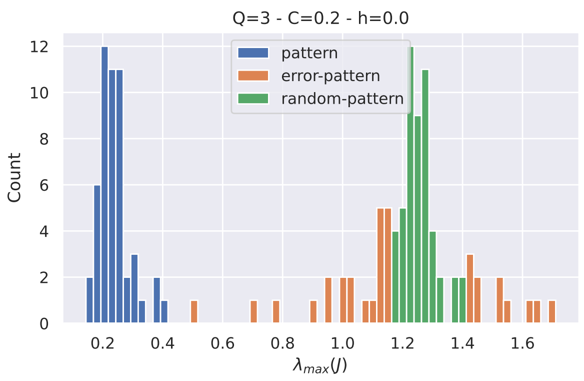
$$J_{ij} = \frac{\partial f_i}{\partial z_j}. \tag{3.19}$$

When $\lambda_{\max}(J) > 0$, the system is unstable and will drift from the current state. In all subfigures in Fig. 3.11, values for $\lambda_{\max}(J)$ are presented for three types of patterns, stored patterns, patterns with errors, and random patterns. Figs. 3.11a and 3.11b plot histogram values for $\lambda_{\max}(J)$ in networks without sub-harmonic injection locking. For all three pattern types, $\lambda_{\max}(J) > 0$. Therefore, the network is unstable. Introducing sub-harmonic injection locking can act to stabilize the system for stored patterns. Figs. 3.11c and 3.11d plot $\lambda_{\max}(J)$ for networks with sub-harmonic injection locking. Now, for stored patterns, $\lambda_{\max}(J) <= 0$. This results in stable network activity. I.e., the stored patterns are fixed points of the dynamical system. For error patterns and random patterns, $\lambda_{\max}(J) > 0$. Again, the network is unstable. For error patterns, the system will move towards a stored pattern.

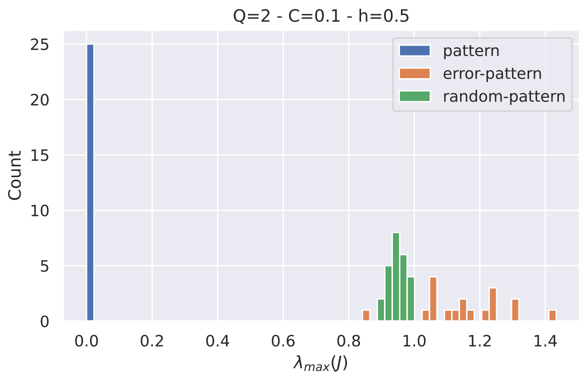
We demonstrate how a network can retrieve a stored pattern from a corrupted pattern. Fig. 3.12 presents the results from the decoding (recall) of a corrupted pattern in a 3-state oscillator neural network. The oscillator phases versus time are plotted in Fig. 3.12a. The network state is initialized to the corrupted pattern. The injection locking coefficient, h, is increased linearly over the course of decoding. As h increases, the oscillator phases are biased towards one of 3 possible states. It can be seen that some of the oscillator phases move between phase states. At the end of decoding, all of the oscillator phases have locked into one of the 3 possible states. Similarity versus time is plotted in Fig. 3.12b. Since the network



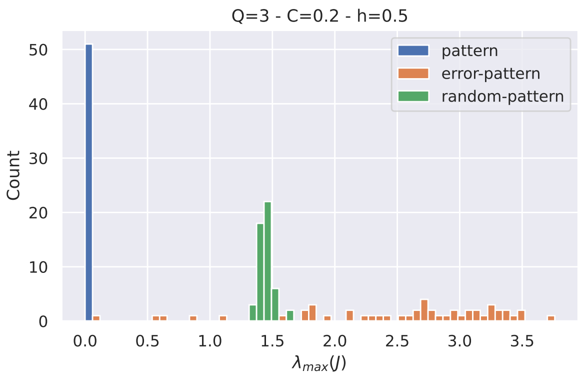
(a)



(b)



(c)



(d)

Figure 3.11: **Network stability.** The distribution of the maximum eigenvalue of Jacobian matrix is plotted for different types of patterns.

is initialized with a corrupted pattern, the initial similarity is low. The similarity approaches one over the course of decoding, indicating error correction and successful decoding.

The extent to which associative memories can retrieve stored patterns depends on the basins of attraction around stored patterns. If the state of a network starts within the basin of attraction for a stored pattern, the network dynamics will move the state toward the stored pattern. The size of the basin determines the amount of noise that can be introduced before an error is made. Fig 3.13 presents results from an error analysis of Q-state oscillator networks. Each subfigure plots the initial versus final similarity for different pattern loadings. In each subfigure, the horizontal axis is the similarity between the stored pattern and the initial network state. The vertical axis is the similarity between the stored pattern and the final network state after simulation. The black line has slope one. Points above the black line indicate error correction, i.e., the final network state is more similar to the stored pattern of the initial network state. When the pattern loading is low, all values of Q correct a significant number of errors. As the pattern loading reaches the pattern capacity, the ability

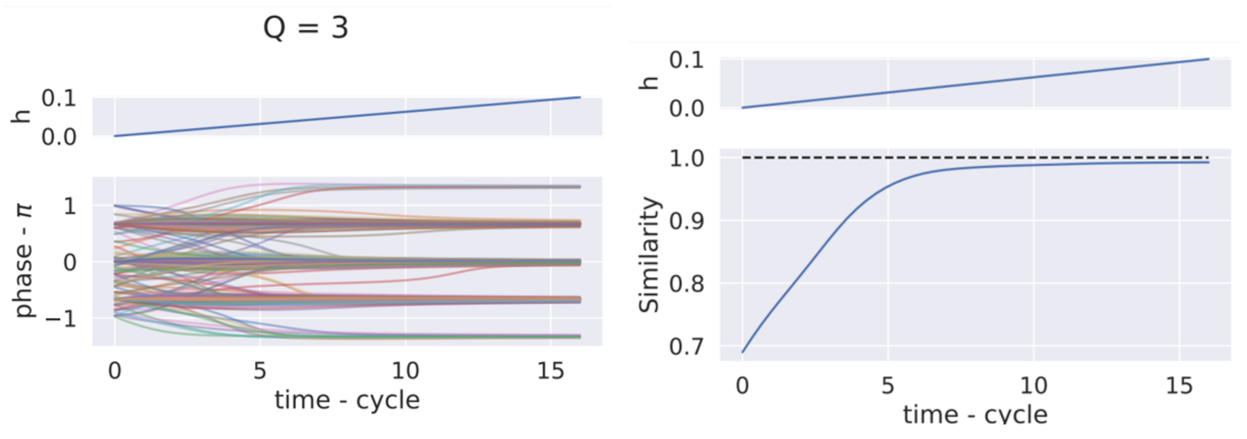


Figure 3.12: **Successful decoding in a 3-state oscillator neural network.** a) Phase versus time is plotted for a 3-state oscillator neural network with sub-harmonic injection locking. The injection locking coefficient, h, is increased linearly over the course of recall. b) Similarity versus time. The network is initialized with a corrupted pattern. The similarity increase to one over the course of recall.

of the network to correct errors decreases.

3.2.9 Thresholding oscillator model

In order to implement the thresholding phasor associative memory in a network of coupled oscillators, the magnitude of the oscillators must be thresholded below a critical value. This can be achieved in several ways depending on the mode. We consider two types of models. One model is a phase-based model like the Kuramoto model that includes a dynamic amplitude variable. The other is a Hopf oscillator model with a dynamic variable that causes a transition through the Hopf bifurcation. For the Hopf oscillator model, the system of differential equations for the thresholding oscillator model is

$$\dot{z}_i = z_i\big(g(\alpha_i) + i\omega_i + \beta_1|z_i|^2\big) + \sum_j w_{ij}z_j \tag{3.20}$$

$$\dot{\alpha}_i = g'(\alpha_i)\big(|\sum_j w_{ij}z_j| - \alpha_t\big). \tag{3.21}$$

Here, z_i is the complex number representing ith oscillator, α_i is a dynamics parameter controlling the oscillator behavior of the ith oscillator, ω_i is the center frequency for the ith oscillator, β_1 is the parameter for the nonlinear part of the oscillator dynamics, $g(\cdot)$ is the hyperbolic tangent function, $g'(\cdot)$ is its derivative, and α_T is the activity threshold value. When the magnitude of the input to the ith oscillator, $|\sum_j w_{ij}z_j|$, is greater than the

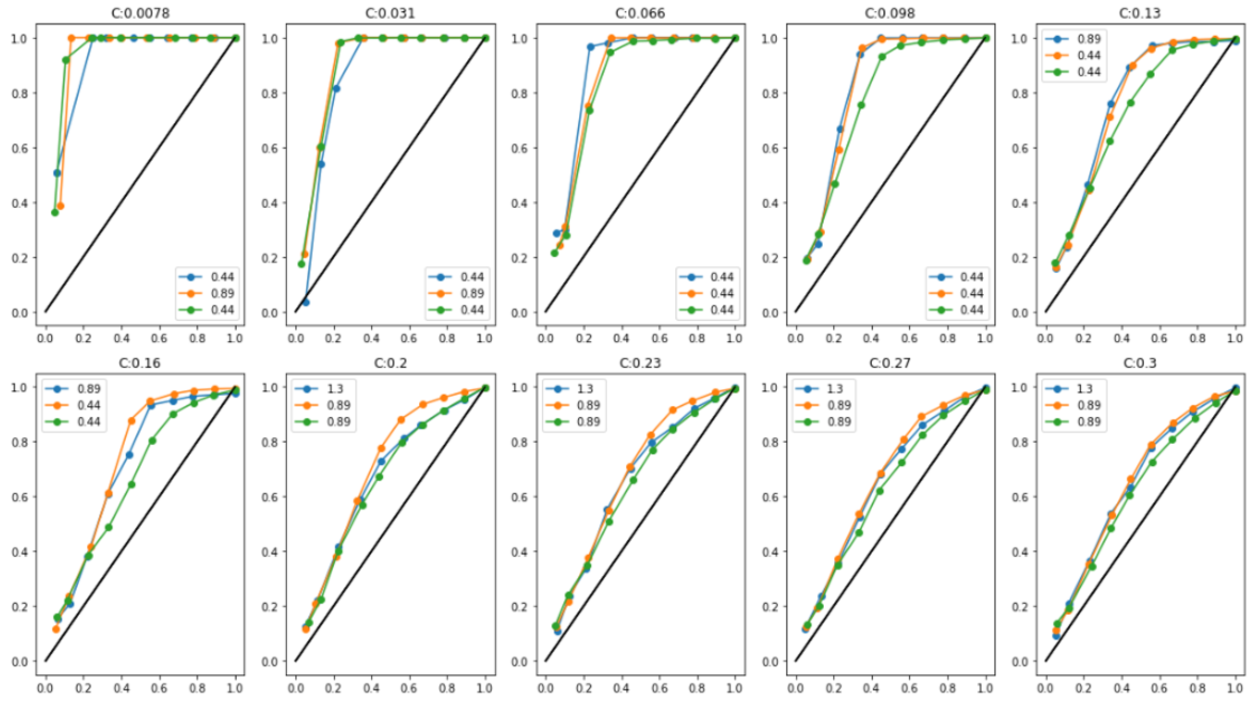


Figure 3.13: **Basins of attraction of Q-state oscillators networks with dense patterns.**

critical value of α_T, α_i increases. Stated another way, when the input is strongly coherent, α_i increases. When the magnitude of the input to the ith oscillator is less than α_T, α_i decreases. The hyperbolic tangent and its derivative act to saturate keep α_i between -1 and 1. The bounds of -1 and 1 are chosen to keep the amplitude of the oscillators around zero and 1 respectively. Though, the amplitude can be adjusted by adding a constant to scale $g(\alpha_i)$ and adjusting β_1.

The activity threshold value, α_i, must be set below the lowest magnitude of the input for active units and above the greatest magnitude of the input for inactive units. If these two values overlap, then there will be errors during recall.

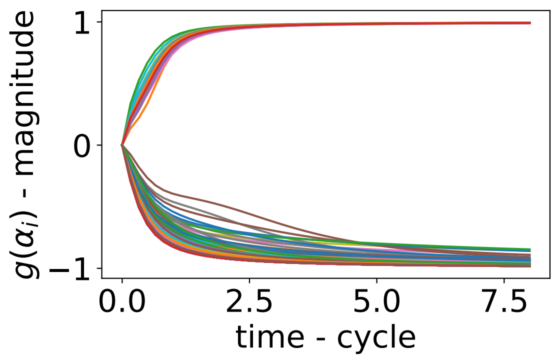
The behavior of the thresholding oscillator model can be understood through the properties of the Hopf bifurcation. When $g(\alpha_i) > 0$ and $\beta_1 < 0$, the is a stable limit cycle with amplitude $\sqrt{-g(\alpha_i)/\beta_1}$. When $g(\alpha_i) < 0$ and $\beta_1 < 0$, the system exhibits damped oscillator behavior and the amplitude decays to zero in the absence of input.

Since the change in α depends on the magnitude of the input, we modify the learning rule to make the magnitude of the input close the magnitude of the stored patterns. The new learning rule is

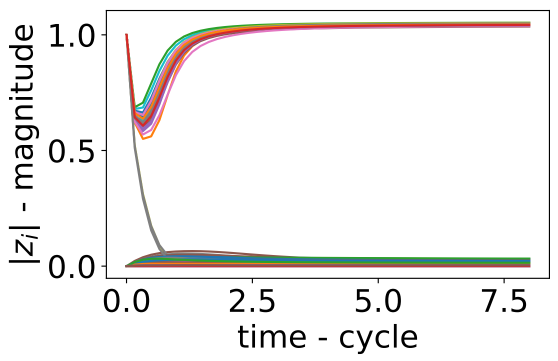
$$W = \sum_{k=1}^{M} \frac{\boldsymbol{\xi}^k \boldsymbol{\xi}^{k*T}}{|\boldsymbol{\xi}^{k*T} \boldsymbol{\xi}^k|^2}. \tag{3.22}$$

A key property of associative memories is the ability to recall stored patterns in the presence of noise. In a thresholding phasor associative memory there can be several types of noise. Two types of noise are considered in this study. One type of noise is cause by false positives, or unit activity when there is non in the stored patterns, and by false negatives, unit which are inactive but should be active. Another type of noise is variation in the phase of the active units of a stored pattern. The two types of noise can be represented as a combination of multiplicative and additive noise. The multiplicative noise adds phase noise to active elements and can deactivate active elements. The additive noise introduces active elements which are inactive in the stored pattern. The individual elements of the corrupted pattern are $\tilde{\xi}_i^k = \eta_i^p \xi_i^k + (1 - |\xi_i^k|)\eta_i^a$. Here, ξ_i^k is the ith element of the kth pattern, η_i^p is the ith element of the phase noise pattern, and η_i^a is the ith element of the amplitude noise. The phase noise is multiplicative. The individual elements of η^p have unit or zero amplitude with phase drawn uniformly from $-\pi$ to π, i.e., $\eta_i^p = r_i^p e^{i\phi_i^p}$ where $\phi_i^p \sim \text{Uniform}(-\pi, \pi)$ and $r_i^p \sim \text{Bernoulli}(P_\text{p})$. The probability of a unit being active is P_p. The individual elements of η^a also have unit or zero amplitude with phase drawn uniformly from $-\pi$ to π, $\eta_i^a = r_i^a e^{i\phi_i^a}$ where $\phi_i^a \sim \text{Uniform}(-\pi, \pi)$ and $r_i^a \sim \text{Bernoulli}(P_a)$. Here, there is a different probability of being active, P_a.

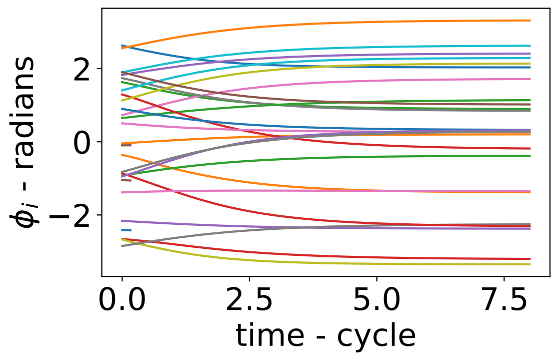
We demonstrate the ability of a thresholding oscillator neural network to implement a thresholding phasor associative memory. Fig. 3.14 presents results from the simulation of a thresholding oscillator associative memory recalling a stored pattern from a corrupted pattern with both phase and amplitude noise. Fig. 3.14a displays the hyperbolic tangent of the parameter controlling the transition through the Hopf bifurcation. α_i is initialized to zero for all oscillators. Over the course of the simulation, the α_i parameters for units with an input magnitude that is greater than the threshold parameter, α_T, increase. The α_i parameters for units with an input magnitude less than the threshold parameter decrease. In both cases, the change in α_i tends towards zero as α_i reaches the saturation regions of the hyperbolic tangent. Fig. 3.14b displays the amplitude of each oscillator over the course of the simulation. At the beginning of the simulation, $\alpha_i = 0\ \forall i$. Therefore, the steady-state amplitude of each oscillator is zero. At the beginning of the simulation, the amplitudes move towards one for all oscillators. Then, for the oscillators which receive input with amplitude greater than α_T, the oscillator amplitude begins to increase, eventually reaching the steady-state limit cycle with an amplitude approximately equal to one. The network of oscillators was initialized with a corrupted pattern that had active units that are not active in the stored pattern. For those units, the input amplitude is less than α_T, As a consequence, for those oscillators, the amplitude decays to approximately zero. Fig. 3.14c displays the oscillator phase over time for those oscillators with amplitude greater than 0.1. The active oscillators at the beginning of the simulation were initialized with phase noise. As the network converges the phase of each oscillator reaches a steady value. Fig 3.14d displays the similarity of the



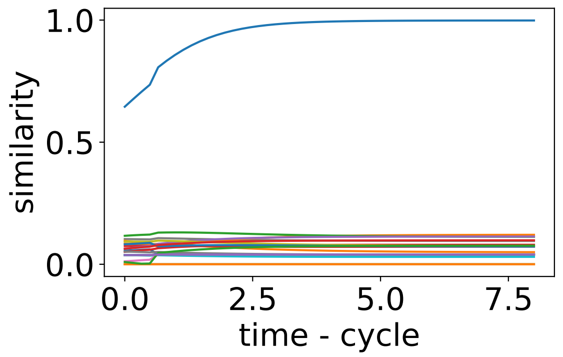
(a)



(b)



(c)



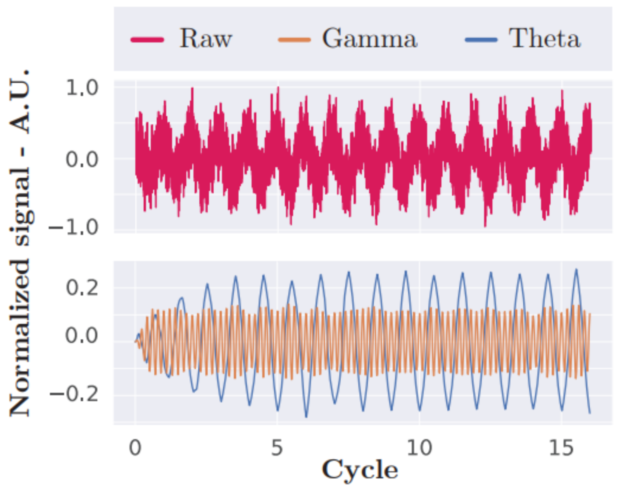
(d)

Figure 3.14: **Thresholding oscillator neural network recalling a stored pattern in the presence of noise** a) The magnitude of $g(\alpha_i)$ plotted over time. b) The magnitude of each oscillator plotted as a function of time. c) The phase of each oscillator is plotted as a function of time. d) The similarity for each stored pattern plotted as a function of time.

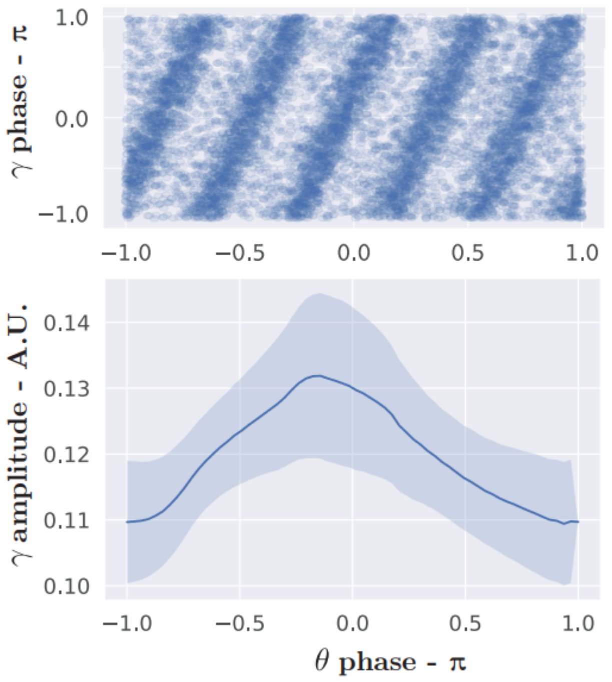
network state to all stored patterns over time. Since the network was initialized with a corrupted pattern, it can be seen that the similarity is higher for one pattern and low for the rest of the patterns. As the network evolves, the similarity for the correct pattern increase, indicating that the network correctly retrieved the stored pattern. Additionally, this shows that the stored patterns are fixed points of the dynamical system.

3.2.10 Cross-frequency coupling

Cross-frequency coupling (CFC) is a phenomenon observed in neural recordings where correlations exist between oscillations at two different frequencies. Two common types of CFC are phase-amplitude coupling (PAC) and phase-phase coupling (PPC). In PAC, the phase of one oscillation is correlated with the amplitude of another oscillation. In PPC, the phases



(a)



(b)

Figure 3.15: **Emergence of cross-frequency coupling in coupled oscillator networks with sub-harmonic injection locking.** (a) Top - The raw signal measured from a population of oscillators containing groups with different frequencies, gamma (γ) and theta (θ). Bottom - Two signals were obtained after filtering at gamma and theta bands. (b) Top - Gamma phase versus theta phase. Bottom - Gamma amplitude versus theta amplitude.

of both oscillations are correlated. CFC has been observed between θ and γ frequencies. θ frequencies range from 4-10 Hz depending on the species and brain region. γ frequencies range from 40-140 Hz.

We demonstrate how Eq. (3.14) can produce CFC in a network of coupled oscillators when the harmonic injection coefficient is modulated by the phase of θ, $h(\phi_\theta)$. Fig. 3.15 presents results from simulations of 5-state oscillator neural networks consisting of two groups of oscillators, θ and γ. Fig. 3.15a plots the normalized amplitude for the raw signal obtained from summing the activity of all the oscillators in the network with additive white noise and filtered signals at the θ and γ frequency bands. Fig. 3.15b plots the PPC and PAC between the filtered θ and γ signals. Q-state oscillator neural networks exhibit PPC and PAC. Here, $Q = 5$. Therefore, in the top plot of Fig. 3.15b, there are 5 bands corresponding to the ratio of γ and θ. The bottom plot of Fig. 3.15b shows PAC between θ and γ.

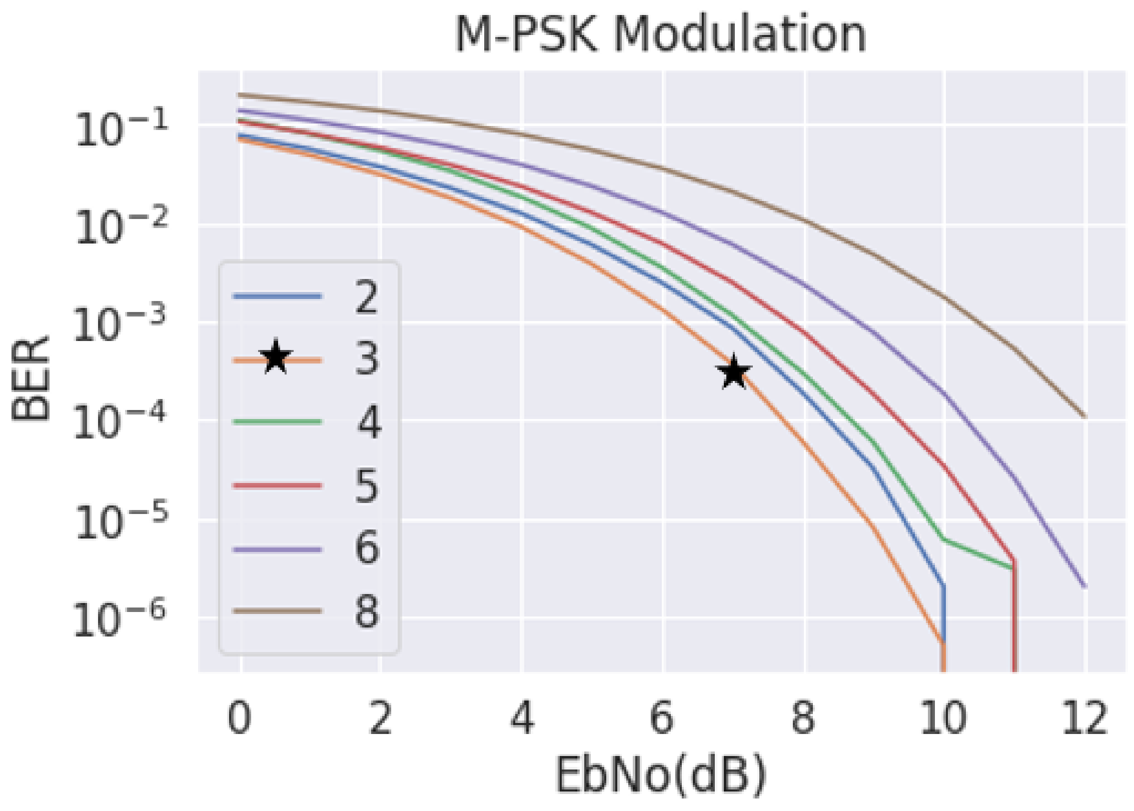


Figure 3.16: **Channel rate for uncoded M-PSK signals.** The bit error rate (BER) is plotted as a function of the energy per bit over the noise power in dB (E_b/N_0). Line represent different values of M, or the number of states per symbol.

3.2.11 Communications

We considered if $Q = 3$ phasors would have useful properties for communicating information. To test this, we computed the bit error rate (BER) for several types of M-ark phase shift keying (M-PSK) sending information on the additive white noise channel (AWGN). It turns out there is a close connection between phasor associative memories and communication of information. The same representation is useful for both robust associative memories and robust communication. Fig. 3.16 shows the results from simulations of sending symbols modulated using M-ary phase shift keying (M-PSK) across the AWGN channel. Notice the similarity to Fig. 3.8a. It appears that $Q = 3$ and $M = 3$ are both optimal. It fact, the advantages of 3-PSK over BPSK and QPSK were discovered only recently [67].

3.3 Discussion

This chapter presents new results for Q-state phasor associative memories and validated predictions on the pattern capacity and information content made by mean-field theory. In addition, we present new models for Q-state oscillatory neural networks that successfully implement Q-state phasor associative memories. This work is also the first to present a sparse coupled oscillator model for TPAMs using Hopf oscillators.

Chapter 4

Efficient Optimization with Higher-Order Ising Machines

A prominent approach to solving combinatorial optimization problems on parallel hardware is Ising machines, i.e., hardware implementations of networks of interacting binary spin variables. Most Ising machines leverage second-order interactions although important classes of optimization problems, such as satisfiability problems, map more seamlessly to Ising networks with higher-order interactions. Here, we demonstrate that higher-order Ising machines can solve satisfiability problems more resource-efficiently in terms of the number of spin variables and their connections when compared to traditional second-order Ising machines. Further, our results show on a benchmark dataset of Boolean k-satisfiability problems that higher-order Ising machines implemented with coupled oscillators rapidly find solutions that are better than second-order Ising machines, thus, improving the current state-of-the-art for Ising machines.

4.1 Introduction

An Ising machine is a type of parallel computer utilizing energy relaxation in a network of interacting binary variables. Ising machines have been proposed as efficient methods for finding optimal or near-optimal solutions to hard combinatorial optimization problems [38, 77, 23, 22, 90, 62]. For a given combinatorial optimization problem, the network interactions are shaped so that the energy minima correspond to the problem solutions. For mapping a given combinatorial optimization problem to a network, a common strategy is to formulate the objective as the energy function of an Ising model, an abstract network of coupled bipolar variables originally proposed to model ferromagnetic material. The Ising model can then be implemented on hardware, referred to as an Ising machine. Ising machines implemented on quantum computers promise optimal solutions [30, 85, 23, 31]. However, due to the challenges of constructing them, Ising machines based on classical physics are reemerging and new technologies are being developed. There is a large variety of possibilities for implementing classical Ising machines, including coupled electrical oscillators [95, 16, 91, 62], optical parametric oscillators [97], stochastic circuits (probabilistic bits) [13], and neuromorphic hardware [38, 47, 20]. Here, we focus on classical Ising machines for approximately solving combinatorial optimization problems at scale and extremely fast.

Casting a combinatorial optimization problem as an Ising model usually takes two or three steps. The first step is to express the combinatorial optimization problem objective as a polynomial in the binary variables. The second step is mapping the polynomial to the energy function of an Ising model. For many combinatorial optimization problems, step one results in a higher-order polynomial [77, 12, 9, 5, 46, 10], i.e., a polynomial with terms that contain products of more than two binary variables. However, most Ising machines utilize second-order polynomial interactions between variables. In this case, a third step, called quadratization [77, 12, 9, 11, 3, 46, 19, 93], is applied for reducing higher-order terms in the polynomial to second-order. The resulting second-order polynomial represents the energy function of a classical Ising model, i.e, a second-order network in which each interaction just couples a pair of variables [59]. Quadratization increases the network size by adding auxiliary variables and it requires increased precision and range of the second-order interaction coefficients compared to higher-order interactions [9, 5].

Higher-order Ising models – models that include polynomial interactions of a degree greater than two – have received little attention because the possible number of interactions grows exponentially with the interaction order. Thus, the training and implementation of higher-order Ising models seemed intractable and impractical [84]. Here, we propose to skip the step of quadratization and instead use higher-order Ising models that directly implement the higher-order polynomials describing the combinatorial optimization problems. Although this proposal seems daunting at first glance, we show that for important classes of combinatorial optimization problems, the corresponding higher-order Ising machines require fewer variables and connections than the second-order Ising machines resulting from the quadratization approach.

Among the proposed Ising machines, coupled electrical oscillators are promising for combinatorial optimization problems in terms of solution quality [94], and the ability to leverage existing technologies such as complementary metal-oxide-semiconductor (CMOS) ring oscillators [96, 65]. To build an oscillator Ising machine, the continuous phases of oscillator variables have to be biased towards two anti-symmetric states, for example, by sub-harmonic injection locking [29, 70, 94]. To demonstrate a concrete higher-order Ising machine, we investigate a network of coupled Hopf oscillators with sub-harmonic injection locking, referred to as a higher-order oscillator Ising machine. Results from our simulations show that the higher-order oscillator Ising machine not only uses fewer network resources compared to the second-order oscillator Ising machine but, importantly, achieves better solutions. All told, our results suggest that, against common beliefs, optimization with higher-order Ising machines can outperform traditional Ising model approaches.

4.2 Results

4.2.1 Mapping constraint satisfaction problems to Ising models

A broad class of combinatorial optimization problems are constraint satisfaction problems, including invertible logic circuits, Boolean satisfiability (SAT) problems, and Boolean maximum satisfiability (MaxSAT) problems. SAT solvers have many direct applications in areas, such as artificial intelligence [17], electronic design automation [92], cryptography [60], and many more. Many Boolean constraint satisfaction problems naturally map to higher-order polynomials [77, 12]. The most common approach for solving constraint satisfaction problems with Ising machines has been first to apply quadratization for translating problems to second-order polynomials, and then use second-order Ising machines to solve them efficiently [77, 9, 5, 11, 59, 19]. However, optimization can also be performed in higher-order Ising machines without quadratization [10, 87, 15]. Here, we aim to construct higher-order Ising machines for Boolean constraint satisfaction problems which are simple, yet, scale to large problems and quickly find near-optimal solutions.

In Boolean constraint satisfaction problems, the Boolean variables must take a state which satisfies a set of pre-defined constraints. For the hth constraint containing k variables, the state space, $\mathbf{S}_h = \{-1, 1\}^k$, can be partitioned into two sets. Let $\mathbf{C}_h = \{\mathbf{c} \in \mathbf{S}_h : \mathbf{c} = \text{satisfied state}\}$ be the set of valid states, i.e., that satisfy the constraint, and $\bar{\mathbf{C}}_h = \mathbf{S}_h \setminus \mathbf{C}_h = \{\mathbf{c} \in \mathbf{S}_h : \mathbf{c} = \text{unsatisfied state}\}$ be the set of invalid states which do not satisfy the constraint. Any logic function can be expressed by a constraint for which the set $\mathbf{C}_h$ represents the truth table of the function. An objective or energy function of the hth constraint, E_h, can be written as the characteristic function of its set of invalid states [77]:

$$E_h(\mathbf{s}) = \sum_{\mathbf{c} \in \bar{\mathbf{C}}_h} \prod_{i=1}^{k} (1 + c_i s_i)/2 \tag{4.1}$$

or, equivalently (Methods 4.4.2), as one minus the characteristic function of its set of valid states:

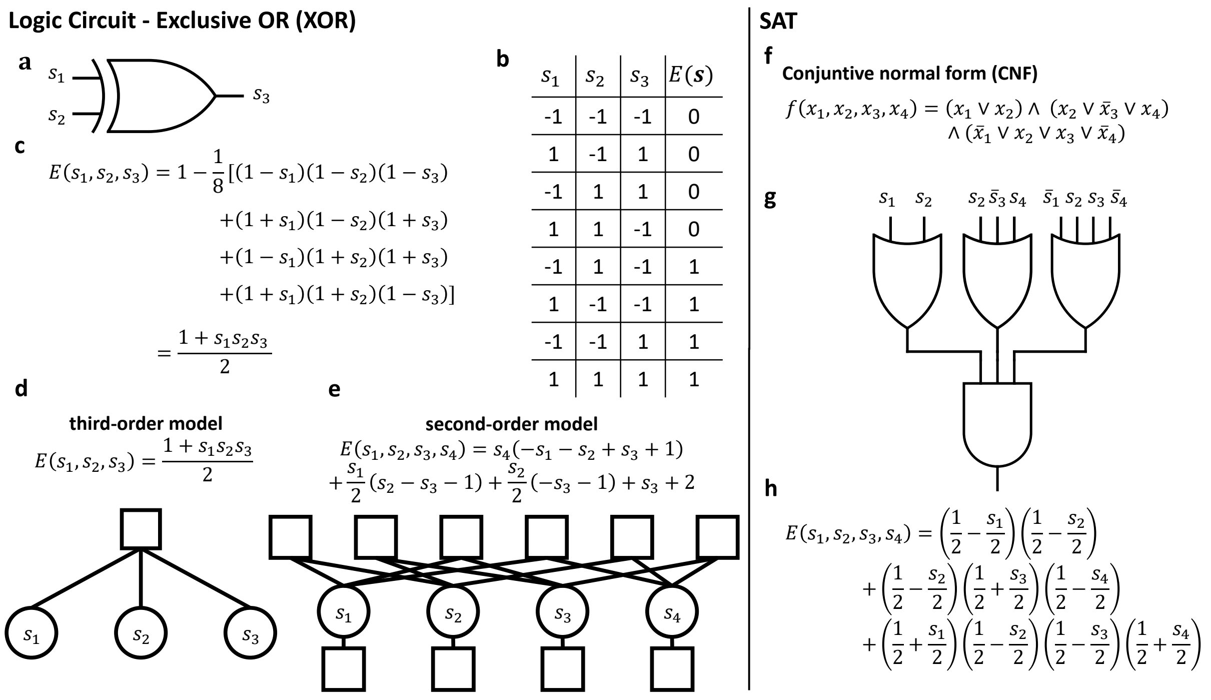
$$E_h(\mathbf{s}) = 1 - \sum_{\mathbf{c} \in \mathbf{C}_h} \prod_{i=1}^{k} (1 + c_i s_i)/2. \tag{4.2}$$

Thus, the sizes of the sets of valid and invalid states may determine which of the two equations is preferable. Let $N_{\mathbf{C}_h} = |\mathbf{C}_h|$ and $N_{\bar{\mathbf{C}}_h} = |\bar{\mathbf{C}}_h|$ denote the size of the set $\mathbf{C}_h$ and $\bar{\mathbf{C}}_h$, respectively. Then, Eqs. (4.1) and (4.2) contain a sum with $N_{\bar{\mathbf{C}}_h}$ and $N_{\mathbf{C}_h}$ terms, respectively. Note that both energies contain higher-order interactions of the order of the size of the constraint.

The total energy for a constraint satisfaction problem is the weighted sum of the individual constraints, Eq. (4.3):

$$E(\mathbf{s}) = \sum_{h \in \mathbf{\Gamma}} w_h E_h(\mathbf{s}). \tag{4.3}$$

Eq. (4.3) generalizes our method to weighted MaxSAT problems, which have many applications [57]. In MaxSAT, each constraint is assigned a weight, w_h, representing the relative importance of satisfying the hth constraint. Here, $\mathbf{\Gamma}$ is the set of indices for the problem constraints, E_h is energy function for the hth constraint formulated according to either Eq. (4.1) or (4.2).


Logic Circuit - Exclusive OR (XOR)
SAT
a
s_1
s_2
s_3
b
c
$E(s_1,s_2,s_3) = 1 - \frac{1}{8}[(1-s_1)(1-s_2)(1-s_3) + (1+s_1)(1-s_2)(1+s_3) + (1-s_1)(1+s_2)(1+s_3) + (1+s_1)(1+s_2)(1-s_3)] = \frac{1+s_1s_2s_3}{2}$
d
third-order model
$E(s_1,s_2,s_3) = \frac{1+s_1s_2s_3}{2}$
e
second-order model
$E(s_1,s_2,s_3,s_4) = s_4(-s_1-s_2+s_3+1) + \frac{s_1}{2}(s_2-s_3-1) + \frac{s_2}{2}(-s_3-1) + s_3 + 2$
s_1
s_2
s_3
s_4
f
Conjuntive normal form (CNF)
$f(x_1,x_2,x_3,x_4) = (x_1 \vee x_2) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee x_2 \vee x_3 \vee \bar{x}_4)$
g
s_1 s_2
$s_2\, \bar{s}_3\, s_4$
$\bar{s}_1\, s_2\, s_3\, \bar{s}_4$
h
$E(s_1,s_2,s_3,s_4) = \left(\frac{1}{2}-\frac{s_1}{2}\right)\left(\frac{1}{2}-\frac{s_2}{2}\right) + \left(\frac{1}{2}-\frac{s_2}{2}\right)\left(\frac{1}{2}+\frac{s_3}{2}\right)\left(\frac{1}{2}-\frac{s_4}{2}\right) + \left(\frac{1}{2}+\frac{s_1}{2}\right)\left(\frac{1}{2}-\frac{s_2}{2}\right)\left(\frac{1}{2}-\frac{s_3}{2}\right)\left(\frac{1}{2}+\frac{s_4}{2}\right)$

s_1	s_2	s_3	$E(\boldsymbol{s})$
-1	-1	-1	0
1	-1	1	0
-1	1	1	0
1	1	-1	0
-1	1	-1	1
1	-1	-1	1
-1	-1	1	1
1	1	1	1

Figure 4.1: **Mapping optimization problems to Ising models.** Two example problems. Left: XOR circuit. **a**, Circuit schematic for the XOR. The XOR gate has two inputs, s_1 and s_2, and one output, s_3. **b**, State table has eight lines. Four lines are input configurations for valid/true output, the other four lines input configurations for invalid/false output. **c**, The higher-order energy function for the XOR in both the factored and simplified form. **d**, Energy and corresponding hypergraph of third-order XOR Ising network, variables nodes, depicted as circles, connected by one interaction, depicted as a square. **e**, Energy and corresponding graph of second-order XOR Ising network, resulting from quadratization. The graph contains four variable nodes (one auxiliary variable), six second-order interactions and four first-order interactions (biases). Right: SAT problem. **f**, SAT problem in CNF. The SAT function is written with binary variables, $x_i \in \{0, 1\}$, where $\bar{x}_i$ denotes the variable negation. **g**, The SAT problem in CNF has an equivalent circuit representation consisting of k-input OR gates which output to one AND gate. **h**, The energy can be succinctly formulated with one term per clause using Eq. (4.1).

Eq. (4.1) or (4.2) are higher-order interactions represented as factored polynomials. The equations can be expanded to coincide with the common formulation of a higher-order Ising model (Eq. 4.5 in Methods 4.4.1). Either the factored or expanded parameterization may be preferred depending on the problem and which form results in the fewest number of terms in the energy. In general, the expanded energy may contain $2^k - 1$ terms or parameters. However, for many practical problems each clause contains only a few literals, hence, k is small. The factored representations require $N_{\mathbf{C}_h} k$ and $N_{\bar{\mathbf{C}}_h} k$ parameters for Eqs. (4.2) and (4.1), respectively. Thus, when k is large or the expanded form does not simplify to a few terms, the factored representation is preferable.

The derivation of Ising models is first explained for two small examples of combinatorial optimization problems, the exclusive OR (XOR) invertible logic gate, and a small SAT problem. The XOR problem can be depicted by the XOR gate symbol (Fig. 4.1**a**), and its state table (Fig. 4.1**b**). The expanded and simplified energy polynomial of XOR contains only one interaction (Fig. 4.1**c**), resulting in a very simple hypergraph of the corresponding third-order Ising network (Fig. 4.1**d**). The quadratization of the third-order XOR polynomial produces a second-order Ising network with one additional auxiliary variable, six second-order interactions, and four biases (Fig. 4.1**e**). The additional network resources required after quadratization may be negligible for small problems but significantly change the scaling behavior of required resources for larger problems (Fig. 4.2).

Any SAT problem can be written as the product (conjunction or AND) of clauses (constraints) where each clause is the Boolean sum (OR) of literals. A literal is a variable or its negation. This form is known as conjunctive normal form (CNF). For a particular 3 clause SAT problem, the CNF (Fig. 4.1**f**) corresponds to a logic gate circuit (Fig. 4.1**g**), and a factored higher-order energy polynomial (Fig. 4.1**h**). The factored energy polynomial of a SAT problem corresponds to Eq. 4.3 with $w_h = 1 \; \forall h$. Therefore, any SAT problem in CNF maps directly to a higher-order Ising model in which each higher-order interaction represents a clause. The order of an interaction corresponds to the size of the corresponding clause.

4.2.2 Model scaling of higher-order and traditional Ising models

Quadratization of higher-order interactions introduces auxiliary variables and adds second-order interactions (XOR example in Fig. 4.1), thereby potentially increasing the total resources required by the corresponding Ising machine. To quantify this effect, Fig. 4.2 compares the resource use of higher-order models versus second-order models on kSAT benchmarks [35, 8]. kSAT is a SAT problem where each clause involves maximally k variables. Quadratization of kSAT proceeds first by reducing a kSAT problem to 3SAT for $k > 3$, which can always be done [49], and then quadratization of the 3SAT problem. We use the D-Wave Ocean software package for quadratization (Methods 4.4.5), which accepts the minimum classical energy gap, ΔE_{min}, as an input parameter. ΔE_{min} is the difference in energy between satisfied states and the lowest energy unsatisfied state. The choice of minimum energy gap value influences the annealing time in quantum adiabatic annealing [23] and the state acceptance probability in simulated annealing [51]. Increasing the minimum energy gap for an Ising machine may improve the optimization, however, it tends to increase the number of auxiliary variables and interactions required (Methods 4.4.5). We compare higher-order to second-order models in terms of the number of variables in the energy function and the number of connections needed to implement all interactions. We consider second-order models with different minimum energy gap values. Nearby values of $\Delta E_{\min}$ result in the same quadratization, therefore, we investigate $\Delta E_{\min}$ settings of 1, 5, 10, and 13 where 1, 2, 3, and 5 auxiliary variables are introduced per clause, respectively. In addition, we found that the method used to perform quadratization increases the required precision or resolution of coupling coefficients from one bit for factored higher-order Ising models to at most six bits. This is another significant difference in resource requirements, as hardware typically offer limited resolution precision for representing interactions [65].

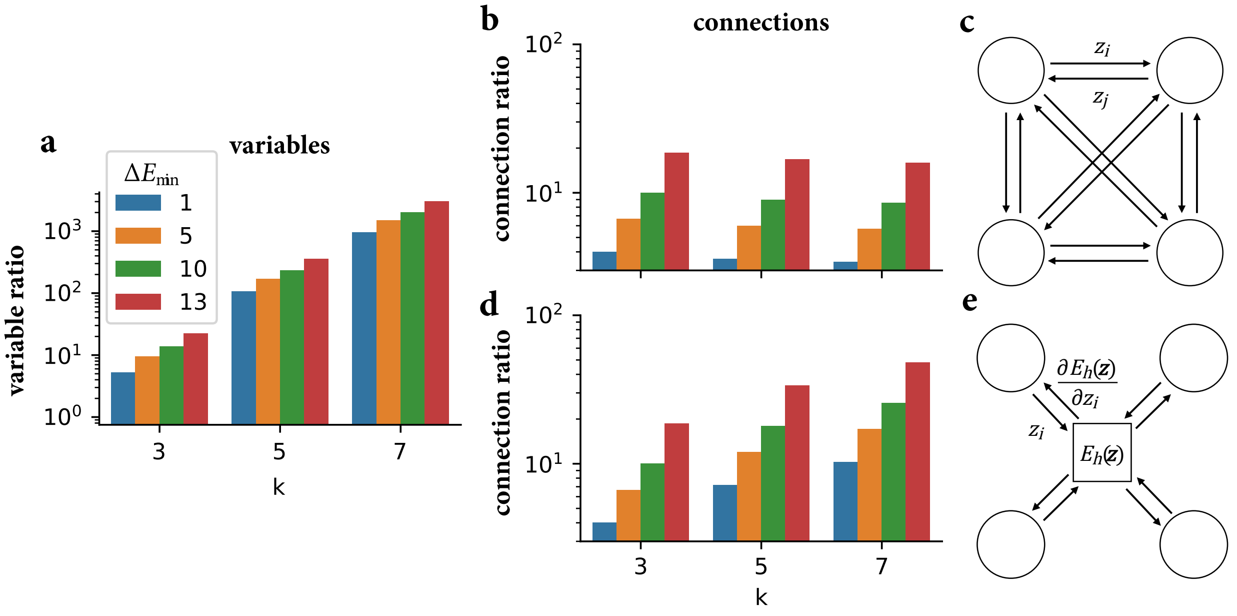


Figure 4.2: **Comparing second-order to third-order model parameters on benchmark kSAT problems.** **a**, **b**, and **d**, The ratio between the number of second-order parameters to higher-order parameters are plotted as a function of the number of variables per constraint in the kSAT problem and different values of $\Delta E_{\min}$. Colors represent reductions with a different minimum energy gap, $\Delta E_{\min}$. The bars are grouped by k, the number of variables per clause. **a**, The ratio of the number of variables required for second-order networks compared to higher-order networks. **c**, A higher-order interaction implemented with all-to-all connectivity. **e**, A higher-order interaction is implemented with a computational node for each constraint. **b**, **d**, The ratio of the number of connections required for second-order networks compared to higher-order networks implemented with all-to-all connectivity, **c**, and intermediate computational nodes, **e**.

To compare the resource use of interactions of different orders we consider the number of connections between nodes that are required for their implementation. The required number of connections depends on the way a higher-order interaction is implemented, here we compare two methods of implementation. The first method is bidirectional connections between all variables participating in the higher-order interaction – a kth-order interaction requires $k(k-1)$ connections (Fig. 4.2**c**). The second method uses an intermediate computational node that receives input from all other variables participating in the interaction and sends output back to all other variables – a kth-order interaction requires $2k$ connections (Fig. 4.2**e**).

Our comparison shows that second-order models based on quadratization of higher-order models require a much greater number of variables and connections compared to higher-order models for kSAT benchmarks Fig. 4.2. In particular, second-order models require three orders of magnitude more variables and one order of magnitude more connections compared to higher-order models. In addition, the number of variables obtained from the D-Wave Ocean software package for $\Delta E_{\text{min}} = 1$ is the same as another method of quadratization based on a circuit decomposition of SAT clauses [1] which introduces one auxiliary variable per clause (Methods 4.4.5).

4.2.3 Solving SAT problems with a higher-order oscillator Ising machine

To compare the computation performance of higher-order Ising machines with corresponding second-order models, we use a concrete network model of coupled oscillators. In our higher-order oscillator Ising machine, each oscillator is described by a complex variable $z_i \in \mathbb{C}$, which evolves according to:

$$\dot{z}_i(t) = f(z_i(t)) - r_i(t)\frac{\partial E(g(\mathbf{z}(t)))}{\partial z_i} + q_i(t)\; l(z_i(t)). \tag{4.4}$$

Here, z_i represents the amplitude and phase of the ith oscillator. On the right-hand side, $f(z_i)$ is the local oscillator dynamics, and $\frac{\partial E(g(\mathbf{z}(t)))}{\partial z_i}$ the partial derivative of the Ising energy with respect to oscillator z_i, with time-dependent coupling coefficient $r_i(t)$, and optional element-wise non-linearity, $g(\mathbf{z}(t)) = \mathbf{z}(t)/|\mathbf{z}(t)|$ for normalizing the amplitude of each oscillator. Further, $l(z_i) = \bar{z}_i$ is the phase quantization signal driving the phase of oscillator z_i to discrete states, with time-dependent "annealing" coefficient, $q_i(t)$. The phase quantization signal is equivalent to sub-harmonic injection locking (Methods 4.4.7).

We compared simulations of higher-order oscillator Ising machines and second-order oscillator Ising machines for solving kSAT benchmarks [35, 8]. Higher-order oscillator Ising machines achieve better solutions than second-order oscillator Ising machines on all 3SAT benchmark problems, as measured by mean energy at the solution points (Fig. 4.3**a**). Only for the smallest problem instances (20 variables), the difference is small. For larger problems, a substantial gap in energy appears and increases with problem size. Interestingly, even second-order oscillator Ising machines with large minimum energy gaps and, correspondingly, high resource use cannot close the performance gap to higher-order oscillator Ising machines. The performance gap amounts to about 0.75 percent of constraints satisfied for the large 3SAT problems (Fig. 4.3**b**). Finding optimal solutions, i.e, states which satisfy all the constraints, is a hard problem as there could be very few satisfying states in the entire

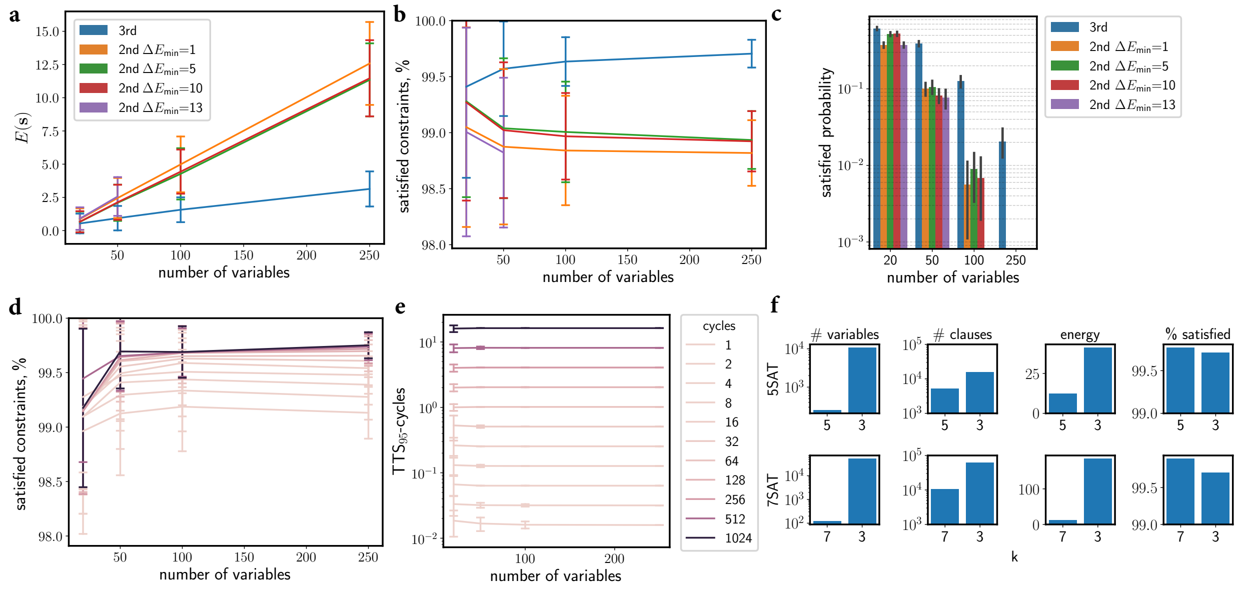


Figure 4.3: **Second-order versus higher-order networks when solving kSAT problems.** **a**, The mean higher-order energy at the end of the simulation is plotted against the number of problem variables for hard instances of 3SAT problems. As the problem size increases, the difference in energy between the second-order oscillator Ising machines and the higher-order oscillator Ising machines increases. **b**, The mean percent of constraints satisfied at the end of simulation versus the problem size for 3SAT problems. **c**, The probability of satisfying all constraints for different problem sizes and models for 3SAT problems. **d**, The mean percent of constraints satisfied at the end of simulation versus the problem size for higher-order oscillator Ising machines for 3SAT problems. **e**, The mean time to satisfy 95% of constraints for higher-order Ising machines for 3SAT problems. **d-e**, Lines indicate different linear annealing schedules for the sub-harmonic injection locking coefficients, q_i. In all plots, error bars represent the sample standard deviation computed over problem instances and trial simulations. **f**, Comparing resources and solutions of 5SAT and 7SAT problems to their 3SAT reductions. Reducing kSAT problems to 3SAT for $k > 3$ increases the number of variables and connections (left two columns). The 5th-order and 7th-order Ising machines find lower energy states corresponding to a greater fraction of constraints satisfied compared to the 3rd-order Ising machine (right two columns).

state space. Nevertheless, for larger problems of the 3SAT benchmarks, higher-order oscillator Ising machines tend to find solutions that satisfy all constraints with greater probability than the second-order oscillator Ising machines, Fig. 4.3**c**. In fact, the higher-order oscillator Ising machine is the first reported Ising machine to find satisfiable solutions to the largest 3SAT problems (250 variables) since the previous efforts with second-order Ising machines have been unable to find solutions satisfying all clauses [1], note the missing bars in Fig. 4.3**c**.

Annealing typically improves the quality of solutions found by Ising machines [46, 10, 94, 1]. In both our higher-order oscillator Ising machine and existing second-order oscillator Ising machines, a process analogous to adiabatic and simulated annealing is achieved by gradually increasing the coefficient in the sub-harmonic injection locking term, q_i [95]. We investigated linear annealing schedules with different duration, measured by the number of cycles of the resonant frequencies of the oscillators. The percentage of constraints satisfied at the end of the annealing schedule improves with the duration of the annealing schedule (Fig. 4.3**d**). The time-to-solution for reaching a fixed target of 95% of constraints satisfied (TTS_{95}) scales linearly with the slope in the annealing schedule (Fig. 4.3**e**). For large slopes, the TTS_{95} can be a fraction of a cycle, consistent with previous findings that oscillator Ising machines rapidly find low energy states [94]. In fact, higher-order oscillator Ising machines can satisfy more than 95% in less than one cycle for all problems.

Many studies on solving kSAT problems for $k > 3$, first use an efficient method for reducing the problem to 3SAT [49] and then focus on solving the resulting 3SAT problem. Here, we use a benchmark dataset of 5SAT and 7SAT problems [33] to assess this strategy for the higher-order oscillator Ising machine in terms of resource efficiency and solution quality (Methods 4.4.4). First, we find that the reduction to 3SAT increases the number of problem variables by one or two orders of magnitude, and there is approximately a 3 and 6 times increase in the number of clauses for 5SAT and 7SAT, respectively (left two columns in Fig. 4.3**f**). Second, we observe that the direct solution of the 5SAT and 7SAT problems satisfy a greater fraction of constraints compared to solutions of corresponding 3SAT reductions (right column in Fig. 4.3**f**). It would be interesting to compare the 5th- and 7th-order oscillator Ising machines to second-order oscillator Ising machines but we were unable to test second-order oscillator Ising machines on these problems due to the large number of auxiliary variables introduced via quadratization.

4.3 Discussion

Much of the existing literature on optimization with Ising machines have focused on second-order Ising networks. Such models were first proposed in [77] for solving constraint satisfaction problems. The authors in [77] originally proposed mapping a SAT problem to a higher-order polynomial but then applied quadratization to map to a second-order Ising model. Our first contribution is to directly compare the resource use of second- and higher-order Ising models for solving SAT problems. Defying common intuition, the comparison reveals that higher-order Ising machines are more resource-efficient than second-order Ising machines for solving large combinatorial optimization problems. The resource efficiency of higher-order models results from the fact that no auxiliary variables are required and many combinatorial optimization problems map to polynomials which correspond to a very sparse higher-order interaction graph. Thus, the savings in higher-order models are in the number of Ising variables, as well as in the number of connections (Fig. 4.2).

Our second contribution is to build a resource-efficient higher-order Ising machine with coupled oscillators and test it on benchmark datasets of SAT problems. Motivated by other recent work [10, 87, 15, 6], we investigated the implementation of a higher-order oscillator Ising machine in a coupled oscillator network. Our model resembles the one in [6], but still differs in several ways. First, we use Hopf oscillators which include amplitude dynamics and capture the dynamics of oscillator hardware [68] more closely than the oscillators modeled by the Kuramoto model in [6]. Second, we introduce a form of annealing, specifically, the gradual increase of the sub-harmonic injection locking coefficient following a linear annealing schedule. Third, our model uses the simplest energy function resulting from the mapping method in [77] (Eqs. (4.1) or (4.2)), a sum of all constraint terms where each constraint term is a product of binary values. In principle, the mapping method specifies an entire family of valid energy functions in which the products in the constraint energy are raised by any positive exponent before summing them. For example, in [22, 6] the constraint terms are squared before summing. Our model choice results in gradient computations with the lowest possible complexity, and, moreover, achieves better solutions on the benchmark problems than a model with squared constraints (Methods 4.4.8).

Higher-order oscillator Ising machines converge to optimal or near-optimal solutions in very few cycles, and importantly, convergence time does not increase with problem size (Fig. 4.3**e**). In some practical cases, solutions are reached in less than one cycle. Further, higher-order oscillator Ising machines outperform second-order Ising machines in solution quality and in some cases find optimal solutions to Boolean constraint satisfaction problems. To our knowledge, this study is the first to report an Ising machine that finds optimal satisfiable solutions for the large 3SAT problems in the benchmark dataset (Fig. 4.3**c**).

It has to be emphasized that our study focuses on optimization methods with a basic Ising model whose only dynamic variables are the spin variables. These methods are extremely fast and resource-efficient, but they sometimes find only near-optimal solutions. Another type of Ising machine with higher-order interactions implements the Lagrange method [66, 22, 63], consisting of two types of dynamic variables, spin variables and Lagrange multipliers. In these models, each constraint term in the objective function is multiplied with a nonnegative variable, the Lagrange multiplier. If a constraint is unsatisfied, the corresponding Lagrange multiplier grows dynamically, until the constraint is satisfied [66, 22, 63]. In theory, the Lagrange models can find optimal solutions in polynomial time but the multipliers can grow exponentially large as a function of time [22]. Further, the time to solution in Lagrange models increases with problem size [22, 63]. The systematic comparison of Lagrange methods with higher-order versus second-order interactions is an interesting topic for future research.

The reported benefits of higher-order Ising machines, and higher-order oscillator Ising machines, in particular, are practically relevant because today many technologies exist for their realization. For example, higher-order interactions require the multiplication of the variables involved in the interaction. The multiplication of coupled electrical ring oscillator voltages can be implemented in the analog domain using existing CMOS technologies [14]. Further, the kth-order interactions of electrical oscillators can be implemented in $\log_2(k)$ stages using a cascade of two-input multipliers or in one stage by a sequence consisting of element-wise log transform, summation, and anti-log transform. Another interesting technology is translinear electronic circuits which make use of the translinear principle [27]. Finally, existing methods for implementing real-valued analog higher-order interactions [99] may be modified for use in higher-order oscillator Ising machines.

4.4 Methods

4.4.1 Mapping optimization problems to higher-order Ising models

The energy function of the generalized Ising model, which includes higher-order interactions, is:

$$E(\mathbf{s}) = -\Big(J^{(0)} + \sum_{i_1} J^{(1)}_{i_1} s_{i_1} + \sum_{i_1<i_2} J^{(2)}_{i_1 i_2} s_{i_1} s_{i_2} + ...+ \sum_{i_1<...<i_k} J^{(k)}_{i_1...i_k} s_{i_1}...s_{i_k} + ... + \sum_{i_1<...<i_n} J^{(n)}_{i_1...i_n} s_{i_1}...s_{i_n}\Big). \quad (4.5)$$

Here, the real-valued variable $J^{(k)}$ represents the k-th order interaction between k spin variables and n is the total number of spin variables in the Ising model. The three groups of terms with 0th to 2nd order interactions on the RHS of (4.5) form the energy function of the traditional Ising model. Note that Eqs. (4.1) and (4.2) can be expanded and reduced into the form of (4.5).

In order to express the objective function of a combinatorial optimization problem as the energy function of an Ising model, binary variables in the optimization problem must be mapped to the spins of the Ising model. In this study, the transformation between problem variables, $x_i \in \{0, 1\}$, and spins, $s_i \in \{-1, 1\}$, uses the standard transformation: $s_i = 2x_i - 1$.

4.4.2 Equivalence of higher-order Ising energy formulations

It is easy to see that Eqs. (4.1) and (4.2) are equivalent. For constraint h, the corresponding sets $\bar{\mathbf{C}}_h$ or $\mathbf{C}_h$ partition the state space. Therefore, any state, s, is an element of one of the two sets and we have:

$$\sum_{\mathbf{c}\in\bar{\mathbf{C}}_h} \prod_{i=1}^{k} (1 + c_i s_i)/2 = 1 - \sum_{\mathbf{c}\in\mathbf{C}_h} \prod_{i=1}^{k} (1 + c_i s_i)/2,$$

with Eq. (4.1) on the LHS and Eq. (4.2) on the RHS. The product terms evaluate to 1 when $\mathbf{s} = \mathbf{c}$ and 0 otherwise. If $\mathbf{s} \in \bar{\mathbf{C}}_h$, both sides equal 1, if $\mathbf{s} \in \mathbf{C}_h$, both sides equal 0. Therefore, Eqs. (4.1) and (4.2) represent the same objective function and can be used interchangeably.

4.4.3 Derivatives of higher-order Ising energy functions

The partial derivatives of Eqs. (4.1) and (4.2) with respect to a complex variable, z_i, can be efficiently computed as:

$$\frac{\partial E_h(\mathbf{z})}{\partial z_i} = -\sum_{c\in\bar{C}_h} c_i \prod_{j\neq i}(1+c_j z_j)/2. \tag{4.6}$$

and

$$\frac{\partial E_h(\mathbf{z})}{\partial z_i} = \sum_{c\in C_h} c_i \prod_{j\neq i}(1+c_j z_j)/2 \tag{4.7}$$

4.4.4 Method for reducing kSAT to 3SAT

In this study, we also investigate a polynomial-time method [49] to reduce kSAT to 3SAT when $k > 3$. The method works as follows. Let $\vee$, $\wedge$, and $\sim$ denote the logical OR, AND, and NOT operations, respectively. Consider a clause with 5 binary variables, $(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5)$. Introduce auxiliary variables y_1 and y_2. Introduce new clauses and insert auxiliary variables as:

$$(x_1 \vee x_2 \vee \tilde{y}_1) \wedge (x_3 \vee x_4 \vee \tilde{y}_2) \wedge (y_1 \vee y_2 \vee x_5).$$

The problem is 3SAT as no clause has greater than 3 variables. Reducing one 5SAT clause to 3SAT form results in 3 clauses and 7 variables.

Consider a clause with 7 variables, $(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6 \vee x_7)$. Introduce auxiliary variables y_1, y_2, and y_3. Introduce new clauses and insert auxiliary variables:

$$(x_1 \vee x_2 \vee \tilde{y}_1) \wedge (x_3 \vee x_4 \vee \tilde{y}_2) \wedge (x_5 \vee x_6 \vee \tilde{y}_3) \wedge (x_7 \vee y_1 \vee y_2 \vee y_3).$$

The last clause contains 4 variables so it has to be reduced further. Introduce auxiliary variables l_1 and l_2. Introduce new clauses and insert auxiliary variables:

$$(x_1 \vee x_2 \vee \tilde{y}_1) \wedge (x_3 \vee x_4 \vee \tilde{y}_2) \wedge (x_5 \vee x_6 \vee \tilde{y}_3) \wedge (x_7 \vee y_1 \vee \tilde{l}_1) \wedge (y_2 \vee y_3 \vee \tilde{l}_2) \wedge (l_1 \vee l_2).$$

The problem is 3SAT as no clause has greater than 3 variables. Reducing one 7SAT clause to 3SAT results in 6 clauses and 12 variables.

4.4.5 Excess resource use by different quadratization methods

Numerous quadratization methods have been proposed for reducing objectives with higher-order interactions to energy functions of second-order Ising energies [77, 12, 11, 19, 1]. In general, the number of auxiliary variables introduced by quadratization depends on the particular combinatorial optimization problem and the method of quadratization. In this study, quadratization was performed with the D-Wave Ocean software package [1]. With the quadratization method in D-Wave Ocean one can adjust the minimum energy gap, $\Delta E_{\min}$,

[1] D-Wave Ocean Software Documentation [online], 2022.–Available online: `https://docs.ocean.dwavesys.com/en/stable`.

for controlling the tradeoff between excess resource use and computation performance of the resulting second-order Ising machine.

With the parameter choice of $\Delta E_{\min} = 1$, one auxiliary variable is introduced per 3SAT clause, the same number as with some other quadratization methods [1]. Thus, the excess resource use of quadratization we report for this parameter choice generalizes to other methods in the literature. In addition, we also assess the resource use with parameter settings of $\Delta E_{\min} > 1$ in D-Wave Ocean. These results are specific to the D-Wave Ocean quadratization method, but informative for exploring whether increased excess resource use could potentially close the performance gap between second-order and higher-order oscillator networks.

4.4.6 Benchmark datasets

We assess the performance of higher-order Ising machines on Boolean satisfiability (k-satisfiability, kSAT) problems, a well-known class of hard combinatorial optimization problems. Specifically, the 3SAT problems used in our experiments were obtained from the SATLIB collection [35][2]. We selected instances of sizes 20, 50, 100, and 250 variables. The first sixteen instances were selected from each problem size to run the simulations. The dynamic variables in the oscillator networks were randomly initialized for each trial simulation. 64 trial simulations were performed for each instance.

To demonstrate the performance of higher-order Ising machines on 5SAT and 7SAT problems, we selected an instance of each problem from the 2018 SAT Competition [33][3]. The 5SAT and 7SAT problems were also reduced to 3SAT using the method described in Section 4.4.4.

4.4.7 Oscillator model and simulation details

In higher-order oscillator Ising machines, each oscillator is represented by the complex Van der Pol or Hopf oscillator as described in Eq. (4.8):

$$f(z_i) = (\lambda_i + i\omega_i)z_i + \rho_i z_i |z_i|^2. \tag{4.8}$$

Here, ω_i is the center frequency for the ith oscillator, λ_i is a parameter determining the oscillator quality, and ρ_i controls the degree of nonlinearity.

In our simulations, the network coupling, $r_i(t)$, was the same for all oscillators and was held constant for the duration of the simulation. The center frequency was held constant at zero for all oscillators, $\omega_i = 0 \ \forall i$. The parameters λ_i and ρ_i were set to produce limit-cycle oscillations with unit amplitude. We used a linear annealing schedule, $q_i(t) = q_{\max}\frac{t}{t_{\text{end}}}$. The phase quantization signal, $l(z_i)$ is equivalent to sub-harmonic injection locking. We show

[2] SATLIB - Benchmark Problems [online], 2022.–Available online:`https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`.

[3] SAT Competition [online], 2018.–Available online:`https://satcompetition.github.io/2018/`.

this by representing each oscillator, z_i, with a real and imaginary part $(a_i + \mathrm{i}b_i)$. By adding the conjugate of z_i to the dynamics, the real part grows and the imaginary part decays to zero. The solutions to the dynamics for each uncoupled oscillator including the limit-cycle dynamics, $f(z_i)$, are $a_i = \sqrt{(h_i + \lambda_i)/\rho_i}$

The results reported in Fig. 4.3 were obtained using a parameter search to find the optimal values of λ, ρ, $q_{\max}$, and r. The best candidates were selected based on the lowest mean energy and the greatest mean probability of satisfying problem instances. The mean energy and percent of constraints satisfied were computed based on the final state of the network after simulation. The mean was computed across random network initializations for all trail simulations across problem instances within each problem size. The error bars in Fig. 4.3 represent the sample standard deviation. Integration of the dynamical system was performed using an adaptive step-size RK4/5 method.[4]

4.4.8 Comparing higher-order constraint energy functions with different exponents

For a kSAT problem, the objective for clause h in our method (4.1) simplifies to:

$$E_h(\mathbf{s}) = \prod_{i=1}^{k} (1 - c_i s_i)/2. \tag{4.9}$$

with $c_i = 1$ if a literal is TRUE and $c_i = -1$ if a literal is FALSE. Since $E_h(\mathbf{s})$ evaluates to either one or zero for all bipolar state vectors, $\mathbf{s}$, an obvious generalization of the clause objective (4.9) is to exponentiate the RHS by a positive number. In [22, 6], the objective of kSAT problems with a higher-order energy function of this type was proposed, with the specific setting of the exponent set to a value of two:

$$E_h(\mathbf{s}) = \big(\prod_{i=1}^{k} (1 - c_i s_i)/2\big)^2. \tag{4.10}$$

We compared the solution quality of higher-order oscillator Ising machines implementing objective (4.9) vs. (4.10). Our experiments included parameter optimization for each method, as described above (Section 4.4.7). Fig. 4.4 shows that networks based on (4.10) obtain worse solutions (with greater energies) and satisfy only a smaller percentage of constraints on benchmark 3SAT problems [35] compared to our method. The systematic analysis of exponent settings in the generalization of our method is left to future research.

[4] JAX: Autograd and XLA [online], 2022.–Available online:`https://github.com/google/jax`.

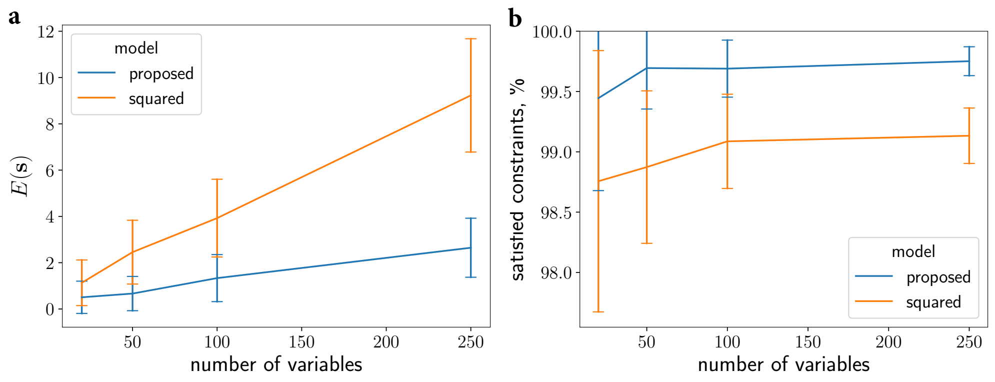


Figure 4.4: **Final energy on benchmarks 3SAT problems for the method proposed in this paper using Eq. (4.9) and the method using the square of the constraint energy (4.10) as in [22, 6].** **a**, Energy versus the number of variables for 3SAT problems. **b**, The percent of constraints satisfied versus the number of variables for 3SAT problems.

Bibliography

[1] Navid Anjum Aadit et al. “Massively parallel probabilistic computing with sparse Ising machines”. In: *Nature Electronics* (2022), pp. 1–9.

[2] Filipp Akopyan et al. “Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34.10 (2015), pp. 1537–1557.

[3] Martin Anthony et al. “Quadratic reformulations of nonlinear binary optimization problems”. In: *Mathematical Programming* 162.1 (2017), pp. 115–144.

[4] Toshio Aoyagi. “Network of neural oscillators for retrieving phase information”. In: *Physical review letters* 74.20 (1995), p. 4075.

[5] Ryan Babbush, Bryan O’Gorman, and Alán Aspuru-Guzik. “Resource efficient gadgets for compiling adiabatic quantum optimization problems”. In: *Annalen der Physik* 525.10-11 (2013), pp. 877–888.

[6] Mohammad Khairul Bashar, Zongli Lin, and Nikhil Shukla. “Formulating Oscillator-Inspired Dynamical Systems to Solve Boolean Satisfiability”. In: *arXiv preprint arXiv:2209.07571* (2022).

[7] Lou Beaulieu-Laroche et al. “Enhanced dendritic compartmentalization in human cortical neurons”. In: *Cell* 175.3 (2018), pp. 643–651.

[8] Olaf Beyersdorff and Christoph M Wintersteiger. *Theory and Applications of Satisfiability Testing: SAT 2018*. Springer, 2018.

[9] J D Biamonte. “Nonperturbative k-body to two-body commuting conversion Hamiltonians and embedding problem instances into Ising spins”. In: *Physical Review A* 77.5 (2008), p. 052331.

[10] William A Borders et al. “Integer factorization using stochastic magnetic tunnel junctions”. In: *Nature* 573.7774 (2019), pp. 390–393.

[11] Endre Boros and Aritanan Gruber. “On quadratization of pseudo-Boolean functions”. In: *arXiv preprint arXiv:1404.6538* (2014).

[12] Endre Boros and Peter L Hammer. “Pseudo-boolean optimization”. In: *Discrete Applied Mathematics* 123.1-3 (2002), pp. 155–225.

[13] Kerem Yunus Camsari et al. “Stochastic p-bits for invertible logic”. In: *Physical Review X* 7.3 (2017), p. 031014.

[14] Chunhong Chen and Zheng Li. “A low-power CMOS analog multiplier”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 53.2 (2006), pp. 100–104.

[15] Dmitry A Chermoshentsev et al. “Polynomial unconstrained binary optimisation inspired by optical simulation”. In: *arXiv preprint arXiv:2106.13167* (2021).

[16] Jeffrey Chou et al. “Analog coupled oscillator based weighted Ising machine”. In: *Scientific Reports* 9.1 (2019), pp. 1–10.

[17] Edmund Clarke et al. “Bounded model checking using satisfiability solving”. In: *Formal Methods in System Design* 19.1 (2001), pp. 7–34.

[18] J. Cook. “The mean-field theory of a Q-state neural network model”. In: *Journal of Physics A: Mathematical and General* 22 (12 June 1989), pp. 2057–2067. ISSN: 13616447. DOI: `10.1088/0305-4470/22/12/011`. URL: `https://iopscience.iop.org/article/10.1088/0305-4470/22/12/011https://iopscience.iop.org/article/10.1088/0305-4470/22/12/011/meta`.

[19] Nike Dattani. “Quadratization in discrete optimization and quantum mechanics”. In: *arXiv preprint arXiv:1901.04405* (2019).

[20] Mike Davies et al. “Advancing neuromorphic computing with Loihi: A survey of results and outlook”. In: *Proceedings of the IEEE* 109.5 (2021), pp. 911–934.

[21] Mike Davies et al. “Loihi: A neuromorphic manycore processor with on-chip learning”. In: *IEEE Micro* 38.1 (2018), pp. 82–99.

[22] Mária Ercsey-Ravasz and Zoltán Toroczkai. “Optimization hardness as transient chaos in an analog approach to constraint satisfaction”. In: *Nature Physics* 7.12 (2011), pp. 966–970.

[23] Edward Farhi et al. “Quantum computation by adiabatic evolution”. In: *arXiv preprint quant-ph/0001106* (2000).

[24] E. Paxon Frady and Friedrich T. Sommer. “Robust computation with rhythmic spike patterns”. In: *Proceedings of the National Academy of Sciences of the United States of America* (2019). ISSN: 10916490. DOI: `10.1073/pnas.1902653116`.

[25] E Paxon Frady and Friedrich T Sommer. “Robust computation with rhythmic spike patterns”. In: *Proceedings of the National Academy of Sciences* 116.36 (2019), pp. 18050–18059.

[26] George M Georgiou and Cris Koutsougeras. “Complex domain backpropagation”. In: *IEEE transactions on Circuits and systems II: analog and digital signal processing* 39.5 (1992), pp. 330–334.

[27] Barrie Gilbert. “Translinear circuits: A proposed classification”. In: *Electronics Letters* 1.11 (1975), pp. 14–16.

[28] Dan FM Goodman and Romain Brette. “The brian simulator”. In: *Frontiers in neuroscience* 3 (2009), p. 26.

[29] Eiichi Goto. “The parametron, a digital computing element which utilizes parametric oscillation”. In: *Proceedings of the IRE* 47.8 (1959), pp. 1304–1316.

[30] Lov K Grover. “A fast quantum mechanical algorithm for database search”. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 212–219.

[31] Philipp Hauke et al. “Perspectives of quantum annealing: Methods and implementations”. In: *Reports on Progress in Physics* 83.5 (2020), p. 054401.

[32] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[33] Marijn JH Heule. “Generating the uniform random benchmarks”. In: *Proceedings of SAT competition* 2018 (2018).

[34] Akira Hirose. “Continuous complex-valued back-propagation learning”. In: *Electronics Letters* 28.20 (1992), pp. 1854–1855.

[35] Holger H Hoos and Thomas Stützle. “SATLIB: An online resource for research on SAT”. In: *SAT2000* (2000), pp. 283–292.

[36] John J Hopfield. “Neural networks and physical systems with emergent collective computational abilities”. In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[37] John J Hopfield. “Neurons with graded response have collective computational properties like those of two-state neurons.” In: *Proceedings of the national academy of sciences* 81.10 (1984), pp. 3088–3092.

[38] John J Hopfield and David W Tank. ““Neural” computation of decisions in optimization problems”. In: *Biological Cybernetics* 52.3 (1985), pp. 141–152.

[39] Frank C Hoppensteadt and Eugene M Izhikevich. “Associative memory of weakly connected oscillators”. In: *Proceedings of International Conference on Neural Networks (ICNN’97)*. Vol. 2. IEEE. 1997, pp. 1135–1138.

[40] Frank C. Hoppensteadt and Eugene M. Izhikevich. “Pattern recognition via synchronization in phase-locked loop neural networks”. In: *IEEE Transactions on Neural Networks* 11 (3 May 2000), pp. 734–738. DOI: `10.1109/72.846744`.

[41] Frank C Hoppensteadt and Eugene M Izhikevich. *Weakly connected neural networks*. Vol. 126. Springer Science & Business Media, 1997.

[42] Eric Hunsberger and Chris Eliasmith. “Spiking deep networks with LIF neurons”. In: *arXiv preprint arXiv:1510.08829* (2015).

[43] E Ising. “Beitrag zur theorie des ferromagnetismus Zeit. fur Physik 31”. In: (1925).

[44] Eugene M Izhikevich. *Dynamical systems in neuroscience*. Ed. by MIT Press. MIT press, 2007.

[45] Eugene M Izhikevich. “Polychronization: computation with spikes”. In: *Neural computation* 18.2 (2006), pp. 245–282.

[46] Shuxian Jiang et al. “Quantum annealing for prime factorization”. In: *Scientific Reports* 8.1 (2018), pp. 1–9.

[47] Zeno Jonke, Stefan Habenschuss, and Wolfgang Maass. “Solving constraint satisfaction problems with networks of spiking neurons”. In: *Frontiers in Neuroscience* 10 (2016), p. 118.

[48] Ido Kanter. “Potts-glass models of neural networks”. In: *Physical Review A* 37.7 (1988), p. 2739.

[49] Richard M Karp. “Reducibility among combinatorial problems”. In: *Complexity of Computer Computations*. Springer, 1972, pp. 85–103.

[50] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[51] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. “Optimization by simulated annealing”. In: *Science* 220.4598 (1983), pp. 671–680.

[52] Alex Krizhevsky and Geoffrey Hinton. *Learning multiple layers of features from tiny images*. Tech. rep. CIFAR, 2009.

[53] Ankit Kumar and Pritiraj Mohanty. “Autoassociative memory and pattern recognition in micromechanical oscillator network”. In: *Scientific reports* 7.1 (2017), pp. 1–9.

[54] Edward W Large. “Neurodynamics of music”. In: *Music perception*. Springer, 2010, pp. 201–231.

[55] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[56] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. “Training deep spiking neural networks using backpropagation”. In: *Frontiers in neuroscience* 10 (2016), p. 508.

[57] Chu-Min Li and Felip Manyà. *Theory and Applications of Satisfiability Testing: SAT 2021*. Springer, 2021.

[58] Michael London et al. “Sensitivity to perturbations in vivo implies high noise and suggests rate coding in cortex”. In: *Nature* 466.7302 (2010), p. 123.

[59] Andrew Lucas. “Ising formulations of many NP problems”. In: *Frontiers in Physics* 2 (2014), pp. 1–15.

[60] Fabio Massacci and Laura Marraro. “Logical cryptanalysis as a SAT problem”. In: *Journal of Automated Reasoning* 24.1 (2000), pp. 165–203.

[61] Yoshiko Matsumoto-Makidono et al. “Ionic basis for membrane potential resonance in neurons of the inferior olive”. In: *Cell reports* 16.4 (2016), pp. 994–1004.

[62] Naeimeh Mohseni, Peter L McMahon, and Tim Byrnes. "Ising machines as hardware solvers of combinatorial optimization problems". In: *Nature Reviews Physics* 4.6 (2022), pp. 363–379.

[63] Botond Molnár et al. "A continuous-time MaxSAT solver with high analog performance". In: *Nature Communications* 9.1 (2018), pp. 1–12.

[64] Hesham Mostafa. "Supervised learning based on temporal coding in spiking neural networks". In: *IEEE transactions on neural networks and learning systems* 29.7 (2018), pp. 3227–3235.

[65] William Moy et al. "A 1,968-node coupled ring oscillator circuit for combinatorial optimization problem solving". In: *Nature Electronics* 5.5 (2022), pp. 310–317.

[66] Masahiro Nagamatu and Torao Yanaru. "On the stability of Lagrange programming neural networks for satisfiability problems of prepositional calculus". In: *Neurocomputing* 13.2-4 (1996), pp. 119–133.

[67] Makoto Nakamura and H Torii. "Ternary phase shift keying and its performance". In: *The 5th International Symposium on Wireless Personal Multimedia Communications.* Vol. 3. IEEE. 2002, pp. 1284–1288.

[68] Dmitri E Nikonov et al. "Coupled-oscillator associative memory array operation for pattern recognition". In: *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 1 (2015), pp. 85–93.

[69] Takashi Nishikawa, Frank C. Hoppensteadt, and Ying Cheng Lai. "Oscillatory associative memory network with perfect retrieval". In: *Physica D: Nonlinear Phenomena* 197 (1-2 Oct. 2004), pp. 134–148. ISSN: 0167-2789. DOI: `10.1016/J.PHYSD.2004.06.011`.

[70] Takashi Nishikawa, Ying-Cheng Lai, and Frank C Hoppensteadt. "Capacity of oscillatory associative-memory networks with error-free retrieval". In: *Physical review letters* 92.10 (2004), p. 108101.

[71] André J Noest. "Discrete-state phasor neural networks". In: *Physical Review A* 38.4 (1988), p. 2196.

[72] André J Noest. "Phasor neural networks". In: *Neural information processing systems.* 1988, pp. 584–591.

[73] André J Noest. "Phasor neural networks". In: 1987, pp. 584–591.

[74] Peter O'Connor, Efstratios Gavves, and Max Welling. "Training a Network of Spiking Neurons with Equilibrium Propagation". In: *arxiv* (2018).

[75] Günther Palm. "On associative memory". In: *Biological cybernetics* 36.1 (1980), pp. 19–31.

[76] Adam Paszke et al. "Automatic differentiation in PyTorch". In: *NeurIPS 2017* (2017).

[77] Gadi Pinkas. "Symmetric neural networks and propositional logic satisfiability". In: *Neural Computation* 3.2 (1991), pp. 282–291.

[78] Renfrey Burnard Potts. “Some generalized order-disorder transformations”. In: *Mathematical proceedings of the cambridge philosophical society.* Vol. 48. 1. Cambridge University Press. 1952, pp. 106–109.

[79] Deboleena Roy, Priyadarshini Panda, and Kaushik Roy. “Learning Spatio-Temporal Representations Using Spike-Based Backpropagation”. In: *arxiv* (2018).

[80] Bodo Rueckauer and Shih-Chii Liu. “Conversion of analog to spiking neural networks using sparse temporal coding”. In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS).* IEEE. 2018, pp. 1–5.

[81] João Sacramento et al. “Dendritic cortical microcircuits approximate the backpropagation algorithm”. In: *Advances in Neural Information Processing Systems.* 2018, pp. 8735–8746.

[82] Benjamin Scellier and Yoshua Bengio. “Equilibrium propagation: Bridging the gap between energy-based models and backpropagation”. In: *Frontiers in computational neuroscience* 11 (2017), p. 24.

[83] Catherine D Schuman et al. “A survey of neuromorphic computing and neural networks in hardware”. In: *arXiv preprint arXiv:1705.06963* (2017).

[84] Terrence J Sejnowski. “Higher-order Boltzmann machines”. In: *AIP Conference Proceedings.* Vol. 151. 1. American Institute of Physics. 1986, pp. 398–403.

[85] Peter W Shor. “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”. In: *SIAM Review* 41.2 (1999), pp. 303–332.

[86] Karl Steinbuch. “Die lernmatrix”. In: *Kybernetik* 1.1 (1961), pp. 36–45.

[87] Nikita Stroev and Natalia G Berloff. “Discrete polynomial optimization with coherent networks of condensates and complex coupling switching”. In: *Physical Review Letters* 126.5 (2021), p. 050504.

[88] David Tingley and György Buzsáki. “Transformation of a spatial map across the hippocampal-lateral septal circuit”. In: *Neuron* 98.6 (2018), pp. 1229–1242.

[89] Chiheb Trabelsi et al. “Deep complex networks”. In: *arXiv preprint arXiv:1705.09792* (2017).

[90] Sri Krishna Vadlamani, Tianyao Patrick Xiao, and Eli Yablonovitch. “Physics successfully implements Lagrange multiplier optimization”. In: *Proceedings of the National Academy of Sciences* 117.43 (2020), pp. 26639–26650.

[91] Jaykumar Vaidya, RS Surya Kanthi, and Nikhil Shukla. “Creating electronic oscillator-based Ising machines without external injection locking”. In: *Scientific Reports* 12.1 (2022), pp. 1–8.

[92] Yakir Vizel, Georg Weissenbacher, and Sharad Malik. “Boolean satisfiability solvers and their applications in model checking”. In: *Proceedings of the IEEE* 103.11 (2015), pp. 2021–2035.

[93] Baonan Wang et al. "Prime factorization algorithm based on parameter optimization of Ising model". In: *Scientific Reports* 10.1 (2020), pp. 1–10.

[94] Tianshi Wang and Jaijeet Roychowdhury. "OIM: Oscillator-based Ising Machines for Solving Combinatorial Optimisation Problems". In: *arXiv preprint arXiv:1903.07163* (2019).

[95] Tianshi Wang and Jaijeet Roychowdhury. "Oscillator-based Ising machine". In: *arXiv preprint arXiv:1709.08102* (2017).

[96] Tianshi Wang et al. "Solving combinatorial optimisation problems using oscillator based Ising machines". In: *Natural Computing* 20.2 (2021), pp. 287–306.

[97] Zhe Wang et al. "Coherent Ising machine based on degenerate optical parametric oscillators". In: *Physical Review A* 88.6 (2013), p. 063853.

[98] David J Willshaw, O Peter Buneman, and Hugh Christopher Longuet-Higgins. "Non-holographic associative memory". In: *Nature* 222.5197 (1969), pp. 960–962.

[99] Xunzhao Yin et al. "Efficient analog circuits for Boolean satisfiability". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26.1 (2017), pp. 155–167.