Dynamics and Control in Spiking Neural Networks

Fuqiang Huang
Washington University in St. Louis

Follow this and additional works at: https://openscholarship.wustl.edu/eng_etds

Part of the Engineering Commons, and the Neuroscience and Neurobiology Commons

Recommended Citation
https://openscholarship.wustl.edu/eng_etds/495

This Dissertation is brought to you for free and open access by the McKelvey School of Engineering at Washington University Open Scholarship. It has been accepted for inclusion in Engineering and Applied Science Theses & Dissertations by an authorized administrator of Washington University Open Scholarship. For more information, please contact digital@wumail.wustl.edu.
WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science
Department of Electrical and Systems Engineering

Dissertation Examination Committee:
ShiNung Ching, Chair
Shantanu Chakrabartty
Jr-Shin Li
Carlos Ponce
Shen Zeng

Dynamics and Control in Spiking Neural Networks
by
Fuqiang Huang

A dissertation presented to
The Graduate School
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

December 2019
Saint Louis, Missouri
Contents

List of Figures ........................................................................................................ v
Acknowledgments ................................................................................................... viii
Abstract ................................................................................................................ xi

1 Introduction ........................................................................................................ 1
  1.1 Background .................................................................................................... 1
  1.2 Linear Tracking Control via SNNs (Chapter 2) ........................................... 3
  1.3 Endpoint Control (Chapter 3) ................................................................. 6
  1.4 Nonlinear Control (Chapter 4) ............................................................... 7
  1.5 Nonlinear Planning via Iterative Learning Control (Chapter 5) .......... 8
  1.6 Summary of Contributions ..................................................................... 9

2 Spiking Networks as Efficient Distributed Tracking Controllers .............. 11
  2.1 Introduction ............................................................................................... 11
  2.2 Preliminaries and Formulation ................................................................ 12
    2.2.1 Spike Train and Firing Rate ............................................................ 13
    2.2.2 Linear Control Assumption ............................................................. 14
    2.2.3 Decoding ......................................................................................... 14
    2.2.4 Optimization Strategy .................................................................. 16
  2.3 Derivation of the Spiking Network ........................................................... 17
    2.3.1 The Emergent Network Dynamics Are Linear With Fixed Spiking Threshold 17
    2.3.2 Output Feedback .......................................................................... 20
    2.3.3 Adding a Self-decay Leads To Exact Integrate-and-Fire Dynamics .... 21
  2.4 Analysis and Performance Characterization ............................................ 22
    2.4.1 Fast versus Slow Network Interactions and Ensuing Control Performance 22
    2.4.2 Network Size and Robustness to Neuronal Failure ....................... 24
    2.4.3 Homogeneous v.s. Heterogeneous Decoding Weights ................. 25
  2.5 Discussion and Conclusions ..................................................................... 26
    2.5.1 Limitations of the Greedy Spiking Assumption: Feasibility .......... 26
    2.5.2 The Role of Noise ........................................................................ 27
    2.5.3 The Effect of the Leak Term ........................................................... 27
    2.5.4 Units and Scaling of Parameters ................................................. 28
    2.5.5 Synaptic Dynamics Allow for Spike Transmission Only ............... 28
3 Integrated Planning, Control and Estimation within Spiking Networks

3.1 Introduction
3.2 A Distributed Spiking Network
  3.2.1 Preliminaries
  3.2.2 Spiking rule of the network
  3.2.3 Characterization
3.3 Planning and Control for End Point Problem – Minimum Energy Control
  3.3.1 Dynamics of networks
  3.3.2 Simulation and Characterization
3.4 Soft Final State Control with LQR
  3.4.1 Introduction to PI Control in LQR
  3.4.2 Trajectory Generation
  3.4.3 Final State Control
  3.4.4 Simulation and Characterization
3.5 Linear Estimator and Kalman Filter
  3.5.1 Introduction to Luenberger Observer and Kalman Filter
  3.5.2 Simulation and Characterization
3.6 Conclusion

4 Distributing a Nonlinear Control Policy across a Network of Threshold-based Agents

4.1 Introduction
4.2 Preliminaries
  4.2.1 Control-Affine Nonlinear Systems
  4.2.2 Optimal Control Strategy
4.3 Dynamics and Properties of the Network
  4.3.1 Dynamics of the Optimal SNN
4.4 Examples
  4.4.1 Tracking Performance
  4.4.2 Robustness to Neuron Failure
4.5 Conclusion

5 Nonlinear Planning via Iterative Spiking Networks

5.1 Introduction
5.2 PD-Type ILC
5.3 Distributed Networks for ILC
  5.3.1 Distributed Recurrent SNN for Control Signal Increment $w_j(k)$
  5.3.2 Distributed Iterative SNN for ILC Control Signal $u_{j+1}(k)$
5.4 Simulation of Learning and Control for Model Uncertainty
  5.4.1 Control Performance
  5.4.2 Convergence
5.5 Simulation of Planning and Control for Nonlinear Systems ........................................ 106
5.6 Conclusion .................................................................................................................. 108

6 Conclusion ...................................................................................................................... 110
6.1 Outcomes ..................................................................................................................... 110
6.2 Outlook and Future Work ............................................................................................. 111

Appendix A Derivation of the Spiking Rule (2.6)-(2.12) .................................................. 112
Appendix B Proof of the Spiking Rule (3.7)-(3.9) of the Distributed Network ............... 117
Appendix C Proof of the dynamics (4.5) .......................................................................... 120
Appendix D Extension of Distributed SNN in Proposition 2 ........................................... 123
Appendix E Iterative SNN for ILC ................................................................................... 128
References ......................................................................................................................... 131
Vita ...................................................................................................................................... 140
List of Figures

2.1 Problem schema. We aim to study the circumstances in which a spiking neural network (SNN) can control a linear system through its decoded activity. To do so, we optimize the timing of the spiking outputs $o(t)$ in order to minimize a tracking objective, thereby, in effect, optimizing the dynamics of the SNN itself. .......................... 13

2.2 Performance of the proposed spiking network (a) without and (b) with rate decoding. The first panels of (a) and (b) demonstrate the tracking performance; the spiking events, i.e. the spiking activity across the population are shown in the second panels while the membrane voltage for the first neuron and its threshold are plotted in the third panels. ......................................................... 30

2.3 Step response of the proposed spiking network for different numbers of neurons. .......................... 31

2.4 Robustness of the proposed network to neuronal failure. Spikes in the red shaded regions are manually suppressed. The internal dynamics of the network result in other neurons producing compensatory spikes, mitigating the effect on tracking. .. 32

2.5 In a multiple input setting, the decoding weights must be heterogenous. Here, performance is shown for (a) homogeneous vs. (b) heterogeneous decoding. .......................... 33

2.6 The magnitude of the extrinsic noise $\sigma_V$ is a key parameter. In this example, too little or too much noise leads to degradation in performance. .......................... 34

2.7 Excessive leak $\lambda_V$ deteriorates performance. .......................... 35

3.1 Schematic of Distributed Spiking Network. The network encodes the time varying signal $\hat{u}(t)$ into neural activity such as spikes and firing rates, while the decoder reproduces a signal $u(t)$ from the neural activity. .......................... 38

3.2 Schematic of Predictive Coding. If we embed the linear dynamic system $\dot{\hat{u}}(t) = A\hat{u}(t) + c(t)$ into the distributed spiking network, we can realize the predictive coding proposal in [24]. In this way, our spiking network can be seen as a general form of predictive coding. .......................... 42

3.3 Control Performance Characterized by the Number of Neurons. More neurons lead to better tracking performance and faster step response. .......................... 44

3.4 Control Performance Characterized by the Rate Decoding Weight. Larger rate decoding weight implies better tracking performance and faster step response. .......................... 45

3.5 Schematic of Minimum Energy Control Networks. The three-layer network, based on minimum energy control policy, aims at steering a linear time-invariant system to a specific termination. .......................... 46

3.6 Schematic of Layer [I]. It reproduces minimum control signal to steer a linear system to a target point. .......................... 47
3.7 Schematic of Layer II. This forward layer generates a template trajectory $x_d$ by feeding a minimum control signal $u_d(t)$ through a linear dynamic system.

3.8 Control Performance of Minimum Energy Control Strategy. The first chart shows the trajectory tracking performance, the second chart demonstrates the template control signal in black solid line and the actual control signal in black dotted line, and the spiking activity of neurons is plotted in the last chart.

3.9 Robustness to Neuron Failure of Minimum Energy Control Strategy. Half of neurons, 26–50, 76–100, and 126–150, are disabled during some time intervals, 200–400ms and 600–800ms.

3.10 Tuning Curve with respect to Target Positions for Minimum Energy Control Strategy. The chart in the center demonstrates the center-out motion where the red star $\hat{x}(t)$ represents the target point, the black line $x_d(t)$ is the template trajectory, while the blue line $x(t)$ is the actual state. The charts in the first row shows the tuning curve of Layer I, the second row comes from Layer II, while Layer III is shown in the last row.

3.11 Tuning Curve with respect to Target Velocities for Minimum Energy Control Strategy.

3.12 Tuning Analysis for Minimum Energy Control Strategy. Each point represents a neuron whose coordinates are the instantaneous weight and whose color represents the number of spikes it fires.


3.14 Robustness to Neuron Failure of LQR Strategy.

3.15 Tuning Curve of LQR Strategy.

3.16 Tuning Analysis of LQR Strategy.

3.17 Control Performance of LQR with Kalman Filter.

3.18 Robustness to Neuron Failure of LQR with Kalman Filter.

3.19 Tuning Curve of LQR with Kalman Filter.

3.20 Tuning Analysis of LQR with Kalman Filter.

4.1 Schematic of the Nonlinear Control Architecture.

4.2 Performance of Instantaneous Decoding.

4.3 Performance of Rate-Instantaneous Decoding $|\Gamma_{ij}| = 200$.

4.4 Robustness of Instantaneous Decoding.

4.5 Robustness of Rate-Instantaneous Decoding $|\Gamma_{ij}| = 200$.

4.6 Robustness Analysis under Different Rate Decoding Weight.

5.1 Control Performance for a Linear System with Exact Model. The left chart represents the control performance without learning, the middle one shows the performance learned by nominal / classical ILC controller, while the right one is learned by the iterative SNN.
5.2 Convergence for a Linear System with Exact Model. Fig (a) demonstrates the tracking performance, the tracking error, the control signal, and spiking distribution of the feedback SNN, while Fig (b) shows the ILC control signal for each iteration, and spiking distribution of the iterative SNN.

5.3 Learning Result of Iterative SNN

5.4 Planning and Control for Nonlinear Systems
Acknowledgments

I would like to express my deep gratitude to my advisor, Professor ShiNung Ching, for his guidance, and support during my studies, research, and life at Washington University in St. Louis. His creative ideas and inspiration into the research, his patience and diligence greatly impress me and lead me to the right orientation. It is my great honor to become his advisee.

I would also like to thank the members of my dissertation defense committee, Dr. Jr-Shin Li, Dr. Shen Zeng, Dr. Shantanu Chakrabartty, and Dr. Carlos Ponce, for their precious time and valuable advice.

I also wish to thank my instructors and mentors from Huazhong University of Science and Technology, Tsinghua University, and Washington University in St. Louis for helping me to establish a foundation on which to build my research.

In addition, I would like to thank my colleagues and collaborators James Riehl, Sina Khanmohamaddi, Peng Yi, Mehdi Kafashan, Anirban Nandi, Sensen Liu, Delsin Menolascino, Matthew Singh, Elham Ghazizadeh, Sruti Mallik, and Maren Loe, for many fruitful discussions.

Most of the credit and my gratitude is due to my lovely family: my parents Jilu Huang and Danger Zhu, my wife Huiling Li, and my kids Elife and Emil, who always provide me with their deep love and strong support. Their encouragement made this work possible.

Fuqiang Huang
Dedicated to my parents.
In the brain, neurons (brain cells) produce electrical impulses, or spikes, that are thought to be the substrate of information processing and computation. Through enigmatic processes, these spikes are ultimately decoded into perceptions and actions. The nature of this encoding and decoding is one of the most pervasive questions in theoretical neuroscience. In other words, what are the specific functions enacted by neural circuits, through their biophysics and dynamics? This thesis examines the dynamics of neural networks from the perspective of control theory and engineering. The pivotal concept is that of the normative synthesis of neural circuits, wherein neural dynamics are built from fundamental mathematical objectives. Emergent properties of the synthesized circuits thus constitute a hypothesis regarding how actual networks in the brain might be achieving the functions in question. Several sub-problems are considered. First, we propose an optimization problem to find the dynamics of neural networks that can generate spike trains over a period of time to drive a linear dynamic system along a prescribed trajectory, a canonical problem in control engineering. It turns out that this problem can be solved by a recurrent spiking network with integrate-and-fire dynamics. The network amounts to an efficient, event-based controller in which
each neuron (node) contributes spikes only when doing so is advantageous to the overall cost. We then turn our attention to another classical control problem: state transfer. We introduce a distributed spiking network with multiple layers that enacts a minimum energy control policy for this problem. We then generalize this approach to enact a classical linear quadratic regulator (LQR) control policy with Kalman filter, thus building networks that achieve another highly prevalent control-theoretic construct. While the above results are restricted to linear systems, we also show how an offline nonlinear control policy can be distributed onto a spiking neural network. Finally, we study the idea of iterative learning control, an adaptive control method, and show how this can be integrated into a spiking network through the introduction of a slow time-scale. It turns out that for each of the above synthesis problems, the networks display emergent properties that are compatible with known features of motor cortical networks. Hence, our results show that classical control architectures, including, planning, feedback control, estimation, and learning, could indeed be enacted by a single recurrent spiking network with biophysically plausible dynamics. Further, from an engineering perspective, these network realizations provide a way to achieve distributed control over networks, which may confer robustness properties above and beyond traditional centralized control designs.
Chapter 1

Introduction

1.1 Background

Overview

A long-standing challenge in theoretical neuroscience involves studying how brain networks underlie complex functions, culminating in our ability to see, hear and think [2, 38, 37]. A specific sub-problem in this domain is that of motor control [5], which is related to how brain networks confer the ability to execute complicated tasks such as manipulating our limbs, executing fine motor tracking, walking, etc. [3, 20, 102]. Networks in the brain are able to achieve an unparalleled level of performance on these tasks with high energy efficiency and robustness that far exceeds what we have been able to engineer [71, 103]. Hence, using brain-like networks, i.e, neural networks, for the purpose of control and estimation has been a significant area of research interest for decades [44, 25, 81, 108, 31, 35, 111]. Exploring this idea has generated novel and powerful tools, such as artificial neural networks (ANNs), which have been proposed and successfully implemented to solve a variety of control tasks such as system identification and feedforward control problems [53, 76, 69, 70].
On the other hand, formalisms from control theory have been used to interpret the role of actual brain circuits, under the premise that these circuits could, in fact, be carrying out certain operations such as estimation, optimization, forward modelings, and optimal control [39, 43, 112, 89, 105, 48, 36, 101]. In this sense, ideas from control engineering can feedback to neuroscience by suggesting hypotheses regarding the function of neural circuits. This framework is sometimes referred to as a normative approach to theoretical neuroscience [15, 16, 17, 18, 100], since mathematical schema are directly used to yield circuit hypotheses.

The synergistic back-and-forth between engineering and neuroscience forms the basis for this thesis, with engineering helping to generate new ideas about neural circuits, which in turn might constitute new means of enacting the engineering itself [95, 95, 22].

Dynamics in (artificial) neural networks and circuits

Nominally, artificial neural networks (ANN) are premised on the architecture and dynamics of actual brain networks [49, 114, 56]. In particular, the idea that brain networks consist of many interconnected computational units – neurons – whose interconnection weights adapt over time-based activity-dependent plasticity rules [109]. In ANNs, the goal of these plasticity rules is to optimize a prescribed objective function. This process of weight adaption is colloquially referred to as learning. After learning is complete, the ANN is able to achieve the function associated with the objective in question [53, 56].

In the present context, we are interested in functional objectives related to control, i.e., ANNs that are able to interact with and manipulate dynamical systems. Although ANNs have been used in this context, especially for feedforward control and system identification, they face limitations because they lack the formal guarantees afforded by modern control engineering techniques [53, 76, 69, 70].
From the normative perspective, ANNs face the fundamental issue of abstraction. Actual neurons possessing far more complex dynamics and especially produce activity primarily in the form of electrical impulses, or spikes. Many ANNs, in contrast, produce real-values graded activity. Our goal is to work strictly in the domain of spiking neural networks (SNNs) [47, 88, 85, 33]. Studying how SNNs encode and decode information and external stimuli to and from spiking activity is one of the most prevalent and fundamental questions in theoretical neuroscience [3, 20]. In other words, why and when are spikes produced by neurons? Are SNNs modular or hierarchical in nature, or is such decomposition not possible? Does the use of spikes confer advantages in robustness and/or efficiency of the network at hand? How might an artificial SNN (aSNN) be ‘designed’ to enact a specific optimal control solution?

This dissertation consists of four thrusts that address the above question, each briefly introduced below.

1.2 Linear Tracking Control via SNNs (Chapter 2)

Motivated by these questions, our first goal in this thesis is to synthesize a spiking neural network whose activity can be decoded into a useful control signal for a linear dynamical system. To do so, we leverage the idea from computational neuroscience of predictive coding [13, 90], which posits that neurons act in a way that best reduces the future uncertainty of extrinsic inputs and latent variables. While often carried out in a statistical framework, recent work [23, 24] has used predictive coding to build deterministic spiking networks that approximate dynamical systems. Here, we use the ideas in [23, 24] to elaborate on our previous work [52] in considering not system approximation, but rather system control. The approach is based on the following theoretical suppositions:
Neurons fire spikes at certain times $t_i, i \in \mathbb{Z}^+$ resulting in a spike train of the form:

$$o(t) = \sum_i \delta(t-t_i),$$

where $\delta(\cdot)$ is a Dirac delta function.

Spike trains of the neural network are converted into a real-valued control signal via

$$u(t) = h(o(\cdot))$$

where $h(\cdot)$ is a decoder that may in general depend on the current and past state of the spike train.

The goal of the network is to emit spikes with timing $t_i$ such that $u(t)$ produces a desired control objective in a known plant model.

In other words, we ask: if spikes are converted to a control signal via $h(\cdot)$, then when and how should those spikes be produced? Should neurons fire many spikes, densely in time; or should only a few neurons produce a small number of spikes, in a more sparse, efficient manner? As intuition might suggest, the answer to these questions will depend on the specific assumptions regarding the form of $h(\cdot)$, as well as the system being controlled.

As will be shown in the first chapter of the dissertation, when we assume that the decoder and plant both have linear dynamics, the answers to the above questions can be analytically obtained through an optimization problem whose solution is realized by endowing each neuron with drift-diffusion dynamics and a fixed spiking threshold that is analogous to the classical integrate-and-fire model neuron [1]. That is, rather than starting a priori with a prespecified neuronal network and learning its connections, we start with a control objective and synthesize the network and its dynamics in
one step. Indeed, there are several other comparable ideas in theoretical neuroscience and machine learning within which our approach should be placed in context. In particular, the concepts of reservoir computing [70] and the neural engineering framework [14] both highlight the use of recurrent networks for a variety of objectives including control [104]. Such strategies work by training a set of decoding weights (analogous to learning the function $h(\cdot)$) that read activity from a fixed network with usually random architecture. The relationship between $h(\cdot)$ and the ‘best’ such architecture/dynamics is not considered and this is the main focus of our work. Further, almost always the dynamics of such networks are formulated in discrete time, versus the overt continuous-time formulation we consider herein.

From a purely control-theoretic standpoint, our results can be interpreted in the context of a type of control problem over networks. Unique in our problem is that we explicitly deduce the particular connection motifs and dynamics of each node for a particular control objective goal. That is, whereas much effort has been directed at understanding how to elicit prescribed dynamics and patterns of networks (e.g., [83]), our work takes the direction of examining control by networks, finding the network dynamics that are most useful for generic control tasks. In this regard, our network can be viewed as an efficient event-based controller [98] where each neuron (node) remains silent unless producing a spike leads to a reduction in feedback error (or, more generally, an error-based cost function). The neurons in the network constitute a set of ‘event detectors’, whose spikes indicate a deviation in the desired cost. The spikes are decoded into control signals that compensate for these deviations.
1.3 Endpoint Control (Chapter 3)

A key facet of control theory is the ability of analytically synthesize signals that produce desired transfers of state end-to-end [12, 32, 87, 94]. Another way to describe this is that given an initial condition and the desired end-state, theoretical formalisms exist to produce a trajectory that takes the controlled system from the former to the latter. At a high level, such formalisms solve the ‘planning’ problem (mapping a route) concurrently with the control problem (executing the route) [19, 30, 63, 115].

Thus, our second line of investigation involves enacting endpoint control within an SNN. In Chapter 3, we propose a multi-layer spiking network strategy to control a linear system from an initial state to a fixed terminal state within a specific time horizon. Specifically, the first two-layer networks are used to generate a trajectory based on classical minimum energy control policies. Then, the final layer of the network enacts the tracking control studied earlier.

Afterwards, we consider a relaxed version of this problem wherein the goal of the controller is to steer the system at hand to an endpoint asymptotically. Such relaxation allows for treatment using a large number of classical control strategies, such as traditional PID control, LQR with integral action, among others [11, 10, 7, 28, 97, 67]. We specifically study the LQR strategy, since it allows us to take a rigorous optimization-based framework to the control synthesis. We are able to embed the entire synthesis procedure directly into an SNN, using similar frameworks to those used above. Moreover, we are also able to embed a Kalman filter to perform robust state estimation, again using an SNN with integrate-and-fire dynamics [91, 107, 105, 74]. The last step allows us to use SNNs to solve asymptotic setpoint tracking with only output feedback with an increased tolerance of noise and uncertainty.
There are several aspects of these results that are of interest. First, they provide a means to distribute classical control policies across an SNN, which confers robustness properties and are generally and interesting instantiation of distributed control theory. From a normative theory standpoint, the results are interesting because the neurons within the network exhibit several key features that are thought to be canonical in motor regions of the brain; for example, the minimum energy control SNNs exhibit ‘cosine’ type position and velocity tuning, distributed throughout the network. Moreover, for the LQR architecture, the control and estimation, while distinct entities as engineering constructs, are actually inextricable at the level of the network so there are not spatially distinct modules associated with these different functions.

1.4 Nonlinear Control (Chapter 4)

The above two research thrusts involve SNNs for control of linear systems. The analytical advantages of linear systems afford analytical advantages for the normative synthesis procedure. Of course, it is of immediate interest as to whether these ideas can be generalized to the case of nonlinear control [55, 77, 99]. Perhaps unsurprisingly, the problem in this case brings forth several nontrivial complexities that are not as easily handled as in the linear case. Nonetheless, a partial solution was achieved.

Specifically, in Chapter 4, we present an approach to construct a spiking network for the purpose of controlling a nonlinear system involving linearization about a pre-planned trajectory. Here, we assume that a nonlinear controller is solved ‘offline’ and that linearization is possible about the associated planned trajectory [62, 72]. In this scenario, we are able to leverage the predictive coding scheme to produce a network that distributed the nonlinear control policy across a population of
neurons. Here, the connections/weights of the network evolve as a function of time, in contrast to the static weights that manifest in the linear case.

### 1.5 Nonlinear Planning via Iterative Learning Control (Chapter 5)

The control schemes developed in the above three research thrusts are all fixed, i.e., non-adapting. However, we know that actual neural circuits constantly adapt over multiple time-scales, ostensibly optimizing themselves over time to different functional needs. Are there control-theoretic ideas that might provide leverage to understand such adaption? Adaptive control is itself a broad area of research, wherein controllers adjust their architecture and dynamics according to errors between expected and actual performance [9, 54, 64]. This is especially apt in situations where model uncertainty prevails. Indeed, in the above contributions, the networks that are synthesized have within them an embedded model of the system at hand, in accordance with the internal model principle of control theory [45, 61, 92, 93]. Adaptive control, in principle, allows for this internal model to be accrued over time, based on observation.

Here, we specifically consider one particular form of adaptive control, known in the literature as iterative learning control (ILC) [4, 26], wherein a controller adapts based on performance errors incurred over repetitive iterations of the same task operation. In contrast to the gradient-based learning that prevails in modern machine learning, ILC leverages the repetitive nature of the task to more rapidly converge to a control solution, usually in just a few iterations [8, 26, 69]. This performance tends to be quite robust, even in the face of large model uncertainty.
Moreover, we observe that perfect trajectory tracking – almost zero dynamic error – can be achieved by learning over some iterations, even for a nonlinear system. Based on this observation, we can utilize ILC to plan a template control signal given a reference trajectory. This drives us to distribute ILC into a network so that such a planning can be executed via a network.

In Chapter 5, we propose a novel approach to construct an iterative spiking network to distribute the ILC learning algorithm so that we can achieve almost zero tracking error with SNNs and further plan template control signals for nonlinear systems. We assume that during each iteration, the SNN encodes the current error information into the spiking train and firing rate and accumulate them to the iterative firing rate, with slow time scale, which can be used to be decoded to the feedforward ILC control signal to improve the tracking performance iteratively.

1.6 Summary of Contributions

In summary, the primary contributions of the dissertation are:

- A method to synthesize networks to control linear time-invariant systems. These networks amount to robust distributed controllers, wherein parts of the network can compensate for others. The synthesized network exhibits integrate-and-fire dynamics with activity that evolves on fast- and slow- time-scales.

- A method to integrate trajectory generation and control within a spiking neural network. The network displays emergent properties that are compatible with known phenomena in motor cortex, suggesting new functional hypotheses regarding these prior observations.
• A method to distribute nonlinear control policies computed offline, into a distributed spiking neural network with time-varying architecture.

• A method to iteratively adapt the architecture of a SNN to plan a trajectory for an arbitrary nonlinear system, motivated by results in iterative learning control (ILC).

In the last of the dissertation, Chapter 6, we make a few concluding remarks and comment of the outlook for this research moving forward.
Chapter 2

Spiking Networks as Efficient Distributed Tracking Controllers

2.1 Introduction

In the brain, networks of neurons produce activity that is decoded into perceptions and actions. How the dynamics of neural networks support this decoding is a major scientific question. That is, while we understand the basic mechanisms by which neurons produce activity in the form of spikes, whether these dynamics reflect an overlying functional objective is not understood [3, 20, 102].

In this chapter, we examine the dynamics of neural networks and decoder from a first-principle control-theoretic viewpoint. For a certain decoder candidate, we construct a spiking network for the purpose of tracking control and then examine tracking performance as well as spike distribution such that a better candidate can be screened. Our approach can be realized in the following steps:
1. We postulate an objective wherein neuronal spiking activity is decoded into a control signal that subsequently drives a linear system. The objective function aims at minimizing tracking error and penalizing on firing rate which accounts for the number of spikes, that is, the objective balances effectiveness and efficiency.

2. Using a recently proposed principle from theoretical neuroscience [24], we optimize the production of spikes so that the linear system in question achieves reference tracking. Specifically, the spiking principle, defining each neuron fires spikes in order to minimize the objective, is a local optimum policy.

3. The solution to the optimization problem is a threshold-based spiking rule which leads to a recurrent network architecture wherein each neuron possesses integrative dynamics. The network amounts to an efficient, distributed event-based controller where each neuron (node) produces a spike if doing so improves tracking performance.

4. By examining the dynamics of networks, tracking performance, and spiking distribution, we can tell which decoder is a better choice.

2.2 Preliminaries and Formulation

The formulation of the problem is similar to that in our prior work [52], the major details of which are included here.
Figure 2.1: Problem schema. We aim to study the circumstances in which a spiking neural network (SNN) can control a linear system through its decoded activity. To do so, we optimize the timing of the spiking outputs \( o(t) \) in order to minimize a tracking objective, thereby, in effect, optimizing the dynamics of the SNN itself.

\subsection*{2.2.1 Spike Train and Firing Rate}

We consider spiking neural networks (SNNs) wherein each neuron produces activity in the form of instantaneous spikes, modeled as Dirac delta functions \cite{38}. Thus, the \( k^{th} \) neuron in the network emits a spike train \( o_k(t) \), where

\[
o_k(t) = \sum_{i \in \{1, 2, \ldots\}} \delta(t - t_k^i),
\]

where \( t_k^i \) denotes time of the \( i^{th} \) spike from the \( k^{th} \) neuron. The collection of spike trains over the network of \( N \) neurons is denoted \( o(t) \in \mathbb{R}^N \), where \( o(t) = (o_1(t), \cdots, o_N(t)) \). It is very important to note that we do not yet assume any dynamics associated with the generation of spikes, only that they occur as instantaneous events.
As is conventional in neuroscience [38], we define a firing rate variable $r(t)$ (spikes per unit time) by low-pass filtering the spike trains via

$$\dot{r}(t) = -\lambda_d r(t) + \lambda_d o(t), \quad (2.1)$$

where $\lambda_d$ determines the (receding horizon) kernel over which rates are computed.

### 2.2.2 Linear Control Assumption

We postulate a putative function for the SNN wherein spikes are decoded for the purpose of controlling a linear time-invariant system of the standard form

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (2.2)$$

with system states $x(t) \in \mathbb{R}^n$ and external input $u(t) \in \mathbb{R}^m$, while $A \in \mathbb{R}^{n \times n}$ is the state matrix and $B \in \mathbb{R}^{n \times m}$ is the input matrix.

We proceed to study the control of such a system, under the tacit assumption that the pair $(A, B)$ is controllable, which means that an external input $u(t)$ is able to steer the internal state between any two points within a finite time interval [59].

### 2.2.3 Decoding

As described in the introduction, the control itself hinges on a dynamical transformation of the spike trains $o(t)$ to the input signal $u(t)$. This transformation is termed the decoder and is elaborated
on below. Endowed with a decoder, we consider the schema illustrated in Figure 2.1, wherein the SNN takes the reference command \( \hat{x}(t) \) as input and produces spikes so that \( x(t) \) follows \( \hat{x}(t) \).

We assume that the spiking activity of the SNN is decoded into a signal \( u(t) \in \mathbb{R}^m \) via a rate-instantaneous type decoder

\[
u(t) = e^{-\lambda_d t} u(t_0) + \Gamma \int_0^t e^{-\lambda_d (t-\tau)} o(\tau) d\tau + \Omega o(t) \\
= \frac{1}{\lambda_d} \Gamma r(t) + \Omega o(t), \tag{2.3}
\]

with rate decoding weight \( \Gamma \in \mathbb{R}^{m \times N} \) and instantaneous decoding weight \( \Omega \in \mathbb{R}^{m \times N} \), where

\[
\Gamma = [\Gamma_1 \Gamma_2 \cdots \Gamma_N], \\
\Omega = [\Omega_1 \Omega_2 \cdots \Omega_N],
\]

in which \( \Gamma_k, \Omega_k \in \mathbb{R}^m \) are decoding vectors associated with each neuron. These weights govern how the spiking activity of each neuron propagate to the inputs of the system to be controlled.

In this work, for a given neuron \( k \), the \( j^{th} \) elements of vectors \( \Gamma_k \) and \( \Omega_k \) will have the same sign, i.e.,

\[
\text{sgn}(\Gamma_{jk}) = \text{sgn}(\Omega_{jk}), \quad \text{where } j = 1, 2, \ldots, m,
\]

so that \( j^{th} \) neuron acts on the decoded signal with the same valence on both instantaneous and rate-based time-scales.

The formulation of our decoder (2.3) and application to a control problem (2.2) deviates in a subtle but important way from the original work of [23, 24]. Specifically, our decoder overtly contains two time-scales and the resultant signal is not simply read-out but used toward a dynamical effector.
As we will see, this shift in formulation allows for a feedback error signal to be used directly by the derived spike-generating network. Indeed, the dynamics in (2.3) encapsulates the two dominant theories regarding how spiking activity is decoded in the brain. Specifically, the $u(t)$ has terms that capture: (i) rate decoding, wherein the decoded signal depends proportionally on the firing rate; and (ii) instantaneous decoding, wherein the decoded signal depends proportionally on the exact timing of spikes via direct dependence on $o(t)$. As we will show herein, it turns out that both decoding principles confer distinct advantages in terms of $u(t)$ being usable as a control signal.

### 2.2.4 Optimization Strategy

We proceed by formulating a control objective that will reward small errors while penalizing excessive neural activation:

$$J(t) = \int_{t_0}^{t} \left( \|x_d(\tau) - x(\tau)\|^2_2 + V\|r(\tau)\|_1 + \mu \|r(\tau)\|^2_2 \right) d\tau. \hspace{1cm} (2.4)$$

Two regularizing terms are present in this function. The penalty on the $\ell_1$ norm of the firing rate, $\|r(\tau)\|_1 = \sum_i |r_i(\tau)| = \sum_i r_i(\tau)$ (for positive firing rates), is intended to prohibit the network from using too many spikes to execute the task, while the $\ell_2$ norm, $\|r(\tau)\|^2_2 = r^T(\tau)r(\tau)$, is introduced to distribute spiking throughout the population.

Finally, we introduce the schema based on which we will optimize the dynamics by which neurons produce spikes. Specifically, as noted above, we adopt the ‘greedy spiking premise’ introduced in [24] so that the objective of neurons is to minimize the cost function $J(t)$. Thus, we enforce the policy that the neuron $k$ fires a spike at time $t^k_\tau$ only if doing so decreases the value of $J(t)$, i.e.,
with a slight abuse of notation,

\[ J((t_k^i + \epsilon)|o_k(t_k^i) \text{ spikes}) < J((t_k^i + \epsilon)|o_k(t_k^i) = 0), \]  

(2.5)

where \(0 < \epsilon \ll \lambda_d\). It is important to emphasize that the optimization will take place over the spike times \(\{t_k^i\}_{k \in \{1, \ldots, N\}, i \in \mathcal{N}}\), and not over the decoding parameters \(\Gamma\) or \(\Omega\), which are assumed known \textit{a priori}.

The policy in (2.5) is the central premise from which we will deduce our ensuing spiking network.

### 2.3 Derivation of the Spiking Network

#### 2.3.1 The Emergent Network Dynamics Are Linear With Fixed Spiking Threshold

The above optimization problem results in a threshold spiking rule, which is our first and key result. Namely, we introduce a latent state variable \(v_k(t)\), termed ‘voltage,’ for each neuron. If \(v_k(t)\) for the \(k\)th neuron exceeds a fixed threshold \(\bar{v}_k\) at time \(t\), then this neuron fires a spike to decrease the value of the objective function (3.5). More formally, we state:

**Proposition 1.** The spike times that greedily minimize (3.5) according to assumption (2.5) are realized through the threshold criteria

\[ v_k(t) > \bar{v}_k, \]  

(2.6)
subject to the neuronal voltage relation

\[ v_k(t) \equiv \Omega_k^T B^T (x_d(t) - x(t)) - \mu \lambda_d \bar{e}_k^T r(t), \tag{2.7} \]
\[ \bar{v}_k \equiv \Omega_k^T B^T \Omega_k + \nu \lambda_d + \mu \lambda_d^2 \]
\[ \frac{2}{2}, \tag{2.8} \]

where \( \bar{e}_k \in \mathbb{R}^N \) is the standard entry vector with zero elements except for the \( k^{th} \) row.

Proof. See Appendix A.

The above equations, in fact, specify a network of interconnected dynamical spiking neurons. To see this, we denote the membrane voltage vector for all of neurons as

\[ V(t) = (v_1(t), \cdots, v_N(t)), \tag{2.9} \]

then we can deduce dynamics for the network of neurons as:

\[ \dot{V}(t) = \Omega^T B^T \Lambda e(t) + \Omega^T B^T c(t) + W^r r(t) + W^f o(t) \tag{2.10} \]

where \( e(t) = x_d(t) - x(t) \) is the feedback error, \( c(t) = \dot{x}_d(t) - Ax_d(t) \) is the feedforward signal and the weight matrix of connections among neurons are

\[ W^r = -\frac{1}{\lambda_d} \Omega^T B^T B \Gamma + \mu \lambda_d^2 I, \tag{2.11} \]
\[ W^f = -\Omega^T B^T B \Omega - \mu \lambda_d^2 I, \tag{2.12} \]

where \( I \) is the identity matrix. Several observations regarding the emergent spiking dynamics are notable. The emergent connectivity dynamics are separated into two parts: a slow, rate-driven
component mediated by $W^s$ (slow connectivity), and an instantaneous, spike-driven component mediated by $W^f$ (fast connectivity). Note that these terms capture, in essence, the network connectivity (i.e., how a neuron is affected by spikes of other neurons). Since the diagonal elements are nonzero, these terms also capture the internal dynamics of each neuron.

Both of the regularization parameters $\nu$ and $\mu$ affect the threshold of $\bar{v}_k$, making it harder for the $k^{th}$ neuron to fire. Further, $\mu$ appears in $W^f$ and, specifically, causes the value of $v_k(t)$ to instantaneously decrease following a firing event (i.e., in essence, a ‘reset’ in the latent variable $v_k(t)$). This prevents dense, repetitive spiking.

To enable tracking, the spiking network must be sensitive to both the desired state trajectory $\hat{x}(t)$ and its derivative $\dot{x}_d(t)$ (i.e., to create $c(t)$). This dependence is consistent with the notion of an instantaneous spike encoder, which has been shown to be optimal from the perspective of signal reconstruction [57].

Finally, there is a duality in the parameters of the system to be controlled, and their manifestation in the SNN. Specifically, the state matrix $A$ appears as, effectively, an input gain on the feedback error $e(t)$ and feedforward signal $c(t)$ on the SNN, whereas the input matrix $B$, modifies the internal connectivity among neurons. However, the number of neurons $N$ depends solely on the dimensions of the decoding weights $\Gamma$ and $\Omega$, and not on the dimension of the system to be controlled.

The use of the feedback error $e(t)$ represents a deviation from our prior results in [52], as well as the original predictive coding network of [23] [24]. This is because in the control formulation herein, the signal to be tracked $\hat{x}$ and the ‘output’ signal $x$ both evolve under (2.2), so that the incremental optimization in (2.5) involves the same set of forwarding dynamics (see also Appendix A). In a non-control formulation, the signal $x$ is simply the decoded activity, which has no dynamical relation to the tracking reference. Consequently, by allowing explicit incorporation of an error
signal, we obviate the need to perform pseudoinverse of the forward dynamics of the system, a potentially error-inducing step.

### 2.3.2 Output Feedback

The above result is readily extended in the case of output feedback

\[ y = Cx, \]

where \( y(t) \in \mathbb{R}^l \) is the output signal and \( C \in \mathbb{R}^{l \times n} \) is the output matrix, based on the revised objective function

\[
J(t) = \int_{t_0}^{t} \| \hat{y}(\tau) - y(\tau) \|^2_2 + \nu \| r(\tau) \|_1 + \mu \| r(\tau) \|^2_2 \, d\tau. \tag{2.13}
\]

The extended spiking rule becomes

\[ v_k(t) > \bar{v}_k, \tag{2.14} \]

with

\[
v_k(t) \equiv \Omega_k^T B^T C^T (\hat{y}(t) - y(t)) - \mu \lambda d \bar{e}_k^T r(t), \tag{2.15}
\]

\[
\bar{v}_k \equiv \frac{\Omega_k^T B^T C^T C B \Omega_k + \nu \lambda d + \mu \lambda d^2}{2}, \tag{2.16}
\]
while the neural dynamics are

\[ \dot{V}(t) = \Omega^T B^T C^T A e(t) + \Omega^T B^T C C(t) + W^s r(t) + W^f o(t), \]  

where

\[ W^s = -\frac{1}{\lambda_d} \Omega^T B^T C B \Gamma + \mu \lambda_d^2 I, \]  

\[ W^f = -\Omega^T B^T C B \Omega - \mu \lambda_d^2 I. \]  

These equations are derived in a similar way as in Appendix A.

### 2.3.3 Adding a Self-decay Leads To Exact Integrate-and-Fire Dynamics

Note that the emergent dynamics (2.10) are similar to those of the classical Integrate and Fire (IF) neuron model[1], a well-known formal biological neuron model. As in [24], to allow the neuronal dynamics to match the classical IF model, one needs to add two additional terms to the dynamics of the latent variable \( V(t) \): (i) a decay/leak action that forces the voltage to decay to zero in the absence of input; and (ii) a noise term that promotes stochastic firing, i.e.,

\[ \dot{V}(t) = -\lambda_V V(t) + \Omega^T B^T A e(t) + \Omega^T B^T c(t) + W^s r(t) + W^f o(t) + \sigma_v w(t) \]  

where, \( \lambda_V \) is the decay constant and \( w(t) \) is a white noise process.

Note that the self-decay and noise term are introduced to match the neuron dynamics with the IF model and do not arise from the direct greedy optimization solution. However, the noise term is very important to promote realistic stochastic firing, which we will discuss later.
2.4 Analysis and Performance Characterization

We proceed to analyze the key dynamical features of the spiking network and characterize necessary conditions for the network to achieve the desired control performance.

2.4.1 Fast versus Slow Network Interactions and Ensuing Control Performance

The network exhibits coupling on two-time scales: fast coupling mediated by $W^f$ and slow coupling mediated by $W^s$. These time-scales are related to the two time-scales of decoding presented in (2.3). We now highlight the different functional roles of these time-scales.

Fast Interactions for Immediate Error Reduction and Feasibility

Our derivation highlights an important insight regarding the putative decoder (2.3) for the purpose of control, namely that the instantaneous term is necessary in order to ensure that command tracking is possible. Indeed, without instantaneous decoding, the spiking condition in (2.6) becomes

$$-\mu \lambda_d e_k^T r(t) > \frac{V \hat{\lambda}_d + \mu \hat{\lambda}_d^2}{2}. \quad (2.21)$$

Two observations can be immediately made: (i) the control error $x(t) - x_d(t)$ no longer appears in this condition, and, (ii) more bluntly, this condition can never be satisfied, since $r(t)$ is always nonnegative and the right-hand side is positive. Thus, under the greedy policy (2.5), a spike can never decrease the cost in this decoding scenario (i.e., with rate decoding only).
This result can be interpreted since the decoder and the system to be controlled both involve the integration of their afferent inputs. Without instantaneous decoding, if a neuron spikes, a delta function will be added to the spike trains \( o(t) \) and the decoded signal \( u(t) \) will exhibit a jump discontinuity. Consequently, \( x(t) \) can produce, at most, a trapezoidal change over the ‘greedy’ horizon \( \varepsilon \).

**Slow Interactions for Efficient Tracking**

Unlike fast interactions, which are pertinent to the feasibility of the control problem, slow interactions can improve performance through more efficient use of spikes, since each spike is able to have a lasting effect on other neurons.

To illustrate this effect, we consider a simple example of tracking in a linear system with \( A = -10 \), \( B = 1 \) and compare the ensuing network performance achieved by a network of 100 neurons for \( |\Gamma_k| = 0 \) (Fig. 2.2(a)) and \( |\Gamma_k| = 200 \) (Fig. 2.2(b)). The parameters for the spiking network are chosen as \( \lambda_V = 0 \text{Hz}, \sigma_v = 50 \mu = 0.03, v = 0.3, \lambda_d = 10 \text{Hz}, \Omega_k = 0.5 \) for \( k = 1 \ldots 50 \), \( \Omega_k = -0.5 \) for \( k = 51 \ldots 100 \), while \( \Gamma_k \) has the same sign as \( \Omega_k \).

By comparing Figure 2.2(a) and 2.2(b), we note substantially sparser and more random spiking outputs are observed when rate decoding is used. In addition to the sparser firing, we also note that the use of rate decoding also leads to generally better tracking performance. This increase in performance can be qualitatively understood since each neuron is able to have a longer-lasting effect on the control signal. In fact, lower firing rates \( r(t) \) are inextricably associated with low errors via (2.7). This rationale for this stems from the fact that the threshold \( \bar{v}_k \) in (2.8) is independent of the rate decoding weight \( \Gamma \). Thus, if the dynamics of the network produce a decrease in \( r(t) \) (as is the
case when $\Gamma$ is utilized), then the resultant error must also decrease, else the membrane potentials $v_k(t)$ incur a contradictory increase above the threshold.

### 2.4.2 Network Size and Robustness to Neuronal Failure

To demonstrate the effect of network size $N$, we fix the weight of rate decoding $|\Gamma_k| = 200$, change the number of neurons and generate corresponding step responses.

As shown in Figure 2.3, with a greater number of neurons, the control performance appears to increase in terms of both the bandwidth and damping associated with a nominal step response. This is perhaps expected since more neurons are able to contribute to the decoded signal. The drawback here is, of course, that greater ‘effort’ is used from the standpoint of the computational burden.

A more important property of the spiking network is its robustness to neuron failure, i.e., its performance when subsets of neurons randomly cease spiking. Indeed, the connectivity within the network serves as a feedback mechanism, wherein neurons compensate for each other by up- or down-regulating their firing.

We show this robustness by disabling some neurons during certain time intervals (300 – 1200 ms and 1800 – 2700 ms). As shown in Figure 2.4(a), when only instantaneous decoding is used and some neurons are disabled, the tracking performance deteriorates because of insufficient neurons for spiking. While in 2.4(b), the rate-instantaneous decoding is implemented, the system is still able to track the desired trajectory well, showing robustness to disabled neurons. This result is not difficult to understand, as mentioned above, with rate decoding, the spiking distribution is sparse, hence only a few neurons are needed to ensure good tracking.
2.4.3 Homogeneous v.s. Heterogeneous Decoding Weights

In the above cases, we considered single input scenarios wherein we used instantaneous-decoding weights $\Omega_k$ that are homogeneous over $k$. However, in a multi-input system, heterogeneous weights are necessary. The reason lies in Eqs. (2.3), (2.6) - (2.8). Specifically, if all $|\Omega_k|$’s are the same, each neuron responds in the same way to the error, so that there is no way to disassociate error in different states. However, if these weights are heterogeneous, neurons will have a sensitivity to an error in different states. Thus, as a whole, the network will act to reduce errors across all controlled states. This argument is conceptually similar to that of the observability of the error with respect to $\Omega$.

To illustrate this notion, we compare step responses of a dual input dual output system,

$$
\begin{bmatrix}
\dot{v}_1 \\
\dot{v}_2
\end{bmatrix} = 
\begin{bmatrix}
-1 & 0 \\
0 & -20
\end{bmatrix}
\begin{bmatrix}
v_1 \\
v_2
\end{bmatrix} + 
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2
\end{bmatrix},
$$

controlled by homogeneous and heterogeneous decoding, respectively. The parameters for the spiking network are chosen as $N = 100$, $\lambda_V = 0$Hz, $\sigma_v = 50$, $\mu = 0.03$, $\nu = 0.3$, $\lambda_d = 10$Hz, $|\Gamma_k| = 200$.

In Figure 2.5(a), we set $\Omega_k = [0.5 \ 0.5]^T$ for $k = 1 \ldots 50$, $\Omega_k = [-0.5 \ -0.5]^T$ for $k = 51 \ldots 100$ and we see that both states cannot stay in the static states while the control signals in the second chart seems exactly the same. In contrast, when $\Omega \sim \mathcal{N}(0, 0.5^2)$ – the normal distribution with 0 mean and 0.5 standard deviation, both states are able to track respective set points (Fig. 2.5(b)).
2.5 Discussion and Conclusions

In this work, we have proposed a dynamical, recurrently connected spiking network whose activity can be decoded for the purposes of controlling a linear system in terms of a tracking objective. The network amounts to an event-based, distributed control strategy. Moreover, the dynamics provide inherent robustness properties, so that if some neurons fail, others will compensate by increasing their activity so that the tracking objective is met. However, the design does rely on knowledge of the system matrices $A$ and $B$, which are embedded in the emergent network dynamics.

2.5.1 Limitations of the Greedy Spiking Assumption: Feasibility

As we saw in Section 2.4.1, the greedy optimization premise underlying our network necessitates the use of instantaneous decoding. This assumption plays out in terms of another, more direct limitation: that the proposed network can only be used to control state variables that are directly actuated. Intuitively, this is because for the greedy assumption to work, a spike must be able to produce a nonzero instantaneous derivative in the error variable. In terms of the derived network, this manifests in the term $B^T(\hat{x} - x)$ in (2.7). Thus, the $k^{th}$ neuron will only produce a spike in response to the error that can propagate through $B^T$ (i.e., variables that are directly actuated).

One way to formalize this notion is to restrict ourselves to the output feedback situation, wherein the objective function is specified in terms of $y$ (2.13). In this case, a necessary condition for feasibility is simply:

$$B^T C^T \neq 0.$$  \hspace{1cm} (2.22)
In fact, the greedy cost reduction policy means that even if the system is fully controllable and observable, we still cannot steer all of the states if $B^T C^T = 0$.

### 2.5.2 The Role of Noise

Another key issue in the proposed network is the role of noise, which we have modeled as additive white noise in (2.20). Such noise is not only needed to promote more realistic stochastic firing but is also needed for the network to attain the desired performance. Indeed, the magnitude of the noise, parameterized by $\sigma_V$ can have a substantial effect on this performance.

Note specifically that without noise, as shown in Figure 2.6(a) all of the neurons with the same kernel will fire spikes together since they will meet the same threshold at the same time according to (2.7) (assuming uniform initial conditions). This results in larger tracking error, non-random spiking distribution and a larger firing rate, compared with Fig. 2.6(b) wherein a small amount of noise is added. However, if the noise magnitude is too large, as seen in Figure 2.6(c), the firing rate becomes aberrantly high and the tracking performance is also deteriorated. Hence, in this sense, there appears to be an ‘optimal’ $\sigma_V$, though ascertaining this value analytically is difficult. Thus, some amount of tuning is needed.

### 2.5.3 The Effect of the Leak Term

We note that the leak term $\lambda_V V(t)$ is not an output of the optimization scheme, but rather a phenomenological addition that allows the neuronal dynamics to match the classical IF model. The incorporation of the leak term deteriorates the control performance as shown in Figure 2.7(a) and 2.7(b).
2.5.4 Units and Scaling of Parameters

The parameters in (2.6)-(2.12) can be expressed in biologically meaningful units. For example, $\lambda_d$ can be specified in Hz while the decoding weights $|\Gamma|$ and $|\Omega|$, determining the membrane voltage and the threshold, can be expressed as post-synaptic potential size in mV. It is important to note, however, that there is a tradeoff in the scale of these parameters in terms of the input gain matrix $B$ and the size $N$ of the network. In particular, with more neurons, the decoding weights can become smaller since more neurons are contributing to the control signal.

2.5.5 Synaptic Dynamics Allow for Spike Transmission Only

Note that in (2.10)-(2.11), interactions among neurons are mediated not only by the spike train $o(t)$, but also through the firing rate $r(t)$. However, the requirement that neurons transmit firing rates can be eliminated by endowing neurons with a synaptic state $S(t)$, which has the dynamics

$$
\dot{S}(t) = -\lambda_d S(t) + \lambda_d W^s o(t),
$$

(2.23)

With the dynamics (2.23), $S(t)$ is equivalent to $W^s r(t)$. In this way, the original dynamics can be rewritten as

$$
\dot{V}(t) = \Omega^T B^T A e(t) + \Omega^T B^T c(t) + S(t) + W^f o(t).
$$
2.5.6 Relationship to Predictive Coding

The control problem we consider is based on the notion of predictive neural coding [24], wherein the goal of the SNN is to mimic the input-output response of a linear system. Our formulation can be viewed as a generalization of this schema in three main ways: (i) we consider the possibility of spiking information being used instantaneously and in terms of rate; (ii) we consider the derived signal (i.e., \( u(t) \)) to be itself controlling a separate dynamical system. There is a potential semantic nuance here, since the system to be controlled could also be formulated as part of the ‘decoder.’ In fact, this nuance is what ultimately enables us to (iii) introduce explicit feedback error to the dynamics of neurons, which avoids the use of pseudo-inversion in obtaining the spiking dynamics [24]. It is worth noting that several features of our derived network do differ in interpretation from [14]. Indeed, the fast and slow interactions in our model arise directly from the formulation of our decoder, which has two time-scales. As we showed in our results, when the goal of the network is system control (or, more specifically, error reduction), these two time-scales are necessary to achieve performance. This is in contrast to the goal of emulating a system, wherein the time-scales arise from forwarding integration of the system dynamics, coupled with the aforementioned pseudo-inversion. It is also important to note that our scheme cannot be reduced to emulating a previously designed dynamical controller (e.g., a classical PID-type scheme). Rather, our network works on the primitive goal of error reduction, with no prior assumption or specification on controller dynamics.
Figure 2.2: Performance of the proposed spiking network (a) without and (b) with rate decoding. The first panels of (a) and (b) demonstrate the tracking performance; the spiking events, i.e. the spiking activity across the population are shown in the second panels while the membrane voltage for the first neuron and its threshold are plotted in the third panels.
Figure 2.3: Step response of the proposed spiking network for different numbers of neurons.
Figure 2.4: Robustness of the proposed network to neuronal failure. Spikes in the red shaded regions are manually suppressed. The internal dynamics of the network result in other neurons producing compensatory spikes, mitigating the effect on tracking.
Figure 2.5: In a multiple input setting, the decoding weights must be heterogeneous. Here, performance is shown for (a) homogeneous vs. (b) heterogeneous decoding.
Figure 2.6: The magnitude of the extrinsic noise $\sigma_V$ is a key parameter. In this example, too little or too much noise leads to degradation in performance.
Figure 2.7: Excessive leak $\lambda_V$ deteriorates performance.
Chapter 3

Integrated Planning, Control and Estimation within Spiking Networks

3.1 Introduction

In the previous chapter, we present an optimization method to construct a spiking network to drive a linear system to track a predefined trajectory. But how to generate such a reference signal online with spiking networks remains an unsolved problem. Moreover, it is also an interesting question to study: can we construct a network to steer the system to a target point directly without generating a trajectory.

Before answering those questions, we construct a distributed spiking network aiming at “copy” a time-varying signal based on predictive coding [24] of neuroscience. Similarly to the network, we build up in the previous chapter,

1. We propose an objective to minimize the difference between the tracking signal and the decoder output and to penalty terms of firing rate.
2. We use the greedy optimization method to solve the above problem and optimizes the spike train.

3. The solution turns out to be a threshold spiking rule defining when and which neuron fires a spike to decrease the objective value.

With this distributed network framework, we can distribute many control methods to this network.

For the first question, we adopt the minimum energy control for trajectory planning [30] and distribute such a control policy to a network of neurons (nodes). More specifically, we develop multi-layer networks to steer a linear system to a terminal.

1. The first layer distributes the minimum control into a distributed spiking network to generate a reference trajectory.

2. The second layer is a feedforward network as in [24] converting control signal to reference trajectory.

3. The third layer, similar to the network we developed in the previous chapter, is used for tracking control.

We then embed the linear quadratic regulator [80] to address the second question – endpoint control.

1. For a linear system, we introduce augmented states which are integral of the difference between fixed end states and original states and then design linear quadratic regulator to penalize on those augmented states [68]. In this way, we can drive the system to the target state based on LQR control.
2. Since the dynamic of augmented states is a linear dynamic system itself, we use the spiking network developed in [24] to realize it.

3. We then bury the LQR policy to a distributed spiking network that serves as an event-based controller.

Considering not all states are measurable, we further design a distributed network to realize a linear estimator or Kalman filter [12]. Combing with LQR, this provides a complete solution for control and estimation.

3.2 A Distributed Spiking Network

Before constructing SNN for minimum energy control or LQR control, we present a distributed spiking network as a fundamental building block, as shown in Fig. 3.1, whose decoded output $u(t)$ can reproduce time-varying input signal $\hat{u}(t)$, based on the framework in [24] and our previous work [52].

Figure 3.1: Schematic of Distributed Spiking Network. The network encodes the time varying signal $\hat{u}(t)$ into neural activity such as spikes and firing rates, while the decoder reproduces a signal $u(t)$ from the neural activity.
3.2.1 Preliminaries

Spike train, firing rate and firing amount of spiking networks

As in [24], we use \( o(t) \) to represent spike trains, which describe the form for activities of neurons in a spiking neural network (SNN) with dimension \( N \) and normally are modeled as

\[
o(t) = (o_1(t), \ldots, o_N(t))^T \in \mathbb{R}^N,
\]

in which \( o_k(t) = \sum_{i \in \{1, 2, \ldots\}} \delta(t - t_{ik}) \), for \( k = 1, \ldots, N \)

where a new Dirac delta function \( \delta(t - t_{ik}) \) is appended to \( o_k(t) \) when the \( k^{th} \) neuron fires the \( i^{th} \) spike at time \( t_{ik} \).

Meanwhile, we denote \( r(t) \) as firing rate [38] of neurons via

\[
\dot{r}(t) = -\lambda r(t) + \lambda o(t), \quad (3.1)
\]

with decaying rate \( \lambda \in \mathbb{R} \).

Moreover, we introduce the firing amount \( a(t) \) [50] by

\[
\dot{a}(t) = -\lambda a(t) + \lambda r(t), \quad (3.2)
\]

which accounts for the number of historical spikes and can be viewed as filtered firing rate.
**Decoder**

We adopt a *rate-instantaneous decoder* [50] to extract useful output signal $u(t)$ from spiking activities of the network by

$$\dot{u}(t) = -\lambda u(t) + \Gamma r(t) + \Omega o(t), \quad (3.3)$$

where $u(t) \in \mathbb{R}^m$, decaying factor $\lambda \in \mathbb{R}$, *rate decoding weight* $\Gamma \in \mathbb{R}^{m \times N}$ and *instantaneous decoding weight* $\Omega \in \mathbb{R}^{m \times N}$. Based on this equation, it is trivial to show that $u(t)$ can be written as a function of $a(t)$ and $r(t)$ [50] when $\Lambda = -\lambda I_m$.

$$u(t) = \frac{1}{\lambda} \Gamma a(t) + \frac{1}{\lambda} \Omega r(t). \quad (3.4)$$

**Optimization strategy**

To construct the SNN, we still need to set up an objective [50] to minimize the difference between output signal $u(t)$ and input signal $\hat{u}(t)$ as well as suppress the magnitude of firing rate $r(t)$, i.e.,

$$E(t) = \int_{t_0}^{t} \|\dot{u}(\tau) - u(\tau)\|_2^2 + \nu \|r(\tau)\|_1 + \mu \|r(\tau)\|_2^2 d\tau, \quad (3.5)$$

where $t_0$ is the initial time while $\nu$ and $\mu$ are penalty weights on the $l_1$ and $l_2$ norm of $r(t)$, respectively.

We also need to introduce the so-called ‘greedy spiking premise’ assumption wherein the $k^{th}$ neuron cannot fire a spike at $t_k^i$ unless it results in the instantaneous decrease of the value of $E(t)$ [24],
i.e.,

\[ E((t_k^0 + \varepsilon)|o_k(t_k^0) \text{ spikes}) < E((t_k^0 + \varepsilon)|o_k(t_k^0) = 0), \]  \hspace{1cm} (3.6)

where \(0 < \varepsilon \ll \lambda_d\).

### 3.2.2 Spiking rule of the network

**Proposition 2.** By solving the greedy optimization problem (3.6), a threshold-based spiking rule with respect to the network can be derived,

\[ v_k(t_k^0) > \bar{v}_k \Rightarrow \delta(t - t_k^0) \text{ is appended to } o_k(t), \]  \hspace{1cm} (3.7)

where \(v_k(t)\) is membrane voltage and \(\bar{v}_k\) is spiking threshold, defined by

\[ v_k(t) \equiv \Omega_k^T (\hat{u}(t) - u(t)) - \mu \lambda \bar{e}_k \bar{T} r(t) \]  \hspace{1cm} (3.8)

\[ \bar{v}_k = \frac{\Omega_k^T \Omega_k + \nu \lambda + \mu \lambda^2}{2}, \]  \hspace{1cm} (3.9)

where \(\bar{e}_k \in \mathbb{R}^N\) is the entry vector with all elements 0 except that the \(k^{th}\) entry is 1.

**Proof.** See Appendix B.

By introducing membrane voltage vector \(V(t) = (v_1(t), \cdots, v_N(t))\), we can rewrite (3.8) as a vectorized form

\[ V(t) \equiv \Omega^T (\hat{u}(t) - u(t)) - \mu \lambda r(t) \]  \hspace{1cm} (3.10)

41
After taking derivative for both sides, we can get

\[
\dot{V}(t) = \Omega^T (\dot{\hat{u}}(t) - \dot{u}(t)) - \mu \lambda \dot{r}(t)
\]

\[
= \Omega^T (\dot{\hat{u}}(t) + \lambda u(t) - \Gamma r(t) - \Omega o(t)) - \mu \lambda (-\lambda r(t) + \lambda o(t))
\]

\[
= \Omega^T \dot{\hat{u}}(t) + \Omega^T \Gamma a(t) + (\Omega^T \Omega - \Omega^T \Gamma + \mu \lambda^2 I) r(t) - + (-\Omega^T \Omega o(t) - \mu \lambda^2 I) o(t)
\] (3.11)

Note that

1. As long as we know the dynamics of the reference signal \( \dot{\hat{u}}(t) \), we can bury such dynamics to this network.

2. The predictive coding proposed in [24] can be generated by our distributed spiking network by embedding the linear system \( \dot{\hat{u}}(t) = A\hat{u}(t) + c(t) \) into the network, as shown in Fig. 3.2.

![Figure 3.2: Schematic of Predictive Coding. If we embed the linear dynamic system \( \dot{\hat{u}}(t) = A\hat{u}(t) + c(t) \) into the distributed spiking network, we can realize the predictive coding proposal in [24]. In this way, our spiking network can be seen as a general form of predictive coding.]
Table 3.1: Effect of the Number of Neurons $N$

<table>
<thead>
<tr>
<th>$\Gamma$</th>
<th>$K$</th>
<th>$\omega_n$</th>
<th>$\zeta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.81</td>
<td>22.48</td>
<td>0.62</td>
</tr>
<tr>
<td>10</td>
<td>0.87</td>
<td>30.63</td>
<td>0.67</td>
</tr>
<tr>
<td>50</td>
<td>0.96</td>
<td>61.35</td>
<td>1.53</td>
</tr>
</tbody>
</table>

3.2.3 Characterization

Neuron number $N$

To demonstrate the effect of neuron number $N$, we change the value of $N$ while keeping other parameters of the network as: $\sigma_v = 100$, $\mu = 2$, $\nu = 20$, $\lambda = 10$, $\Omega \sim \mathcal{N}(\mu_N, \sigma_N^2)$ where $\mathcal{N}$ is the normal distribution with mean $\mu_N = 0$ and standard deviation $\sigma_N = 5$, while $\|\Gamma_k\| = 100$ for $k = 1, \ldots, N$.

As the number of neurons $N$ increases, the output signal $u(t)$ tracks the input one $\hat{u}(t)$ with better precision in Fig. 3.3(a) or ramps up faster when $\hat{u}(t)$ is a step signal as in Fig. 3.3(b). Moreover, we model the step response as a second order system $\frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$ and use MATLAB identification toolbox `ident` to identify its parameters, as shown in Tab. 3.1, from which we can see that more neurons means wider bandwidth and lower damping ratio. Hence, more neurons enhance the ability of the network to keep up with a faster input command.

Rate Decoding Weight $|\Gamma|$

Now we study the effect of the rate decoding weight by fixing the neuron number as $N = 50$ while other parameters keep the same as above.

43
Figure 3.3: Control Performance Characterized by the Number of Neurons. More neurons lead to better tracking performance and faster step response.

As shown in Fig. 3.4(a), as the amplitude $|\Gamma|$ is increased, the tracking precision becomes better. While from the step response in Fig. 3.4(b), we observe the rising time is similar but the static error with the largest $\Gamma$ is much smaller.
From the identification result in Tab. 3.2, when the amplitude increases, the plant gain $K$ becomes closer to 1, the bandwidth $\omega_n$ decreases while the damping ratio $\zeta$ increases.

Figure 3.4: Control Performance Characterized by the Rate Decoding Weight. Larger rate decoding weight implies better tracking performance and faster step response.
### Table 3.2: Effect of the Rate Decoding Weight $|\Gamma|$

<table>
<thead>
<tr>
<th>$\Gamma$</th>
<th>$K$</th>
<th>$\omega_n$</th>
<th>$\zeta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.73</td>
<td>350.64</td>
<td>0.69</td>
</tr>
<tr>
<td>10</td>
<td>0.87</td>
<td>103.46</td>
<td>1.31</td>
</tr>
<tr>
<td>100</td>
<td>0.96</td>
<td>67.76</td>
<td>1.40</td>
</tr>
</tbody>
</table>

#### 3.3 Planning and Control for End Point Problem – Minimum Energy Control

We propose a three-layer-network, as shown in Fig. 3.5, to steer a linear system from an initial state $\hat{x}_0$ at time $t_0$ to an arbitrary target $\hat{x}_1$ at time $t_1$. Specifically, Layer [I], as seen in Fig. 3.6, takes the desired points $(\hat{x}_0, t_0)$ and $(\hat{x}_1, t_1)$ as input signals and generates the template control signal $u_d(t)$ and, afterwards, Layer [II], as in Fig. 3.7, gives birth to the template state trajectory $x_d(t)$ and passes it with $u_d(t)$ to Layer [III] which is just the optimal spiking network in Chapter 2.
3.3.1 Dynamics of networks

Layer [I]: producing the template control signal $u_d$

For a template linear system

$$\dot{x}_d(t) = Ax_d(t) + Bu_d(t),$$

we can design the template signal $\hat{u}_d$ by adopting minimum energy control,

$$\hat{u}_d(t) = B^T \Phi^T(t_1, t) W^{-1}_{c} [t_1, t_0] (\hat{x}_1 - \Phi(t_1, t_0) \hat{x}_0),$$

(3.12)
where $\Phi(t_{1}, t_{0}) = e^{A(t_{1} - t_{0})}$ is the state transition matrix and $W_{c}[t_{1}, t_{0}] = \int_{t_{0}}^{t_{1}} \Phi(t_{1}, \tau)BB^{T}\Phi^{T}(t_{1}, \tau) d\tau$ is the controllability Gramian. Then we can embed its derivative

$$\dot{\hat{u}}_{d}(t) = B^{T}\Phi^{T}(t_{1}, t)A^{T}W_{c}^{-1}[t_{1}, t_{0}](\hat{x}_{1} - \Phi(t_{1}, t_{0})\hat{x}_{0}),$$

to the “copy” spiking network, similar to (3.10),

$$V^{[I]}(t) = \Omega^{[I]T}(\dot{\hat{u}}_{d}(t) - u_{d}(t)) - \mu^{[I]}\lambda^{[I]}r^{[I]}(t),$$

combined with the decoder

$$\dot{\hat{u}}_{d}(t) = -\lambda^{[I]}u_{d}(t) + \Gamma^{[I]}r^{[I]}(t) + \Omega^{[I]}o^{[I]}(t),$$

(3.13)

to derive the dynamics of Layer $[I]$,

$$\dot{V}^{[I]} = \Omega^{[I]T}(\dot{\hat{u}}_{d}(t) - \dot{\hat{u}}_{d}(t)) - \mu^{[I]}\lambda^{[I]}r^{[I]}(t)$$

$$= \Omega^{[I]T}B^{T}\Phi^{T}(t_{1}, t)A^{T}W_{c}^{-1}[t_{1}, t_{0}](\hat{x}_{1} - \Phi(t_{1}, t_{0})\hat{x}_{0})$$

$$+ \Omega^{[I]T}\Gamma^{[I]}r^{[I]}(t)$$

$$+ \left(\Omega^{[I]T}\Omega^{[I]} - \Omega^{[I]T}\Gamma^{[I]} + \mu^{[I]}\lambda^{[I]}r^{[I]}I\right)r^{[I]}(t)$$

$$+ \left(-\Omega^{[I]T}\Omega^{[I]} + \mu^{[I]}\lambda^{[I]}r^{[I]}I\right)o^{[I]}(t),$$

(3.14)

where the superscript $[I]$ denotes variables or parameters of Layer $[I]$. 

48
Layer [II]: producing the template state trajectory $x_d$

Given a control signal $u_d(t)$, Layer [II] predicts the output of a linear system [24]. Similarly as above, embedding this dynamic system

$$\dot{x}_d(t) = A\hat{x}_d(t) + Bu_d(t),$$

to the distributed network, as in (3.10),

$$V^{[II]}(t) = \Omega^{[II]}T(\hat{x}_d(t) - x_d(t)) - \mu^{[II]}\lambda^{[II]}r^{[II]}(t),$$

(3.15)

combined with the decoder

$$\dot{x}_d(t) = -\lambda^{[II]}x_d(t) + \Gamma^{[II]}r^{[II]}(t) + \Omega^{[II]}a^{[II]}(t),$$

(3.16)

we can derive the dynamics of Layer [II],

$$V^{[II]}(t) = \Omega^{[II]}T(A\hat{x}_d(t) - x_d(t)) + \Omega^{[II]}T Bu_d(t)$$
$$+ \left(-\Omega^{[II]}T\Omega^{[II]} - \mu^{[II]}\lambda_{[II]}^2 I\right) a^{[II]}(t)$$
$$+ \left(-\Omega^{[II]}T\Gamma^{[II]} + \mu^{[II]}\lambda_{[II]}^2 I + \frac{1}{\lambda_{[II]}}\Omega^{[II]}T A\Omega^{[II]} + \Omega^{[II]}T \Omega^{[II]} \right) r^{[II]}(t)$$
$$+ \left(\frac{1}{\lambda_{[II]}}\Omega^{[II]}T A\Gamma^{[II]} + \Omega^{[II]}T \Gamma^{[II]}\right) a^{[II]}(t)$$

where the superscript [II] denotes variables or parameters of Layer [II].
To kick the decoder output $u_d(t)$ of the previous layer out of the above differential equation, we need to represent $u_d(t)$ in terms of the spike train $o^{[I]}(t)$ and the firing rate $r^{[I]}(t)$, that is, to incorporate (3.4) into the above dynamics

$$
\dot{V}^{[II]}(t) = \Omega^{[II]^T} A(\hat{x}_d(t) - x_d(t)) \\
\quad \quad + \frac{1}{\lambda^{[I]}} \Omega^{[II]^T} B\Omega^{[I]} r^{[I]}(t) + \frac{1}{\lambda^{[I]}} \Omega^{[II]^T} B\Gamma^{[I]} a^{[I]}(t) \\
\quad \quad + \left(-\Omega^{[II]^T} \Omega^{[II]} - \mu^{[II]} \lambda^{[II]^2} I\right) o^{[II]}(t) \\
\quad \quad + \left(-\Omega^{[II]^T} \Gamma^{[II]} + \mu^{[II]} \lambda^{[II]^2} I + \frac{1}{\lambda^{[II]}} \Omega^{[II]^T} A\Omega^{[II]} + \Omega^{[II]^T} \Omega^{[II]} \right) r^{[II]}(t) \\
\quad \quad + \left(\frac{1}{\lambda^{[II]}} \Omega^{[II]^T} A\Gamma^{[II]} + \Omega^{[II]^T} \Gamma^{[II]}\right) a^{[II]}(t)
$$

Moreover, to remove the term $\hat{x}_d(t)$ and $x_d(t)$, we have to introduce the pseudo-inverse of $\Omega^{[II]^T}$ and derive the following equation based on (3.15),

$$
\hat{x}_d(t) - x_d(t) \approx \Omega^{[II]^T} V^{[II]}(t) + \mu^{[II]} \lambda^{[II]} \Omega^{[II]^T} r^{[II]}(t).
$$

Then we can get

$$
\dot{V}^{[II]}(t) = \Omega^{[II]^T} A\Omega^{[II]^T} V^{[II]}(t) \\
\quad \quad + \frac{1}{\lambda^{[I]}} \Omega^{[II]^T} B\Omega^{[I]} r^{[I]}(t) \\
\quad \quad + \frac{1}{\lambda^{[I]}} \Omega^{[II]^T} B\Gamma^{[I]} a^{[I]}(t) \\
\quad \quad + \left(-\Omega^{[II]^T} \Omega^{[II]} - \mu^{[II]} \lambda^{[II]^2} I\right) o^{[II]}(t) \\
\quad \quad + \left(-\Omega^{[II]^T} \Gamma^{[II]} + \mu^{[II]} \lambda^{[II]^2} I + \frac{1}{\lambda^{[II]}} \Omega^{[II]^T} A\Omega^{[II]} + \Omega^{[II]^T} \Omega^{[II]} \right) r^{[II]}(t) \\
\quad \quad + \left(\frac{1}{\lambda^{[II]}} \Omega^{[II]^T} A\Gamma^{[II]} + \Omega^{[II]^T} \Gamma^{[II]}\right) a^{[II]}(t)
$$
Layer [III]: tracking control

As in our first work, the dynamics of this network is

$$
\dot{V}^{[III]}(t) = \Omega^{[III]}^T B^T A e(t) + \Omega^{[III]}^T B^T B u_d(t)
+ \left(-\frac{1}{\lambda^{[III]}} \Omega^{[III]}^T B^T B \Gamma^{[III]} + \mu \lambda^{[III]^2 I}\right) r^{[III]}(t)
- \left(\Omega^{[III]}^T B^T B \Omega^{[III]} + \mu \lambda^{[III]^2 I}\right) o^{[III]}(t)
= \Omega^{[III]}^T B^T B u_d(t)
+ \Omega^{[III]}^T B^T A x_d(t)
+ \left(-\frac{1}{\lambda^{[III]}} \Omega^{[III]}^T B^T A \Gamma^{[III]}\right) a^{[III]}(t)
+ \left(-\frac{1}{\lambda^{[III]}} \Omega^{[III]}^T B^T (A \Omega^{[III]} + B \Gamma^{[III]} + \mu \lambda^{[III]^2 I})\right) r^{[III]}(t)
+ \left(-\Omega^{[III]}^T B^T B \Omega^{[III]} - \mu \lambda^{[III]^2 I}\right) o^{[III]}(t)
$$

Similarly as above, by inserting the following equations

$$
u_d(t) = \frac{1}{\lambda^{[II]}} \Gamma^{[II]} a^{[II]}(t) + \frac{1}{\lambda^{[II]}} \Omega^{[II]} r^{[II]}(t),
\quad x_d(t) = \frac{1}{\lambda^{[III]}} \Gamma^{[III]} a^{[III]}(t) + \frac{1}{\lambda^{[III]}} \Omega^{[III]} r^{[III]}(t),
$$
we can get the dynamics of Layer [III],

\[
\dot{V}^{[III]}(t) = \frac{1}{\lambda^{[I]}} \Omega^{[III]}T B^T B \Gamma^{-[I]} a^{[I]}(t) \\
+ \frac{1}{\lambda^{[I]}} \Omega^{[III]}T B^T B \Omega^{[I]} r^{[I]}(t) \\
+ \frac{1}{\lambda^{[II]}} \Omega^{[III]}T A \Gamma^{-[II]} a^{[II]}(t) \\
+ \left( -\frac{1}{\lambda^{[III]}} \Omega^{[III]}T B^T A \Gamma^{-[III]} \right) a^{[III]}(t) \\
+ \left( -\frac{1}{\lambda^{[III]}} \Omega^{[III]}T B^T (A \Omega^{[III]} + B \Gamma^{[III]}) + \mu \lambda^{[III]} \right) r^{[III]}(t) \\
+ \left(-\Omega^{[III]}T B^T B \Omega^{[III]} - \mu \lambda^{[III]} \right) o^{[III]}(t)
\]

From the dynamics of the spiking networks, we can see that the intermediate signals – template control signal \( u_d(t) \) and trajectories \( x_d(t) \) are replaced by neural activities, implying that the decoders are not necessary to communicate signals from one layer to another layer.

### 3.3.2 Simulation and Characterization

To characterize the performance of this approach, we consider a planar motor system with viscous friction and coupled axis represented by

\[
\dot{x}(t) = Ax(t) + Bu(t),
\]
where \( x = [p_1, v_1, p_2, v_2]^T \),

\[
A = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & -2 & 0 & 3 \\
0 & 0 & 0 & 1 \\
0 & -3 & 0 & -2 \\
\end{bmatrix}, \quad B = \begin{bmatrix}
0 & 0 \\
1 & 0 \\
0 & 0 \\
0 & 1 \\
\end{bmatrix}.
\]

Each axis \( x_i = [p_i, v_i]^T \) is a typical linear motor system.

Parameters for the layer [I] are chosen as \( N^{[I]} = 50, \lambda_{v}^{[I]} = 0 \text{Hz}, \sigma_{v}^{[I]} = 500, \mu^{[I]} = 10, v^{[I]} = 100, \lambda^{[I]} = 10\text{Hz}, \Omega^{[I]} \sim \mathcal{N}(0, 5^2) \) – the normal distribution with 0 mean and 5 standard deviation, while \( \Gamma_{k}^{[I]} \) has the same sign as \( \Omega_{k}^{[I]} \) and has the amplitude 1000.

Parameters for the layer [II] are tuned as \( N^{[II]} = 50, \lambda_{v}^{[II]} = 0 \text{Hz}, \sigma_{v}^{[II]} = 500, \mu^{[II]} = 2, v^{[II]} = 20, \lambda^{[II]} = 10\text{Hz}, \Omega^{[II]} \sim \mathcal{N}(0, 5^2) \) – the normal distribution with 0 mean and 5 standard deviation, while \( \Gamma_{k}^{[II]} \) has the same sign as \( \Omega_{k}^{[II]} \) and has the amplitude 500.

Parameters for the layer [III] are tuned as \( N^{[III]} = 50, \lambda_{v}^{[III]} = 0 \text{Hz}, \sigma_{v}^{[III]} = 1000, \mu^{[III]} = 3, v^{[III]} = 40, \lambda^{[III]} = 10\text{Hz}, \Omega^{[III]} \sim \mathcal{N}(0, 5^2) \) – the normal distribution with 0 mean and 5 standard deviation, while \( \Gamma_{k}^{[III]} \) has the same sign as \( \Omega_{k}^{[III]} \) and has the amplitude 1000.

Our control goal is to steer the system from the original at time 0 ms to the target \((1000, 1000)\) at time 1000 ms.

**Analysis of Performance**

As shown in the second chart of Fig. 3.8, the decoded output \( u_d(t) \) of Layer [I] in black solid line follows the minimum energy control policy \( \hat{u}_d(t) \) in gray line very well.
Figure 3.8: Control Performance of Minimum Energy Control Strategy. The first chart shows the trajectory tracking performance, the second chart demonstrates the template control signal in black solid line and the actual control signal in black dotted line, and the spiking activity of neurons is plotted in the last chart.

For Layer [II], the decoded output of template trajectory $x_d(t)$ in black solid line also tracks the desired reference $\hat{x}_d(t)$ precisely, as shown in the first chart.
It is interesting to note, the actual control signal $u(t)$ does not follow the template control $u_d(t)$ very well in the second chart, but the actual state $x(t)$ is able to track the template trajectory $x_d(t)$. This fulfilled the target of the distributed network of Layer [III] driving $x(t)$ to follow $x_d(t)$.

**Robustness to Neuron Failure**

To demonstrate robustness to neuron failure, that is, how the networks respond when subsets of neurons fail to fire spike randomly, we disable half of the neurons during some time intervals by setting their voltages to zero, as in Fig. (3.9). And we can observe that our networks can still drive the system to the target point well.

**Center-out control**

We then conduct the canonical motor control task of center-out point-to-point planar motion, i.e., steering the system from the origin to some circular targets sharing the same center point and radius but having different orientations (see in the center chart of Fig. (3.10)). We can see that for different targets, our system can still be steered to them.

But for this center-out motion, we focus on neuronal tuning, which refers to neurons in the networks that are selectively tuned or responded to external stimuli. Tuning curves are widely used to characterize the responses of neurons to external stimuli [29]. To demonstrate this tuning property, we count the number of spikes according to different stimuli (orientation of target position in Fig. (3.10), or orientation of trajectory velocity in Fig. (3.11)) and plot the tuning curves by calling Matlab function `rose` and showing the angle histogram.
Figure 3.9: Robustness to Neuron Failure of Minimum Energy Control Strategy. Half of neurons, 26 – 50, 76 – 100, and 126 – 150, are disabled during some time intervals, 200 – 400ms and 600 – 800ms.

In Fig. (3.10), we can see that Neuron 1 fire the most spikes when the target points are in the west, Neuron 45 is active when the target points are in the north, etc. As it to say, neurons are selected to respond to certain stimuli or the orientation of target point. This can be understood since the voltage of each neuron depends on its own instantaneous decoding weight $\Omega_k$ which is generated randomly in Gaussian Distribution and the difference related to an objective as in Eq. (3.8). To
Figure 3.10: Tuning Curve with respect to Target Positions for Minimum Energy Control Strategy. The chart in the center demonstrates the center-out motion where the red star \( \hat{x}(t) \) represents the target point. The black line \( x_d(t) \) is the template trajectory, while the blue line \( x(t) \) is the actual state. The charts in the first row shows the tuning curve of Layer [I], the second row comes from Layer [II], while Layer [III] is shown in the last row.

To help understand, we write this equation again

\[
v_k(t) \equiv \Omega_k^T (\hat{u}(t) - u(t)) - \mu \lambda \bar{e}_k^T r(t).
\]
Figure 3.11: Tuning Curve with respect to Target Velocities for Minimum Energy Control Strategy.

Since the difference $\hat{u}(t) - u(t)$ is a vector with an orientation, hence neurons whose decoding weight $\Omega_k$ can maximize the dot production of $\Omega_k$ and $\hat{u}(t) - u(t)$, that is, neurons whose orientation of $\Omega_k$ are the same as or similar to the angle of $\hat{u}(t) - u(t)$, will be selected to fire spikes in order to decrease the difference $\hat{u}(t) - u(t)$ in the most efficient or optimal way.

When we select the external stimuli as trajectory velocities instead of target orientations, as in Fig. (3.11), we find that neurons do not show apparent tuning properties or are not tuned to such stimuli.
For example, Neuron 45 becomes active for almost all of the angles and hence does not present distinct spikes for a certain concentrated angle.

Returning to Fig. (3.10), we observe that some neurons, such as Neuron 18, 107, and 150, do not fire spikes at all for any external stimulus. To analyze such phenomena, we plot the total number of spikes versus the plane represented by $\Omega_k$, as in Fig (3.12). We can see that neurons with a larger amplitude of instantaneous decoding weight $|\Omega_k|$ fire more spikes. Actually, even if two neurons have similar orientation $\angle \Omega_k$ to match $\hat{u}(t) - u(t)$, the neuron with larger amplitude leads to higher voltage $v_k$ and got more opportunities to fire spikes. On the other side, the neuron with a smaller amplitude can become active or useful when the other one fails to work randomly. This ensures the robustness of neuron failure of the network.

### 3.4 Soft Final State Control with LQR

In the previous section, we propose a spiking network to generate a trajectory based on the minimum energy control policy where the control objective is to drive the system from an initial state exactly to a given fixed reference at a fixed final time [68]. This hard constraint on the final state hinders us from generating trajectory to fulfill a general objective, not just the concern of minimum energy. In addition, for some control applications, we do not put a hard constraint on the final time. Hence we need to develop a framework for a soft constraint on the final state without a fixed final time.
Figure 3.12: Tuning Analysis for Minimum Energy Control Strategy. Each point represents a neuron whose coordinates are the instantaneous weight and whose color represents the number of spikes it fires.

3.4.1 Introduction to PI Control in LQR

Before developing the framework, we introduce the PI control in LQR from the perspective of control engineering [68].

Our control target is to drive the steady state of a linear system $\dot{x}(t) = Ax(t) + Bu(t)$ to $x_r$ in infinite time horizon, that is $x(t) \to x_r$ as $t \to \infty$. 
To realize such a control goal, we augment the system with an integrator $z(t)$ of the difference between the final state $x_r$ and the current state $x(t)$. For the steady state, $\dot{z}(t)$ tends to zero, implying that $x(t)$ tends to the steady state $x_r$ finally. The augmented system can be written as

$$\dot{x}(t) = Ax(t) + Bu(t),$$

$$\dot{z}(t) = x_r - x(t),$$

or, in state space form,

$$\begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} A & 0 \\ -I & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ x_r \end{bmatrix}$$

Let $\tilde{x}(t) = \begin{bmatrix} x(t) \\ z(t) \end{bmatrix}$ be the augmented state, $\tilde{A} = \begin{bmatrix} A & 0 \\ -I & 0 \end{bmatrix}$ be the augmented state transition matrix, $\tilde{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}$ be the augmented input matrix, and $\tilde{x}_r = \begin{bmatrix} 0 \\ x_r \end{bmatrix}$ be the augmented final state, then the above system can be written in a simpler form

$$\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t) + \tilde{B}u(t) + \tilde{x}_r$$  \hspace{1cm} (3.17)

with initial condition $x(0) = x_0$

For the general linear quadratic regulator, the objective function is

$$\min_{u(t)} J(u) = \frac{1}{2} \int_0^T \dot{\tilde{x}}^T(t)Q\tilde{x}(t) + u^T(t)Ru(t)dt,$$  \hspace{1cm} (3.18)
the optimal controller has the form of

$$u(t) = -K \bar{x}(t) + u_r,$$

where $u_r = -B^{-1}Ax_r$ is the desired control to achieve the final state while $K = R^{-1}\bar{B}^TP$ is the optimal gain and $P$ is the solution to the algebraic Riccati equation,

$$P\bar{A} + \bar{A}^TP - P\bar{B}R^{-1}\bar{B}^TP + Q = 0.$$

Above all, the optimal solution to the LQR problem Eq. (3.18) has the close-loop form

$$u(t) = -R^{-1}\bar{B}^TP\bar{x}(t) - B^{-1}Ax_r.$$

(3.19)

Note that $B^{-1}$ here is the pseudo-inverse of input matrix $B$. Since $B^{-1}$ might not have only one solution, the final state control $u_r$ might not be unique.

### 3.4.2 Trajectory Generation

Now that we have the closed-form control solution, we can use it to generate a template trajectory.
Inserting the control solution $u(t)$ Eq. (3.19) into the augmented system Eq. (3.17) leads to

\[
\dot{x}(t) = \tilde{A}\ddot{x}(t) - \tilde{B}R^{-1}\tilde{B}^TP\ddot{x}(t) - \tilde{B}B^{-1}Ax_r + \ddot{x}_r \\
= \tilde{A}\ddot{x}(t) - \tilde{B}R^{-1}\tilde{B}^TP\ddot{x}(t) - \begin{bmatrix}
Ax_r \\
-x_r
\end{bmatrix}
\]

\[
= (\tilde{A} - \tilde{B}R^{-1}\tilde{B}^TP)\ddot{x}(t) - \tilde{A}\ddot{x}_r
\]

(3.20)

This is a still linear dynamic system and can be distributed into a spiking network.

We still consider the rate-instantaneous decoder,

\[
\dot{x}(t) = -\lambda \ddot{x}(t) + \Gamma r(t) + \Omega o(t),
\]

and the voltage for distributed spiking network,

\[
V(t) = \Omega^T(\ddot{x}(t) - \ddot{x}(t)) - \mu \lambda r(t),
\]
By taking derivative for both sides of the voltage, we can get dynamics of the network

\[
\dot{V}(t) = \Omega^T (\bar{A} - \bar{B}K) \bar{x}(t) - \Omega^T \bar{A}\hat{x}_r \\
- \Omega^T (-\lambda \hat{x}(t) + \Gamma r(t) + \Omega o(t)) - \mu \lambda (-\lambda r(t) + \lambda o(t)) \\
= \Omega^T (\bar{A} - \bar{B}K) (\bar{x}(t) - \hat{x}(t)) - \Omega^T \bar{A}\hat{x}_r \\
+ \Omega^T (\bar{A} - \bar{B}K + \lambda I) \hat{x}(t) - \Omega^T (\Gamma r(t) + \Omega o(t)) - \mu \lambda (-\lambda r(t) + \lambda o(t)) \\
= \Omega^T (\bar{A} - \bar{B}K) (\bar{x}(t) - \hat{x}(t)) - \Omega^T \bar{A}\hat{x}_r \\
+ \Omega^T (\bar{A} - \bar{B}K + \lambda I) \left( \frac{1}{\lambda} \Gamma a(t) + \frac{1}{\lambda} \Omega r(t) \right) \\
- \Omega^T (\Gamma r(t) + \Omega o(t)) - \mu \lambda (-\lambda r(t) + \lambda o(t)) \\
= \Omega^T (\bar{A} - \bar{B}K) (\bar{x}(t) - \hat{x}(t)) - \Omega^T \bar{A}\hat{x}_r \\
+ \frac{1}{\lambda} \Omega^T (\bar{A} - \bar{B}K + \lambda I) \Gamma a(t) \\
+ \left( \frac{1}{\lambda} \Omega^T (\bar{A} - \bar{B}K + \lambda I) \Omega - \Omega^T \Gamma + \mu \lambda^2 I \right) r(t) \\
+ (-\Omega^T \Omega - \mu \lambda^2 I) o(t)
\]

With the dynamics of the spiking network, we can generate the reference trajectory and then send it to the distributed control network for reference tracking such that we get another framework to drive a linear system to a target state.
### 3.4.3 Final State Control

However, since the optimal solution Eq. (3.19) is itself close-loop control. We can distribute the control in a spiking network for the purpose of final state control directly, that is, we can combine the planning and control problems in only one network.

For consistency of notation, we write the optimal control signal as

$$\hat{u}(t) = -R^{-1} \bar{B}^T P \bar{x}(t) - B^{-1} A x_r$$

Taking derivate for $\hat{u}(t)$, we obtain

$$\dot{\hat{u}}(t) = -R^{-1} \bar{B}^T P (\bar{A} \bar{x}(t) + \bar{B} u(t) + \bar{x}_r)$$

where $u(t)$ is the output of the decoder

$$\dot{u}(t) = -\lambda u(t) + \Gamma r(t) + \Omega o(t),$$

which converts neuron activity to a useful control signal to control a linear system.

Considering the voltage vector of the distributed network,

$$V(t) = \Omega^T (\hat{u}(t) - u(t)) - \mu \lambda r(t),$$
and taking derivate for both sides, we get dynamics of the network

\[
\dot{V}(t) = \Omega^T (\dot{u}(t) - \dot{u}(t)) - \mu \lambda \dot{r}(t)
\]

where

\[
\dot{V}(t) = \Omega^T R^{-1} B^T P (\bar{A} \bar{x}(t) + \bar{B} u(t) + \bar{x}_r) - \Omega^T (\lambda \bar{x}(t) + \Gamma \bar{r}(t) + \Omega \bar{O}) - \mu \lambda (\lambda \bar{r}(t) + \lambda \bar{o}(t))
\]

\[
= - \Omega^T R^{-1} B^T P (\bar{A} \bar{x}(t) + \bar{x}_r) + (\Omega^T R^{-1} B^T P \bar{B} + \lambda \Omega^T) \left( \frac{1}{\lambda} \bar{a}(t) + \frac{1}{\lambda} \Omega \bar{r}(t) \right) + (-\Omega^T \Gamma + \mu \lambda^2 I) r(t) + (-\Omega^T \Omega - \mu \lambda^2 I) o(t)
\]

\[
= - \Omega^T R^{-1} B^T P (\bar{A} \bar{x}(t) + \bar{x}_r) + \frac{1}{\lambda} (-\Omega^T R^{-1} B^T P \bar{B} + \lambda \Omega^T) \bar{a}(t)
\]

\[
+ \left( \frac{1}{\lambda} (-\Omega^T R^{-1} B^T P \bar{B} + \lambda \Omega^T) \Omega - \Omega^T \Gamma + \mu \lambda^2 I \right) r(t)
\]

\[
+ (-\Omega^T \Omega - \mu \lambda^2 I) o(t)
\]

(3.21)

However, even for the full state measurement, we can only get full information of the state \(x(t)\) instead of the augmented state \(\bar{x}(t)\), that is, we still need a network to estimate \(z(t)\) online.

Assume that we have a spiking network to encoder the stimulus \(z(t)\) and a decoder

\[
\dot{\hat{z}}(t) = -\lambda_z \hat{z}(t) + \Gamma_z \bar{r}(t) + \Omega_z \bar{o}(t),
\]

whose output \(\hat{z}(t)\) will be used for the control in Eq. (3.21).

With the same voltage vector in Eq. (3.10)

\[
V_z(t) = \Omega_z^T (z(t) - \hat{z}(t)) - \mu_z \lambda_z \bar{r}(t),
\]
we can get the dynamics of the network

\[
\dot{V}_z(t) = \Omega_z^T (\dot{z}(t) - \dot{\hat{z}}(t)) - \mu_z \lambda z \dot{r}_z(t)
\]

\[
= \Omega_z^T (x_r - x(t))
\]

\[+ \Omega_z^T \Gamma z a_z(t)
\]

\[+ (\Omega_z^T \Omega_z - \Omega_z^T \Gamma z + \mu_z \lambda^2 \Omega z) r_z(t)
\]

\[+ (\Omega_z^T \Omega_z o_z(t) - \mu_z \lambda^2 \Omega z) o_z(t)
\]  (3.22)

In this way, the LQR control network Eq. (3.21) can be updated as

\[
\dot{V}(t) = -\Omega^T R^{-1} \bar{B}^T P (Ax(t) - \dot{\hat{z}}(t) + \bar{x}_r)
\]

\[+ \frac{1}{\lambda} \left( -\Omega^T R^{-1} \bar{B}^T P \bar{B} + \lambda \Omega^T \right) \bar{a}(t)
\]

\[+ \left( \frac{1}{\lambda} \left( -\Omega^T R^{-1} \bar{B}^T P \bar{B} + \lambda \Omega^T \right) \Omega - \Omega^T \Gamma + \mu \lambda^2 \Omega \right) r(t)
\]

\[+ \left( -\Omega^T \Omega - \mu \lambda^2 \Omega \right) o(t)
\]  (3.23)
3.4.4 Simulation and Characterization

We consider the same model and the same approach to characterize the performance of the distributed networks.

Parameters for the LQR network in Eq (3.23) are chosen as $N = 50$, $\lambda_V = 0Hz$, $\sigma_v = 5000 \mu = 5$, $\nu = 50$, $\lambda = 10Hz, \Omega \sim \mathcal{N}(0, 5^2)$, while $\Gamma_k$ has the same sign as $\Omega_k$ and has the amplitude 500. And the $Q$ and $R$ weight matrix are chosen as

$$Q = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1000 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1000 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}, \quad R = \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}. $$

The reason to choose such a $Q$ lies that we would like to steer the system to the final state $(1000, 0, 1000, 0)$.

Parameters for the network of the augmented $z$ are tuned as $N_z = 100$, $\lambda_{zv} = 0Hz$, $\sigma_{zv} = 5000 \mu = 3$, $\nu_z = 30$, $\lambda_z = 10Hz, \Omega_z \sim \mathcal{N}(0, 5^2)$, while $\Gamma_{zk}$ has the same sign as $\Omega_{zk}$ and has the amplitude 150.
Analysis of Performance

The control performance of such networks is shown as in Fig. (3.13). The distributed networks successfully steer the system to the final state. But we also note that

1. Compared with the control curve of the minimum control policy in Fig. (3.8), the overshoot here is apparent, even up to 100%. We have tried tuning $Q$ and $R$ to compress the overshoot,

Figure 3.13: Control Performance of LQR Strategy.
but hardly seek a good set of parameters. Especially that our augmented system is not fully controllable, the solver to the Riccati equation limits the arbitrary tuning of $Q$ and $R$. In addition, one main disadvantage of LQR is that there is no theoretical connection between parameters of $Q, R$ and performance index such as overshoot and settling time.

2. Neurons of the LQR network fire too many spikes, as seen as Neuron 1 - 50 in the third chart. The purpose of this network is to follow the template control signal $\hat{u}$ in black solid line which is noise and sensitive to the augmented state $\hat{z}(t)$.

**Robustness to Neuron Failure**

We demonstrate the robustness of neuron failure by disabling half of the neurons for some time-intervals. As in Fig. (3.14), even some neurons fail to work for some time, the networks can still steer the system to the final state $x_r$.

**Center-out control**

We adopt the same approach to characterize the tuning property. From Fig. (3.15), we see that Neuron 1, 18, and 45, from the LQR networks, do not tend to be tuned to external stimuli. This is related to the noisy reference $\hat{u}(t)$ which interferes the orientation of the target state. However, Neuron 65, 86, 107, and 129 still shows typical tuning features.

Taking a look at Fig. (3.16), where the number of spikes (the color of each node) versus to the decoding weight plane is plotted, we can find that a neuron with a larger amplitude of $\Omega_k$ tends to be more active.
3.5 Linear Estimator and Kalman Filter

In many applications, due to limited available sensors, it is not practical to measure all of the states directly. Hence we need to use output feedback via observers or state estimators to design the control [12].
3.5.1 Introduction to Luenberger Observer and Kalman Filter

Considering the system with the measurement model and Gaussian noise

\[
\dot{x}(t) = \bar{A} \tilde{x}(t) + \bar{B} u(t) + \bar{x}_r + FW \\
y = \bar{C} x(t) + V,
\]
where $W$ representing dynamics noise and $V$ representing measurement noise are zero-mean, Gaussian white noise with covariance matrices $R_W$ and $R_V$, we can have the optimal linear observer [12]

$$\dot{\hat{x}}_o = \hat{A}\hat{x}_o + \hat{B}u + L(y - \hat{C}\hat{x}_o) + \bar{x}_r$$

$$= \hat{A}\hat{x}_o + \hat{B}(-K\hat{x}_o + u_r) + L(y - \hat{C}\hat{x}_o) + \bar{x}_r$$

$$= (\hat{A} - \hat{B}K - L\hat{C})\hat{x}_o + Ly + \bar{x}_r + \bar{B}u_r$$

$$= (\hat{A} - \hat{B}K - L\hat{C})\hat{x}_o + Ly - \bar{A}\bar{x}_r,$$
where \( L(t) = P(t)\tilde{C}^T R^{-1}_v(t) \) and \( P(t) = E[(\tilde{x}(t) - \hat{x}_o(t))(\tilde{x}(t) - \hat{x}_o^T(t))] \) satisfies

\[
\dot{P} = \tilde{A}P + P\tilde{A}^T - P\tilde{C}^T R^{-1}_v(t)\tilde{C}P + FR_W(t)F^T, \quad P(0) = E[x(0)x^T(0)].
\]

Specifically, if we have stationary noise and state dynamics for \( P(t) \), then \( P(t) \) converges to a steady solution and satisfies the following algebraic Riccati equation [12]:

\[
0 = \tilde{A}P + P\tilde{A}^T - P\tilde{C}^T R^{-1}_v(t)\tilde{C}P + FR_W F^T.
\]

Actually, the observer is a dynamical system whose inputs are the process output \( y \) and the reference target state \( \tilde{x}_r \). And, of course, we can bury this dynamic system to a distributed spiking network.

Assume that we have a rate-instantaneous decoder

\[
\dot{\hat{x}}_o(t) = -\lambda_o \tilde{x}_o(t) + \Gamma_o r_o(t) + \Omega_o o_o(t)
\]

and we hope that \( \hat{x}_o(t) \) can follow \( \tilde{x}_o \) with the SNN network whose voltage is

\[
V_o(t) = \Omega_o^T(\hat{x}_o(t) - \tilde{x}_o(t)) - \mu_o \lambda_o r_o(t).
\]

We can get an approximation of \( \hat{x}_o(t) \) by

\[
\hat{x}_o(t) = \Omega_o^{T-1}(V_o(t) + \mu_o \lambda_o r_o(t)) + \bar{x}_o(t).
\]
By taking the derivative of \( V_o(t) \), we can get

\[
\dot{V}_o(t) = \Omega_o^T (\dot{x}_o(t) - \dot{x}_o(t)) - \mu_o \lambda_o r_o(t)
\]

\[
= \Omega_o^T \left( (\bar{A} - \bar{B}K - L\bar{C}) \dot{x}_o + Ly + \ddot{x}_r + \bar{B}u_r \right)
\]

\[
- \Omega_o^T (\lambda_o \ddot{x}_o(t) + \Gamma_o r_o(t) + \Omega_o o_o(t)) - \mu_o \lambda_o (-\lambda_o r_o(t) + \lambda_o o_o(t))
\]

\[
= \Omega_o^T (\bar{A} - \bar{B}K - L\bar{C}) \left( \Omega_o^{T-1} (V_o(t) + \mu_o \lambda_o r_o(t)) + \ddot{x}_o(t) \right)
\]

\[
+ \Omega_o^T Ly + \Omega_o^T \ddot{x}_r + \Omega_o^T \bar{B}u_r
\]

\[
- \Omega_o^T (\lambda_o \ddot{x}_o(t) + \Gamma_o r_o(t) + \Omega_o o_o(t)) - \mu_o \lambda_o (-\lambda_o r_o(t) + \lambda_o o_o(t))
\]

\[
= \Omega_o^T (\bar{A} - \bar{B}K - L\bar{C}) \Omega_o^{T-1} V_o(t)
\]

\[
+ \Omega_o^T Ly + \Omega_o^T \ddot{x}_r + \Omega_o^T \bar{B}u_r
\]

\[
+(\Omega_o^T (\bar{A} - \bar{B}K - L\bar{C}) + \lambda_o \Omega_o^T) \ddot{x}_o(t)
\]

\[
+ \left( \mu_o \lambda_o \Omega_o^T (\bar{A} - \bar{B}K - L\bar{C}) \Omega_o^{T-1} - \Omega_o^T \Gamma_o + \mu_o \lambda_o^2 I \right) r_o(t)
\]

\[
+ \left( -\Omega_o^T \Omega_o - \mu_o \lambda_o^2 I \right) o_o(t)
\]

\[
= \Omega_o^T (\bar{A} - \bar{B}K - L\bar{C}) \Omega_o^{T-1} V_o(t)
\]

\[
+ \Omega_o^T Ly + \Omega_o^T \ddot{x}_r + \Omega_o^T \bar{B}u_r
\]

\[
+ \frac{1}{\lambda_o} \left( \Omega_o^T (\bar{A} - \bar{B}K - L\bar{C}) + \lambda_o \Omega_o^T \right) \Gamma_o a_o(t)
\]

\[
+ \frac{1}{\lambda_o} \left( \Omega_o^T (\bar{A} - \bar{B}K - L\bar{C}) + \lambda_o \Omega_o^T \right) \Omega_o r_o(t)
\]

\[
+ \left( \mu_o \lambda_o \Omega_o^T (\bar{A} - \bar{B}K - L\bar{C}) \Omega_o^{T-1} - \Omega_o^T \Gamma_o + \mu_o \lambda_o^2 I \right) r_o(t)
\]

\[
+ \left( -\Omega_o^T \Omega_o - \mu_o \lambda_o^2 I \right) o_o(t)
\]
Now the LQR controller can be updated as

\[ \hat{u}(t) = -R^{-1}B^T \hat{P} \hat{x}_o(t) + u_r \]

and its corresponding SNN can be rewritten as

\[
\begin{align*}
\dot{V}(t) &= -\Omega^T R^{-1} \hat{P} \hat{A} (\hat{A} \hat{x}_o(t) + \tilde{x}_r) \\
&\quad + \frac{1}{\lambda} (-\Omega^T R^{-1} \hat{P} \hat{B} + \lambda \Omega^T) \Gamma a(t) \\
&\quad + \left( \frac{1}{\lambda} (-\Omega^T R^{-1} \hat{P} \hat{B} + \lambda \Omega^T) \Omega - \Omega^T \Gamma + \mu \lambda^2 I \right) r(t) \\
&\quad + (-\Omega^T \Omega - \mu \lambda^2 I) o(t) \\
&= -\Omega^T R^{-1} \hat{P} \hat{A} \left( \frac{1}{\lambda_o} \Gamma_o a_o(t) + \frac{1}{\lambda_o} \Omega_o r_o(t) \right) \\
&\quad - \Omega^T R^{-1} \hat{P} \hat{P} \hat{x}_r \\
&\quad + \frac{1}{\lambda} (-\Omega^T R^{-1} \hat{P} \hat{B} + \lambda \Omega^T) \Gamma a(t) \\
&\quad + \left( \frac{1}{\lambda} (-\Omega^T R^{-1} \hat{P} \hat{B} + \lambda \Omega^T) \Omega - \Omega^T \Gamma + \mu \lambda^2 I \right) r(t) \\
&\quad + (-\Omega^T \Omega - \mu \lambda^2 I) o(t) \\
&= -\Omega^T R^{-1} \hat{P} \hat{x}_r \\
&\quad - \left( \Omega^T R^{-1} \hat{P} \hat{P} \frac{1}{\lambda_o} \Gamma_o \right) a_o(t) \\
&\quad - \left( \Omega^T R^{-1} \hat{P} \hat{P} \frac{1}{\lambda_o} \Omega_o \right) r_o(t) \\
&\quad + \frac{1}{\lambda} (-\Omega^T R^{-1} \hat{P} \hat{B} + \lambda \Omega^T) \Gamma a(t) \\
&\quad + \left( \frac{1}{\lambda} (-\Omega^T R^{-1} \hat{P} \hat{B} + \lambda \Omega^T) \Omega - \Omega^T \Gamma + \mu \lambda^2 I \right) r(t) \\
&\quad + (-\Omega^T \Omega - \mu \lambda^2 I) o(t)
\end{align*}
\]
3.5.2 Simulation and Characterization

We still use the same dynamic model of a planar motor, but we have the measurement mode with measure matrix

\[ \tilde{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]

i.e., we can only measure the positions. The covariance of Gaussian noise are set as \( R_v = R_w = 1 \).

Parameters for the LQR SNN network are chosen as \( N = 50, \lambda_V = 0 \text{Hz}, \sigma_v = 5000 \mu = 6, \nu = 60, \lambda = 10 \text{Hz}, \Omega \sim \mathcal{N}(0, 5^2) \), while \( \Gamma_k \) has the same sign as \( \Omega_k \) and has the amplitude 500. And the \( Q \) and \( R \) weight matrix are chosen as the same as before.

Parameters for the network of linear observer are tuned as \( N_o = 100, \lambda_oV = 0 \text{Hz}, \sigma_{ov} = 5000 \mu_o = 5, \nu_o = 50, \lambda_o = 10 \text{Hz}, \Omega_o \sim \mathcal{N}(0, 5^2) \), while \( \Gamma_{ok} \) has the same sign as \( \Omega_{ok} \) and has the amplitude 150.

Analysis of Performance

The control performance of networks for the LQR controller and the linear observer is shown as in Fig. (3.17). Even only a subset of states are measurable, we can still use output feedback with the linear estimator to drive the system to the desired final state.
Figure 3.17: Control Performance of LQR with Kalman Filter.

**Robustness to Neuron Failure**

From Fig. (3.18), we can see that our networks are robust to neuron failure since the system can still be steered to the desired final state even some neurons fail to work for some time.
Figure 3.18: Robustness to Neuron Failure of LQR with Kalman Filter.

Center-out control

Similarly as before, although the LQR network does not show apparent tuning property, neurons of the observer network can be tuned to external stimuli – the orientation of the target state, as plotted in Fig (3.19).
3.6 Conclusion

In this chapter, we present a multi-layer SNN based on minimum energy control to steer a linear system to a fixed endpoint within a specific time horizon. We then consider a relaxed version of this problem and embed LQR with integral action in distributed SNN. Moreover, we distribute a Kalman Filter for the purpose of state estimation.
Figure 3.20: Tuning Analysis of LQR with Kalman Filter.
Chapter 4

Distributing a Nonlinear Control Policy across a Network of Threshold-based Agents

4.1 Introduction

A long-standing line of study in neuroscience pertains to the so-called motor control, or how our brains enable the execution of complex motions such as manipulating our hands or walking [3, 20]. In this regard, many questions can be postulated, e.g., are particular brain networks performing specific ‘functions’ (e.g., optimization, disturbance rejection, etc.)? And, if so, how exactly do the dynamics of those networks (and, neural spiking in particular) achieve the function in question? In understanding these questions, control-theoretic concepts have sometimes been used as a framework within which to ascribe functions to certain brain regions. For example, the notion that the region of the brain known as the cerebellum performs state estimation and inference [43, 75].
Recently, another approach has emerged as a theoretical schema within which to study the above questions. Termed the predictive coding method [24], this approach aims to synthesize spiking networks from a first-principles optimization problem. The basic version of the problem is thus: given a way to ‘read out’ (or, decode) the activity of a spiking network, how should spikes be produced so that the decoded signal tracks that of a prescribed dynamical system? It turns out that this question can be answered by endowing the spiking network with a specific set of integrative dynamics that to a first approximation bear resemblance to those observed in the brain [24].

In our previous work, we adapted the predictive coding approach to handle control objectives. Rather than asking a network to mimic a dynamical system, we examined whether a network could be used for control. Specifically, we constructed a spiking network that can control a linear dynamic system [52]. The network is interesting not only because its dynamics are biophysically salient, but because it produces notable robustness properties.

In this paper, we aim to further explore the synthesis of biophysically meaningful spiking networks for control of general, nonlinear control-affine systems. The construction of the spiking network proceeds in two parts:

- Under the assumption that the nonlinear system can be linearized about a template trajectory, a design (e.g., linear quadratic regulator) is obtained for the linearized system.
- Spikes are produced in order to ‘copy’ the optimal control signal (for the linearized system).

Both parts can be fully embodied in a single, recurrent spiking network wherein each neuron amounts to an integrate-and-fire dynamical model [1]. The connections between neurons also have dynamics on several time-scales and can be obtained in closed form. It turns out that the features
of the network have quite interesting interpretations from a neuroscience perspective, though in this paper we only focus on the control-theoretic developments.

It is worth mentioning that our approach is distinct from the use of artificial neural networks (ANNs) for control (e.g., [81]). While such approaches have proven successful for practical control problems, they usually rely on networks that produce graded (i.e., non-spiking) activity with prescribed feedforward network architecture. Thus, the biophysical meaning of these networks is often limited to high-level conceptual interpretations. In contrast, our goal is to examine control by means of fully recurrent networks that have more naturalistic spiking dynamics.

4.2 Preliminaries

The purpose of our work is to control a nonlinear plant with an event-based distributed spiking network employing the schema in Fig. 4.1, whose components are presented as follows:

![Figure 4.1: Schematic of the Nonlinear Control Architecture.](image-url)
4.2.1 Control-Affine Nonlinear Systems

The plant to be controlled is a standard control-affine system,

\[ \dot{x}(t) = f(x(t)) + u(t)g(x(t)) = f(x(t)) + \sum_{i=1}^{m} u_i(t)g_i(x(t)) \]

where \( x(t) \in \mathbb{R}^n \) represents the system state, \( u(t) \in \mathbb{R}^m \) is the control signal, while \( f(x(t)) \) and \( g(x(t)) \) are vector fields over \( x(t) \).

We assume the control objective is to drive the actual state \( x(t) \) along a template trajectory \( x_d(t) \in \mathbb{R}^n \) produced by a template control input \( u_d(t) \in \mathbb{R}^m \) via the template system

\[ \dot{x}_d(t) = f(x_d(t)) + u_d(t)g(x_d(t)). \] (4.1)

4.2.2 Optimal Control Strategy

We aim at constructing a distributed controller to replicate a nominal control signal \( u_\ast(t) \) (i.e., \( u(t) = u_\ast(t) \)) from:

\[ u_\ast(t) = u_d(t) - w_\ast(t), \]

where \( w_\ast(t) \) is the optimal solution to the following optimization problem

\[ \min_{w(t)} J(w) = \frac{1}{2} \int_0^T e^T(t)Qe(t) + w^T(t)Rw(t)dt, \] (4.2)

s.t.

\[ e(t) = x_d(t) - x(t), \]

\[ w(t) = u_d(t) - u(t), \]
where the system state $x(t)$ is driven closely to the template command $x_d(t)$ while the control signal $u(t)$ is expected to approach to the template signal $u_d(t)$.

Note that the dynamics of $e(t)$ is a nonlinear system with respect to $x_d(t)$ and $u_d(t)$, i.e.,

$$\dot{e}(t) = \dot{x}_d(t) - \dot{x}(t) = F(e(t), w(t), x_d(t), u_d(t)).$$

Since our control task is to track a trajectory, as long as the controller performance well, it is nature to make a assumption that $e(t)$ is so small that the above dynamics can be linearized around $e(t) = 0$ [12],

$$\dot{e}(t) \approx \left. \frac{\partial F}{\partial e} \right|_{(x_d, u_d)} e(t) + \left. \frac{\partial F}{\partial w} \right|_{(x_d, u_d)} w(t)
= A(t)e(t) + B(t)w(t).$$

With this linearization, the optimization problem (4.2) becomes a standard Linear Quadratic Regulator (LQR) whose solution is

$$w_*(t) = -R^{-1}B(t)^TP(t)e(t),$$

where $P(t)$ is the solution to the Riccati differential equation. Therefore, we get the explicit form for the nominal control signal $u_*(t)$:

$$u_*(t) = u_d(t) + R^{-1}B(t)^TP(t)(x_d(t) - x(t)).$$ (4.3)
4.3 Dynamics and Properties of the Network

4.3.1 Dynamics of the Optimal SNN

To derive the dynamics of the network, we introduce the membrane voltage vector $V(t) = (v_1(t), \cdots, v_N(t))$ so that (3.8) can be expressed as a vectorized form

$$V(t) \equiv \Omega^T (u^*(t) - u(t)) - \mu \lambda r(t). \quad (4.4)$$

**Proposition 3.** The greedy optimization (3.5) and the optimal control (4.2) are achieved by a SNN with the dynamics which has the form of integrate and fire (IF) model,

$$\dot{V}(t) = -\alpha V(t) + W^e e(t) + W^u u_d(t) + \Omega^T \dot{u}_d(t) + W^a a(t) + W^r r(t) + W^o o(t), \quad (4.5)$$
where $\alpha$ is decaying rate of the membrane voltage, while $W^e \in \mathbb{R}^{N \times n}, W^u \in \mathbb{R}^{N \times m}, W^a \in \mathbb{R}^{N \times N}, W^r \in \mathbb{R}^{N \times N}, W^\omega \in \mathbb{R}^{N \times N}$ are weighting matrices specified by

\[ W^e = \alpha \Omega^T R^{-1} B^T(t) P(t) \]
\[ + \Omega^T R^{-1} B^T(t) P(t) - \Omega^T R^{-1} B^T(t) M(t), \tag{4.6} \]
\[ W^u = \Omega^T R^{-1} B^T(t) P(t) B(t) + \alpha \Omega^T, \tag{4.7} \]
\[ W^a = \left( 1 - \frac{\alpha}{\lambda} \right) \Omega^T \Gamma - \frac{1}{\lambda} \Omega^T R^{-1} B^T(t) P(t) B(t) \Gamma, \tag{4.8} \]
\[ W^r = \left( 1 - \frac{\alpha}{\lambda} \right) \Omega^T \Omega - \Omega^T \Gamma + \left( 1 - \frac{\alpha}{\lambda} \right) \mu \lambda^2 I \]
\[ - \frac{1}{\lambda} \Omega^T R^{-1} B^T(t) P(t) B(t) \Omega, \tag{4.9} \]
\[ W^\omega = -\Omega^T \Omega - \mu \lambda^2 I, \tag{4.10} \]

where

\[ M(t) = A(t)^T P(t) - P(t) B(t) R(t)^{-1} B(t)^T P(t) + Q(t). \]

**Proof.** See Appendix C. \qed

Note,

1. the nominal control signal $\hat{u}(t)$ does not appear explicitly in the dynamics (4.5). Actually, the optimal control policy is fully embedded within the dynamics of neurons and their connections. This implies that optimal control is realized by a distributed network controller.
2. when the decaying rate $\alpha$ of the membrane voltage is equal to the forgetting rate $\lambda$ of the firing rate, dynamics of the membrane voltage can be simplified to

$$
\dot{V}(t) = -\lambda V(t) + \lambda \Omega^T R^{-1} B^T (t) P(t) e(t) \\
+ \Omega^T \left( R^{-1} \dot{B}^T (t) P(t) - R^{-1} B^T (t) M(t) \right) e(t) \\
+ (\Omega^T R^{-1} B^T (t) P(t) B(t) + \lambda \Omega^T) u_d(t) \\
+ \Omega^T \dot{u}_d(t) \\
- \frac{1}{\lambda} \Omega^T R^{-1} B^T (t) P(t) B(t) \Gamma a(t) \\
+ \left( -\Omega^T \Gamma - \frac{1}{\lambda} \Omega^T R^{-1} B^T (t) P(t) B(t) \Omega \right) r(t) \\
+ (-\Omega^T \Omega - \mu \lambda^2 I) o(t).$$  \quad \text{(4.11)}

### 4.4 Examples

In this section, we use several examples to highlight the features of the derived spiking network.

We consider a classic bistable attractor,

$$
\dot{x}(t) = f(x) + u g(x) = x(0.5 - x)(0.5 + x) + u \cos x,
$$

which is a classic nonlinear system in computational neuroscience. It can be linearized as

$$
A = 0.25 - 9x_d^2 - u_d \sin x_d, \\
B = \cos x_d.
$$
In the following examples, to avoid designing a template control strategy $u_d$ which is not our concern in this work, we simply set it *a priori* as a sinusoid which leads to a periodic waveform $x_d(t)$ as depicted in Figure 4.2.

The parameters for the designed distributed controller are set as following: $Q = I_n, R = 10^{-3}I_m, N = 100, \sigma_v = 100, \mu = 3, \nu = 30, \lambda = 10, \Omega \sim \mathcal{N}(0, 5^2)$, and $\Gamma_{i,j}$ has the same sign as $\Omega_{i,j}$, where $I_n$ represent the identity matrix with dimension $n$.

### 4.4.1 Tracking Performance

From Figure 4.2 and 4.3, we can see that the actual state $x(t)$ can track the template trajectory $x_d(t)$ well, which demonstrates that the derived SNN is effective to drive a nonlinear system along its prescribed trajectory.

Comparing Figure 4.3 with Figure 4.2, the existence of the rate decoding leads to better trajectory tracking (note the error between $x_d(t)$ and $x(t)$ and the difference between $u_d(t)$ and $u(t)$) and sparser spiking distribution of neurons, since the rate decoding employs additional historical information to build up the control signal $u(t)$.

### 4.4.2 Robustness to Neuron Failure

One important feature of the distributed network is its robustness to node failure. To verify such a feature, we disable half neurons in the network for the time intervals $[300, 1200]$ ms and $[1800, 2700]$ ms. Fig. 4.4 and Fig. 4.5 shows that even some neurons fail to work for some time, others compensate by increasing their own activities so that the control object is still achieved.
Moreover, we find that the rate decoding makes the network much more robust to neuron failure. Comparing Figure 4.5 and Figure 4.4 where only the instantaneous decoding is used, when we disable some neurons, the tracking performance is deteriorated by at least 50% even other neurons try to compensate. But when the rate decoding is equipped, as in the comparison between Figure 4.5 and Figure 4.4, the actual state $x(t)$ is still able to approach to the template signal $x_d(t)$ with high precision. This is intuitive: If the rate decoding is used, fewer neurons are needed to fire.
spikes to achieve a control requirement. Hence when some neurons fail to work, it is much easier for other neurons to take over the responsibility.

To further study the advantage of the rate decoding, we plot the RMS tracking error with respect of the percentage of disabled neurons for different rate decoding weights, as shown in Fig. 4.6. From this figure:

Figure 4.3: Performance of Rate-Instantaneous Decoding $|\Gamma_{ij}| = 200$. 
Figure 4.4: Robustness of Instantaneous Decoding.

1. as more neurons fail, it is reasonable to see that the RMS error for all three cases ($|\Gamma_{ij}| = 0, 50, \text{and} 200$) becomes larger.

2. when the rate decoding plays a role, for example, $|\Gamma_{ij}| = 200$, the RMS error is always much less than that of $|\Gamma_{ij}| = 0$. Even when 80% neurons do not work, the RMS error is still less than the scenario of none neurons are disabled without the rate decoding.
Figure 4.5: Robustness of Rate-Instantaneous Decoding $|\Gamma_{ij}| = 200$.

3. The trend of these three curves are very similar.
4.5 Conclusion

We further answered the question of how neuronal networks decode spiking activities into usable signals from the point of view of control theory by extending our previous work [51] of controlling a linear system to a control-affine nonlinear plant. We derived a spiking neural network whose activity is decoded by a specific rate-instantaneous decoder into a control signal to drive a dynamic system along a predefined trajectory. We proposed an approach involving two nested optimization problems, analogous to a dual loop feedback control strategy, to construct an SNN with threshold-based integrative dynamics and connections that exchange signals at three time-scales. Actually, the two optimization policies can be embedded into one spiking network, serving as an event-based distributed controller. We provided several examples to examine the control capabilities and robustness of such a network. From a biological perspective, however, a question arises as to how...
the template trajectory is generated and how to learn internal connections among neurons. These will be our future work.
Chapter 5

Nonlinear Planning via Iterative Spiking Networks

5.1 Introduction

Most real-world operations are repeatedly performed, implying that the control tracking error is also repeated during each pass. Although that tracking error from past iterations contains rich information, they cannot be used by a non-learning controller. Learning control strategies, such as iterative learning control (ILC), can take full advantage of the previous error information into the control for next iterations. In this way, perfect tracking performance can be achieved by learning from previous executions [26, 79].

Moreover, in our previous work, we always assume that we have the exact model information. But in reality, this is always impossible because of the existence of model uncertainty and disturbance. However, as long as the process is repetitive, we can use ILC to achieve low transient error despite large model uncertainty and repeating disturbances [26].
For instance, when a basketball player is practicing the set shoot – shooting from a fixed position, he improves his or her ability to score iteratively by observing the trajectory of the ball in previous trials and consciously make corrections for the next attempt. With more and more practice, the correct motion (control signal) is iteratively learned and ingrained into the muscle memory so that the shooting becomes more and more accurate. From the standpoint of control engineering, this is the essence of ILC which is a mature and widely used technology in the industry. However, from the perspective of neuroscience, how the dynamics of neural networks, especially spiking networks, support such a learning process, is not yet discovered.

Artificial neural network control can also be used to improve tracking performance despite model uncertainty by learning from training data and modifying controller parameters instead of a control signal. Hence, larger networks of nonlinear neurons often require massive training data and lead to slow convergence [53], whereas ILC usually is able to converge in just a few iterations.

In addition, ILC can also be used to plan trajectories for nonlinear systems based on the fact that ILC can achieve perfect tracking error after learning over some iterations. With the assistance of ILC, we can plan the template signals $x_d$ and $u_d$ for our nonlinear control in the previous chapter.

In this chapter, we start from a popular PD-Type ILC and present an iterative framework to construct a network of nodes to distribute such an ILC. We then demonstrate control performance and convergence with an example.
5.2 PD-Type ILC

A widely used PD-type ILC learning algorithm \([21, 26, 79]\) can be expressed as

\[
u_{j+1}(k) = u_j(k) + w_j(k)
= u_j(k) + Q(k_p e_j(k) + k_d (e_j(k) - e_j(k-1)))
\]

where \(u_{j+1}(k) \in \mathbb{R}^m\) is the control signal for the \((j + 1)^{th}\) iteration at time \(k\), \(e(k)\) is the control error, \(Q\) represents a filter, mostly a low pass filter in practice, \(k_p\) is the proportional gain, and \(k_d\) is the derivative gain.

Note that, in this learning algorithm,

1. The control signal of the \((j + 1)^{th}\) iteration \(u_{j+1}\) is composed of the control signal of the previous iteration \(u_j\) “history information”, and the control signal increment of the current iteration \(w_j\), “current information”.

2. The derivative term \(k_d(e_j(k) - e_j(k-1))\) improves the stability margin of the system but causes high frequency noise. Hence, in practice, we usually use \(Q\) as a low pass filter to filter out the high frequency noise.

5.3 Distributed Networks for ILC

Based on the structure of the learning algorithm, we can distribute the iterative learning control via spiking networks in the following steps,
1. Construct a recurrent SNN to realize $w_j(k) = Q(k_p e_j(k) + k_d(e_j(k) - e_j(k - 1)))$,

2. Develop an iterative SNN to realize $u_{j+1}(k) = u_j(k) + w_j(k)$.

### 5.3.1 Distributed Recurrent SNN for Control Signal Increment $w_j(k)$

First of all, we need to convert $\hat{w}(t) = Q(k_p e(t) + k_d \dot{e}(t))$ to a state space representation so that we can realize it via a distributed SNN as we developed in Chapter 3.

Before doing that, we use a simple and typical case to exemplify the process. Assume that the filter $Q$ is a low pass filter and has the form of

$$Q(s) = \frac{\omega_n^2}{s^2 + 2 \zeta \omega_n s + \omega_n^2},$$

where $\omega_n$ is the cut-off frequency while $\zeta$ is the damping ratio, we can write the transfer function between $W(s)$ (Laplace transform of $\hat{w}(t)$) and $E(s)$ (Laplace transform of $e(t)$) as

$$G_{WE} = \frac{W(s)}{E(s)} = \frac{k_d \omega_n^2 s + k_p \omega_n^2}{s^2 + 2 \zeta \omega_n s + \omega_n^2}.$$

Afterwards, we can convert the above transfer function to a state space representation,

$$\dot{\hat{q}}(t) = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta \omega_n \end{bmatrix} \hat{q}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} e(t)$$

$$\hat{w}(t) = \begin{bmatrix} k_p \omega_n^2 & k_d \omega_n^2 \end{bmatrix} \hat{q}(t)$$
For a general case, the control increment $\hat{w}(t) = Q(k_pe(t) + k_d\dot{e}(t))$ can be converted to a state space representation

$$\dot{\hat{q}}(t) = F\hat{q}(t) + Ge(t),$$
$$\hat{w}(t) = H\hat{q}(t),$$

which is a linear dynamic system with a measurement model.

To realize such a dynamic system in a distributed SNN, we need to generalize our work of to cover the measurement model.

Assume that our decoder of the SNN takes the form of

$$\dot{w}(t) = -\lambda w(t) + \Gamma r(t) + \Omega o(t) \tag{5.1}$$

and the objective of our network is to minimize the output error and penalize terms of firing rate, that is,

$$E(t) = \int_{t_0}^{t} \|\hat{w}(\tau) - w(\tau)\|_2^2 + V \|r(\tau)\|_1 + \mu \|r(\tau)\|_2^2 d\tau, \tag{5.2}$$

**Proposition 4.** By solving the greedy optimization problem (5.2), a threshold-based spiking rule with respect to the network can be derived,

$$v_k(t^*_k) > \bar{v}_k \Rightarrow \delta(t - t^*_k) \text{ is appended to } o_k(t), \tag{5.3}$$
where \( v_k(t) \) is the membrane voltage for the \( k^{th} \) neuron and \( \bar{v}_k \) is the spiking threshold, defined by

\[
v_k(t) \equiv \Omega_k^T (\hat{w}(t) - w(t)) - \mu \lambda \bar{e}_k^T r(t) \quad \text{(5.4)}
\]

\[
\bar{v}_k = \frac{\Omega_k^T \Omega_k + \nu \lambda + \mu \lambda^2}{2}, \quad \text{(5.5)}
\]

where \( \bar{e}_k \in \mathbb{R}^N \) is the entry vector with all elements 0 except that the \( k^{th} \) entry is 1.

If we Denote the membrane voltage vector as \( V = (v_1, \cdots, v_N) \), we have

\[
V(t) = \Omega^T (\hat{w}(t) - w(t)) - \mu \lambda r(t) \quad \text{(5.6)}
\]

and then the dynamics of the SNN become

\[
\dot{V}(t) = \Omega^T HF (\Omega^T H)^{-1} V(t) + \Omega^T H Ge(t) \\
+ \left( \frac{1}{\lambda} \Omega^T HF (\Omega^T H)^{-1} \Omega^T \Gamma + \Omega^T \Gamma \right) a(t) \\
+ \left( \mu \lambda \Omega^T HF (\Omega^T H)^{-1} + \frac{1}{\lambda} \Omega^T HF (\Omega^T H)^{-1} \Omega^T \Omega + \Omega^T \Omega - \Omega^T \Gamma - \mu \lambda^2 I \right) r(t) \\
+ (-\Omega^T \Omega - \mu \lambda^2 I) o(t). \quad \text{(5.7)}
\]

Proof. See Appendix D.

\[
\]

5.3.2 Distributed Iterative SNN for ILC Control Signal \( u_{j+1}(k) \)

Now we need to construct an iterative SNN to mimic the reference system

\[
\hat{u}_{j+1}(t) = \hat{u}_j(t) + w_j(t) \quad \text{(5.8)}
\]
The SNN constitutes of a derivative firing rate

\[ \dot{r}_j(t) = -\lambda r_j(t) + \lambda o_j(t) \]

and an iterative firing rate

\[ R_{j+1}(t) = R_j(t) + r_j(t). \] (5.9)

Neural activity can be decoded into iterative control signal via the decoder

\[ u_{j+1}(t) = u_j(t) + \Omega r_j(t) \]

The objective of the spiking network is defined as

\[ E_j(t) = \|\hat{u}_j(t) - u_j(t)\|^2 + \nu \|R_j(t)\|_1 + \mu \|R_j(t)\|^2. \] (5.10)

**Proposition 5.** By solving the greedy optimization problem (5.10), a threshold-based spiking rule with respect to the network can be derived,

\[ v_{j,k}(t^i_k) > \bar{v}_{j,k} \Rightarrow \delta(t-t^i_k) \text{ is appended to } o_{j,k}(t), \] (5.11)

where \( v_{j,k}(t) \) is the membrane voltage for the \( k^{th} \) neuron under the \( j^{th} \) iteration and \( \bar{v}_{j,k} \) is the spiking threshold, defined by
\[ v_{j,k}(t) \equiv \lambda \Omega_k^T (\hat{u}_j(t) - u_j(t)) - \mu \lambda \tilde{e}_k^T R_j(t) \quad (5.12) \]

\[ \tilde{v}_{j,k} \equiv \frac{\lambda^2 \Omega_k^T \Omega_k + \nu \lambda + \mu \lambda^2}{2}, \quad (5.13) \]

where \( \tilde{e}_k \in \mathbb{R}^N \) is the entry vector with all elements 0 except that the \( k^{th} \) entry is 1.

If we denote the membrane voltage vector as \( V = (v_1, \cdots, v_N) \), we have

\[ V_j(t) = \lambda \Omega^T (\hat{u}_j(t) - u_j(t)) - \mu \lambda R_j(t). \quad (5.14) \]

and then the dynamics of the SNN become

\[ V_{j+1}(t) = V_j(t) + \lambda \Omega^T w_j(t) - (\lambda \Omega^T \Omega + \mu \lambda I) r_j(t). \quad (5.15) \]

Proof. See Appendix E.

Note that in this iterative SNN, the voltage of the next iteration depends on the current iteration ("history information"), the current increment ("current update"), and the current firing rate ("current neural activity").

### 5.4 Simulation of Learning and Control for Model Uncertainty

We use a simple example to demonstrate the effectiveness of our SNNs. Here we consider to control a SISO linear system with nonlinear model uncertainty, \( \dot{x}(t) = -10x(t) - 0.001x^3(t) + u(t) \).
Note that the main part is a linear system, but there is one nonlinear part that is unknown. To conduct the simulation, we use a distributed SNN developed in Chapter 2 as a feedback controller to deal with the linear part and an iterative SNN as a feedforward learning controller for the nonlinear model uncertainty.

Parameters for the distributed SNN of feedback controller are tuned as \( N = 100, \lambda_V = 0 \text{Hz}, \sigma_v = 50 \mu = 3, \nu = 0.3, \lambda = 0.03 \text{Hz}, \Omega \sim N(0, 0.5^2) \), while \( \Gamma_k \) has the same sign as \( \Omega_k \) and has the amplitude 10.

Parameters for the iterative SNN of ILC controller are tuned as \( N = 100, \lambda_V = 0 \text{Hz}, \sigma_v = 500 \mu = 0.1, \nu = 1, \lambda = 1 \text{Hz}, \Omega_k = 2.5 \) for \( k = 1, \ldots, 50 \), and \( \Omega_k = -2.5 \) for \( k = 51, \ldots, 100 \).

The parameters for the ILC are set as \( kp = 10, kd = 100, \omega_n = 50, \zeta = 1.0 \) and the sampling frequency \( f_s = 1000 \text{ Hz} \).

### 5.4.1 Control Performance

As shown in Fig. 5.1, after 10 iterations, the maximum tracking error decrease by 75% so that the actual state \( x(t) \) can track the desired trajectory \( \hat{x}(t) \) very well. This demonstrates that our SNN can learn and benefit from repetitive tracking error.

### 5.4.2 Convergence

Fig. 5.2 demonstrates the convergence by showing the whole learning process over iterations. We can see that, after each iteration of learning, tracking error becomes smaller and smaller, implying that the error converges to almost zero at last.
5.5 Simulation of Planning and Control for Nonlinear Systems

In this section, we use iterative SNN to plan a trajectory for a classic bi-stable attractor in computational neuroscience

\[
\dot{x}(t) = f(x) + u g(x) = 0.01 \times x(10 - x)(10 + x) + 10 \times u.
\]

Afterwards, we combine this planning with the nonlinear control work in the previous chapter.

First, we generate a template control signal given a periodic sinusoidal trajectory, as shown in Figure 5.3. After learning over some iterations, the actual state \(x(t)\) follows the reference signal \(\hat{x}(t)\) very well, implying that the learned control signal, as in the third chart, can match the reference trajectory.

Second, we can pass the planned signals to the SNN we developed in the previous chapter for nonlinear control. From Figure 5.4, the actual state \(x(t)\) can track the desired reference signal \(x_d(t)\)
Figure 5.2: Convergence for a Linear System with Exact Model. Fig (a) demonstrates the tracking performance, the tracking error, the control signal, and spiking distribution of the feedback SNN, while Fig (b) shows the ILC control signal for each iteration, and spiking distribution of the iterative SNN.

very well while the actual control $u(t)$ can also follow the planned signal $u_d(t)$. This demonstrates the effectiveness of our approaches in nonlinear planning and control.
Figure 5.3: Learning Result of Iterative SNN

5.6 Conclusion

In this chapter, we distribute an ILC learning algorithm into an iterative SNN. We use an example to demonstrate the effectiveness and convergence of the network. We also use such a network to plan template trajectories for nonlinear systems and combine it with our previous work in nonlinear control. The future work will be proving the convergence in theory. Furthermore, it is of interest to study how a spiking network supports reinforcement learning (RL).
Figure 5.4: Planning and Control for Nonlinear Systems
Chapter 6

Conclusion

6.1 Outcomes

This work has examined the dynamics of spiking neural networks from the perspective of normative synthesis. Using classical control-theoretic objectives, we have synthesized spiking networks that achieve a number of canonical tasks, such as reference tracking and endpoint control. We have done this by building networks that enact proven theories regarding these functions. The synthesis leverages a key idea in theoretical neuroscience, namely that neurons fire spikes in an efficient manner and only when doing so leads to an increase in the prescribed objective function. We have shown that under this assumption, networks can be built near closed-form. The dynamics that emerge in this situation mimic the well-known integrate-and-fire neural model. Further, the networks display certain properties that are known to exist biologically, including position and velocity tuning. Because these properties were not prescribed \textit{a priori}, our synthesis procedure provides a quantitative hypothesis regarding their functional significance.
The second outcome of this dissertation is that shows how we might create distributed, spike-based versions of classical control solutions. With the rise of network-based control systems, understanding how to distribute a control policy across a set of interacting agents is an increasing problem of interest [46, 86, 27, 25, 84, 34, 106]. Said another way, there is an alternative interpretation of this research that has to do with creating cooperative decision policies for agents interacting toward a common goal. In this regard, our solutions could be viewed as a way to prescribe network architecture and a set of local decision rules for agents to take actions that ultimately lead to an overall objective.

### 6.2 Outlook and Future Work

Clearly, the most significant limitation of the current work is that it predominantly treats linear control problems. Treating a broader class of nonlinear problems represents a clear direction for future exploration, though there are certainly very real limits to the developed theory in this regard. As such, making traction of this problem is likely to require modifying some of the fundamental predictive coding assumptions that underlie the synthesis procedures studied herein.

There is also an opportunity to expand on placing this work in the context of multi-agent systems. Whereas that body of control theory typically focuses on problems related to controlling such systems, our framework would examine control *by means* of such systems. By generalizing the idea of a decoder and considering a broader class of events beyond just impulsive spikes, many additional problems related to emerging engineering areas in cyber-physical systems and distributed sensing and control [65, 66, 6, 113, 40, 42, 78, 60].
Appendix A

Derivation of the Spiking Rule (2.6)-(2.12)

The methodology to derive the dynamics of the spiking network is based on the schema originally developed in [24]. Our derivation deviates insofar as we utilize the feedback error directly, consistent with our considering of control rather than prediction objective.

We begin by quantifying the effect of any added spike on the overall cost. Assume that the $k_{ih}$ neuron is silent at time $t_s$ and there are no spikes since time $t_s$, then we can get the expression

$$
\tilde{r}(t) = e^{-\lambda_d(t-t_s)} r(t_s),
$$

where $\tilde{r}(t)$ denotes the firing rate at time $t$ assuming no spikes fired since time $t_s$.

If the $k^{th}$ neuron spikes at time $t_s$, then a delta function $\delta(t-t_s)$ is added to $a_k(t)$ resulting in

$$
r(t) = e^{-\lambda_d(t-t_s)} r(t_s) + \int_{t_s}^{t} e^{-\lambda_d(t-\tau)} \lambda_d \delta(\tau-t_s) d\tau
$$

$$
= \tilde{r}(t) + \int_{t_s}^{t} e^{-\lambda_d(t-\tau)} \lambda_d \delta(\tau-t_s) d\tau
$$

$$
= \tilde{r}(t) + e^{-\lambda_d(t-t_s)} \lambda_d \delta_k.
$$
Define $\tilde{u}(t)$ and $\tilde{x}(t) = e^{A(t-t_s)}x(t_s) + \int_{t_s}^{t} e^{A(t-\tau)}B\tilde{u}(\tau)\,d\tau$ as the decoded output and the system states when there is no spike since time $t_s$, then according to the relationship between $u(t)$ and $r(t)$, $o(t)$, i.e., Eq. (2.3), we have

\[
\begin{align*}
    u(t) &= \tilde{u}(t) + \frac{1}{\lambda_d} \Gamma e^{-\lambda_d(t-t_s)} \lambda_d \bar{e}_k + \Omega_k o_k(t) \\
    &= \tilde{u}(t) + e^{-\lambda_d(t-t_s)} \Gamma_k + \Omega_k o_k(t),
\end{align*}
\]

where $\Gamma_k$ is the $k^{th}$ column of $\Gamma$ while $\Omega_k$ is the $k^{th}$ column of $\Omega$. Similarly, by Eq. (2.2), we obtain

\[
\begin{align*}
    x(t) &= \tilde{x}(t) + \int_{t_s}^{t} e^{A(t-\tau)}B e^{-\lambda_d(t-t_s)} \Gamma_k + B\Omega_k o_k(\tau)\,d\tau \\
    &= \tilde{x}(t) + e^{-\lambda_d(t-t_s)} \left( \int_{t_s}^{t} e^{\lambda_d(\tau-t_s)} e^{A(t-\tau)}\,d\tau \right) B\Gamma_k + \int_{t_s}^{t} e^{A(t-\tau)}B\Omega_k o_k(\tau)\,d\tau \\
    &= \tilde{x}(t) + e^{-\lambda_d(t-t_s)} \left( \int_{0}^{t-t_s} e^{(A+\lambda_d)\tau} \,d\tau \right) B\Gamma_k + e^{A(t-t_s)}B\Omega_k.
\end{align*}
\]

In summary, when there is a new spike from the $k^{th}$ neuron at time $t_s$, the firing rate, decoded output and system states have sudden changes as

\[
\begin{align*}
    r(t) &\rightarrow r(t) + h(t-t_s) \lambda_d \bar{e}_k \\
    u(t) &\rightarrow u(t) + h(t-t_s) \Gamma_k + \Omega_k o_k(t) \\
    x(t) &\rightarrow x(t) + h(t-t_s) H(t-t_s) B\Gamma_k + e^{A(t-t_s)}B\Omega_k,
\end{align*}
\]

(A.1)

where

\[
\begin{align*}
    h(t) &= e^{-\lambda_d t} 1(t) \\
    H(t) &= \int_{0}^{t} e^{(A+\lambda_d)\tau} \,d\tau,
\end{align*}
\]

113
where \(1(t)\) denotes the unit Heaviside function. For convenience, from this point onwards, we will use \(h\) and \(H\) to denote \(h(t-t_s)\) and \(H(t-t_s)\), respectively.

With the above equations, the spiking assumption (3.6) can be translated into

\[
\int_{t_o}^{t_s+\epsilon} \|x_d - x - hH\Gamma_k - e^{A(t-t_s)}B\Omega_k\|^2_2 + \nu\|r + h\lambda_d\hat{e}_k\|_1 + \mu\|r + h\lambda_d\hat{e}_k\|^2_2 d\tau \\
< \int_{t_o}^{t_s+\epsilon} \|x_d - x\|^2_2 + \nu\|r(\tau)\|_1 + \mu\|r(\tau)\|^2_2 d\tau.
\]

With the definitions of \(\ell_1\) and \(\ell_2\) norms, we get

\[
\int_{t_s}^{t_s+\epsilon} 2h\Gamma_k^T B^T H^T (x_d - x) + h^2\Gamma_k^T B^T H^T HBG_k - 2\Omega_k^T B^T e^{A^T(t-t_s)} (x_d - x) \\
+ 2h\Gamma_k^T B^T e^{A(t-t_s)} B\Omega_k + \Omega_k^T B^T e^{A^T(t-t_s)} e^{A(t-t_s)} B\Omega_k + \nu h\lambda_d + 2\mu h\lambda_d\hat{e}_k^T r + \mu h^2\lambda_d^2 d\tau < 0.
\]

Note that \(h(t-t_s) = e^{-\lambda_d(t-t_s)} = 0\) and \(e^{A(t-t_s)} = 0\) for \(\tau < t_s\), and rearrange the inequality to obtain

\[
\int_{t_s}^{t_s+\epsilon} 2h\Gamma_k^T B^T H^T (x_d - x) + 2\Omega_k^T B^T e^{A^T(t-t_s)} (x_d - x) - 2\mu h\lambda_d\hat{e}_k^T r d\tau \\
> \int_{t_s}^{t_s+\epsilon} h^2\Gamma_k^T B^T H^T HBG_k + 2h\Gamma_k^T B^T H^T e^{A(t-t_s)} B\Omega_k + \Omega_k^T B^T e^{A^T(t-t_s)} e^{A(t-t_s)} B\Omega_k + \nu h\lambda_d + \mu h^2\lambda_d^2 d\tau.
\]

By examining \(\epsilon \ll \lambda_d\) into the future, we can then approximate the integrands as constants so that (using \(h(t-t_s) \approx 1\), \(H(t-t_s) \approx 0\) and \(e^{A(t-t_s)} \approx I\) for \(\tau - t_s \approx \epsilon\))

\[
\Omega_k^T B^T (x_d - x) - \mu \lambda_d \hat{e}_k^T r > \frac{\Omega_k^T B^T B\Omega_k + \nu \lambda_d + \mu \lambda_d^2}{2}.
\]

114
Defining

\[ v_k(t) \equiv \Omega_k^T B^T (x_d - x) - \mu \lambda_d \tilde{e}_k^T r \]
\[ \tilde{v}_k \equiv \frac{\Omega_k^T B^T B \Omega_k + \nu \lambda_d + \mu \lambda_d^2}{2}, \]

the spiking rule becomes

\[ v_k > \tilde{v}_k. \]

This implies that when \( v_k(t) \) is larger than \( \tilde{v}_k \), the \( k^{th} \) neuron fires a spike, thus decreasing the value of the cost function.

It now remains to deduce the differential form of the dynamics on the latent variable \( v_k(t) \). With \( V = (v_1, \ldots, v_N) \), we can write

\[ V(t) = \Omega^T B^T (x_d(t) - x(t)) - \mu \lambda_d r(t). \] (A.2)

Let \( e(t) = x_d(t) - x(t) \) and take derivatives of Eq.(A.2), we could get that

\[ \dot{V}(t) = \Omega^T B^T (\dot{x}_d(t) - \dot{x}(t)) - \mu \lambda_d \dot{r}(t). \]

Note that \( u(t) = \frac{1}{\lambda_d} \Gamma r(t) + \Omega o(t) \), and

\[ \dot{x}_d(t) = Ax_d(t) + c(t) \]
\[ \dot{x}(t) = Ax(t) + Bu(t) = Ax(t) + \frac{1}{\lambda_d} B \Gamma r(t) + B \Omega o(t) \]
\[ \dot{r}(t) = -\lambda_d r(t) + \lambda_d o(t), \]
then,

\[ \dot{V}(t) = \Omega^T B^T (\dot{x}_d(t) - \dot{x}(t)) - \mu \lambda_d \dot{r}(t) \]

\[ = \Omega^T B^T (Ax_d(t) + c(t)) - \Omega^T B^T \left( Ax(t) + \frac{1}{\lambda_d} B \Gamma r(t) + B \Omega o(t) \right) - \mu \lambda_d (-\lambda_d r(t) + \lambda_d o(t)) \]

\[ = \Omega^T B^T A e(t) + \Omega^T B^T c(t) + \left( -\frac{1}{\lambda_d} \Omega^T B^T B \Gamma + \mu \lambda_d^2 I \right) r(t) - (\Omega^T B^T B \Omega + \mu \lambda_d^2 I) o(t). \]

This last step highlights the core difference in the network dynamics under the control objective versus the original predictive coding framework. Because \( \hat{x} \) and \( \dot{x} \) are both subject to the same (linear) dynamics in our case, the feedback error \( (\hat{x} - x) \) can be retained explicitly here.

With the definition in (2.11) and (2.12), the voltage differential equation can be finally written as

\[ \dot{V}(t) = \Omega^T B^T A e(t) + \Omega^T B^T c(t) + W^s_1 r(t) + W^f_1 o(t). \]
Appendix B

Proof of the Spiking Rule (3.7)-(3.9) of the Distributed Network

We adopt the methodology originally proposed in [24] to derive the threshold-based spiking rule of the distributed spiking network.

Proof. According to (3.1) and (3.3), the new spike at time $t^i_k$ from the $k^{th}$ neuron leads to instantaneous increment of the firing rate $r(t)$ and the decoded output of and $u(t)$:

$$
\begin{align*}
    r(t) &\rightarrow r(t) + h_\lambda (t-t^i_k)\lambda \bar{e}_k, \\
u(t) &\rightarrow u(t) + h_\Lambda (t-t^i_k)\Omega_k + \lambda h_\lambda (t-t^i_k)H(t-t^i_k)\Gamma_k,
\end{align*}
$$

where

$$
\begin{align*}
h_\lambda (t) &= e^{-\lambda t}1(t), \\
h_\Lambda (t) &= e^{\Lambda t}1(t), \\
H(t) &= \int_0^t e^{(\Lambda+\lambda)\zeta}d\zeta
\end{align*}
$$
where $1(t)$ is the unit Heaviside function which represents the current spikes only influence the future. From now on, we denote $h_\lambda(\tau-t^i_k), h_\Lambda(\tau-t^i_k)$, and $H(\tau-t^i_k)$ as $h_\lambda, h_\Lambda$ and $H$ for simplicity. With the above equations, we can interpret (3.6) as

$$\int_{t_o}^{t^i_k+\epsilon} \|u_* - u - h_\Lambda \Omega_k - \lambda h_\lambda H \Gamma_k\|^2_2 + v \|r + h_\lambda \lambda \bar{e}_k\|_1$$

$$+ \mu \|r + h_\lambda \lambda \bar{e}_k\|^2_2 \mathrm{d}\tau$$

$$< \int_{t_o}^{t^i_k+\epsilon} \|u_* - u\|^2_2 + v \|r(\tau)\|_1 + \mu \|r(\tau)\|^2_2 \mathrm{d}\tau.$$

We can move forward with definitions of norms, $\|z\|^2_2 = z^T z$ and $\|z\|_1 = \sum_i |z_i| = \sum_i z_i$ for positive firing rates,

$$\int_{t_o}^{t^i_k+\epsilon} -2\lambda h_\lambda \Gamma_k^T H^T (u_* - u) + \lambda^2 h_\lambda^2 \Gamma_k^T H^T H \Gamma_k$$

$$- 2\Omega_k^T h_\lambda^T (u_* - u) + \Omega_k^T h_\lambda \Omega_k$$

$$+ 2\lambda h_\lambda \Gamma_k^T H^T h_\Lambda \Omega_k + v h_\lambda \lambda + 2\mu h_\lambda \lambda \bar{e}_k^T r$$

$$+ \mu h_\lambda^2 \lambda^2 \mathrm{d}\tau < 0.$$

Since any spike cannot change the history, $h_\lambda(t-t^i_k) = 0$, $h_\Lambda(t-t^i_k) = 0$, and $H(t-t^i_k) = 0$ when $t < t^i_k$, then

$$\int_{t_k^i}^{t^i_k+\epsilon} 2\lambda h_\lambda \Gamma_k^T H^T (u_* - u) + 2\Omega_k^T h_\lambda^T (u_* - u)$$

$$- 2\mu h_\lambda \lambda \bar{e}_k^T \mathrm{d}\tau$$

$$> \int_{t_k^i}^{t^i_k+\epsilon} \lambda^2 h_\lambda^2 \Gamma_k^T H^T H \Gamma_k + \Omega_k^T h_\lambda \Omega_k$$

$$+ 2\lambda h_\lambda \Gamma_k^T H^T h_\Lambda \Omega_k + v h_\lambda \lambda + \mu h_\lambda^2 \lambda^2 \mathrm{d}\tau.$$
With $\varepsilon \ll \lambda$ noted in (3.6), we approximate $h_{\lambda}(t - t^i_k) \approx 1$, $h_{\lambda}(t - t^i_k) \approx I_m$ and $H(t - t^i_k) \approx 0$ for $t - t^i_k \sim \varepsilon$. Similarly, we approximate the integrands as constants to get

$$\Omega^T_k (u - u) - \mu \lambda \varepsilon^T_k r > \frac{\Omega^T_k \Omega_k + \nu + \mu \lambda^2}{2}.$$

We can rewrite the above spiking rule as

$$\nu_k > \bar{\nu}_k$$

by introducing the membrane voltage $\nu_k(t)$ and the threshold $\bar{\nu}_k(t)$ as

$$\nu_k(t) \equiv \Omega^T_k (u - u) - \mu \lambda \varepsilon^T_k r$$

$$\bar{\nu}_k \equiv \frac{\Omega^T_k \Omega_k + \nu + \mu \lambda^2}{2},$$

$\square$
Appendix C

Proof of the dynamics (4.5)

Proof. By taking derivatives of (3.10) and referring to equations (3.1), (3.3) and (3.4), we get

\[ V(t) = -\alpha V(t) + \alpha (\Omega^T u_0(t) - u(t)) - \mu\lambda r(t) \]
\[ + \Omega^T (\dot{u}_0(t) - \dot{u}(t)) - \mu\lambda \dot{r}(t) \]
\[ = -\alpha V(t) \]
\[ + \alpha \left( \Omega^T \left( u_0(t) - \frac{1}{\lambda} \Gamma a(t) - \frac{1}{\lambda} \Omega r(t) \right) - \mu\lambda r(t) \right) \]
\[ + \Omega^T (\dot{u}_0(t) + \Gamma a(t) + \Omega r(t) - \Gamma r(t) - \Omega o(t)) \]
\[ - \mu\lambda^2 (-r(t) + o(t)) \]
\[ = -\alpha V(t) + \Omega^T \dot{u}_0(t) + \alpha \Omega^T u_0(t) + \left( 1 - \frac{\alpha}{\lambda} \right) \Omega^T \Gamma a(t) \]
\[ + \left( \left( 1 - \frac{\alpha}{\lambda} \right) \Omega^T \Omega - \Omega^T \Gamma + \left( 1 - \frac{\alpha}{\lambda} \right) \mu\lambda^2 I \right) r(t) \]
\[ + \left( -\Omega^T \Omega - \mu\lambda^2 I \right) o(t) \]
Next, taking derivative of the nominal control signal (4.3) and considering the Riccati differential equation for the LQR problem, we can obtain

\[ \dot{u}_* (t) = \dot{u}_d (t) + R^{-1} B(t)^T P(t) e(t) + R^{-1} B(t)^T \dot{P}(t) e(t) \]
\[ + R^{-1} B(t)^T P(t) \dot{e}(t) \]
\[ = (R^{-1} B^T (t) P(t) - R^{-1} B^T (t) M(t)) e(t) \]
\[ + R^{-1} B^T (t) P(t) B(t) w(t) + \dot{u}_d (t) \]
\[ = (R^{-1} B(t)^T P(t) - R^{-1} B^T (t) M(t)) e(t) \]
\[ + R^{-1} B^T (t) P(t) B(t) u_d (t) + \dot{u}_d (t) \]
\[ - \frac{1}{\lambda} R^{-1} B^T (t) P(t) B(t) (\Gamma a(t) + \Omega r(t)) \]

Finally, embedding the above expression of \( \dot{u}_* (t) \) and (4.3) into the network dynamics results in

\[ \dot{V} (t) = - \alpha V(t) + \alpha \Omega^T R^{-1} B^T (t) P(t) e(t) \]
\[ + \Omega^T (R^{-1} B^T (t) P(t) - R^{-1} B^T (t) M(t)) e(t) \]
\[ + (\Omega^T R^{-1} B^T (t) P(t) B(t) + \alpha \Omega^T) u_d (t) \]
\[ + \Omega^T \dot{u}_d (t) \]
\[ + \left(1 - \frac{\alpha}{\lambda}\right) \Omega^T \Gamma a(t) \]
\[ - \frac{1}{\lambda} \Omega^T R^{-1} B^T (t) P(t) B(t) \Gamma a(t) \]
\[ + \left(1 - \frac{\alpha}{\lambda}\right) \Omega^T \Omega - \Omega^T \Gamma + \left(1 - \frac{\alpha}{\lambda}\right) \mu \lambda^2 I \right) r(t) \]
\[ - \frac{1}{\lambda} \Omega^T R^{-1} B^T (t) P(t) B(t) \Omega r(t) \]
\[ + (- \Omega^T \Omega - \mu \lambda^2 I) o(t). \]
Appendix D

Extension of Distributed SNN in Proposition 2

For a linear dynamic system with a measurement model

\[
\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t)
\]
\[
\hat{y}(t) = C\hat{x}(t),
\]

we can predict the output by a spiking network with a decoder

\[
\dot{\hat{y}}(t) = -\lambda \hat{y}(t) + \Gamma r(t) + \Omega o(t)
\]

And the spiking network has a threshold based spike rule which can be expressed as

\[
v_k(t) \equiv \Omega_k^T (\hat{y} - y) - \mu \lambda e_k^T r > \bar{v}_k \equiv \frac{\Omega_k^T \Omega_k + \nu \lambda + \mu \lambda^2}{2}.
\]
and the corresponding neuronal dynamics is
\[
\dot{V}(t) = \Omega^T CA(\Omega^T C)^{-1} V(t) + \Omega^T CBu(t) \\
+ \left( \frac{1}{\lambda} \Omega^T CA(\Omega^T C)^{-1} \Omega^T \Gamma + \Omega^T \Gamma \right) a(t) \\
+ \left( \mu \lambda \Omega^T CA(\Omega^T C)^{-1} \Omega^T \Omega \right. \\
+ \left. \Omega^T \Omega - \Omega^T \Gamma + \mu \lambda^2 I \right) r(t) \\
+ \left( -\Omega^T \Omega - \mu \lambda^2 I \right) o(t)
\] (D.1)
(D.2)
(D.3)
(D.4)

**Proof.** To substantiate the proposition, we begin by quantifying the effect of any added spike on the overall cost. Specifically, if the \(k^{th}\) neuron spikes at time \(t'_k\), then a delta function \(\delta(t - t'_k)\) is added to \(o_k(t)\), and then

\[
r(t) \rightarrow r(t) + h(t - t'_k) \lambda \bar{e}_k \\
y(t) \rightarrow y(t) + h(t - t'_k) \lambda H(t - t'_k) \Gamma_k + e^{-\lambda(t - t'_k)} \Omega_k
\]

where \(\Gamma_k\) is the \(k^{th}\) column of \(\Gamma\), \(\Omega_k\) is the \(k^{th}\) column of \(\Omega\), \(\bar{e}_k\) is the vector of all zeros except the \(k^{th}\) entry, and

\[
h(t) = e^{-\lambda t} \mathbf{1}(t) \\
H(t) = \int_0^t e^{-(\lambda + \lambda_d)\zeta} d\zeta = t,
\]

where \(\mathbf{1}(t)\) denotes the unit Heaviside function. For convenience, from this point afterwards, we will use \(h\) and \(H\) to denote \(h(\tau - t'_k)\) and \(H(\tau - t'_k)\), respectively.
With the above equations, the spiking assumption can be translated into

\[ \int_{t_o}^{t_i^j+e} ||\hat{y} - y - h\lambda HT_k - e^{-\lambda(t-t_k^j)}\Omega_k||^2 + v ||r + h\lambda e_k||_1 + \mu ||r + h\lambda e_k||_2^2 d\tau \]

\[ < \int_{t_o}^{t_i^j+e} ||\hat{y} - y||^2 + v ||r(\tau)||_1 + \mu ||r(\tau)||_2^2 d\tau. \]

With the norm \( ||z||^2 = z^T z \) and \( ||z||_1 = \sum_i |z_i| = \sum_i z_i \) for positive firing rates, we get

\[ \int_{t_k^i}^{t_k^{i+e}} -2h\lambda \Gamma^T_k HT_k (\hat{y} - y) + h^2 \lambda^2 \Gamma^T_k HT_k \Gamma_k - 2\Omega^T_k e^{-\lambda(t-t_k^j)} (\hat{y} - y) \]

\[ + 2h\lambda \Gamma^T_k HT_k e^{-\lambda(t-t_k^j)} \Omega_k + \Omega^T_k e^{\lambda(t-t_k^j)} e^{-\lambda(t-t_k^j)} \Omega_k \]

\[ + vh\lambda + 2\mu h\lambda e^T_k r + \mu h^2 \lambda^2 d\tau < 0. \]

Note that \( h(t - t_k^j) = e^{-\lambda_d(t-t_k^j)} = 0 \) and \( e^{\lambda(t-t_k^j)} 1(t - t_k^j) = 0 \) for \( t < t_k^i \), and rearrange the inequality to obtain

\[ \int_{t_k^i}^{t_k^{i+e}} 2h\lambda \Gamma^T_k HT_k (\hat{y} - y) + 2\Omega^T_k e^{-\lambda(t-t_k^j)} (\hat{y} - y) - 2\mu h\lambda e^T_k rd\tau \]

\[ > \int_{t_k^i}^{t_k^{i+e}} h^2 \lambda^2 \Gamma^T_k HT_k \Gamma_k + 2h\lambda_d \Gamma^T_k HT_k e^{-\lambda(t-t_k^j)} \Omega_k \]

\[ + \Omega^T_k e^{-\lambda(t-t_k^j)} e^{-\lambda(t-t_k^j)} \Omega_k + vh\lambda + \mu h^2 \lambda^2 d\tau. \]

By examining \( \epsilon \ll \lambda_d \) into the future, we can then approximate the integrands as constants so that (using \( h(t - t_k^j) \approx 1, H(t - t_k^j) \approx 0 \) and \( e^{-\lambda(t-t_k^j)} \approx I \) for \( t - t_k^j \sim \epsilon \))

\[ \Omega^T_k (\hat{y} - y) - \mu \lambda e^T_k r > \frac{\Omega^T_k \Omega_k + v\lambda + \mu \lambda^2}{2}. \]
Defining

\[ v_k(t) \equiv \Omega_k^T (\hat{y} - y) - \mu \lambda \hat{e}_k^T r \]

\[ \bar{v}_k \equiv \frac{\Omega_k^T \Omega_k + \nu \lambda + \mu \lambda^2}{2}, \]

the spiking rule becomes

\[ v_k > \bar{v}_k. \]

This implies that when \( v_k(t) \) is larger than \( \bar{v}_k \), the \( k^{th} \) neuron fires a spike, thus decreasing the value of the cost function.

It now remains to derive the differential form of the dynamics on the latent variable \( v_k(t) \). With \( V = (v_1, \cdots, v_N) \), we can write

\[ V(t) = \Omega^T (\hat{y}(t) - y(t)) - \mu \lambda r(t) \]

\[ = \Omega^T C \hat{x}(t) - \Omega^T y(t) - \mu \lambda r(t) \]

Note that we can estimate \( \hat{x}(t) \) by taking pseudo-inverse of \( \Omega^T C \) for both sides of the above equation

\[ \hat{x}(t) = (\Omega^T C)^{-1} V(t) + \mu \lambda (\Omega^T C)^{-1} r(t) + (\Omega^T C)^{-1} \Omega^T y(t). \]

Then taking derivatives of \( V(t) \), we could get that

\[ \dot{V}(t) = \Omega^T (C \hat{x}(t) - \hat{x}(t)) - \mu \lambda \dot{r}(t). \]
Note that

\[
\dot{x}(t) = A\dot{x}(t) + Bu(t) \\
= A(\Omega^T C)^{-1}V(t) + \mu \lambda A(\Omega^T C)^{-1}r(t) + A(\Omega^T C)^{-1}\Omega^T y(t) + Bu(t) \\
= A(\Omega^T C)^{-1}V(t) + \mu \lambda A(\Omega^T C)^{-1}r(t) + \frac{1}{\lambda} A(\Omega^T C)^{-1}\Omega^T (\Gamma a(t) + \Omega r(t)) + Bu(t) \\
\dot{y}(t) = -\lambda y(t) + \Gamma r(t) + \Omega o(t) \\
= -\Gamma a(t) - \Omega r(t) + \Gamma r(t) + \Omega o(t) \\
\dot{r}(t) = -\lambda r(t) + \lambda o(t),
\]

then,

\[
\dot{V}(t) = \Omega^T C \left( A(\Omega^T C)^{-1}V(t) + \mu \lambda A(\Omega^T C)^{-1}r(t) + \frac{1}{\lambda} A(\Omega^T C)^{-1}\Omega^T (\Gamma a(t) + \Omega r(t)) + Bu(t) \right) \\
+ \Omega^T (\Gamma a(t) + \Omega r(t) - \Gamma r(t) - \Omega o(t)) - \mu \lambda (-\lambda r(t) + \lambda o(t)) \\
= \Omega^T CA(\Omega^T C)^{-1}V(t) + \Omega^T CBu(t) \\
+ \left( \frac{1}{\lambda} \Omega^T CA(\Omega^T C)^{-1}\Omega^T \Gamma + \Omega^T \Gamma \right) a(t) \\
+ \left( \mu \lambda \Omega^T CA(\Omega^T C)^{-1} + \frac{1}{\lambda} \Omega^T CA(\Omega^T C)^{-1}\Omega^T \Omega + \Omega^T \Omega - \Omega^T \Gamma + \mu \lambda^2 I \right) r(t) \\
+ (-\Omega^T \Omega - \mu \lambda^2 I) o(t).
\]
Appendix E

Iterative SNN for ILC

Based on this objective function (5.10) and the spiking assumption, we can deduce the dynamics of the iterative SNN. We begin by quantifying the effect of any added spike on the overall cost. Specifically, if the $k^{th}$ neuron spikes at time $t_k$, then a delta function $\delta(t - t_k)$ is added to $o_k(t)$, and then

\[
\begin{align*}
    r_j(t) &\rightarrow r_j(t) + h(t - t_k^i)\lambda \tilde{e}_k \\
    R_j(t) &\rightarrow R_j(t) + h(t - t_k^i)\lambda \tilde{e}_k \\
    u_j(t) &\rightarrow u_j(t) + h(t - t_k^i)\lambda \Omega_k
\end{align*}
\]

where $\Gamma_k$ is the $k^{th}$ column of $\Gamma$, $\Omega_k$ is the $k^{th}$ column of $\Omega$, $\tilde{e}_k$ is the vector of all zeros except the $k^{th}$ entry, and $h(t) = e^{-\lambda t}1(t)$ where $1(t)$ denotes the unit Heaviside function. For convenience, from this point afterwards, we will use $h$ to denote $h(\tau - t_k)$ and use $\hat{u}$, $u$, $R$ to denote $\hat{u}_j$, $u_j$ and $R_j$, respectively.

With the above equations, the spiking assumption can be translated into

\[
\|\hat{u} - u - h\lambda \Omega_k\|_2^2 + \nu \|R + h\lambda \tilde{e}_k\|_1 + \mu \|R + h\lambda \tilde{e}_k\|^2 \leq \|\hat{u} - u\|^2_2 + \nu \|R(\tau)\|_1 + \mu \|R(\tau)\|^2_2.
\]
With the norm $\|z\|_2^2 = z^T z$ and $\|z\|_1 = \sum_i z_i = \sum_i z_i$ for positive firing rates, we get

$$2h\lambda \Omega_k^T (\hat{u} - u) - 2\mu h\lambda \bar{e}_k^T R > h^2 \lambda^2 \Omega_k^T \Omega_k + \nu h\lambda + \mu h^2 \lambda^2.$$ 

By examining $\epsilon \ll \lambda$ into the future, we can then approximate the integrands as constants so that (using $h(\tau - t_k) \approx 1$ for $\tau - t_k \sim \epsilon$)

$$\lambda \Omega_k^T (\hat{u} - u) - \mu \lambda \bar{e}_k^T R > \frac{\lambda^2 \Omega_k^T \Omega_k + \nu \lambda + \mu \lambda^2}{2}.$$ 

Defining

$$v_k(t) \equiv \lambda \Omega_k^T (\hat{u} - u) - \mu \lambda \bar{e}_k^T R$$

$$\bar{v}_k \equiv \frac{\lambda^2 \Omega_k^T \Omega_k + \nu \lambda + \mu \lambda^2}{2},$$

the spiking rule becomes

$$v_k > \bar{v}_k.$$ 

This implies that when $v_k(t)$ is larger than $\bar{v}_k$, the $k^{th}$ neuron fires a spike, thus decreasing the value of the cost function.

It now remains to deduce the iterative form of the dynamics on the latent variable $v_k(t)$. With $V = (v_1, \cdots, v_N)$, we can write

$$V_j(t) = \lambda \Omega^T (\hat{u}_j(t) - u_j(t)) - \mu \lambda R_j(t).$$
Take the next iteration of $V_j(t)$, we could get that

$$V_{j+1}(t) = \lambda \Omega^T (\tilde{u}_{j+1}(t) - u_{j+1}(t)) - \mu \lambda R_{j+1}(t).$$

Note that

$$\tilde{u}_{j+1}(t) = \tilde{u}_j(t) + w_j(t)$$
$$R_{j+1}(t) = R_j(t) + r_j(t)$$
$$u_{j+1}(t) = \Omega R_{j+1}(t) = \Omega (R_j(t) + r_j(t)) = u_j(t) + \Omega r_j(t),$$

then,

$$V_{j+1}(t) = \lambda \Omega^T (\tilde{u}_{j+1}(t) - u_{j+1}(t)) - \mu \lambda R_{j+1}(t)$$
$$= \lambda \Omega^T (\tilde{u}_j(t) + w_j(t) - u_j(t) - \Omega r_j(t)) - \mu \lambda (R_j(t) + r_j(t))$$
$$= \lambda \Omega^T (\tilde{u}_j(t) - u_j(t)) - \mu \lambda R_j(t) + \lambda \Omega^T w_j(t) - (\lambda \Omega^T \Omega + \mu \lambda I) r_j(t)$$
$$= V_j(t) + \lambda \Omega^T w_j(t) - (\lambda \Omega^T \Omega + \mu \lambda I) r_j(t).$$

And note that the solution indeed has the form of

$$V_j(t) = \lambda \Omega^T (\tilde{u}_j(t) - u_j(t)) - \mu \lambda R_j(t)$$
$$= \lambda \Omega^T (\tilde{u}_j(t) - \Omega R_j(t)) - \mu \lambda R_j(t)$$
$$= \lambda \Omega^T \tilde{u}_j(t) - (\lambda \Omega^T \Omega + \mu \lambda I) R_j(t).$$
References


[40] Levi DeVries, Francis D Lagor, Hong Lei, Xiaobo Tan, and Derek A Paley. Distributed flow estimation and closed-loop control of an underwater vehicle with a multi-modal artificial lateral line. *Bioinspiration & biomimetics*, 10(2):025002, 2015.


# Vita

**Fuqiang Huang**

| Degrees       | Ph.D. Systems Science and Mathematics, December 2019  
|               | M.S. Systems Science and Mathematics, December 2017  
|               | M.S. Automation, May 2010  
|               | B.S. Automation, July 2007  
| Professional Societies | IEEE Society  
|               | F. Huang, S. Ching (2019). Distributed spiking networks for nonlinear control.  

December 2019