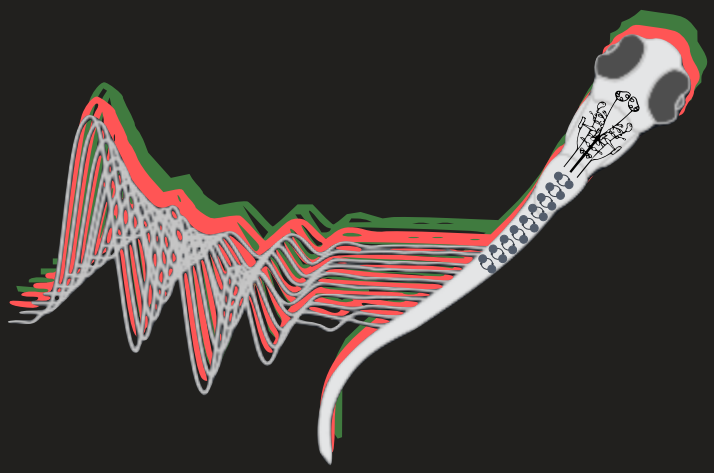
A Dynamical Systems Approach To Understanding Motor Control



Thomas Nathaniel
Soares Mullen

Dissertation presented to obtain the Ph.D degree in
International Neuroscience Doctoral Programme

Institute de Tecnologia Química e Biológica I Universidade Nova de Lisboa

Oeiras,
March, 2025

INSTITUTO
DE TECNOLOGIA
QUÍMICA E BIOLÓGICA
/UNL

Knowledge Creation

A Dynamical Systems Approach to Understanding Motor Control

Thomas Nathaniel Soares Mullen

Dissertation presented to obtain the Ph.D degree in International Neuroscience Doctoral Programme

Institute de Tecnologia Química e Biológica | Universidade Nova de Lisboa

Research work coordinated by: Michael Orger

Jury Panel: Dr Lea Duncker
Dr Tosif Ahamed
Dr James Fitzgerald
Prof. Alfonso Renart

Fundação Champalimaud

FCT
Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

FUNDAÇÃO CALOUSTE GULBENKIAN
Instituto Gulbenkian de Ciência

Oeiras,
March, 2025

A dynamical systems approach to understanding motor control

Thomas Nathaniel Soares Mullen

A Dissertation
Presented to the Faculty
of Universidade Nova de Lisboa
in Candidacy for the Degree
of Doctor of Philosophy

Supervisor: Michael B. Orger

March 2025

To Freedom

Acknowledgments

The past four years of my PhD have been a complex journey that I could not have navigated without the love, care, and support of those closest to me. As Mike once said during my uncertain days, "A PhD is like walking into the woods with the expectation of exiting in one place, only to find yourself leaving from a completely different spot." Either way - thank God I'm out!

I would like to give a special thank you to:

Mike, my supervisor, for his patience and support in all the decisions I made. I truly valued the freedom you gave me to explore the field of neuroscience.

Adrien, my mentor, who was always available to offer advice and guidance on both my research and future plans.

Marine, one of my closest collaborators and a good friend. It has been great working with you, and I've learned so much from you.

My collaborators, who supported me with their data and scientific insights.

A big thanks to the Zenith ETN. Starting my PhD during COVID and in a field I had never worked in before was daunting, but this network provided me with a community of fellow PhD students and PIs where I could discuss both personal and academic challenges. To my fellow students - Tahnee, Verity, Gautam, Shuhong, Giulia, Eduard, and Sharbat - I always felt like I could call on you for help. I am also deeply grateful to Claire and Joana for leading this network and for offering support and solutions when I was struggling.

The Imbizo network was a pivotal experience that kept me engaged in neuroscience. It reinforced the importance of diversity in science and

representation while also expanding my network of scientists and friends across the world.

The Science on the Wall network - I have loved being part of a team that fought for equality in science and reached out to the Cova da Moura community, which is far too often marginalized in Portugal. In particular - Rory, Charlotte, Violetta, Marga, Denise, Vitinho, Virus and Giuliano. As a minority in science, and within the CCU, I deeply connected with this initiative.

I would also like to thank Elena, who has lived with me for the past six months and supported me in every possible way during this tough period of finishing my PhD.

And finally, to my family - Mum and Dad, and my brothers Jack, Ollie, and Archie. During the difficult times when I questioned my path, they always supported me, no matter what decision I made.

Título

Uma abordagem de sistemas dinâmicos para compreender o controlo motor

Resumo

Uma das principais funções do cérebro é o controlo dos movimentos. Nas larvas de peixe-zebra, os movimentos são controlados por entradas descendentes do tronco cerebral para a medula espinal. A forma como estas entradas controlam os diferentes parâmetros do movimento não é completamente compreendida. As abordagens categóricas anteriores limitam a capacidade de modelar eficazmente os esquemas de controlo contínuo. Aqui, adoptamos uma abordagem da teoria do controlo ótimo para a modelação comportamental e defendemos que os movimentos podem ser entendidos como a saída de um sistema dinâmico controlado. Utilizando um rico conjunto de dados comportamentais, o nosso objetivo é identificar os sinais de controlo latentes e a dinâmica subjacente que constituem o repertório locomotor das larvas de peixe-zebra. Aprender a dinâmica de um sistema que funciona com dados não observados é um problema difícil de identificação de sistemas. Abordamos este último usando o método iLQR-VAE, proposto por Schimel, M., et al. (ICLR, 2022), que combina inferência variacional com controle de feedback ideal para aprender um sistema dinâmico latente orientado por entrada que captura a saída do comportamento. Os nossos resultados demonstram que este repertório pode ser eficazmente gerado através de um sinal de controlo esparso que conduz a uma RNN latente. Além disso, o espaço latente da rede codifica as caraterísticas cinemáticas da natação e preserva as estruturas das categorias de movimentos previamente definidas. Curiosamente, verificamos que a dinâmica latente para a geração de movimentos da cauda é partilhada com a dinâmica da barbatana peitoral para determinados tipos de bout. Esta estrutura permitiu-nos compreender como o espaço de controlo

mudou para reconstruir o comportamento que foi exposto a perturbações farmacológicas, fornecendo um método para avaliar como o controlo foi atribuído a drogas que podem alterar a função neural. Curiosamente, a transparência do peixe-zebra permite medições da atividade neural em todo o cérebro. Embora estas medições exijam normalmente que o peixe esteja com a cabeça fixa, o que limita o seu repertório locomotor, pudemos aqui utilizar o sistema dinâmico treinado no comportamento de natação livre e efetuar inferências sobre os movimentos com a cabeça fixa sob a restrição dessa dinâmica. Isto permitiu-nos comparar o espaço de controlo dos movimentos de cabeça fixa com o comportamento de natação livre. Realizámos análises preliminares para explorar a relação entre os sinais de controlo inferidos e a atividade neuronal. A nossa abordagem baseada em modelos oferece um novo paradigma para a separação da atividade neuronal supra-espinal de peixes fixados à cabeça com base em parâmetros do comportamento naturalista de natação livre.

Abstract

A key function of the brain is to control movements. In larval zebrafish, movements are controlled by descending input from the brainstem to the spinal cord. The way these inputs control different parameters of movement is not completely understood. Prior categorical approaches limit the ability to model continuous control schemes effectively. Here, we take an optimal control theory approach to behavioral modeling and argue that movements can be understood as the output of a controlled dynamical system. Using a rich behavioral dataset, our goal is to identify both the latent control signals and the underlying dynamics that make up the locomotor repertoire of the zebrafish larvae. Learning the dynamics of a system driven by unobserved inputs is a challenging system identification problem. We addressed the latter using the iLQR-VAE method, proposed by Schimel, M., et al. (ICLR, 2022), which combines variational inference with optimal feedback control to learn a latent input-driven dynamical system that captures behavior output. Our findings demonstrate that this repertoire can be effectively generated through a sparse control signal driving a latent RNN. In addition, the network's latent space encodes kinematic swim features and preserves structures of previously defined bout categories. Interestingly, we see that the latent dynamics for generating tail movements are shared with pectoral fin dynamics for certain bout types. The generative nature of this framework provided a method to evaluate how inferred control attributed to behavioral changes under pharmacological manipulations. Interestingly, zebrafish transparency allows for measurements of brain-wide neural activity. While these typically require the fish to be head-fixed, which limits their locomotor repertoire, we were able here to use the dynamical system trained on free-swimming behavior, and perform inference on head-restrained movements under the constraint of those dynamics. This allowed us to compare the control space of head-fixed movements with free-swimming behavior. We con-

ducted preliminary analyzes to explore the relationship between inferred control signals and neuronal activity. Our model-based approach offers a new paradigm for uncovering supraspinal neuronal activity from head-fixed fish based on parameters from naturalistic free-swimming behavior.

Author Contributions

TSM wrote the manuscript that is included in this thesis (Mullen et al., 2024). TSM also developed several open source software packages that were used in the analysis and data visualization. This included the `DiffiLQRAX` package for differential optimal control (Mullen & Schimel, 2025), and Swimulator for simulating zebrafish swimming (Mullen, 2024).

Financial Support

This work was carried out under the International Neuroscience Doctoral Programme (INDP), funded by the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement 813457. Michael Orger's group received support from the European Research Council (ERC NEUROFISH 773012) and Volkswagen Stiftung "Life?" initiatives.

Fish husbandry was supported by the Champalimaud Foundation Fish Facility, which received support from the research infrastructure CONGENTO, co-financed by the Lisboa Regional Operational Programme (Lisboa2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF), and the Fundação para a Ciência e Tecnologia (Portugal) under the project LISBOA-01- 0145-FEDER-022170.

Overview of the Thesis

This thesis is structured into six chapters. The first chapter provides an introduction to motor control. It outlines a review of the different perspectives of motor control and neural basis, and provides a brief overview of the methods to learn dynamical systems from data. The chapter also introduces the zebrafish as a model organism for studying motor control in neuroscience.

The second chapter presents the foundation of optimal control and a Python package that implements a differentiable version of the iterative Linear Quadratic Regulator (iLQR) algorithm.

The next two chapters, three and four, present the application of the iLQR integrated with Variational Autoencoder (VAE) to learn a sparse control model for zebrafish locomotion. The third chapter focuses on interpreting the sparse control space and relating it to environment and other features of behavior. While the fourth chapter concentrates on interpreting the dynamics of the network and understanding how the network computes different movements. We show how the control drives different dynamics for swim types and how it is modulated by pharmacological perturbations.

The fifth chapter applies the models from chapter three to neural data and relates the control space to neural activity patterns in the hindbrain.

The sixth, and final, chapter provides a summary discussion of the thesis and outlines future directions for the work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Chapter Summary

This chapter provides an introduction to the motor system. In Section 1.2, we outline the structural organization of the motor system, describing key regions in the brain and nervous system responsible for controlling and generating behavior. In Section 1.3, we outline the different perspectives on modeling how neural responses shape movement and the impact it has had on understanding the neural computation in generating movement. Then, Section 1.4 discusses the idea of optimal control and how moving the body to achieve a goal can be framed as a control problem. Section 1.5 addresses locomotion and how rhythmic behaviors are orchestrated from central pattern generators and the brainstem control that shapes these patterns. Afterward, Section 1.6 describes different computational methods to learn these patterns of behavior as dynamical systems, including the framework central to this thesis, iLQR-VAE. Then, in Section 1.7, we discuss the larval zebrafish as a biological model in neuroscience and our understanding of the underlying neural basis for locomotion. Finally, we outline the thesis objectives in Section 1.8 and bridge the dynamical systems framework to behavior in the larval zebrafish.

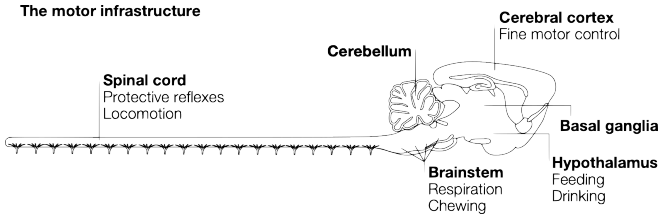


Figure 1.1. The neuronal organization underpinning control of locomotion in vertebrates. This schematic was adapted from (Grillner, 2003).

1.2 Control of our motor system

To survive, it is essential to interact with the world, and we do so primarily through movement. Acting appropriately requires us to integrate sensory information and select the correct motor program through our motor system. The generation of movement relies on the coordinated contraction of muscle ensembles and is mediated through motor neurons. Strikingly, the spinal cord can generate stereotyped motor patterns, even when input from the brain is removed (Brown, 1911; Mori, 1987). However, these movements deteriorate when faced with perturbations or tasks that require integration of sensory information from the surroundings (MacKay-Lyons, 2002). This suggests that the brain has an important role to control movements in a constantly changing environment.

Control is one of the brain's key roles in the motor system. It is crucial for performing and coordinating complex tasks. There are two forms of control: feedback (Scott, 2004), and feedforward (Wolpert et al., 1995). Feedback involves adjusting our actions based on sensory receptors measuring the outcome. Feedforward, which is more proactive, allows the body to anticipate and predict the sensory outcome before it occurs.

1.2.1 Organization of the nervous system for movement

The structure of the nervous system, essential for movement generation, is highly conserved in vertebrates. In general, the system is organized as shown in Figure 1.1, with the cortex, basal ganglia, and thalamus in the forebrain; the Diencephalic Locomotor Region (DLR) and Mesencephalic Locomotor Region (MLR) in the midbrain; and the nucleus of the Medial Longitudinal Fasciculus (nMLF) and Reticulospinal (RS) neurons in the hindbrain, which descend to the spinal cord (Grillner & El Manira, 2020).

Motor Cortex The motor cortex has long-range direct projections to the spinal cord (directly activating motor neurons as well as brainstem interneurons Bhattacharjee et al. (2020); Pearson (2000)) and plays a role in precise movements. Its activity is heavily modulated during intricate movements or when altering a typical movement trajectory to overcome an obstacle.

Basal Ganglia The basal ganglia receives input from the cortex, thalamus, and dopamine system. It is involved primarily in the selection of actions, the association of actions with rewards, and the initiation of movement. It projects continuous inhibition to the MLR and DLR until disinhibition of these regions allows a specific pattern of behavior to be generated.

Diencephalon & Mesencephalon These regions are responsible for initiating and maintaining repetitive locomotion. Spontaneous actions and simple goal-directed behaviors, such as foraging, are regulated in the DLR region (Squire et al., 2008). The DLR and MLR project excitatory outputs to RS neurons in the brainstem (Ryczko et al., 2015). Importantly, different input gains for MLR modulate the walking gait from trot to gallop (Capelli et al., 2017; Grillner & El Manira, 2020).

Brain stem and spinal cord The brainstem and spinal cord are essential for driving basic motor programs in most vertebrates (Lemieux & Bretzner, 2019; Ryczko, 2022). RS neurons and the nMLF have excitatory projections that activate spinal Central Pattern Generator (CPG)s (Grillner, 1985).

Cerebellum The cerebellum is a densely packed crystalline structure of specialized neurons that plays an important role in motor coordination. It dynamically receives detailed information from the spinal cord about ongoing movements and therefore plays a crucial role in motor adaptation. The cerebellum is believed to predict likely errors and makes compensatory adjustments (Wolpert et al., 1998).

1.3 Neural response to motor output

The brain is composed of complex networks of neurons that communicate with each other to generate patterns of activity. These patterns of activity shape how information is processed and computed to generate behavior. Extensive work throughout the past century has focused on understanding how neurons output these patterns of activity across multiple scales and levels of abstraction; with efforts scaling from the biophysical Hodgkin-Huxley modeling action potentials to rate-based network models that capture rich dynamics of neural populations.

A key question in neuroscience is how neural responses are related to movement. The motor control community has long debated how the brain represents movement and encodes these representations. This debate has given rise to two distinct perspectives for understanding the mechanisms of neural population responses during motor tasks: the representational approach and the dynamical systems approach (Shenoy et al., 2013). Most of this debate has been largely focused on the motor cortex, specifically the

primary motor cortex (M1), which is responsible for generating motor commands and executing movements. However, the views have been adopted across different brain regions, scales, and functions, from the spinal cord to the visual system.

1.3.1 Representation view

The classical representational approach examines how neurons encode task parameters and how their tuning influences the accuracy of neural representation. Typically, the neural code in a specific region is characterized by the tuning of individual neurons. A tuning curve illustrates how a neuron's firing rate, $\mathbf{r}(\mathbf{t})$, depends on a particular set of parameters thought to be encoded in the brain,

$$r_n(t) = f_n(\text{param}_1, \text{param}_2, \dots) \tag{1.1}$$

These parameters can be sensory, cognitive, or motor features, such as the direction of movement (Georgopoulos et al., 1982), the orientation of a visual stimulus (Hubel & Wiesel, 1959), or the frequency of a sound (Goldberg & Brown, 1969). This paradigm has been explored from singular neurons (Georgopoulos et al., 1982) to population-wide neural encoding (Georgopoulos et al., 1986).

A significant success of this framework was demonstrated by Hubel and Wiesel (1959, 1968, 1962), showing that the firing rates of V1 neurons were largely inactive unless aligned with the retinal position and edge orientation of an object. This provided a tuning curve and offered a quantitative description of the relationship between neuronal activity and stimulus features. This paved the way for identifying the encoding related to distance (Riehle et al., 1994) and direction (Maunsell & Van Essen, 1983), amongst other modalities.

In the context of motor control, Georgopoulos et al. (1982, 1983, 1986) demonstrated that motor cortex cells exhibit strong directional preferences. These preferences are distributed across the population, allowing the decoding of motor output direction by measuring neuronal activity collectively.

Subsequent work within this framework demonstrated population encoding for various motor-related processes, including directional target prediction (Georgopoulos et al., 1989), directional transformations (Lurito et al., 1991), storing memories of stimulus target direction and planned movements (Smyrnis et al., 1992), sensory-motor transformations (Funahashi & Takeda, 2002), limb spatial location (Graziano et al., 2004), movement kinematics such as speed and direction (Moran & Schwartz, 1999; Tankus et al., 2009; Fu et al., 1995), and muscle activity (Crutcher & Alexander, 1990; Morrow et al., 2007).

However, the relationship between M1 neurons and the wide range of parameters they encode has made it difficult to reach a consensus on their functional role in motor control, unlike the more straightforward neural population encoding observed in the visual cortex. Furthermore, a key limitation of population encoding in this context is its inability to explain the temporal evolution of neural responses that lead to motor output (Churchland & Shenoy, 2007; Kaufman et al., 2016; Russo et al., 2018).

1.3.2 Dynamical systems view

A broader framework, inspired by engineering and physical sciences, uses dynamical systems to model the neural responses underlying motor outputs. Rather than identifying specific parameters tuned to the neural population response, this approach focuses on understanding how the activity

of the population varies over time, $\mathbf{r(t)}$, and corresponds to time-varying motor outputs, $\mathbf{m(t)}$,

$$\mathbf{m(t)} = \mathbf{\Phi}[\mathbf{r(t)}] \tag{1.2}$$

while the mapping function Φ captures all the non-linear interactions of circuits between the cortex and muscle activity. This approach bypasses the assumption of a direct one-to-one relationship between neural population activity and muscle forces by abstracting these interactions into a functional mapping. Notably, it complements the representational approach by embedding tuning properties within a broader dynamical framework. Under the dynamical systems model, the evolution of population activity is determined by its time derivative, $\dot{\mathbf{r}}$, scaled by time constant τ. The dynamics is governed by the current activity state, the local recurrent dynamics of the network (represented by $h()$) and external input $\mathbf{u(t)}$,

$$\tau\ \dot{\mathbf{r}}(\mathbf{t}) = \mathbf{h}(\mathbf{r(t)}) + \mathbf{u(t)} \tag{1.3}$$

External inputs provide flexibility to the framework, allowing it to represent intercommunication between neural populations (Russo et al., 2020; Shenoy & Kao, 2021), behavior and stimulus parameters (Churchland & Shenoy, 2007; Saxena et al., 2022), or optimal feedback control (Scott, 2004).

The dynamical systems framework has been able to describe multiple phases and transitions of the delay-reach task, which were limited by the representational approach (Churchland et al., 2012). Churchland and Shenoy (2007) showed the complex temporal activity patterns of M1 neurons during a delayed-reach behavioral task with slightly different reach speeds (Churchland, Yu, et al., 2006). In the population dynamics frame-

work, Churchland and Shenoy (2007) could recover the spatial tuning of preferred directions like Georgopoulos et al. (1982) but also predict the temporal patterns of activity averaged across similar types of reaches and speeds.

Modeling neural populations as a dynamical system, as shown in Equation (1.3), explains how neural state-space trajectories evolve over time, shaped by recurrent interactions that create flow fields in the activity space. Studies suggest that the intrinsic recurrent dynamics of the motor cortex act as a reservoir of rotations that form a basis set for generating complex patterns, which control muscle output during movement execution (Sussillo et al., 2015; Michaels et al., 2016; Vyas et al., 2020; Rouse et al., 2022). One hypothesis is that the role of external input is to initialize the neural state within the flow field, after which the recurrent dynamics generate the necessary rotations for the motor task (Sussillo et al., 2015). This initialization corresponds to the motor preparation period before the "go" cue in delay-reach tasks (Churchland, Santhanam, & Shenoy, 2006). Kaufman et al. (2014) described the dynamics of this phase as the neural population transitioning through a null-space manifold, priming the state for motor execution along the potent-space. The null-space consists of specific directions in neural state space where output variance remains constant, allowing trajectories to evolve in these directions without directly driving motor commands. Conversely, the potent-space is orthogonal to the null-space and represents directions where neural activity directly projects to motor output.

Another intriguing aspect is the large population activity of Condition-independent signals (CIS), which are recruited to transition from a preparatory to movement command, rather than in relation to an external parameter (Kaufman et al., 2016). CIS is a result of externally delivered trigger signals that drive a switch in neural dynamics to gen-

erate movement, that is, to drive the state to a fixed point with specific rotation dynamics (Sussillo et al., 2015). These rotational dynamics do not obviously map to representational or movement parameters, but can be seen as a basis set in constructing complex temporal patterns in generating muscle activity.

Exploration of the geometric properties of the neural population dynamics provides a language to understand the neural computation at hand in generating motor output (Durstewitz et al., 2023). The large dimensionality of M1 can mitigate 'tangling' - a phenomenon that arises in input-driven systems when trajectories overlap thus supporting stable and distinct motor outputs. Russo et al. (2018) quantified the tangling of population trajectories in the motor cortex and found that low tangling in M1 is crucial for robust output generation through internal dynamics. This low degree of tangling suggests that M1 primarily sustains and shapes internal dynamics, rather than directly encoding external inputs or muscle output (Russo et al., 2018; Pandarinath, Ames, et al., 2018; Saxena et al., 2022).

Separate from the delay-reach task, Russo et al. (2020) studied a cycling task and noticed a different type of population dynamics in the M1 that was not relevant for tracking the task. M1 neurons had high "divergence", meaning that similar initial states led to very different trajectory. Importantly, supplementary motor area (SMA) had low divergence and could track the closed-loop context. This population projects to M1, illustrating an alternative perspective in which the motor cortex continuously receives persistent inputs from other brain regions to guide motor output, rather than simply setting an initial state. This was also supported in works from Sauerbrei et al. (2019). These studies illustrate the importance of $\mathbf{u}(\mathbf{t})$ in Equation (1.3) encoding intra-regional communication in motor system.

1.4 Optimal Control in Motor System

Another class of dynamical system models used in motor control, separate from M1 responses, is optimal feedback control models. Optimal controllers have been used to understand how the brain governs motor control and coordination (Scott, 2004). This framework provides insights into how the brain functions as a controller to achieve task-related goals efficiently and robustly.

1.4.1 Control Theory and the Brain as a Feedback Controller

In control theory, the body's musculoskeletal system is modeled as a plant governed by physical laws. Applying a sequence of torques to the limbs generates trajectories that execute behavior for a given task. The brain is conceptualized as the controller that computes and generates these torques to achieve the desired trajectory.

To achieve task-related goals, the brain relies on feedback for control. A key framework for providing sensorimotor feedback within realistic timescales involves internal models (Wolpert et al., 1995; Kawato, 1999).

1. *Inverse models* reverse the causal flow, computing the motor command to achieve the desired state trajectory.

2. *Forward models* predict the sensory consequences of motor commands, allowing rapid optimization of control signals during motor preparation before they are executed (Churchland, Yu, et al., 2006).

1.4.2 Optimal Feedback Control: A Shift in Perspective

Optimal feedback control has become an influential framework in motor control, particularly due to its ability to explain trial-to-trial variability in motor tasks while consistently achieving goals (Todorov & Jordan, 2002).

Unlike earlier models that assumed the brain explicitly plans and controls precise trajectories, optimal feedback control suggests that hand or limb trajectories emerge naturally from the optimal control laws of a task. This means that variability arises because the controller ignores noise in the trajectory that does not interfere with task performance.

Optimal control provides a mathematical framework to find the optimal sequence of inputs $\mathbf{u}$ to the underlying dynamical system $\mathbf{x}(\mathbf{t})$, which minimizes a cost function $J(\mathbf{x}, \mathbf{u})$ that usually measures the goal of the task.

1.4.3 Key Features of Optimal Feedback Control

Optimal State Estimation Optimal feedback control is based on an accurate estimate of the current state of the system. This state estimate is generated from two sources: afferent feedback from sensory receptors and efferent copies of motor commands, which predict the sensory consequences of actions. The state estimate integrates information about the body, movements of the limbs, and external environment, allowing the controller to distinguish between errors that affect the performance of the task and those that do not (Scott, 2004).

Optimal Control Laws The controller transforms state variables (e.g., position, velocity, acceleration) into motor commands based on behavioral goals. This involves solving an optimization problem to adjust feedback gains in a way that minimizes a performance metric, such as the energy expenditure or time required to complete a task.

Minimal Intervention Principle An essential property of optimal feedback control is the "minimal intervention principle" (Todorov & Jordan, 2002). This principle states that only noise or errors that interfere with task performance are corrected, while redundant variations are ignored. This feature explains inter-trial variability in motor output: noise that does not compromise task goals is left uncorrected, leading to variability that appears random but is actually optimized for efficiency. It also ensures robustness in systems subject to noise in both motor outputs and sensory inputs (Scott, 2004).

1.4.4 Implications for Motor Coordination

Optimal feedback control highlights the brain's role as an adaptive and efficient controller, modifying feedback signals to achieve optimal performance metrics. In contrast, some dynamical system models portray the motor cortex as largely autonomous (Sussillo et al., 2015; Churchland et al., 2012), optimal control emphasizes that dynamics are input driven via sensory feedback integrated from other brain regions (Sauerbrei et al., 2019; Kalidindi et al., 2021) Bridging these ideas with the emergent dynamics described in Section 1.3.2, an optimal feedback controller may rely on neural population dynamics to implement or refine control laws. In this view, recurrent interactions in the motor cortex could serve as a substrate through which feedforward predictions and real-time feedback are integrated.

1.5 Locomotion

1.5.1 Rhythmic locomotion via central pattern generators

In the absence of sensory input or supraspinal control, the spinal cord can generate rhythmic patterns of movement, such as walking, swimming, and

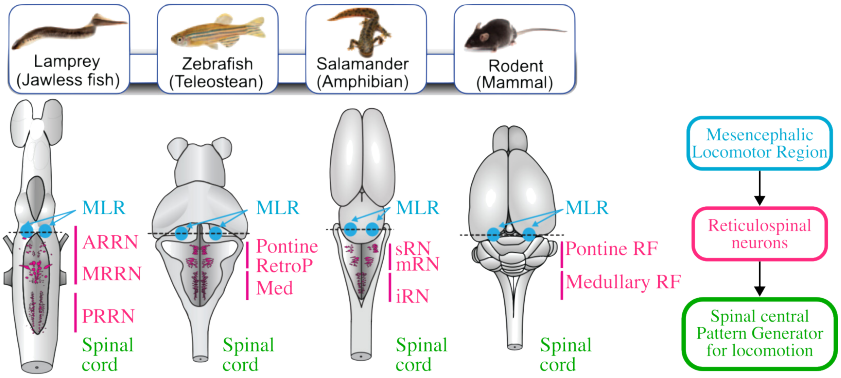


Figure 1.2. Conserved brainstem locomotor circuits across vertebrates. This schematic was adapted from (Ryczko, 2022).

crawling (Dubuc et al., 2023; Grillner & Zangger, 1975; Grillner, 1981). In both vertebrates and invertebrates, there are intrinsic networks of interneurons that produce distinct sequences of activity, which project to motor neurons while inhibiting neurons that would counteract these movements (Marder & Bucher, 2001). These networks, known as Central Pattern Generators (CPGs), are crucial for driving motor patterns (Grillner, 1985; Delvolvé et al., 1999; Stein, 2004; Cohen & Wallén, 1980).

CPGs in the spinal cord coordinate rhythmic movements and are activated by descending control from the brainstem (Selverston & Moulins, 1985). This descending control shapes how CPGs are activated and initiates the movement. Sensory feedback is used to regulate periodic movements (Katz, 1996), while for ballistic movements, predictive control of the model (feedback) is used to account for the timing constraints (Squire et al., 2008).

Dynamical systems had already been adopted to understanding low-level neural control and modeling the CPG in the spinal cord. CPGs generate rhythmic patterns that shape muscle activity to encode locomotion, such

as crawling, swimming, and walking. This phenomenon is conserved across vertebrates (Grillner, 1985; Delvolvé et al., 1999; Stein, 2004; Cohen & Wallén, 1980) and invertebrates (Berkowitz & Laurent, 1996; Marder et al., 2005). Although sensory feedback can modulate CPG activity, much of the pattern generation stems from intrinsic connectivity and descending reticulospinal drive.

1.5.2 Brainstem control of locomotion

Stimulating the spinal cord alone, without brain involvement, is sufficient to generate complex patterns of muscle activity and behavior. However, for locomotion to be appropriate for the environment, descending inputs from Reticulospinal (RS) neurons are required. These RS neurons relay locomotor commands from the MLR and nMLF. The midbrain and hindbrain regions are responsible for the initiation of movement and gait control (Orlovskiĭ et al., 1966). The Mesencephalic Locomotor Region (MLR) is a part of the midbrain and is responsible for initiating movement and controlling gait (Ml et al., 1966). Increased stimulation to the MLR has shown to increase the speed of the rhythmic motor pattern and eventually switch gaits. The MLR projects downstream to the reticulospinal neurons in the hindbrain, which relay locomotor commands to the spinal cord (Grillner, 2008).The MLR and reticulospinal neurons are structures widely conserved in vertebrates (Ryczko, 2022); see Figure 1.2.

1.6 Learning dynamics of behavior

Section 1.3.2 discussed the neural dynamics view in governing behavior. Dynamic equations are often generalized by:

$$\dot{x} = f(x, u) \tag{1.4}$$

$$y = g(x) \tag{1.5}$$

where x is the latent/neural state of the system, u is the control input, f is the dynamics function, and g is the output function where y is the readout of the observed behavior. A key challenge in understanding behavior is to learn the dynamics of the system from the observed data. Often this requires us to learn f and g from the data, and fit it to a set of parameters θ, such that

$$\dot{x} = f(x, u; \theta) \quad \text{and} \quad y = g(x; \theta) \tag{1.6}$$

It is often infeasible to measure all neural activity with high spatial and temporal resolution. Therefore one must look elsewhere, such as restricting measurements to a small regions of the brain, or strictly measuring the behavior of the animal. Advancements in behavioral tracking, e.g., SLEAP (Pereira et al., 2022) and DeepLabCut (Nath et al., 2019), have made it possible to acquire long behavior recordings with high dimensionality of the data. Furthermore, naturalistic behaviors can be captured with detailed posture recordings to track the dynamics of the system. With rich behavioral datasets and high throughput, new methods have been developed to learn the underlying dynamics of the data.

1.6.1 Data-driven methods

Delayed-embedding Delay embedding methods are commonly used to model time series of observations from systems whose underlying state is not fully observed. These methods model the dynamics in an augmented

space in which the state is extended to include multiple time points, and in which the dynamics are assumed to take a more simple form but can still be mapped to the dynamics of the original system (Takens, 2006). This approach was implemented in Ahamed et al. (2020), where postural time series data from C. Elegans were time-embedded to construct a highly predictive state space. However, delay embedding methods are often sensitive to the choice of embedding parameters (Tan et al., 2023). Moreover, this autonomous view of dynamics fails to consider the impact of unknown control signals that may arise either endogenously (e.g., hunger, motivation) or exogenously (e.g., oxygen levels, temperature), otherwise infer from the residuals (Fieseler, Zimmer, & Kutz, 2020; Brunton, Brunton, Proctor, Kaiser, & Kutz, 2017).

1.6.2 Model-based methods

Latent dynamical system models

Alternatively, one can take a generative approach and model observations as arising from a latent dynamical system, whose parameters can be learned from data.

Switching linear dynamical systems In the context of behavioral data modeling, switching Linear Dynamical System (LDS) models have been particularly popular and are at the core of some of the most widely used algorithms for behavioral segmentation (Datta et al., 2019). These models assume the existence of a hidden, discrete state that parametrizes the processes generating the data, with the system using different dynamics depending on the value of the hidden state. For instance, MoSeq (Datta et al., 2019) employs an auto-regressive generative process coupled with a sticky Hidden Markov Model (HMM). An alternative strategy employs adaptive segmentation algorithms that rely on statistical model testing

instead of explicit state transitions (Costa et al., 2019). Such models have been used to successfully segment behavioral time series in various contexts (Batty et al., 2019; Johnson et al., 2020), and relating switches in behavior to concurrent neural recordings has yielded interesting insights into the way the brain controls behavior (Markowitz et al., 2023). However, a potential limitation of switching LDS models is their reliance on multiple underlying dynamical systems, which complicates the interpretation of the learned dynamics.

Input-driven latent dynamics Another approach to modeling behavioral dynamics, which accounts for potential discontinuities caused by factors such as transient changes to the environment and changes in strategies from the brain, is to use input-driven latent dynamical systems models. However, learning these dynamics while simultaneously modeling the unobserved inputs driving them is a highly challenging and often degenerate problem. Two methods, LFADS (Pandarinath, O'Shea, et al., 2018) and iLQR-VAE (Schimel et al., 2022), have recently been proposed to address this issue and have shown promising results on various synthetic and real-world datasets. Both of these methods harness the framework of variational inference by formulating the learning problem as a Variational Autoencoder (VAE) (Kingma & Welling, 2013), and approximating the posterior distribution over unobserved inputs with a Gaussian distribution. The key difference between the two methods is the way in which the posterior inputs are computed: in LFADS, the posterior mean is amortized using a bidirectional Recurrent Neural Network (RNN), while iLQR-VAE uses iterative Linear Quadratic Regulator (iLQR) (Li & Todorov, 2004) to obtain the inputs most likely to have generated the observations, given the model parameters. Additionally, iLQR has been shown to work with several choices of priors over the inputs (and in particular with sparse priors), while LFADS has largely been used with a choice of auto-regressive

prior. Although to our knowledge, there have been very few examples of work modeling behavioral observations using such input-driven, latent dynamical models, we note that Wimalasena et al. (2022) used AutoLFADS (Keshtkaran et al., 2022) to model muscle activity in mice and monkeys. Importantly, however, they made the assumption that the muscles were driven by continuous auto-regressive inputs, as required by the LFADS formulation.

1.7 The larval zebrafish model

In the following section, we shift our attention from modeling strategies to a specific experimental model, larval zebrafish, where these tools and theoretical frameworks can be directly applied to investigate locomotor behavior and its neural basis.

1.7.1 Behavior

Larval zebrafish have long been studied as a biological model in motor system neuroscience due to their genetic accessibility and rapid development (Budick & O'Malley, 2000), allowing them to perform complex tasks (Dreosti et al., 2015; Marques et al., 2019; Markov et al., 2021; Petrucco et al., 2023; Yang et al., 2024). The larvae move in discrete episodes of propulsion known as swim bouts (see Figure 1.3), which typically last between 200-300 ms. Most of their behavior can be described by measuring tail angle along segments of tail; see Figure 1.4. In contrast to adult zebrafish, which is also studied and exhibits an anguilliform gait, larval zebrafish typically employs a carangiform gait (Agha et al., 2024). This form of locomotion concentrates undulatory movements mainly in the rear portion of the tail, thereby optimizing for efficient acceleration and propulsion (Kuo & Eliasmith, 2005).

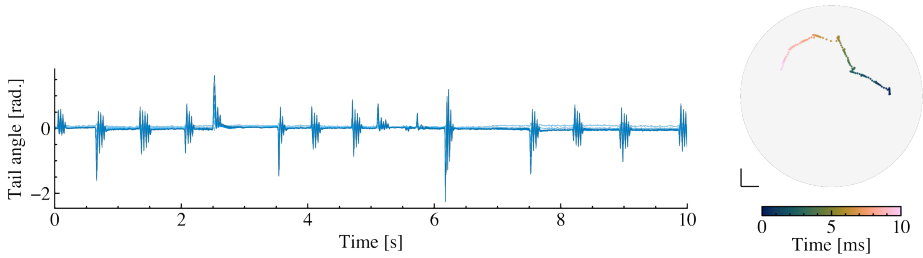


Figure 1.3. Larval zebrafish move with short tail bursts for stabilization. (Left) Tail angle tracked across segments during bouts. (Right) Fish x-y coordinates during tail movements, dots every 10 ms.

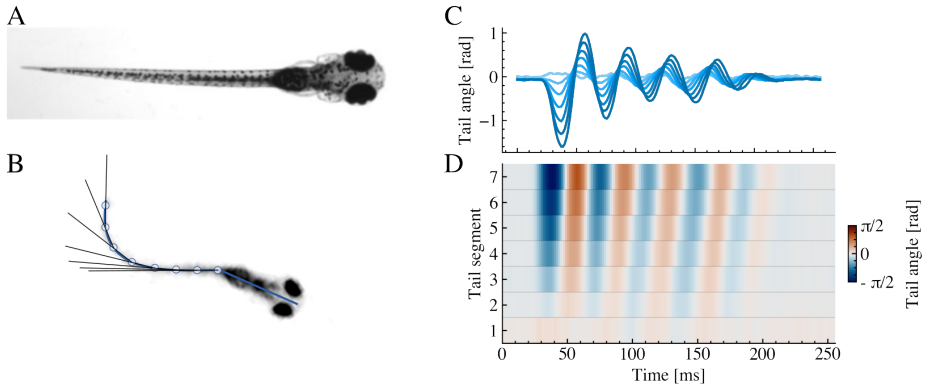


Figure 1.4. (A) Dorsal image of the posture of a 7 dpf larval zebrafish. (B) Behavior tracking software measures the segment angle with respect to the swim bladder. Seven tail segment angles captures a compressed representation of the image posture in (A). It is possible to acquire long tail recordings ($\sim$ 1hr) at fast acquisition (700 Hz) with the aid of online segment tracking extraction. (C) The angle of each track segment produces a time series. The ensemble of these segments is an abstraction of swim which reflects the behavior of the fish in time. (D) The kymograph representation of these tracked segments show coupling of the tail angle oscillations, and uncovers a group wave that propagates backwards along the fish. This explains how side-to-side tail oscillations generate a forward propulsion in the movement of the fish.

1.7.2 Neural basis of locomotion

A combination of genetic tools and optical techniques has enabled non-invasive neuronal recordings of the whole brain in larval zebrafish (Ahrens et al., 2013). Transgenic lines that specifically label hindbrain neurons, like nMLF and RS neurons, have provided insight into the functional role of neural circuits that control locomotion in zebrafish (Severi et al., 2014, 2018; Kimura et al., 2013). Through these techniques, early studies showed that Mauthner cells and their homolog pairs are responsible for initiating escape swims (O'Malley, Kao, & Fetcho, 1996). Since then, studies have continued to further our understanding of hindbrain neurons that modulate kinematic characteristics of movement, such as onset (Severi et al., 2018; Berg et al., 2023), amplitude of turn (Orger et al., 2008; Huang et al., 2013; Thiele et al., 2014), frequency of beat of the tail (Severi et al., 2014; Carbo-Tano et al., 2023), duration of swimming (Berg et al., 2023), and swimming speed (Kimura et al., 2013; Severi et al., 2014; Berg et al., 2023). Behavioral set-ups with closed-loop paradigms have been used to see how sensory inputs modulate behavioral response, and how sensory integration of hindbrain neurons determines the appropriate motor command for the fish to swim (Dunn et al., 2016; Severi et al., 2014).

In particular, this circuit identification approach to kinematic features is widely adopted in the larval zebrafish community. Although larval zebrafish do not have a motor cortex, the structures of the midbrain and hindbrain are conserved with those of other vertebrates, and kinematic feature analysis has helped develop our understanding of the neural basis for locomotor control (Carbo-Tano et al., 2023). However, these approaches are largely driven by kinematic features relevant to the task which are chosen subjectively based on perceived importance. A dynamical systems approach enables generative methods that can propose hypotheses for the underlying computation of locomotion (Vyas et al., 2020). Finally, in

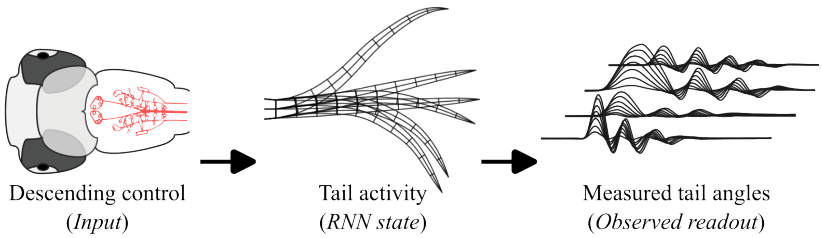


Figure 1.5. Model proposal: Input-driven dynamical system of behavior in the larval zebrafish.

the absence of a dynamical model, proposing hypotheses on how behavior adapts to perturbations or how actions are compensated for becomes a challenge.

1.8 Thesis objectives

The present work was designed to identify the behavior of the larval zebrafish as a dynamical system controlled by sparse input. We first outline the optimal control framework and an implementation of the differentiable iLQR algorithm in JAX, `DiffiLQRAX` (Mullen & Schimel, 2025). Optimal control is central to inferring which external inputs are necessary to generate desired behaviors in the zebrafish.

To learn the dynamics of the zebrafish, we used the iLQR-VAE framework. This fitted distributions over the measured tail angle of swims through a sparse, low-dimensional control space. This framework allowed us to learn a mapping across two different spaces: upstream control from the downstream tail dynamics; see Figure 1.5.

We aimed to make sense of the learned dynamics through the sparse control space. We evaluated how the sparse control mapped to kinematic features of movement linking representation of behavior. We also tested

if the control could predict features the model was not trained on, such as navigational features or dynamics of pectoral fin movements.

To understand the computation of the learned dynamics, we fitted the behavior using the iLQR-VAE with a linear network model. We used balanced model reduction to compress the underlying dynamics. We then explored how different dynamical components were recruited in movement generation. Using a dataset with pharmacological manipulation, we illustrated via model reduction that the learned dynamics were invariant to the perturbation despite changes in behavior. However, the control space showed significant differences in their sparsity to generate the same behavior type.

The final objective was to compare the inferred control space of the model to neuronal recordings in the hindbrain of larval zebrafish. We applied the model to free-swimming behavior and embedded head-restrained movements within this space to relate the findings to naturalistic behaviors.

Chapter 2

`DiffiLQRAX`: Efficient and scalable differentiable controller

2.1 Summary

In this chapter, we describe a non-linear control algorithm and its implementation, `DiffiLQRAX` , in Python using the JAX library. Section 2.3 outlines the mathematical framework for the package. Specifically, in Sections 2.3.1 and 2.3.2 we introduce the mathematical framework of the Linear Quadratic Regulator (LQR) problem, then we adapt the LQR problem for non-linear problems using the iterative Linear Quadratic Regulator (iLQR) method in Section 2.3.3. Finally, we introduce the theory of parallelizing sequential algorithms using associative dynamic programming in Sections 2.3.4 to 2.3.6.

In Section 2.4, we describe the high-level design structure of the software package and the use of the different modules to solve the LQR and iLQR problems. We illustrate in Sections 2.5.1 and 2.5.2 the solutions to the solvers are numerically sound and have been compared with other well-known methods to solve such problems. In Section 2.5.3, we compare the time profile with the sequential and parallel solvers. Then we demonstrated how the differentiable solvers can be integrated in common deep-learning tools such as VAEs with toy examples, in Section 2.5.4.

2.2 Introduction

The underlying principle of optimal control is to steer a system to a target with minimal cost. Thus, optimal control has an important role in systems stabilization, state feedback and probabilistic inference (Todorov, 2006; Schimel et al., 2022). This field of engineering has been applied to many areas, such as autonomous vehicles (Varma et al., 2020), robotics (Recht, 2019), earth science (Soldatenko & Yusupov, 2021; Minchala-Avila et al., 2015), and finance (Fotoohi Bafghi et al., 2022; Soner, 2004). Importantly, optimal control has seen traction in systems neuroscience (Castiñeiras & Renart, 2022; Hennequin et al., 2014).

The optimal control sequence can be efficiently obtained for problems with distinct structures, such as solving the Linear Quadratic Regulator (LQR) problem using quadratic programming to minimize a quadratic cost under a linear equality constraint. The challenge arises when the system is non-linear, and the cost function is non-convex. In this case, the optimal control sequence can be approximated using iterative methods, such as iterative Linear Quadratic Regulator (iLQR) (Li & Todorov, 2004) and Adaptive Dynamic Programming (ADP) (Bian et al., 2014). Automatic Differentiation (autodiff) and Graphical Processing Unit (GPU) scalability have enabled the development of efficient numerical solvers for optimal control problems.

The JAX (Bradbury et al., 2018) Python library is widely adopted in the ML community due to its Numpy-like syntax, GPU scalability, functional programming style with Just-In-Time (JIT) compilation, and autodiff. Unlike other frameworks such as TensorFlow and PyTorch, JAX allows users to customize low-level functions, such as defining specific backward and forward gradients. These features make JAX an ideal environment for developing and creating open source software (DeepMind et al., 2020).

Dynamic Programming (DP) can solve recursive problems with linear time complexity, like with the Hamilton-Jacobi-Bellman equation (Todorov, 2008). However, the sequential nature of recursive problems typically resists parallelization. Blelloch (1989) introduced associative scanning, a paradigm that enables parallelization of sequential algorithms providing the operator is associative. This approach has recently been applied in deep learning, including in linear RNNs (Gonzalez et al., 2024; Smith et al., 2023), Kalman smoothers (Sarkka & Garcia-Fernandez, 2021), and other networks (Feng et al., 2024).

We develop a differentiable, non-linear optimal control package, `DiffilQRAX`[1], which leverages the JAX library to solve optimal control problems. We follow the functional programming style and implement an LQR solver that solves the Riccati equations using DP. Exploiting JAX's autodiff capabilities, we implement an iLQR solver that approximates the optimal control sequence for non-linear systems. Iteratively solving the local LQR problem until a local optimum is achieved. There is an additional back-tracking line search feature for faster convergence. Both the LQR and iLQR solvers are fully differentiable, and gradients are implicitly obtained, reducing memory overhead due to excessive rollouts (Blondel et al., 2021; Schimel et al., 2022; Amos & Kolter, 2017). Differentiable solvers allow for integration with deep learning models and can be used as inference layers in variational auto-encoders (Kingma & Welling, 2013; Schimel et al., 2022). Finally, we implement an associative DP solver that parallelizes the Bellman equation (Sarkka & Garcia-Fernandez, 2023), enabling efficient computation of the optimal control sequence for large-scale problems to be scalable on GPU.

[1] https://diffilqrax.readthedocs.io

2.3 Background

In general, a control optimization problem consists of a cost function, $\mathcal{J}$, and a dynamical constraint, f_t, that describes how the state evolves at time t,

$$\underset{\{u_0,\dots,u_{T-1}\}}{\text{minimize}} \quad \mathcal{J}(x,u) = \ell_f(x_T) + \sum_{t=0}^{T-1} \ell_t(x_t,u_t) \tag{2.1}$$

$$\text{subject to} \quad x_{t+1} = f_t(x_t,u_t), \quad t = 0,\dots,T-1 \tag{2.2}$$

$$x_0 = x_{\text{init}} \tag{2.3}$$

where $\ell_t(\cdot,\cdot)$ is the momentary cost function at time t for a given input $u_t \in \mathbb{R}^m$ and state $x_t \in \mathbb{R}^n$. The goal is to find an optimal state feedback law $u_t^*(x_t)$, that minimizes the cost function $\mathcal{J}$ given the control sequence $u^* = \{u_0^*,\dots,u_{T-1}^*\}$.

2.3.1 Dynamic programming

Dynamic Programming (DP) is a special case of optimization that exploits the self-consistent structure of the optimal solution in a sequential problem. Given the *principle of optimality*, optimal control is obtained by recursively sweeping back through time using the previous optimal solution to solve the current subset of problems (Kirk, 2012). Therefore, DP constrains the problem so that the optimal control sequence $u_{0:T-1}^*$ is derived by reducing the minimization over the entire sequence of controls to a sequence of minimization over a single control.

Providing that the optimization of the cost in the control problem satisfies a Bellman equation, DP can be used to solve the optimization. We define a Value function at time t, $\mathcal{V}(x_t,t)$, as

$$\mathcal{V}(x_t, t) = \min_{u_t \in \mathcal{U}} \{\ell_t(x_t, u_t) + \mathcal{V}(x_{t+1}, t+1)\} \tag{2.4}$$

this is the optimal (minimal) cost-to-go from time t to the terminal time T. In order to initialize the Value function, the terminal boundary condition is defined as, $\mathcal{V}(x_T, T) = \ell_T(x_T)$. Therefore, we can dynamically sweep back through time.

Using the Value function, we can find the optimal control law, by finding the argmin of the Value function,

$$u_t(x_t) = \arg\min_{u_t \in \mathcal{U}} \{\ell_t(x_t, u_t) + \mathcal{V}(x_{t+1}, t+1)\} \tag{2.5}$$

and successively compute the optimal trajectory $\{x^*_{t:T}\}$ in a single forward pass,

$$x^*_{t+1} = f_t(x^*_t, u^*_t(x^*_t)) = f^*_t(x^*_t). \tag{2.6}$$

2.3.2 Linear quadratic program

Linear Quadratic Regulator (LQR) is a particular class of optimal control problems that are well-structured and can be solved efficiently using quadratic programming. For the LQR problem, the ℓ_t term in Equation (2.1) is defined as

$$\ell_t(x_t, u_t) = \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} Q_t & S_t \\ S_t^T & R_t \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} q_t \\ r_t \end{bmatrix} \tag{2.7}$$

and terminal cost term,

$$\ell_T(x_T) = x_T^T Q_T x_T + x_T^T q_T \tag{2.8}$$

The state cost $Q_t = Q_t^T \succcurlyeq 0$ and input cost $R_t = R_t^T \succ 0$ are respectively positive semi-definite and positive definite, and both symmetric matrices, which ensure stability and uniqueness. The f_t terms in Equation (2.2) are defined as,

$$x_{t+1} = A_t x_t + B_t u_t \tag{2.9}$$

The optimization is constrained by the causal linear dynamics, where the successive state $x_{t+1} \in \mathbb{R}^n$ is determined by the state space matrix $A \in \mathbb{R}^{n \times n}$ and the input matrix $B \in \mathbb{R}^{n \times m}$, which act on the current state $x_t \in \mathbb{R}^n$ and control $u_t \in \mathbb{R}^m$, respectively. The state space matrix A and input matrix B can be time-varying.

LQR Value function

The objective of the Value function, $\mathcal{V}(x, t)$, is to minimize the cost-to-go from time t to the terminal time T, that is,

$$\mathcal{V}(x, t) = \min_{u_{t:T}} \quad \{l_f(x_T) + \sum_{t}^{T-1} l_t(x_t, u_t)\} \tag{2.10}$$

The principal of optimality in Equation (2.4) allows us to solve the LQR problem by recursively solving the Value function backwards in time. Therefore, the Value function at time t in Equation (2.10) can be rewritten as a Bellman equation,

$$\mathcal{V}(x, t) = \min_{u_t} \left(\begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} Q_t & S_t \\ S_t^T & R_t \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} q_t \\ r_t \end{bmatrix} + \mathcal{V}(x_{t+1}, t+1) \right) \tag{2.11}$$

where the Value function has a boundary condition at the final timepoint $t = T$, that is,

$$\mathcal{V}(x, T) = \frac{1}{2} x_T^T V_T v_T + x_T^T v_T \tag{2.12}$$

parameterized by the terminal state cost, $V_T = Q_T$ and $v_T = q_T$. Importantly, to solve the LQR with DP we need the Value function to maintain the same structure at all timepoints, t. Therefore to solve the Value function, $\mathcal{V}(x_t, t)$, let,

$$\mathcal{V}(x_t, t+1) = \frac{1}{2} x_{t+1}^T V_{t+1} x_{t+1} + x_{t+1}^T v_{t+1} \tag{2.13}$$

and substituting into Equation (2.11) yields,

$$\begin{aligned} \mathcal{V}(x, t) = \min_{u_t} \Bigg(& \frac{1}{2} \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} Q_t + A_t^T V_{t+1} A_t & S_t + A_t^T V_{t+1} B_t \\ S_t^T + B_t^T V_{t+1} A_t & R_t + B_t^T V_{t+1} B_t \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \\ & \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} q_t + A_t^T v_{t+1} \\ r_t + B_t^T v_{t+1} \end{bmatrix} \Bigg) \end{aligned} \tag{2.14}$$

$$= \min_{u_t} \left(\frac{1}{2} \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} Q_t^{xx} & Q_t^{xu} \\ Q_t^{ux} & Q_t^{uu} \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} q_t^x \\ q_t^u \end{bmatrix} \right) \tag{2.15}$$

having re-arranged variables x and u in terms of time t, and aggregated the block matrix and vector terms in Equation (2.14) to $Q_t^{xx}, Q_t^{xu}, Q_t^{ux}, Q_t^{uu}, q_t^x, q_t^u$. The optimal inputs minimize this function using a state feedback law, $u_t^*(x_t)$,

$$u_t^*(x_t) = K_t x_t + k_t \tag{2.16}$$

where $K_t = -{Q_t^{uu}}^{-1} Q_t^{ux}$ and $k_t = -{Q_t^{uu}}^{-1} q_t^u$ are the optimal gain and bias terms. By substituting the optimal state feedback law Equation (2.16) into Equation (2.15), we can obtain the optimal Value function at time t,

$$\mathcal{V}(x_t, t) = \frac{1}{2} x_t^T V_t x_t + x_t^T v_t \tag{2.17}$$

preserving the structure of the Value function as in Equation (2.13). Once we have solved the Value function at time t, we can compute the gains K and k at time $t-1$, thereby determining the optimal control policy and finding the entire optimal control sequence backward in time. Details of solving the Value function and gain matrices can be found in Appendix A.1.

2.3.3 Iterative linear quadratic regulator

The iterative Linear Quadratic Regulator (iLQR) is an extension of the LQR that solves optimal control problems with a nonlinear cost, constrained by nonlinear dynamics. The iLQR algorithm is an iterative method that solves LQR problems around a local trajectory by updating the control sequence with optimal deviations until the cost converges to a local optimum. Input deviations minimize a local quadratic approximation of the cost and can therefore be interpreted as a quasi-Newton gradient descent method on the input sequence (Xie & Gang, 2021).

The general form of the iLQR problem is defined as,

$$\begin{aligned} \underset{u,x}{\text{minimize}} \quad & \mathcal{J}(x,u) = \sum_{t=0}^{T-1} \mathcal{C}(x_t, u_t) + \mathcal{C}_f(x_T) \\ \text{subject to} \quad & x_{t+1} = f(x_t, u_t), \quad t \in [0, T] \end{aligned} \tag{2.18}$$

At each iteration $i = 1, 2, 3, \ldots$ of the iLQR algorithm, the cost and dynamics functions are locally approximated around the current state $x^{(i)}$ and input $u^{(i)}$ sequence. The initial $x^{(i)}$ trajectory is obtained by applying the initialized control sequence $u^{(i)}$ to the dynamics.

The linearized dynamics, $\hat{f}$, is defined as,

$$\hat{f}^{(i)}(\delta x_t, \delta u_t) = f(x_t^{(i)}, u_t^{(i)}) + A_t \delta x_t + B_t \delta u_t \tag{2.19}$$

where $A = \nabla_x f$ and $B = \nabla_u f$ are the Jacobians of the dynamics function f (Li & Todorov, 2004). Similarly, the cost function is linearized $\hat{\mathcal{J}}$ as,

$$\hat{\mathcal{J}}^{(i)}(\delta x_t, \delta u_t) = \mathcal{J}(x_t^{(i)}, u_t^{(i)}) + \frac{1}{2} \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix}^T \begin{bmatrix} C_t^{xx} & C_t^{xu} \\ C_t^{ux} & C_t^{uu} \end{bmatrix} \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix} + \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix}^T \begin{bmatrix} C_t^{x} \\ C_t^{u} \end{bmatrix} \tag{2.20}$$

where δx and δu are deviations from the trajectories. The cost parameters are Hessians and Jacobians around the state and input trajectories, $\mathcal{C}^{xx} = \nabla_x^2 \mathcal{C}$, $\mathcal{C}^{uu} = \nabla_u^2 \mathcal{C}$, $\mathcal{C}^{ux} = \nabla_{ux} \mathcal{C}$, $\mathcal{C}^{x} = \nabla_x \mathcal{C}$, $\mathcal{C}^{u} = \nabla_u \mathcal{C}$, and $\mathcal{C}^{xx} = \nabla_x^2 \mathcal{C}$. To ensure there exists a global optimal in the cost Levenberg-Marquardt regularization is enforced on the cost matrices $\mathcal{C}^{xx}$ and $\mathcal{C}^{uu}$ (Tassa et al., 2012). Therefore, we add a small non-negative offset ζ along the diagonal, $\mathcal{C}^{xx} \leftarrow \mathcal{C}^{xx} + \zeta \mathbb{I}$, such that $\mathcal{C}^{xx} \succ 0$ and $\mathcal{C}^{uu} \succ 0$.

Equations (2.19) and (2.20) have the same form as the LQR in Equations (2.7) to (2.9). However, now evaluating the linear-quadratic problem for small deviations around the input and state trajectories. Therefore, inputs can be improved by solving LQR and finding optimal input perturbations around the current i-th iterations, hence,

$$u_t^{(i+1)} = u_t^{(i)} + \delta u_t^* \tag{2.21}$$

This can generate another trajectory with a forward rollout, and the process of local approximations can be repeated until the input sequence converges to a local optimum. The optimal input deviation, δu_t^*, is obtained, similar to solving the LQR Value function in Equation (2.11), but this time is around state deviations,

$$\mathcal{V}(\delta x_t, t) = \min_{\delta u_t} \Bigg\{ \frac{1}{2} \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix}^T \begin{bmatrix} \mathcal{C}_t^{xx} & \mathcal{C}_t^{xu} \\ \mathcal{C}_t^{ux} & \mathcal{C}_t^{uu} \end{bmatrix} \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix} + \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix}^T \begin{bmatrix} \mathcal{C}_t^{x} \\ \mathcal{C}_t^{u} \end{bmatrix} + \mathcal{V}(f(x_t + \delta x_t, u_t + \delta u_t), t+1) \Bigg\} \tag{2.22}$$

and,

$$\delta u_t^* = K_t \delta x_t + k_t \tag{2.23}$$

which is then used to update Equation (2.21) (Aaqaoui & Mohammed, 2024; Li & Todorov, 2004). The updated state trajectory $x^{(i+1)}$ and the control input sequence $u^{(i+1)}$ can be acquired once all gain terms have been collected. The non-linear cost value is evaluated using the new $i+1$ iteration of the state and input trajectories. These steps are repeated until the costs reach a convergence threshold; further details are in Appendix A.4.

2.3.4 Associative dynamic programming

Dynamic programming is a sequential algorithm where the solution is obtained in linear time $\mathcal{O}(T)$. Blelloch (1989) developed the associative scan algorithm that allows the parallelization of certain sequential algorithms,

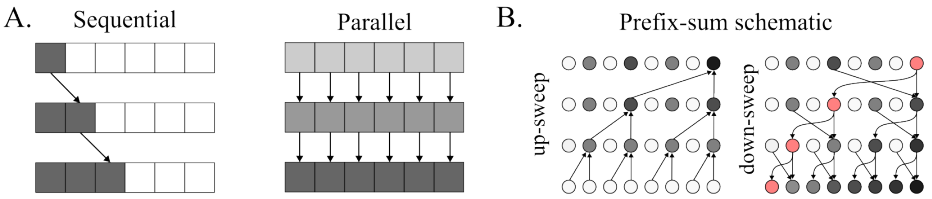


Figure 2.1. (A) The flow of information in a sequential recursive algorithm (left) is based on its left-neighboring element. In contrast, the associative algorithm (right) computes values in parallel and stores them in intermediate layers. (B) An example of an associative algorithm is prefix summation. Parallelization occurs in two steps, the up-sweep (left) and the down sweep (right).

reducing the time complexity to $\mathcal{O}(\log T)$. Associative scans can be formed if the function is a binary operator, i.e. $f : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$, and the operator is associative, e.g. $(a \oplus b) \oplus c = a \oplus (b \oplus c)$.

Prefix sum algorithm

The first demonstration of this case was the all-prefix summation (Blelloch, 1989). Given an array length N,

$$[a_1, a_2, \ldots, a_{N-1}] \tag{2.24}$$

the associative binary operator $\oplus$ sequentially scans through the array, producing

$$[a_1, (a_1 \oplus a_2), \ldots, (a_1 \oplus a_2 \oplus \cdots \oplus a_{N-1})] \tag{2.25}$$

The associative scan consists of two phases, the up-sweep (reduction) and down-sweep (Sengupta et al., 2006); see Algorithm 1. As schematized in Figure 2.1b, for the *up-sweep* phase, each pair of elements is summed

Algorithm 1 Parallel all-prefix sum algorithm. The up-sweep phase followed by the down sweep phase

```
1: procedure UP-SWEEP
2:     for d := 0 to log_2 n − 1 do
3:         for all k from 0 to n − 1 by 2^{d+1} do in parallel
4:             a[k + 2^{d+1} − 1] := a[k + 2^d − 1] + a[k + 2^{d+1} − 1]
5:         end for
6:     end for
7: end procedure

8: procedure DOWN-SWEEP
9:     for d := log_2 n − 1 to 0 do
10:        for all k from 0 to n − 1 by 2^{d+1} do in parallel
11:            t := a[k + 2^d − 1]
12:            a[k + 2^d − 1] := a[k + 2^{d+1} − 1]
13:            a[k + 2^{d+1} − 1] := t + a[k + 2^{d+1} − 1]
14:        end for
15:    end for
16: end procedure
```

in parallel, with their intermediate results stored to form a new, sparse partial layer. This process is repeated: the elements of each new partial layer are summed and stored until the sequence is reduced to a final sum.

The *down-sweep* phase uses a different parallel operator, which requires both partial sums and odd-indexed values, but follows the same communication pattern. Note, as shown in Figure 2.1b, the final sum from the up-sweep is replaced with 0 before the down-sweep. This operator takes two inputs and produces two outputs: the left output is the right input copied over, and the right output is the sum of both inputs (Blelloch, 1990).

2.3.5 Parallel LQR backward pass (Riccati recursion)

Section 2.3.2 has shown that the value function can be solved using the Bellman equation, a first-order recurrence equation; see Equation (2.4). Hence, optimal control can be obtained using the state feedback law; see Equation (2.5). Blelloch (1989) showed that first-order recurrence equations can be framed as a prefix sum operation, and thus, solved in parallel using the associative scan algorithm; see Appendix A.5.1 for more details.

Therefore, given the minimization operator is associative, an alternative to solving the value function sequentially in reverse time is to follow a method similar to the Blelloch (1989) prefix summation algorithm. In this approach, partial value functions, known as Conditional Value Function (CVF), are independently solved in parallel, and these partial solutions are then combined to recover the full value function. Since the Riccati recursion operator is associative (Sarkka & Garcia-Fernandez, 2023), we can generate a set of sub-problems to find the CVF.

Conditional Value function The CVF is the cost function of the optimal trajectory from x_i to x_k, and is denoted as $\mathcal{V}_{i\to k}(x_i, x_k)$. It is defined as,

$$\begin{aligned}\mathcal{V}_{i\to k}(x_i, x_k) = \underset{u_{i:k-1}}{\text{minimize}} \quad & \sum_{t=i}^{k-1} l_t(x_t, u_t) \\ \text{subject to} \quad & x_{t+1} = A_t x_t + B_t u_t + a_t, \quad t \in [i, k-1]\end{aligned} \tag{2.26}$$

Due to the principle of optimality, we can partition the Value function into CVFs, minimized over given states. Therefore, combination of the CVF can be written as,

$$\mathcal{V}_{i\to k}(x_i, x_k) = \min_{x_j}\{\mathcal{V}_{i\to j}(x_i, x_j) + \mathcal{V}_{j\to k}(x_j, x_k)\} \tag{2.27}$$

where $i < j < k \leq T$ and so we can obtain the Value function at time $t = i$ from the CVFs by minimizing over state x_k,

$$\mathcal{V}(x_i, i) = \min_{x_k}\{\mathcal{V}_{i\to k}(x_i, x_k) + \mathcal{V}(x_k, k)\}. \tag{2.28}$$

The minimization in Equation (2.28) also returns the minimizing state x_k. According to the principle of optimality, this value represents the state at time step k, situated on the optimal trajectory from x_i to time T. Likewise for Equation (2.27), minimizing over x_j forms part of the optimal trajectory from x_i to x_k.

Value functions associative operator The associative elements for the parallel backward scan are the CVFs. As in Equation (2.27), the minimization function over the state x_j is the binary and associative operator; see proof for associativity in Appendix A.5.2. For shorthand notation, the set of CVFs elements, a_i, are defined as,

$$a_i = \mathcal{V}_{i\to i+1}(x_i, x_{i+1}) \tag{2.29}$$

and are initialised for $i \in [0, 1, \ldots, T]$. Noting that the terminal CVF element is defined as $\mathcal{V}_{T\to T+1}(x_T, x_{T+1}) := \mathcal{V}(x_T, T)$. Then to roll out the parallel scan as in Equation (2.27), elements are combined as,

$$a_i \circ a_{i+1} \circ \cdots \circ a_{k-1} = \mathcal{V}_{i\to k}(x_i, x_k) \tag{2.30}$$

where the associative min operator is denoted as $\circ$. The terminal element can be used to recover the Value function as,

$$a_k \circ a_{k+1} \circ \cdots \circ a_T = \mathcal{V}(x_k, K) \tag{2.31}$$

corresponding to the value function recursion in Equation (2.28). For further details see Appendix A.5.2.

2.3.6 Parallel LQR forward pass (State integration)

Once the value functions are solved and optimal control law obtained, the optimal trajectory can be recovered, i.e. $\{x^*\} = (x_0^*, \ldots, x_T^*)$. Equation (2.6) can be written as a composition of functions,

$$x_{k+1}^* = (f_k^* \circ f_{k-1}^* \circ \cdots \circ f_0^*)(x_0) \tag{2.32}$$

where f_k^* is the optimal control law at time k. As shown in Appendix A.5.1, linear state dynamics is a first-order recurrence problem that can be solved in parallel.

The optimal control law can be written as an effective autonomous dynamical system when considering the optimal gains from Equations (2.9) and (2.16),

$$f_k^* = \tilde{F}_k x_k + \tilde{c}_k \tag{2.33}$$

where,

$$\tilde{F}_k = A_k - B_k K_k \tag{2.34}$$

$$\tilde{c}_k = a_k + B_k k_k \tag{2.35}$$

Conditional optimal trajectory The conditional optimal trajectory function is a fragment of the optimal trajectory. For example the optimal trajectory from x_i to x_k, driven by dynamic function $f^*_{i\to k}(x_i, x_k)$, is composed by the functional composite

$$f^*_{i\to k} = (f^*_{k-1} \circ \cdots \circ f^*_{i+1} \circ f^*_i)(x_i) \tag{2.36}$$

$$= \tilde{F}_{i\to k} x_i + \tilde{c}_{i\to k} \tag{2.37}$$

The optimal trajectory driven by $f^*_{i\to k}(x_i, x_k)$, can be composed of partial conditional optimal trajectories functions $f^*_{i\to j}(x_i, x_j)$ and $f^*_{j\to k}(x_j, x_k)$. Therefore, the conditional coefficients in Equation (2.37) can be combined as,

$$\tilde{F}_{i\to k} = \tilde{F}_{j\to k}\tilde{F}_{i\to j} \tag{2.38}$$

$$\tilde{c}_{i\to k} = \tilde{F}_{j\to k}\tilde{c}_{i\to j} + \tilde{c}_{j\to k} \tag{2.39}$$

where $i < j < k \leq T$, details shown in Appendix A.5.1.

2.4 Design principle

`DiffiLQRAX` follows the functional programming style as in the JAX library. The main design principle is to create a generic, associative, and differentiable solver for the LQR and iLQR problems Table 2.1. Each solver is optimized for specific purposes: generic solver for CPU, differentiable solver for optimization layer in deep networks or meta-learning, and associative for scalable parallel computation on GPU.

The LQR solvers, require specification of the cost and dynamics parameters, as in Equations (2.7) to (2.9), and the solver will return the optimal state trajectory, x, control sequence, u, and costate trajectory, λ, see Listing 1.

Table 2.1. Selection of `DiffiLQRAX` solvers and optimal device to run solvers

Solver type	Module	Optimal device	Arguments
Generic	`lqr`	CPU	`LQRParams`
	`ilqr`	CPU	`ThetaParams`, `System`
Differentiable	`dlqr`	CPU/GPU	`LQRParams`
	`dilqr`	CPU/GPU	`ThetaParams`, `System`
Associative	`plqr`	GPU	`LQRParams`
	`pilqr`	GPU	`ThetaParams`, `System`

Listing 1 The format of LQR solver which applies to generic, associative, and differentiable.

```
import jax.numpy as jnp
from diffilqrax.typs import LQRParams
from diffilqrax import lqr

lqr_params = LQRParams(x_init=jnp.zeros(3), lqr=mat_params)
opt_xs, opt_us, opt_adjoints = lqr.solve(lqr_params)
```

The iLQR solvers follow a similar structure to LQR, except that the solver requires the non-linear dynamics and cost functions to be defined in the `System` class. These functions will be Taylor approximated around the nominal input and state trajectories, where the local LQR problem is solved. Listing 2 also shows that optional line search hyperparameters can be passed to speed up convergences Section 2.5.2.

Control flow in `DiffiLQRAX` Solvers To utilize the power of JAX JIT-compilation and XLA compiler, control flows are defined in a specific form. Firstly, `if-else` statements are replaced with `vmap` and `jax.lax.cond` functions. Secondly, `for` loops are replaced with `jax.lax.scan` function. This allows the JIT compiler to parallelize the computation and optimize the code for the target device. Finally, iterative solvers that rely on a

Listing 2 The format of iLQR solver which applies to generic, associative, and differentiable.

```
import jax.numpy as jnp
import jax.random as jr
from diffilqrax import ilqr
from diffilqrax.typs import iLQRParams, Theta, ModelDims, System
from diffilqrax.utils import initialise_stable_dynamics, keygen

dims = ModelDims(8, 2, 100, dt=0.1)

key = jr.PRNGKey(seed=234)
key, skeys = keygen(key, 5)

Uh = initialise_stable_dynamics(next(skeys), dims.n, dims.horizon, 0.6)[0]
Wh = jr.normal(next(skeys), (dims.n, dims.m))
theta = Theta(Uh=Uh, Wh=Wh, sigma=jnp.zeros(dims.n), Q=jnp.eye(dims.n))
params = iLQRParams(x0=jr.normal(next(skeys), dims.n), theta=theta)
Us = jnp.zeros((dims.horizon, dims.m))
# define linesearch hyper parameters
ls_kwargs = {
   "beta":0.8,
   "max_iter_linesearch":16,
   "tol":1e0,
   "alpha_min":0.0001,
   }
def cost(t, x, u, theta):
   return jnp.sum(x**2) + jnp.sum(u**2)

def costf(x, theta):
   return jnp.sum(x**2)

def dynamics(t, x, u, theta):
   return jnp.tanh(theta.Uh @ x + theta.Wh @ u)

model = System(cost, costf, dynamics, dims)

(opt_xs, opt_us, opt_adjoints), total_cost = ilqr.ilqr_solver(params, model, Us,
↪   **ls_kwargs)
```

convergence criteria rely on a `while` loop. As this loop could be infinite it is not possible to reverse differentiate through the solver. Instead, we replace the `jax.lax.while_loop` with a `jax.lax.scan` and additional `max_iteration` parameter. In this case there is a conditional step function that returns identity if the convergence criteria is met, otherwise continues with the iterative step function.

2.5 Results

`DiffiLQRAX` was developed to provide a light-weight and efficient solver for the LQR and iLQR problems (Mullen & Schimel, 2025). We tested the performance of the solver on a range of control problems. The bulk of the solver was developed using DP techniques, to reduce the complexity of the problem. We validated the solution by comparing the results with other methods and analyzing the convergence of the solver.

2.5.1 Optimal solution of LQR solver

The LQR algorithm relies on two recursive passes, Algorithm 2. First, the backward pass that starts from the terminal cost and sweeps back through time solving for the optimal gains. Then, the forward pass integrates through the dynamics using the optimal feedback rule.

We ensured the LQR solution of the `DiffiLQRAX` solver was numerically sound by (1) benchmarking its solution against alternative methods and (2) verifying that its solution satisfied the KKT optimality conditions; see Appendix A.2. The generic LQR solver is the core of `DiffiLQRAX` library, as the non-linear, differentiable, and associative versions adopt a lot of the functionality from the sequential backward-forward Riccati sweep version.

Algorithm 2 Riccati Recursion for Solving LQR with KKT Conditions

procedure BACKWARD PASS
  $V_N \leftarrow Q_f$
  $v_N \leftarrow q_f$
  for $n = N-1 \rightarrow 0$ **do**
    $\tilde{R}_n \leftarrow R_n + B_n^T V_{n+1} B_n$
    $K_n \leftarrow -\tilde{R}_n^{-1}(S_n + B_n^T V_{n+1} A_n)$
    $V_n \leftarrow Q_n + A_n^T V_{n+1} A_n - K_n^T \tilde{R}_n K_n$
    $k_n \leftarrow -\tilde{R}_n^{-1}(s_n + B_n^T (V_{n+1} a_n + v_{n+1}))$
    $v_n \leftarrow q_n + A_n^T (V_{n+1} a_n + v_{n+1}) - K_n^T \tilde{R}_n k_n$
  end for
  $\lambda_0 \leftarrow V_0 x_0 + v_0$
end procedure

procedure FORWARD PASS
  for $n = N-1 \rightarrow 0$ **do**
    $u_n \leftarrow K_n x_n + k_n$
    $x_{n+1} \leftarrow A_n x_n + B_n u_n + a_n$
    $\lambda_{n+1} \leftarrow V_{n+1} x_{n+1} + v_{n+1}$
  end for
end procedure

Numerical Stability To encourage stability in the solver, we add Levenberg-Marquardt regularization to enforce non-zero diagonal on $\tilde{R}$ before inversion (Tassa et al., 2012). We also explicitly symmetrize the block-diagonal quadratic coefficients in Equation (2.15) to ensure that $A^T = A$ is preserved after numerical operations.

Comparison of `DiffiLQRAX` solver with direct inversion and conjugate gradient methods

We compared the solution of the `DiffiLQRAX` solver with the direct inversion and conjugate gradient methods; see Figure 2.2. The exact solution of the LQR problem can be obtained by aggregating the cost and dynamic

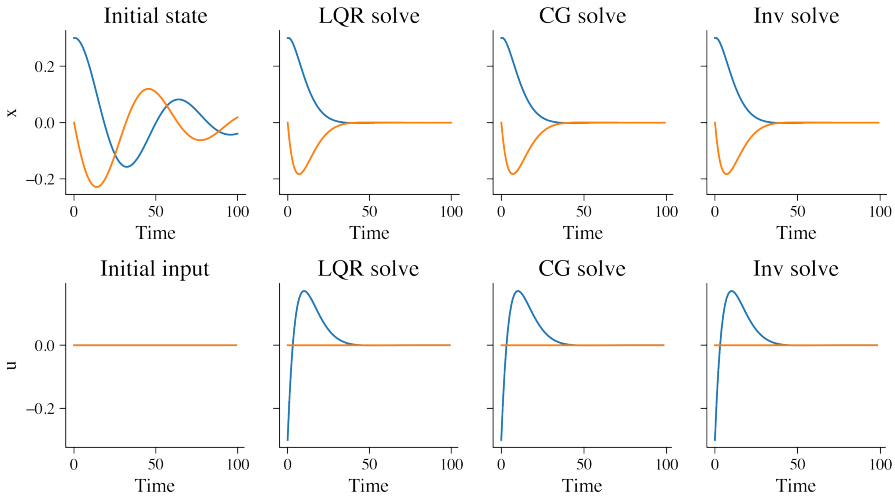


Figure 2.2. Generic `DiffiLQRAX` LQR solver solution compared to inverting the KKT matrix directly (using `JAX.numpy.linalg`) and via conjugate gradients (using `jaxopt` (Blondel et al., 2021)). The `DiffiLQRAX` solution was equivalent to both methods within relative error of 10^{-6} and absolute error of 10^{-8}. Top row shows the evolution of the state trajectory of x without input, with `DiffiLQRAX` solution, direct inversion and conjugate gradient, respectively. Bottom row shows the evolution of the input trajectory of u with `DiffiLQRAX` solution, direct inversion and conjugate gradient, respectively.

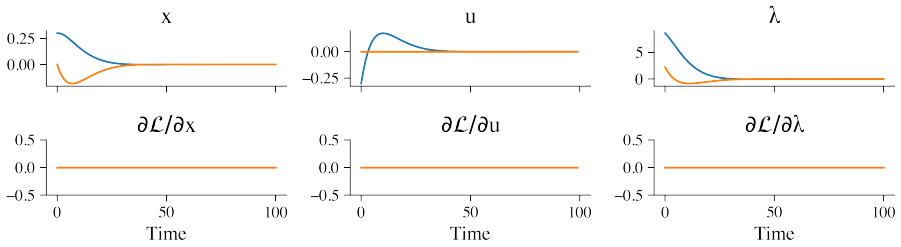


Figure 2.3. Generic `DiffiLQRAX` LQR solver solution satisfies the KKT optimality conditions. The top row shows the evolution of the optimal state trajectory, x, the optimal input trajectory, u, and costate trajectory, λ. The bottom row shows the Lagrangian constraints evaluated at each time step, for optimality conditions all constraints should be zero.

constraints to form a band-diagonal KKT matrix, Appendix A.2. The KKT matrix can then be inverted directly or implicitly using conjugate gradient methods. After jit-compiling the LQR solver obtained the optimal solution 376 ms ±27.7 ms per loop ($\mu \pm \sigma$ of 7 runs, 1 loop each). The direct inversion method took 230 ms ±18 ms per loop ($\mu \pm \sigma$ of 7 runs, 1 loop each) and the conjugate gradient method took 307 ms ±15.8 ms per loop ($\mu \pm \sigma$ of 7 runs, 1 loop each). Although the other methods outperformed the `DiffiLQRAX` solver, the complexity scales with $\mathcal{O}((Tnm)^3)$, so the solver is more efficient for large-scale problems.

Solution validation with Lagrangian optimality conditions

We then confirmed that the `DiffiLQRAX` inferred control sequence met the KKT optimality conditions. In the LQR case, satisfying the KKT condition is Sufficient Optimal Condition (SOC). The KKT conditions are used to verify the solution of an optimization problem with both equality and inequality constraints satisfying Necessary Optimal Condition (NOC). With access to the LQR parameters, we explicitly computed the partial

derivatives of the Lagrangian with respect to the state, input and costate corresponding to the KKT equations; see Appendix A.2 for details. The optimality conditions are satisfied when the Lagrangian constraints are zero, see Figure 2.3.

2.5.2 iLQR solver: optimal control of coupled pendulum

We extended the generic LQR solver to handle non-linear problems by iteratively solving local LQR subproblem, resulting in the iLQR solver. While the core computations remain similar to the LQR solver, the key difference is that the non-linear dynamics and cost functions are linearized and quadratized around an initial state and input sequence.

At each iteration, the state and control variables are updated, and the cost is recalculated. The updated cost is then compared to the previous one to assess convergence. To improve the convergence rate, like Schimel et al. (2022), we added a line search algorithm. This also prevents overshooting when dealing with very non-convex problems.

Backtracking line search An additional feature to improve convergence to a local optimum in the iLQR algorithm is to apply a 1D line search on the linear gain term of Equation (2.23). The line search backtracking algorithm finds the optimal step size in the gain update to prevent overshooting in the perturbation update (Tassa et al., 2012). This becomes useful when the control problem is highly non-convex. Equation (2.23) is adapted to have an addition hyperparameter, α, such that

$$\delta u_t^* = K_t \delta x_t + \alpha k_t \tag{2.40}$$

The backtrack line search performs a series of forward rollouts using Equation (2.6) with the function $f(x, u^*(x, \alpha))$, and the cost $\mathcal{J}(x, u; \alpha)$ is eval-

uated. Initially set with $\alpha = 1.$, that is, the largest step size, then geometrically decrements until the local minimum is found.

2.5.3 Parallelization of LQR solver

In order to define an associative algorithm, we need to define (1) pairwise elements parsed for parallel computation, and (2) the binary associative operation to combine the results of the parallel computation. The JAX library has an `associative_scan` function that scans through tuples and computes the operator in parallel alike the prefix summation in Algorithm 1.

Value function associative elements and combinatorics operator

Sarkka and Garcia-Fernandez (2023) showed the combinatorics rules can be obtained by maximizing the dual function of the LQR problem. Given two elements obtained from the LQR parameters, their combination obtained from Equations (2.27) and (2.28) is characterized by Listing 3. Details of this derivation can be found in (Sarkka & Garcia-Fernandez, 2023) which outlines the use of the dual function solution of the Lagrangian.

Importantly, the associative elements defined as in Equation (2.29) need a terminal element which also yields the Value function Equation (2.31). These elements are defined in Listing 4.

Forward pass associative elements and combinatorics operator

Similarly to the backward pass, the associative optimal trajectory defines the forward elements and combinatorics rules. The generic elements use the value function to obtain the optimal gains, and then the LQR parameters are rearranged to form the optimal state dynamics as in Sarkka and Garcia-Fernandez (2023). The first associative element includes the initial state, implemented as shown in Listing 5.

Listing 3 Element combination rule for associative scan of value functions.

```
from jax import vmap
from jax.numpy import eye
from jax.scipy.linalg import solve


@vmap
def assoc_riccati_operator(elem1, elem2):
    A1, a1, BRinvB1, v1, V1 = elem1 # unpacking elements
    A2, a2, BRinvB2, v2, V2 = elem2

    I = eye(x_dim) # state dimension

    I_BRinvB1V2 = I + BRinvB1 @ V2
    temp = solve(I_BRinvB1V2.T, A2.T).T
    A = temp @ A1
    a = temp @ (a1 + BRinvB1 @ v2) + a2
    BRinvB = temp @ BRinvB1 @ A2.T + BRinvB2

    I_V2BRinvB1 = I + V2 @ BRinvB1
    temp = solve(I_V2BRinvB1.T, A1).T
    v = temp @ (v2 - V2 @ a1) + v1
    V = temp @ V2 @ A1 + V1
    return A, a, BRinvB, v, V


# parallellised riccati scan
def parallel_riccati_scan(model: LQRParams):
    first_elements = build_associative_riccati_elements(model)

    final_elements = associative_scan(
        assoc_riccati_operator, first_elements, reverse = True
    )
    vs = final_elements[-2]
    Vs = final_elements[-1]
    return vs, Vs, v[0] + V[0]@model.x0
```

Listing 4 Defining associative elements from LQR parameters to solve value functions in parallel.

```python
from typing import Tuple
from jax import vmap, Array
import jax.numpy as jnp
import jax.scipy as jsc

from diffilqrax.typs import LQRParams


def build_associative_riccati_elements(
    model: LQRParams,
) -> Tuple[Tuple[Array, Array, Array, Array, Array]]:
    """Join set of elements for associative scan."""
    last_elem = last_riccati_element(model)
    generic_elems = generic_riccati_element(model)
    return tuple(
        jnp.concatenate([jnp.expand_dims(last_e, 0), gen_es])
        for gen_es, last_e in zip(generic_elems, last_elem)
    )


# last riccati element
def last_riccati_element(model: LQRParams):
    """Define last element of Riccati recursion"""
    n_dims = model.lqr.Q.shape[1]
    A = jnp.zeros((n_dims, n_dims), dtype=float)
    a = jnp.zeros((n_dims,), dtype=float)
    BTRinvB = jnp.zeros((n_dims, n_dims), dtype=float)
    v = -model.lqr.qf
    V = model.lqr.Qf
    return A, a, BTRinvB, v, V


# generic riccati element
def generic_riccati_element(model: LQRParams):
    """Generate generic Riccati element"""
    n_dims = model.lqr.Q.shape[1]
    A = model.lqr.A
    a = model.lqr.a
    R_invs = vmap(jsc.linalg.inv)(model.lqr.R)
    BTRinvB = jnp.einsum("ijk,ikl,iml->ijm", model.lqr.B, R_invs, model.lqr.B)
    v = -model.lqr.q
    V = model.lqr.Q
    return A, a, BTRinvB, v, V
```

Listing 5 Defining associative elements from LQR parameters to solve optimal trajectory in parallel.

```python
from typing import Tuple
from functools import partial
from jax import Array, vmap, tree_map
import jax.numpy as jnp
import jax.scipy as jsc

from diffilqrax.typs import LQRParams, LQR
pop_first = partial(tree_map, lambda x: x[1:])

def generic_lin_dyn_elements(lqr: LQR, v:Array, V:Array):
    A, B, R, a =  lqr.A, lqr.B, lqr.R, lqr.a
    pinv = jsc.linalg.inv(B.T@V@B + R)
    # define gain coeffs
    Kv = pinv@B.T
    Kc = Kv@V
    Kx = Kc@A
    Ft = A - B@Kx
    ct = a + B@Kv@v - B@Kc@a
    return Ft, ct

def first_lin_dyn_element(model: LQRParams, v0:Array, V0:Array, alpha:float):
    A, B, R, a =  model.lqr.A[0], model.lqr.B[0], model.lqr.R[0], model.lqr.a[0]
    pinv = jsc.linalg.inv(B.T@V0@B + R)
    Kv = pinv@B.T
    Kc = Kv@V0
    Kx = Kc@A
    F0 = A - B@Kx
    c0 = a + alpha*(B@Kv@v0 - B@Kc@a)
    return jnp.zeros_like(V0), F0@model.x0 + c0, (Kx, Kv, Kc)


def build_associative_lin_dyn_elements(model: LQRParams, vs, Vs, alpha:float):
    """Join set of elements for associative scan."""
    first_elem = first_lin_dyn_element(model, vs[1], Vs[1], alpha) #this is at
    ↪  k+1
    generic_elems = vmap(generic_lin_dyn_elements, in_axes =
    ↪  (LQR(0,0,0,0,0,0,0,0,None,None), 0, 0))(pop_first(model.lqr), vs[2:],
    ↪  Vs[2:])
    return tuple(jnp.r_[jnp.expand_dims(first_e, 0), gen_es]
                 for first_e, gen_es in zip(first_elem, generic_elems))
```

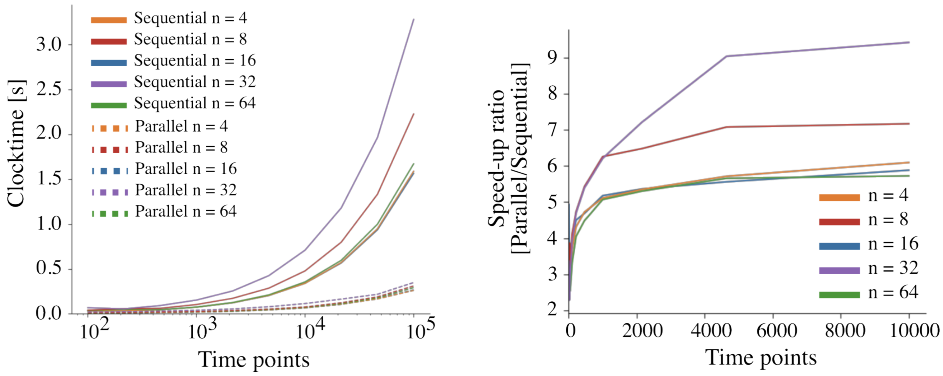


Figure 2.4. GPU runtime comparison of sequential (solid lines) and parallel LQR (dashed lines) solver of combined control law and optimal trajectory. Tested with different state dimensions, n, and input dimensions $m = n$.

Evaluation of parallel vs sequential LQR solver

As the LQR problem is unique and both results were equivalent, we used this experiment to compare the computational speed. The computational speed was calculated by averaging over 5 trials excluding time for jit-compilation. We tested parallel and sequential LQR solvers using a NVIDIA A100-SXM4 with 80GB memory.

We compared the sequential and parallel LQR algorithms (including the backward and forward pass) using random state dynamics initialized with spectral radius 0.6, and cost,

$$Q_k = 10 I_{n \times n}, \qquad R_k = I_{m \times m}. \tag{2.41}$$

The results compare the problem with different horizons, $T = \{10^0, 10^1, \ldots, 10^5\}$ in Figure 2.4 . The parallel LQR solver is faster than the sequential solver in all dimensions by a factor of 100.

2.5.4 Integrate differential optimal control layer in VAE

We demonstrated that iLQR can be used as an optimization algorithm. When made differentiable, it can be integrated into a Variational Autoencoder (VAE), allowing the system's dynamics and cost to be learned from data. Since iLQR is iterative, direct differentiation is computationally expensive because all iterations must be unrolled during back propagation.

However, Amos and Kolter (2017) showed that the KKT conditions can be used for implicit differentiation through the solver, which avoids unrolling all the solver's iterations. Later, Schimel et al. (2022) introduced an efficient method to back-propagate the implicit gradients by defining the adjoints of the KKT as another LQR problem, thus preventing the need to invert large matrices. This breakthrough made it feasible to integrate the differentiable iLQR solver as an optimization layer within a VAE model.

We replicated the work of Schimel et al. (2022) and demonstrated that the `DiffiLQRAX` library can be integrated with a VAE built in JAX. We also extended this work by adapting the differentiable iLQR to use associative scan, a key improvement. This approach enables GPU-scaled parallelization, making it efficient to infer from samples with long horizons, capturing more information about long-term dynamics.

iLQR-VAE model

Variational Autoencoder (VAE) are a probabilistic family of models used to learn the underlying distribution of data via a low-dimensional latent probability space (Kingma & Welling, 2013; Pandarinath, O'Shea, et al., 2018). The model is composed of two parts: a generative model and a recognition model. Once the VAE is trained on data, inferences can be made on unseen data embedding them in the latent space. Also, sampling

from the latent space can generate new data samples. Schimel et al. (2022) proposed iLQR-VAE that learns to capture observational time series of input-driven latent dynamics. This relies on the use of the iLQR algorithm as part of the recognition model.

Generative model The generative model is characterized by a set of equations,

$$x_{t+1} = f_\theta(x_t, u_t) \tag{2.42}$$

$$o_t|x_t \sim p_\theta(o_t|x_t) \tag{2.43}$$

$$u_t \sim p_\theta(u_t) \tag{2.44}$$

where, $x_t \in \mathbb{R}^n$ is the latent state at time t, $u_t \in \mathbb{R}^m$ is the control input, and $o_t \in \mathbb{R}^{n_o}$ is the observation. The latent dynamics transition function f_θ is parameterized by θ. The dynamics are deterministic which generates a sequence of states $\boldsymbol{x} = \{x_0, \ldots, x_T\}$ given an initial state x_0 and input sequence $\boldsymbol{u} = \{u_0, \ldots, u_{T-1}\}$. The observations are captured over the distribution $p_\theta(o_t|x_t)$ with a state readout function $g_\theta(x_t) = Cx_t + b$. Here, we modeled the observational distribution as Gaussian, $p_\theta(o_t|x_t) = \mathcal{N}(g_\theta(x_t), \Sigma^2)$.

Variational Inference With a set of N observations $\mathcal{O} = \{\boldsymbol{o}^{(1)}, \ldots, \boldsymbol{o}^{(N)}\}$ the objective of the model is to maximize the marginal log-likelihood, L,

$$L = \sum_{n=1}^{N} \log p_\theta(\boldsymbol{o}^{(n)}) = \sum_{n=1}^{N} \log \int p_\theta(\boldsymbol{o}^{(n)}|\boldsymbol{x}(\boldsymbol{u}))p_\theta(u)d\boldsymbol{u} \tag{2.45}$$

however, the integral in Equation 2.45 is often intractable. To avoid this, we introduce a variational distribution $q_\phi(x|u)$ parameterized by ϕ that ap-

proximates the true posterior distribution $p_\theta(x|u)$. The approximate posterior modifies the loss to be a lower bound on the marginal log-likelihood,

$$\begin{aligned} Ł(\phi, \theta|\mathcal{O}) &= \sum_{n=1}^{N} \mathbb{E}_{q_\phi(\boldsymbol{u}|\boldsymbol{o}^{(n)})} \left[\sum_{t=1}^{T} \log p_\theta(o_t^{(n)}|z_t) + \log p_\theta(u_t) - \log q_\phi(u_t|\boldsymbol{o}^{(n)}) \right] \\ &\leq \log p_\theta(\mathcal{O}) \end{aligned} \tag{2.46}$$

where Ł is known as the Evidence Lower Bound (ELBO). The model is trained by maximizing the ELBO with respect to both θ and ϕ parameters which is a proxy for maximizing the log-likelihood.

Recognition model The recognition model is parameterized as

$$q_\phi(\boldsymbol{u}|\boldsymbol{o}^{(n)}) = \mathcal{N}(\boldsymbol{u}^*(\boldsymbol{o}^{(n)}), \Sigma_S \otimes \Sigma_T) \tag{2.47}$$

$$\boldsymbol{u}^*(\boldsymbol{o}^{(n)}) = \text{iLQR}(\boldsymbol{o}^{(n)}, \theta) \tag{2.48}$$

$$= \arg\max_{\boldsymbol{u}} \left[\sum_{t=1}^{T} \log p_\theta(\boldsymbol{o_t^{(n)}}|\boldsymbol{u}) + \log p_\theta(u_t) \right] \tag{2.49}$$

The iLQR-VAE differs from the standard VAE by using the iLQR algorithm to find the Maximum A Posteriori (MAP) rather than another mapping function of observation to latent space. The iLQR algorithm is used to find the optimal control sequence u_t that minimizes the cost function given the current dynamics parameters θ. The MAP estimate is then centered around a Gaussian distribution with the mean as the optimal control sequence, $\boldsymbol{u}^*$, and the covariance are composed of spatial factors, Σ_S, and temporal factors, Σ_T

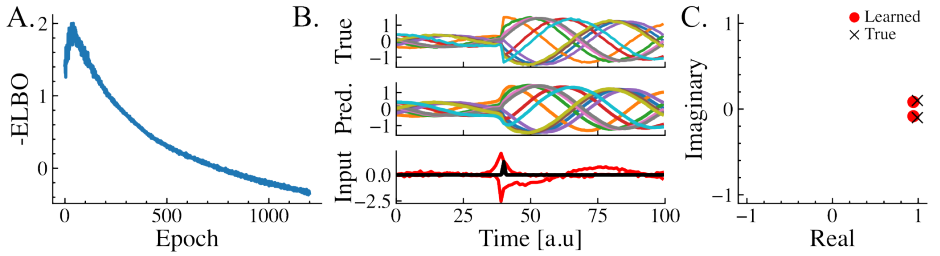


Figure 2.5. **VAE-iLQR model fit with Gaussian prior recovers underlying rotational dynamics.** (A.) A linear RNN trained by maximising ELBO loss converges over each epoch iteration. (B.) An example of an observed trajectory (top) that is reconstructed from the trained model (middle). The inferred control (bottom) used to generate the trajectory showed similar onset to the true external impulse (black). (C.) The eigenvalues of the model (red) show that the model learns similar dynamics to that of the toy model (black).

Recovery of underlying dynamics and control of toy model

To demonstrate the differentiable iLQR algorithm in `DiffiLQRAX` can be used as an inference layer for the VAE model, we trained the model on a toy dataset generated from a 2D linear dynamical system. Trajectories were generated from a 2D linear system which received a random impulse force, they were then projected onto a 10-dimensional observable space and corrupted with Gaussian noise. We fitted a linear RNN with a Gaussian prior on the input space; see Figure 2.5a. We were able to recover similar eigenvalues as the true latent dynamics of the toy model, shown in Figure 2.5c. Post-training, the iLQR could be used to infer optimal control sequences to generate unseen data samples, Figure 2.5b. If the generative model in the VAE was linear, the associative iLQR could be used for training datasets with longer time sequences.

2.6 Discussion

In summary, `DiffiLQRAX` is an open source software package that takes advantage of modern computer technology for efficient and scalable optimizations. This toolbox is designed to solve linear and non-linear optimization problems with equality constraint using LQR. This toolbox differs from other Python LQR packages in that it is differentiable and therefore can be integrated in deep learning networks which had been demonstrated using the Schimel et al. (2022) VAE model. Additional features like back-track linesearch (Tassa et al., 2012), using the DP back propagate gradients (Schimel et al., 2022), and associative LQR (Sarkka & Garcia-Fernandez, 2023) significantly optimized time expense in the solver and integration with other deep learning models.

A drawback with the current iLQR-VAE model is that the iLQR solver finds the optimal solution and sets this sequence as the mean of a predefined Gaussian distribution through time. This assumes that the covariance does not change through time and space and is not necessarily a good representation of the posterior distribution. As the LQR has a duality to the Kalman smoother (Todorov, 2008), where the cost Hessians and the Kalman covariance estimate both solve the Ricatti equations. The LQR inference could also estimate the covariance of input sequence avoiding the need to impose an outer-product Gaussian spatial-temporal product. Furthermore, it would be beneficial for the community to have the additional functionality of a Kalman filter and smoother in the toolbox for state estimation.

Some current drawbacks of the `DiffiLQRAX` toolbox is numerical stability. As the network state space increases, numerical errors are propagated in the matrix inversion regardless of adding the Levenberg-Marquardt regularization. An additional challenge is that the GPU-accelerated LQR solver consumes a lot of memory, so while it is time efficient, compu-

tational resources are compromised, and an economical computer would struggle to see the time benefit when running large models.

2.7 Acknowledgements

The toolbox was jointly developed with M. Schimel and inspired from T. C. Kao. The associative programming method of LQR was sketched out by works from S. Sarkka and A. F. Garcia-Fernandez.

Chapter 3

Learning sparse encoding of movement in Larval Zebrafish

3.1 Chapter Summary

A quantitative description of behavior is essential to understand the functions of the brain. It is believed that behavior is divided into stereotyped units of movement, known as motor primitives, which are combined to produce a wide and flexible range of behavior. The larval zebrafish's ballistic swimming behavior already provides a simple way to discretize behavior into motifs, known as bouts. However, these bouts vary within different behaviors. Traditionally, gross kinematic features are used to describe bout movements, but it is unclear which features are necessary to describe the full behavioral repertoire and whether they are directly encoded in the brain. A more general approach is to learn a dynamical system that can capture the variability of tail dynamics.

In this chapter, we fitted an input-driven generative model capable of capturing the behavioral repertoire of the larval zebrafish. We estimated the model's parameters using iLQR-VAE, which performed variational inference through optimal control, namely the iterative Linear Quadratic Regulator (iLQR) algorithm. The iLQR algorithm constrained the inferred control to be sparse while effectively reconstructing the observed tail movements. We showed that most movements are driven by sparse control inputs to the network. We detailed how the initial control peak encodes kinematic features of movement and is predictive of navigational parameters that were not used in training the model. Finally, we demonstrated that the latent dynamics used to generate tail movements were also involved in producing fin dynamics for certain behaviors. Overall, this chapter presents a framework to understand locomotor behavior through the lens of a low-dimensional control space.

3.2 Introduction

In recent years, there has been a resurgence in the study of neuroethology, which focuses on understanding the structure of behavior in natural environments and the underlying neural mechanisms (Anderson & Perona, 2014; Datta et al., 2019). This renewed interest is in part driven by advancements in optical technology that enable large-scale neuronal recordings (Steinmetz et al., 2018), as well as the development of pioneering computer vision tools capable of capturing detailed descriptions of animal movements (Pereira et al., 2020, 2022; Stih et al., 2019). At the same time, new methods are being devised to interpret complex patterns of brain-wide neural activity (Stringer et al., 2024; McInnes et al., 2018; Cunningham & Yu, 2014). A growing field within this area, known as computational ethology, is developing unsupervised methods to classify and quantify the behavioral repertoires of animals (Wiltschko et al., 2015; Datta et al., 2019).

The larval zebrafish was the first vertebrate in which it was possible to make whole-brain neuronal recordings (Ahrens et al., 2013). As a result, the zebrafish has become an important model organism for studying the neural basis of behavior in "naturalistic" settings (Privat & Sumbre, 2020; Mu et al., 2020). The larvae move in discrete bursts of propulsion, known as bouts, which are highly varied in order to execute a range of behaviors (Marques et al., 2018). The short bursts of tail oscillations address the challenge of determining the timescale and timing for segmenting motifs, which is a common issue for animals with continuous motion where motifs are not as clearly deliminated one from the next (Stephens, Johnson-Kerner, Bialek, & Ryu, 2008; Wiltschko et al., 2015; Berman, Choi, Bialek, & Shaevitz, 2014). Therefore each swim bout can be treated as a distinct unit of behavior. The next challenge is classifying these units of behavior. Studies by Marques et al. (2018); Mearns et al. (2020); John-

son et al. (2020) used unsupervised techniques to capture and classify the variability of bout types. These studies have shown that different bout types are associated with different contexts, such as prey capture, escape, and navigation.

However, these methods have some drawbacks. First, there is disagreement among the methods regarding the number of bout types, and they are constrained by specific tasks, limiting their generalization. Second, these methods rely on selecting gross kinematic features of bouts, which introduces bias in classification. Finally, these methods are not generative, meaning that they cannot produce similar bouts to simulate behavior.

These challenges motivated us to model behavior using a more flexible dynamical system. We adopted the iLQR-VAE framework from Schimel et al. (2022) to learn the underlying latent dynamics that capture the naturalistic behavior space of free-swimming zebrafish. Additionally, bouts generated from this dynamical system are initiated with sparse external input or, alternatively, upstream control.

Learning a dynamical system that recapitulates the behavioral repertoire addresses the three highlighted challenges. On condition that the training data included a large sample of fish and stimuli, the trained dynamics will generalize across all bout types, and thus unobserved bouts can be smoothly interpolated between existing bout types. This removes the need to specify the number of bout types. The training uses the raw tail angle recordings, which over-comes the need to select kinematic features of movement as relevant in describing behavior. iLQR-VAE learns a low-dimensional, smooth control space that can be sampled to generate new bouts, also bridging the challenge between analyzing behavior as a categorical or continuous problem.

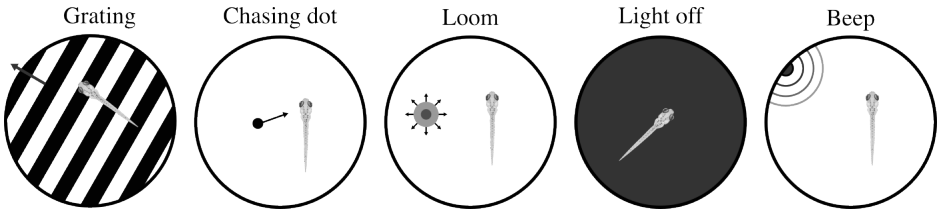


Figure 3.1. **Range of visual and acoustic stimuli to mimic different environments, encouraging the fish to display a variety of behaviors.** From left to right: forward-moving and rotating gratings to elicit the Opto-motor Response (OMR); an approaching fixed-size dot in a virtual open-loop setup, evoking escape sequences; a dot that gradually increases in size, with the looming dot eliciting an escape response; a transition from a light to a dark environment; a beep sound triggering an acoustic startle response.

3.3 Background

3.3.1 Visual stimuli to replicate environmental contexts

In order to elicit the full repertoire of zebrafish behavior, we presented a range of visual stimuli to the fish. The stimuli were designed to replicate different environmental situations they encounter in the wild. The stimuli presented consisted of:

1. **OMR grating**, the fish prefers to align to the grating and minimize optical flow, and therefore elicits a series of forward or turn swims (Orger et al., 2000; Naumann et al., 2016).

2. **Chasing dot** (approaching) the fish avoids the dot stimuli with a sequence of escape and burst swims (Groneberg et al., 2020).

3. **Looming dot** (radially expanding), the fish frequently responds with an O-bend or spot avoidance turn (Dunn et al., 2016).

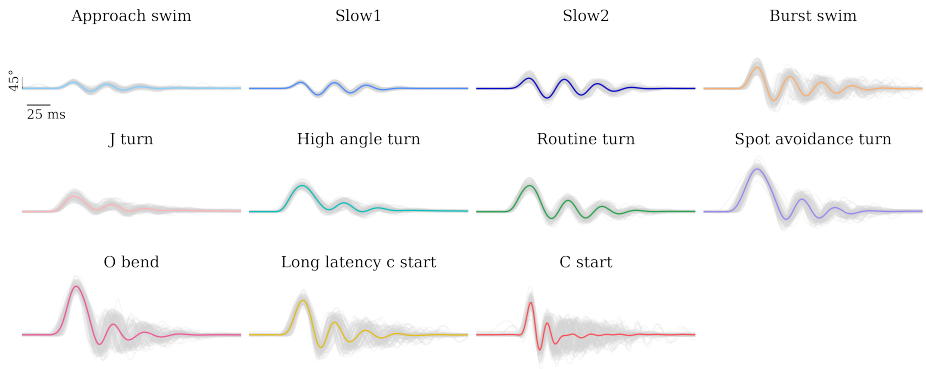


Figure 3.2. Tracked tail angle contain a variety of kinematic features that differ in different contexts to ensure the motor output facilitates the goal. The tail segments time series can be further abstracted to a vector of kinematic parameters, e.g., tail beat frequency, maximum tail amplitude. Unsupervised bout classification determined from bout kinematic features using this key for different bout types.

4. **Light to dark transition** triggers an O-bend turn at the onset (Burgess & Granato, 2007).

5. **Sound beep** triggers an acoustic startle response Marques et al. (2018).

3.3.2 Balanced training dataset

To train the model, we used a dataset collected by Pedro Tomás Silva (Jouary et al., 2024). The dataset was collected using a high-speed camera recording at 700 frames per second and tracked seven tail segments along the fish body. A total of 120 fish were exposed to different visual stimuli, as detailed in Section 3.3.1. Tail tracking data was segmented and classified into 11 distinct bout types using Marques et al. (2018) bout classification algorithm, Figure 3.2. We excluded capture swim categories, as they were largely absent from this dataset. We then equally

sampled 1200 bouts of each category and truncated to length 200 ms, accumulating a dataset of 36000 swim bouts.

3.3.3 Details of the generative model

The generative model, which captures the dynamics of the observed data, is governed by a deterministic latent dynamical system, for more details see Section 2.5.4. We selected a Minimal Gated Unit (MGU) RNN to fit the larval zebrafish dataset (Zhou et al., 2016). The deterministic latent dynamics in Equation (2.42) are defined as,

$$f_{t+1} = \sigma(U_f x_t) \tag{3.1}$$

$$\hat{x}_{t+1} = \rho(U_h(f_{t+1}) \odot x_t) + W u_{t+1} \tag{3.2}$$

$$x_{t+1} = (1 - f_{t+1}) \odot x_t + f_{t+1} \odot \hat{x}_{t+1} \tag{3.3}$$

where the non-linearity functions σ and ρ were defined as tanh functions.

Additionally, the generative model has a sparsity prior on the external input u_t, where the inputs are sampled from a Student-t distribution, defined as,

$$p_\theta(u_t) = \frac{\Gamma\left[(\nu + m)/2\right]}{\Gamma\left[\nu/2\right](\nu\pi)^{m2}|S|}\left[1 + \frac{1}{\nu}u_t^T S u_t\right]^{-(\nu+m)/2} \tag{3.4}$$

where S is a diagonal matrix with sparsity coefficients, m is the number of input dimensions, and ν is the degrees of freedom. Penalizing non-sparse control encourages the dynamics to be stored in the network, rather than the inputs driving dynamics and reflecting the bout.

Table 3.1. Benchmark model parameters (mean ± SEM)

n	**m**	**MGU**	**Linear**	$\mathbf{R}^2$	$\|\|\boldsymbol{u}\|\|_1/\|\|\boldsymbol{u}\|\|_\infty$
60	5	✓	-	0.974	21.7
60	10	✓	-	0.975	13.9
90	5	✓	-	0.976	7.88
90	10	✓	-	0.976	9.43
90	15	✓	-	0.976	4.59
120	5	✓	-	0.976	8.26
120	10	✓	-	0.983 ± 0.000	2.29 ± 0.05
120	15	✓	-	0.986 ± 0.000	4.71 ± 0.03
120	20	✓	-	0.974	3.78
120	10	-	✓	0.971 ± 0.000	2.75 ± 0.03
80	10	-	✓	0.979 ± 0.00	3.06 ± 0.08
60	10	-	✓	0.985 ± 0.00	3.97 ± 0.10
40	10	-	✓	0.986 ± 0.00	5.76 ± 0.20
20	10	-	✓	0.987	15.84
18	10	-	✓	0.985 ± 0.002	23.5 ± 6.2
15	10	-	✓	0.985 ± 0.000	28.7 ± 5.2

3.4 Results

3.4.1 Model selection

The dimensionality of the state space and control space was investigated to determine which models could reconstruct the variety of behavior, and do so in a sparse manner. We performed a grid search over the number of states and controls, and compared the R^2-score of tail reconstruction and the sparsity of the control space. As shown in Table 3.1, across the parameters space, the generative model recaptured the entire bout space with R^2-score$>$ 0.97. However, sparser control was favored with a larger network and control space. We selected a model with state dimension $n = 120$ and control size $m = 10$ as it achieved the high R^2-score of 0.983

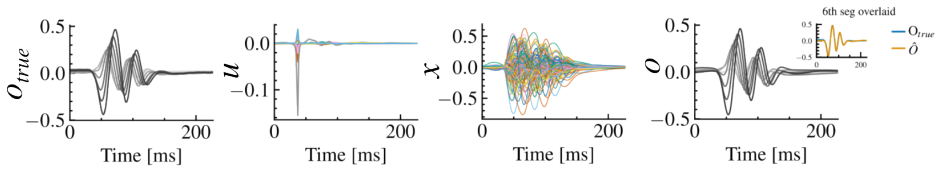


Figure 3.3. **Impulse-like control sufficient at generating bout in learnt dynamics**. Example of single bout reconstructed with impulse-like control and fitting with $R^2 = 0.99$.

with the sparsest control $||\boldsymbol{u}||_1/|\boldsymbol{u}||_\infty = 2.29$ to generate bouts throughout the dataset, an example bout shown in Figure 3.3.

Model robustness

The model fitted the data and inferred sparse control to generate bouts. To ensure that the model's solution was robust to input perturbations, we smoothed the inferred control signal used to generate a bout in Figure 3.4a. Throughout the broadening of the input peak, bouts were still recapitulated.

We compared the embedding of swims in the latent space with multiple RNNs of same dimensionality but trained with different initializations; see Figure 3.4b-d. We observed similar embedding in 2 dimensions using a dynamical adaptation of Canonical Component Analysis (CCA) known as SVCCA (Raghu et al., 2017).

3.4.2 Sparse control representation of zebrafish behavior

A key feature of the iLQR-VAE model is that it produces a model that can generate entire bouts from a sparse control signal Schimel et al. (2022). The ballistic nature of bouts encouraged most of the external inputs in the model to be captured at the beginning of a bout, as shown in Figure 3.5. Across the entire dataset reconstructing the tail angle solely from

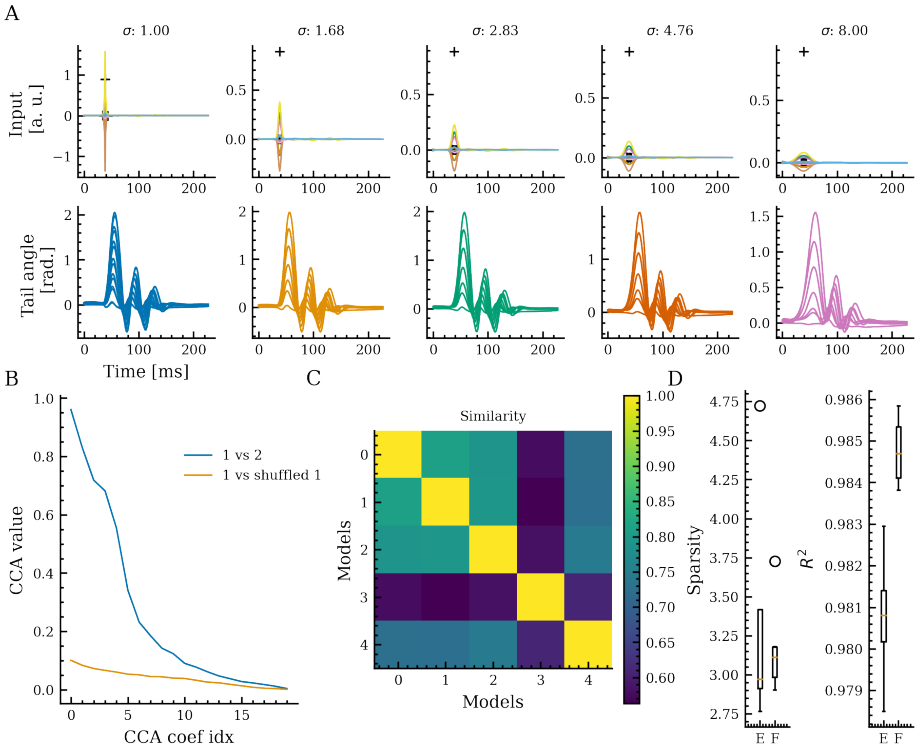


Figure 3.4. **Robustness of the learned model**. (a) Smoothing the input signal to generate movements using a Gaussian kernel $\sigma = \{1.00, 1.68, 2.83, 4.76, 8.00\}$ had small effect on reconstructing the original movement. (b) Using SVCCA to compare the initial latent state representation of two independently initialized models (blue) and a control with shuffled labels for the initial state of the same model (orange). (c) The similarity matrix of the top 3 principal components of the initial state across 5 independently initialized models. (d) A comparison of sparsity and reconstruction for these models when trained on the full bout dataset (F) and the dataset excluding a specific swim type during training (E).

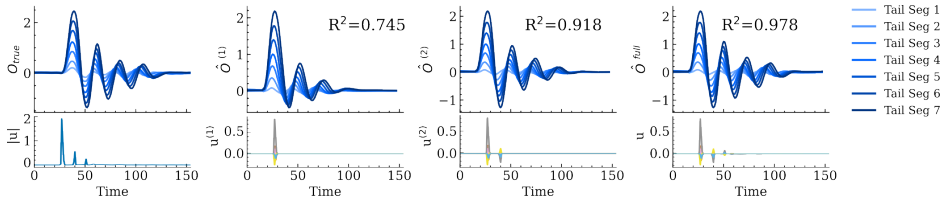


Figure 3.5. The initial control peak is essential for generating ballistic bouts and align with the onset. Additional peaks contribute to the finer control of locomotion. Left to right, illustrates the true observed bout, the generated bout with a single peak, generated bout from two peaks, the generated bout from the full control sequence.

the initial peak of the control signal resulted in a R^2 score of 0.82. Figure 3.5 has a secondary smaller peak in constructing the bout, simulating the bout without this additional input mode misses the finer, subsequent dynamics of the tail movement. The additional input peaks could reflect feedback correcting for tail dynamics, or the generation of more complex movements.

Therefore, after the initial control peak, the dynamics of the generated swim bout can be approximated as an autonomous system. We can interpret the initial control spike as setting the initial state of the network (Kaufman et al., 2014), which then evolves autonomously to generate the rest of the bout. In Figure 3.6b, we aligned all bouts on the initial peak of the control signal, then obtained the 120-dimensional state of the RNN.

With the prior classification method of Marques et al. (2018), we sorted the inferred control peaks on different bout types. Distinct control patterns reflected different bout types, Figure 3.6a. This reflects similar concepts in the literature that describe locomotion generation as being coordinated by distributed networks of hindbrain neurons (Gahtan et al., 2002; Carbo-Tano et al., 2023). Figure 3.6a shows that there is not a single control that

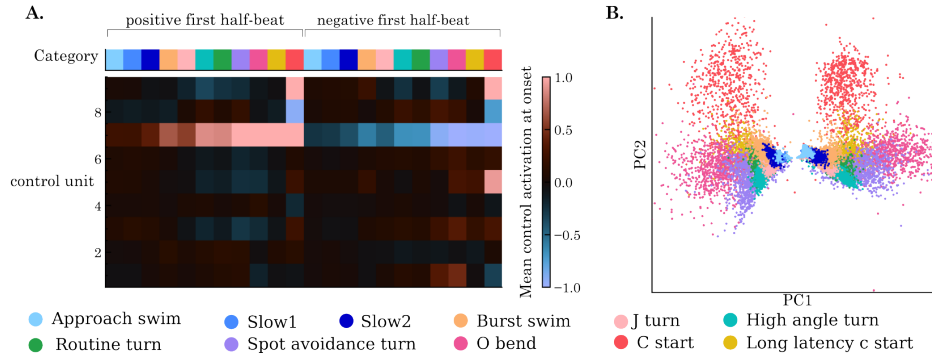


Figure 3.6. **Recruitment of distinct control pattern in generating different bout types**. (a) The average activation during the first peak of each bout category of tail movement. (b) 2-D projection of the 10-dimensional peak in control signal.

is specific to one bout type. Rather, the controls are recruited collectively in constructing different bout types, and individual control components are dominant for different types of swims. The initial input peak provides a low-dimensional sparse description of movement that can be smoothly interpolated to generate a wide range of swim bouts. Aligning swim bouts after the initial inferred control peak, we projected the 120-dimensional state into a 2-D PC space, Figure 3.6b. The PC map shows distinct clusters that corresponded to different bout types. We benchmarked the model's ability to classify bout types depending on the network state, Figure 3.7. Interestingly, from a single snapshot of the network's state, the model discriminated bout types better than if the entire posture time series was used to classify bouts.

3.4.3 Decoding kinematic features of movement from control

Previous studies have successfully classified swims into bout types by clustering on kinematic features of the swim (Marques et al., 2018). We used

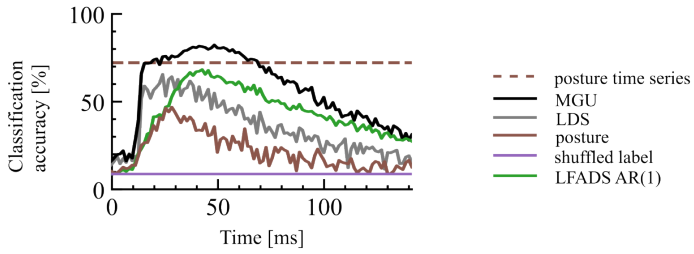


Figure 3.7. **Non-linear network classification out-performs posture space and other models**. Performed a linear bout classification to labeled bout types of the network state at different timepoints. There was a period where the non-linear network start out-performed the entire time series of the tail posture.

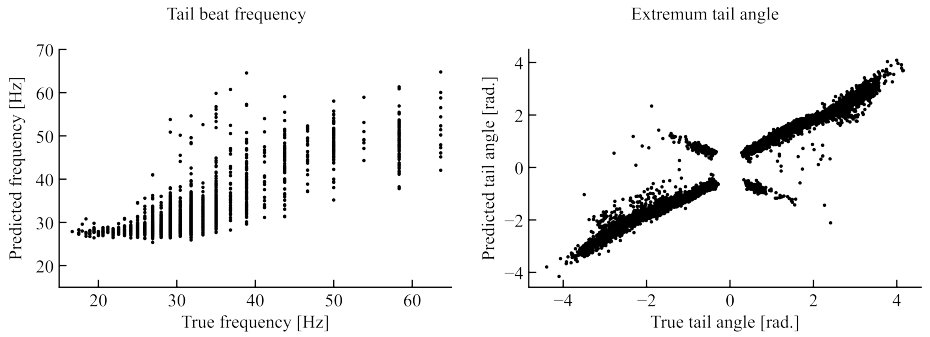


Figure 3.8. **Linear decoding of tail oscillations features following the first control peak of the network.** Linear regression with L1-Norm constraint of the initial network state (state reached after the peak control signal) to predict the tail beat frequency ($R^2 = 0.85$) and maximum tail beat amplitude ($R^2 = 0.96$).

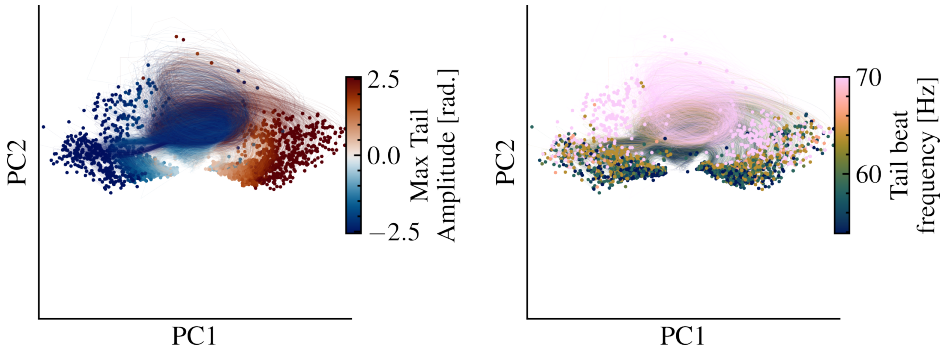


Figure 3.9. **Tail amplitude and tail beat frequency encoded in latent space**. The first PC encode for the maximum tail amplitude of the generated swim (left). The second PC encodes for the tail beat frequency (right).

a different approach that learned a dynamical system trained on many tail angle time series. We tested the model's ability to decode kinematic features of movements from the inferred control signal. This would demonstrate that the low-dimensional control space learned a representation that captures information relevant to the kinematics of the swim bout. The network state after the initial peak of the control signal was able to linearly predict the tail beat frequency ($R^2 = 0.85$) and the maximum tail beat amplitude ($R^2 = 0.96$), as shown in Figure 3.8. Furthermore, the latent trajectories projected in the PC space showed that the first PC captured the right-left tail amplitude, and the second PC captured the tail beat frequency, as shown in Figure 3.9D. This suggests that the model captures information relevant to the kinematics of the swim bout.

3.4.4 Linear decoding of swim trajectory given initial control peak

We have shown the capability of sparse control in capturing kinematic swim features and classifying swim types. All of these features are implic-

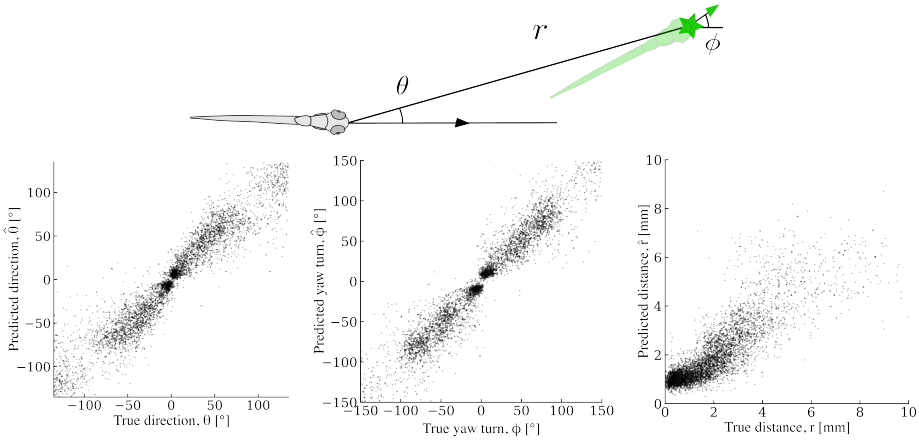


Figure 3.10. **Linear decoding of navigation features following the first control peak of the network**. Schematic outlining navigation features swim bout decodes to achieve target future positions, turn direction θ, swim distance r, swim yaw ϕ. Initial state decode swim direction with $R^2 = 0.95$ (a), swim distance with $R^2 = 0.85$ (b), swim yaw with $R^2 = 0.97$ (c), and turn direction with $R^2 = 0.99$.

itly encoded in the tail angle trajectories and thus in the training. We considered the control signal's ability to predict features not directly encoded in the tail angle, and that were not explicitly used for training. For example, goal-directed movements depend on control of trajectory parameters. These include the change in heading direction, ϕ, angle from the starting position, θ, and distance to the target, r. These parameters should be adequate for it to generate the necessary bout to perform the goal-directed task successfully, see Figure 3.10. We fitted a linear regression model to predict the swim direction ($R^2 = 0.95$), swim distance ($R^2 = 0.85$), and swim yaw ($R^2 = 0.97$) from the initial position of the network. This approach did not require the model to resolve the inverse problem, but relied solely on the initial state following the first input peak. We believe that this method might be ineffective for highly controlled swims, such

as capture swims, which require continuous sensory feedback. However, this approach provides a simple strategy for controlling spatial navigation composed of stereotyped, ballistic swims.

3.4.5 Synchronization of tail and pectoral fin dynamics in larval zebrafish locomotion

The primary role of the pectoral fins in larval zebrafish is respiration and motor balance (Hale, 2014). The pectoral fins move in synchrony with the body during locomotion; however, they are not essential for driving movement (Green & Hale, 2012). While the fins help maintain body orientation and stability (Ehrlich & Schoppik, 2019), different combinations of body and fin movements generate distinct trajectories. For example, during steering maneuvers, fin activity increases while the body's contribution decreases (Ehrlich & Schoppik, 2019). During steady swimming, the pectoral fins move rhythmically (Green et al., 2011; Budick & O'Malley, 2000) and can function either independently or in synchrony with each other.

While we previously explored how upstream descending control drives CPGs that activate body movement, we were interested in how these CPGs are shared across pectoral fin movements. We addressed this using a dataset of 64 larval zebrafish swimming spontaneously for 10 minutes, collected by Katharina Kötter, Portugues lab. Pectoral fins were tracked relative to the body centerline as well as the tracked tail angles, shown in Figure 3.11. We performed inferences using the pre-trained RNN from Section 3.4.1 to generate latent trajectories and reconstruct the corresponding tail movements. In these bouts, the initial tail beat aligned with a half-beat of the pectoral fin, with right-to-left leading bouts showing a corresponding left-to-right pectoral fin beat (Green et al., 2011), as illustrated in Figure 3.11b.

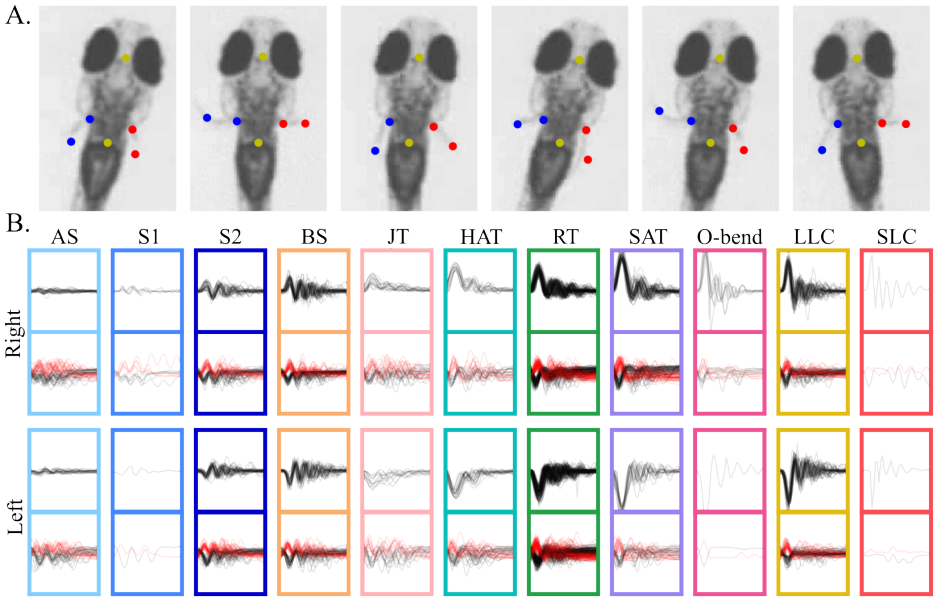


Figure 3.11. **Tracking pectoral fin dynamics across different bout types.** (A) Pectoral fin tracking was performed using SLEAP (Pereira et al., 2022) to track the position of the right (red) and left (blue) pectoral fins relative to the body centerline (yellow). (B) The top row shows 200 ms bout traces of the 7th segment tail angle during spontaneous swimming. Each bout was categorized into a bout type using Megabouts. Below are the corresponding left (red) and right (black) segmented fin traces. The fin and tail traces were upsampled from 500 Hz to 700 Hz.

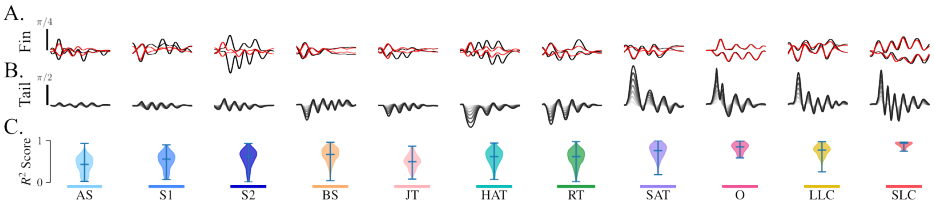


Figure 3.12. **Latent trajectory reconstruction of pectoral fin dynamics highlights movement coupling with body dynamics in specific swim bout types.** (A) Are examples of the predicted pectoral fin time series (red) and the actual pectoral fin time series (black) from latent trajectories used to construct for a swim bout. (B) Examples of the swim bout corresponding to the pectoral fin dynamics in (A). (C) The R^2 reconstruction of pectoral fin dynamics value was calculated for each swim type. More propulsive swims, such as burst and escapes, have higher R^2 values, indicating that the latent dynamics in generating movement are more predictive of the pectoral fin dynamics. Highlighting regime where the body movement is coupled with fin movement.

To investigate whether pectoral fins shared similar underlying dynamics used to generate tail movements, we linearly projected inferred latent dynamics onto pectoral fin dynamics using L2-regularization (test MSE = 0.0082; train MSE = 0.0278). Notably, pectoral fins display distinct gaits, often tucking along the body during high-velocity swimming for hydrodynamic efficiency (Thorsen et al., 2004). We compared pectoral fin reconstruction across different swim types. In Figure 3.12, slow swimming modes, including approach swims, Slow 1 and 2, and J-turns, showed low R^2 scores, suggesting an uncoupled dynamic from that of tail movements. This significant variability in slow swims indicated an independent upstream control distinct from tail kinematics.

3.5 Discussion

In this chapter we demonstrated that a range of movements in the larval zebrafish can be captured from a single dynamical system. External control input used to drive these movements were mostly sparse and centered around the onset of the movement. Using the observation that most control was recruited during the onset, essentially initiating the movement, we were able to define an initial state of the network for movement generation. Regression analysis revealed that the initial control peak could predict kinematic features of the movements, suggesting that for ballistic movements, the initial control peak captures essential features of the movement.

A significant contribution of our model is its ability to infer control that aligns with existing bout classification methods (Marques et al., 2018; Johnson et al., 2020; Mearns et al., 2020). The ability of the state network to better classify bout types compared to the entire tail angle time series underscores the potential of our approach to simplify complex behavioral time series data into a single vector-encoded representation. The approach offers a promising avenue for future behavioral modeling studies, particularly in terms of simplifying complex kinematic data into more manageable representations (Wiltschko et al., 2015; Berman et al., 2014). Additionally, the generative nature of the model allows for synthetic behavior to be generated and manipulated, providing a platform for future studies to explore the effects of different control inputs on the locomotor repertoire.

Moreover, the low-dimensional vector representation of behavior our model produces provides a novel approach to behavior classification. This representation effectively discards the traditional problem of discretizing behavior over time, which often involves cumbersome kinematic feature extraction or clustering. Instead, our model presents a framework where

behavior can be characterized by a reduced set of continuous parameters. This advance is particularly valuable for overcoming the limitations of traditional kinematic clustering methods, which often struggle to capture the continuous nature of movement over different timescales. By doing so, our model not only enhances classification but also contributes to a more holistic understanding of how animals organize their behaviors in response to varying environmental and internal cues.

An advantage of our approach is that there is a separation between the upstream driving inputs and dynamics. By having a general dynamical system flexible to capture a wide range of movements, distinct control sequences can be mapped to different bout types. This reflects a similar structure to the brain's descending control to CPGs in movement generation (Grillner & El Manira, 2020). The sparsity constraint on the inputs helped further isolate the control from dynamics, instead of the inputs driving most of the dynamics. The choice of sparse control was not just an engineering decision but based on empirical finding which demonstrated impulses of electrical stimulation to nMLF (Severi et al., 2014), and RS neurons elicit distinct locomotor patterns (Xu et al., 2021). One caveat is that the trained non-linear model, although capturing a range of movements, may be too flexible and so inferred control cannot be translated to descending control neurons in the hindbrain. An improvement would be to train the dynamics on behavior and neural data such that the learning implicitly captures neural activity, similar to other recent works that have used joint behavior-neural model in reconstructing behavior Hurwitz et al. (2021); Gondur et al. (2024); Abbaspourazad et al. (2023). Alternatively, the dynamical system could have a bio-inspired architecture (Knüsel et al., 2020) where the parameters learned reflect descending coupled-oscillators.

In conclusion, this approach offers a versatile tool for both understanding and classifying behavior in a way that is biologically plausible and compu-

tationally efficient. Future studies may expand on this framework by exploring its application to other species and behavioral paradigms (Ahamed et al., 2020; Markov et al., 2021), ultimately enriching our understanding of the principles governing animal movement and control.

3.6 Acknowledgements

The data was collected by P. T. Silva and A. Jouary. The data was labelled using bout classification developed from Marques et al. (2018). The data with pectoral fin tracking was collected by K. Kötter under the supervision of R. Portugues. The data was preprocessed by A. Jouary. The iLQR-VAE toolbox was developed and adapted by Schimel et al. (2022). This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service. I am grateful for the discussions with G. Hennequin, A. Jouary, M., C. Machens and M. B. Orger.

Chapter 4

Interpretable dynamics of movement generation in Larval Zebrafish

4.1 Chapter Summary

An important aspect of system identification is the ability to interpret the dynamics of the system. In the context of linear dynamical systems, matrix decomposition provides a way to interpret the dynamics of the system as collective modes. These collective modes can be used to see how the system behaves under different conditions. Furthermore, regions in the state space that are hard to steer or readout can be identified through the controllability and observability Gramians of the system. Often, we can compress the system to discard these regions with little input-output variance using balanced model reduction to form a minimal model of our linear dynamical system.

In this chapter, we discuss methods for interpreting linear dynamical systems using matrix decomposition and balanced model reduction. We fit a linear RNN to larval zebrafish behavior using the iLQR-VAE and derive a reduced model that highlights distinct dynamical modes recruited for generating different types of swims. Moreover, this reduced model reflects a compressed signature of the underlying dynamics of the behavior. We used this signature to compare changes in behavior dynamics in fish exposed to ethanol. We showed that while the model learned a similar reduced representation, the inferred control space was adjusted with more persistent activity to capture ethanol-induced movements.

4.2 Introduction

The brain's ability to encode and process information stems from its network's rich variety of temporal activity patterns. Recurrent network models have proven to be a powerful tool for modeling non-linear dynamics of neural and behavioral processes (Sussillo et al., 2015; Hennequin et al., 2014). They show a range of output patterns, including transient and persistent dynamics, as well as periodic and chaotic behaviors (Vyas et al., 2020; Vogels et al., 2005). Often they are trained on measured neural and behavior datasets and, therefore, inherit their dynamical properties (Sussillo & Barak, 2013; Sussillo et al., 2015; Mante et al., 2013; Churchland et al., 2012). Much of the functional properties are encoded in the network architecture, yet mapping structure to function is often a challenge (Mastrogiuseppe & Ostojic, 2018; Litwin-Kumar et al., 2017).

Interpreting the RNN's solution is important in understanding how different model components contribute to its output. One approach is to force the RNN to preserve a network structure that can be factorized into different modes (Dubreuil et al., 2022), or have known fixed points (Linderman et al., 2016). For linear dynamical systems, the state dynamics can be factorized into interpretable components using matrix decomposition methods, such as spectral decomposition and Schur decomposition (Hennequin et al., 2012). Factorizing the state dynamics reduces the problem to understanding how a set of modes evolve and interact with each other. Often, sorting these modes provides a systematic way to discard redundant modes, further reducing the dimensionality of the problem (Schmid, 2022; Kutz et al., 2016).

We fitted an expressive linear RNN to model the dynamics of zebrafish behavior using iLQR-VAE (Schimel et al., 2022). Then, we applied balanced model reduction to reduce an over-parameterized model to a minimal form, and interpreted the dynamics of the system. Initially, the trained network

had a high degree of non-normal dynamics but after model truncation the dynamics were normal. This allowed us to interpret the dynamics of the reduced system using spectral decomposition. We compared the spectral frequencies of the reduced model to identify the dominant modes of the system and their recruitment for different bout types. The spectral decomposition of the reduced model provided a compressed representation of the underlying dynamics of the behavioral dataset. We used this to investigate differences in underlying dynamics between datasets with pharmacological manipulations, specifically ethanol-treated fish, using a dataset collected by Lackner (2018). Balanced model reduction showed that the reduced dynamics of the drug-induced model was similar to those of the untreated control model. We performed cross-model inferences and showed the average inferred control for the Slow 2, routine turns, and long-latency C-starts in ethanol-treated swims were significantly more persistent than the untreated cohort. This reverse engineering approach provides a method for learning interpretable dynamics from high-dimensional models and can be used to investigate the effects of pharmaceuticals on neural control of behavior.

4.3 Background

In this section, we provide an overview of methods used to interpret dynamics fitted to non-linear systems. Next, we briefly highlight the architecture of the linear iLQR-VAE model used to capture the behavior of zebrafish. Finally, we outline the method used to reduce the linear model using balanced model reduction.

4.3.1 Understanding the language of dynamics

Complex patterns of activity generated by non-linear systems are best described by their dynamical structures, such as line attractors, fixed points,

and limit cycles. Identifying these features provides a language for describing the properties of the system (Durstewitz et al., 2023). Interpreting the dynamics of non-linear systems often requires methods that identify regimes where the dynamics can be linearly approximated. In such cases, standard linear methods can be applied to analyze the system's dynamics; see Appendix B.1. Here are a few methods used to linearly approximate non-linear systems.

Fixed point detection One approach is to find fixed points in the system, that is, when the dynamics are stationary, $\dot{x} = f(x^*, t) = 0$. Qualitative properties of the fixed points can be obtained, such as the stability of the fixed points (Golub & Sussillo, 2018; Mante et al., 2013; Duncker & Sahani, 2021). Furthermore, the local dynamics around the fixed points can be linearly approximated using their Jacobians. Sussillo and Barak (2013) developed a method to scan for slow points in the state space of a non-linear dynamical system and detect fixed points for such analysis.

Piecewise linear systems Another method for interpretable dynamics is training a piecewise switching Linear Dynamical System (sLDS) (Smith et al., 2021). Smith et al. (2021) trains in parallel to the RNN, a sLDS network which approximates the RNN dynamics and an auxiliary function producing the RNN's fixed-points. The loss has a regularization term that forces the sLDS to jump to different fixed points in the RNN. This method integrates the fixed point detection and linear approximation into a single model (Sussillo & Barak, 2013).

Delay-embedding linear manifolds Is a method that uses time-delay embedding to find a shadow manifold of the non-linear dynamics. According to Taken's theorem (Takens, 1981), increasing the spatial dimension-

ality with time-delays of the measurement, untangles the non-linearity of the dynamics, allowing for a linear shadow manifold to be formed. This is a common method in time series analysis and has been used in, for example, weather, economics, and neuroscience (Ahamed et al., 2020; Brunton et al., 2017).

Training large enough linear RNN to approximate non-linear behavior Similar to delayed-embedding, which increases the dimensionality of the state space, we proposed to train a high-dimensional linear RNN with "plenty of room" to capture the non-linear dynamics (Mullen et al., 2024). This approach starts with a highly expressive linear RNN to learn the complex behavior. Given the high dimensionality of the model, interpretability can only be achieved once the model is compressed. Balanced model reduction (Moore, 1981) is a PCA-type method that preserves axes of maximal variance in the input-output space of the LDS. So, as in PCA, regions of the state space that have minimal input-output variance are discarded.

4.3.2 Modeling behavior dynamics with iLQR-VAE

We use iLQR-VAE (Schimel et al., 2022) to model the dynamics of the behavior of zebrafish. As outlined in Section 2.5.4, the generative model has a sparse input prior, as in Equation (3.4), and a linear Gaussian readout. However, in this case, the deterministic linear latent dynamics are modeled as

$$x_{t+1} = f_\theta(x_t, u_t) = Ax_t + Bu_t \tag{4.1}$$

where $x_t \in \mathbb{R}^n$ is the state vector at time t, $u_t \in \mathbb{R}^m$ is the control input at time t, and $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ are the state transition and control

matrices, respectively. The latent trajectories are projected on the tail from a Gaussian distribution,

$$o_t \sim \mathcal{N}(Cx_t, \Sigma) \tag{4.2}$$

where $o_t \in \mathbb{R}^{n_o}$ is the observation at time t, $C \in \mathbb{R}^{n_o \times n}$ is the observation matrix and $\Sigma \in \mathbb{R}^{n_o \times n_o}$ is the observation noise covariance matrix. The parameters $(A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}, C \in \mathbb{R}^{n_o \times n})$ describe the linear dynamical system.

4.4 Balanced model reduction

The Linear Dynamical System (LDS), outlined in Equations (4.1) and (4.2) are characterized by parameters ($A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{n_o \times n}$). Methods like Schur decomposition or eigen decomposition can be used to interpret the state dynamics, A; see Appendix B.1 for more details. However, these methods do not capture the extent to which inputs affect the system and how sensitive observations are to dynamical changes in the network. We can investigate this relationship using controllability and observability, which identify properties of how the input matrix B and the readout matrix C propagate through the state dynamics A, for details see Appendix B.2. A basis can be found that rotates the LDS to optimally capture the input-output relationship of the state space model, this is known as balanced realization.

Balanced model reduction is performed in two stages; in the first step, the original dynamics (A, B, C) are transformed into an equivalent balanced system ($\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{B}}$, $\tilde{\boldsymbol{C}}$). That is, a dynamical system with the same transfer function as the original model, but in which each mode is as controllable

as it is observable. In the second stage, this balanced realization of the original system is truncated to a desired order $r < n$ (Laub et al., 1987).

4.4.1 Balanced realization

Balancing the system in Step 1 relies on computing its observability and controllability Gramians (Scherpen, 2011). Those are positive-definite matrices defined, respectively, as

$$W_o = \int_0^\infty e^{A^T t} C^T C e^{At} dt, \tag{4.3}$$

$$W_c = \int_0^\infty e^{At} B B^T e^{A^T t} dt. \tag{4.4}$$

They capture the information about the state space directions in which the system is most sensitive to inputs, and those in which it is most likely to elicit outputs. More precisely, the top eigen modes of W_o correspond to the most observable directions of the system, that is, the directions that will lead to the most variance in the output when the network is initialized along them. The top eigen modes of W_c are the most controllable directions of the linear system, that is, correspond to directions along which the dynamics can be steered with minimal input energy cost (Kao & Hennequin, 2019). Importantly, given (A, B, C), W_o and W_c can be computed in closed form as solutions to two dual Lyapunov equations,

$$A^T W_o + W_o A = -C^T C \tag{4.5}$$

$$A W_c + W_c A^T = -B B^T. \tag{4.6}$$

Balanced realization relied on a T basis transformation such that both Gramians are diagonal and equivalent, that is,

$$\tilde{W}_o = \tilde{W}_c = \Sigma^2 = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2) \tag{4.7}$$

where $\tilde{W}_o = T^T W_o T$, $\tilde{W}_c = T^{-1} W_c (T^{-1})^T$, and $\sigma_i, i = 1, \ldots, n$ are the ordered, positive Hankel Singular Values (HSV) of the system; see Appendix B.3 for details in finding the basis T.

4.4.2 Balanced Truncation

The HSVs have the same role for the dynamical system as singular values do for constant, finite-size matrices. Thus, in the same way as the decay of singular values can be used to choose the cutoff point of dimensionality in PCA, evaluating the HSV spectrum provides a principled way of deciding the dimension of truncation r. Once r has been set, the system is truncated by selecting the first $r \times r$ block of $\tilde{A}$, $\tilde{A}^{(r)}$, the first r rows of $\tilde{B}$, $\tilde{B}^{(r)}$, and the first r columns of $\tilde{C}$, $\tilde{C}^{(r)}$. The dynamics of the reduced dynamical system can then be summarized as,

$$\tilde{x}_{t+1}^{(k)} = f_\theta(\tilde{x}_t^{(k)}, u_t^{(k)}) = \tilde{A}^{(r)} \tilde{x}_t^{(k)} + \tilde{B}^{(r)} u_t^{(k)}. \tag{4.8}$$

The removal of dimensions that are uncontrollable and unobservable provides a simple, yet effective way to reduce the latent dynamics of the system to a minimal form that can be interpreted and analyzed with more ease.

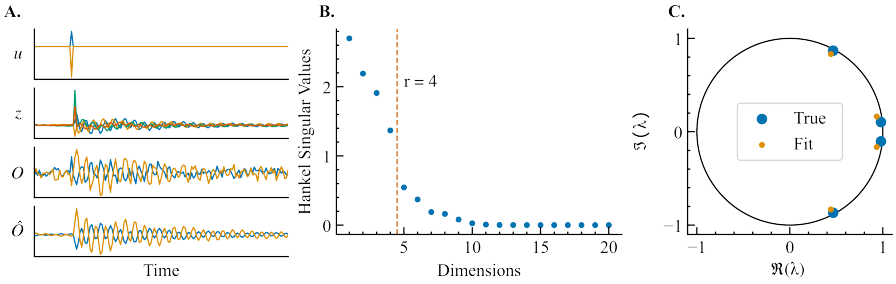


Figure 4.1. **Example case of balanced ordered model reduction in systems identification and recover true dimensionality of system.** (A) Latent coupled oscillator driven by unknown sparse forcing inputs. The control signal u is fed to coupled pendulums parameterized by z. iLQR-VAE learns from observations O and can reconstruct $\hat{O}$ after training. Here, blue and orange are used to denote the two input channels of the model, and the two observed outputs of the model. (B) Spectrum of the Hankel Singular Values (HSV). (C) Ground truth eigenvalues of the dynamics (blue), and eigenvalues of the truncated model (orange).

4.4.3 Balanced model reduction recovers the true dimensionality for a toy model

We illustrate the application of balanced model reduction as a method to recover the true dimensionality using a toy example. Specifically, we considered a toy example of a 2-dimensional latent coupled oscillator driven by unknown sparse forcing inputs; see Figure 4.1. We fitted an expressive 20-dimensional linear model and then reverse-engineered the trained model to find the minimal ordered model that captures the full dynamics of the true system. As shown in Figure 4.1B, the HSVs indicate the input-output variance of the system, and the sudden drop after the fourth component suggests that much of the dynamics can be captured in four components, consistent with the true system. The HSVs can be systematically used to determine the truncation dimension, r, of the balanced model.

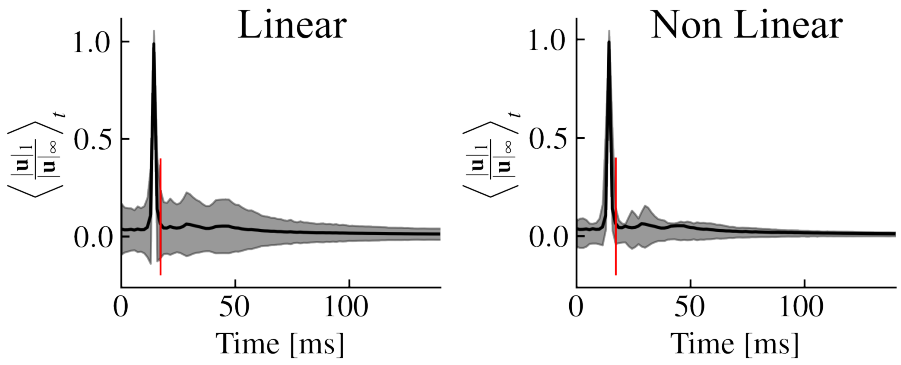


Figure 4.2. **Comparative input sparsity distribution through the linear model compared with the non-linear model.** Mean ± standard deviation across the dataset of the normalized L2-norm of u aligned on the first peak. (right) MGU RNN learns a more transient input to generate movements across the dataset compared to LDS (left). However, both models show that most of the input energy is contained in the first peak to generate bout.

4.5 Results

Here, we fitted the linear model to two different larval zebrafish datasets. The first was the balanced dataset described in Section 3.3.2. The other was a dataset of spontaneous behavior of ethanol-treated zebrafish larvae collected by (Lackner, 2018).

4.5.1 Linear model captures zebrafish behavior with sparse control

We fitted the iLQR-VAE to capture the behavior repertoire of the zebrafish, as in Chapter 3. However, we used a linear network of 120 dimensions with 10 control inputs to capture the system dynamics instead of the non-linear MGU network. The trained linear model was still capable of reconstructing bouts with sparse control Figure 4.2. Similar to

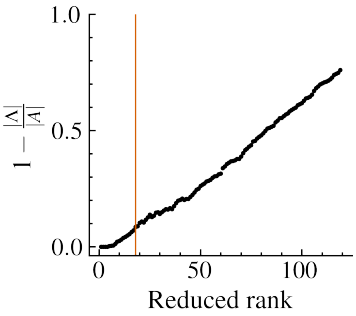


Figure 4.3. Henrici's departure metric is a measure of non-normal dynamics. It compares the ratio of the Frobenius norm of the diagonal to the full matrix measuring the remainder that lies in the triangle. The measurements taken at different ranks of balanced truncation show that minimal models display normal dynamics, but this drifts to non-normal dynamics as the rank tends closer to the full-rank model.

the non-linear model, most of the driving input was centered on the first peak to generate bouts. However, there were usually additional peaks to complete the bout compared to the non-linear model in Chapter 3.

4.5.2 Rank of balanced truncation changes dynamical regime

The ballistic nature of zebrafish locomotion is characterized by oscillations with transient growth followed by longer decay (Marques et al., 2018). The transient dynamics are usually captured by the non-normal dynamics in the Linear Dynamical System (LDS). We investigated the degree of non-normal dynamics in the trained model and after different orders of balanced truncation using the Henrici's departure metric. As shown in Figure 4.3, the full-rank model exhibited a high degree of non-normal dynamics, but balanced truncation forced the dynamics to become more normal with increasing rank of truncation.

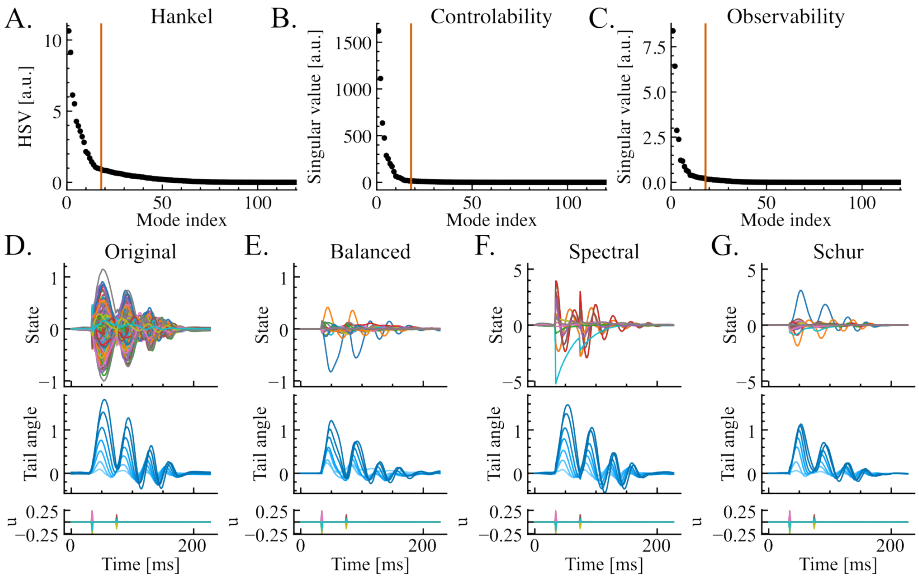


Figure 4.4. **System analysis and dynamics comparison for model reduction methods.** (A-C) The respective modes of the HSVs, the controllability and observability Gramians. The red vertical line indicates the threshold for mode truncation, which is set at mode 18. (D-G) Are the reconstructions of an example bout tail angle using of the full-rank model (D), the balanced model (E), the spectral decomposition on the full-rank model (F), and the Schur decomposition on full-rank model (G). The balanced, spectral and schur decomposition were all truncated to 18 modes. Each reconstruction shows the state trajectories (top), tail angle dynamics (middle), and control input u over time (bottom) for the respective methods.

4.5.3 Interpretable network dynamics in minimal zebrafish model

We selected the order of truncation to be 18, based on the drop-off of Hankel Singular Values (HSV)s in Figure 4.4a. The HSV spectrum is sorted by input-output variance of the balanced state space, which is weighted by

the controllability and observability Gramians; see Figure 4.4b–c. The reduced 18-dimensional balanced model was capable of reconstructing most bouts from the dataset, with a R^2-score$= 0.69 \pm 0.15$ compared to the full-rank model with R^2-score$= 0.93 \pm 0.05$. However, this excluded escape swims, such as long- and short-latency C-starts, which it struggled to reconstruct with the original inputs. A possible explanation could be the discarding of high-frequency modes in balanced model truncation (Green & Limebeer, 2012).

In order for the model to generate a swim bout, the inferred input must be projected into the state space, and complex interactions form patterns of activity that are eventually read out onto the tail segments; see Figure 4.4d. The trajectories of the latent states are often mixed in complex patterns due to the canonical structure of the state dynamics. The balanced transformation in Figure 4.4e rotates the state space to align the network in directions that capture the maximal input-output variation, which also unmixes the latent trajectories into balanced modes. In this example, there are two distinct balanced modes: one that captures a symmetric oscillation, and the other displays an oscillation with a decaying bias.

As Figure 4.3 suggested, the full-rank model was non-normal, we applied Schur decomposition directly to the full-rank state dynamics matrix and then truncated the model at rank 18 in Figure 4.4g. Despite not constraining the decomposition on it's controllable and observable regions, the Schur modes reflected a similar pattern to the balanced model reduction in Figure 4.4e, with transient growth and decay in the state space. This could not be observed when directly applying spectral decomposition to the full-rank state dynamics in Figure 4.4f. However, the non-uniqueness of Schur decomposition made it difficult to interpret and select the ordering of the eigenvalues.

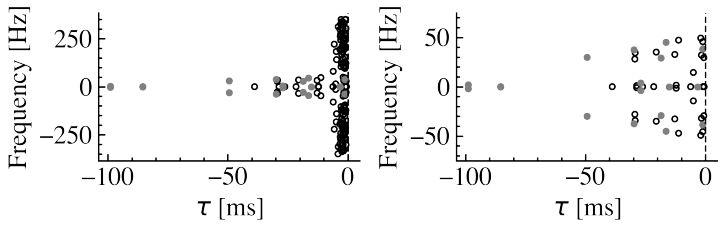


Figure 4.5. Comparative eigen spectrum of the full-rank (open-dots) and reduced model (gray dots). The full range of frequencies are shown in the left figure, which highlights the reduced model filters out high frequency oscillations. The lower frequencies in the reduced model have similar timescales as the full-rank model (right).

4.5.4 Spectral decomposition on reduced model reveal principal oscillators in zebrafish behavior

The normal dynamical regime of the reduced balanced model, shown in Figure 4.3, justified the use of spectral decomposition to further understand the different modes used to capture the dataset. The spectral decomposition of the reduced model identified 10 unique modes, excluding the complex conjugates; see Figure 4.5. The oscillatory modes in the frequency range of 20-50 Hz align with empirical observations of the tail oscillations (Severi et al., 2014), and cover a range of long time-constant frequencies, similar to the full-rank model. The decay time constants of these modes range from 17-92 ms Figure 4.6. The spectral decomposition of these modes in Figure 4.6 can be superposed to generate different bouts.

The contribution of each mode can be obtained by projecting the input through each eigenvector. We explored the contribution of each mode for Routine turns and Slow 2 swims, common in exploration and navigation (Marques et al., 2018; Mearns et al., 2020; Johnson et al., 2020). We found that both swim types recruit similar oscillatory modes, but strikingly, the routine turns additionally recruit a slow decaying flip mode. This

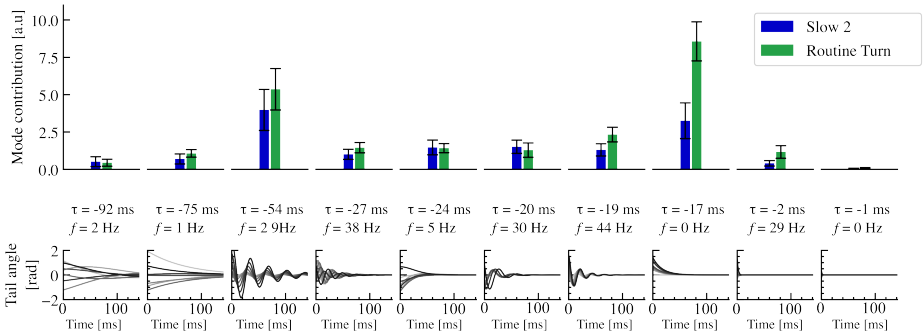


Figure 4.6. (Top) Distribution of the L^2-norm of each mode across routine turns (Green) and slow swims (Blue). (Bottom) Projection of impulse response into the readout for each of the 10 modes (8 complex conjugate eigenvalues and 2 real eigenvalues).

finding matches empirical observation in Huang et al. (2013), where ablations of ventromedial spinal projection neurons (vSPNs) RoV3, MiV1, MiV2 prevent fish from executing turns, so they respond with forward swims instead. This suggests that the flip mode is essential for the fish to execute turns, and oscillatory modes are recruited for both swim types.

4.5.5 Ethanol pharmaceutical perturbation shifts the dominant modes of zebrafish behavior

Zebrafish are used for drug screening, but often with behavioral fingerprints that only allow you to identify phenotypes (Wiltschko et al., 2020; Rihel et al., 2010). Previous works have highlighted that pharmacological perturbations, such as ethanol, modulate neural control of behavior (Dreosti et al., 2015; Bergeron et al., 2014; Ikeda et al., 2013; Guo et al., 2015). Previously, we demonstrated that balanced model reduction provides a compressed representation of the underlying dynamics of zebrafish locomotion, essentially a fingerprint for behavior. Furthermore, using our method, we can perform inference to obtain the control that

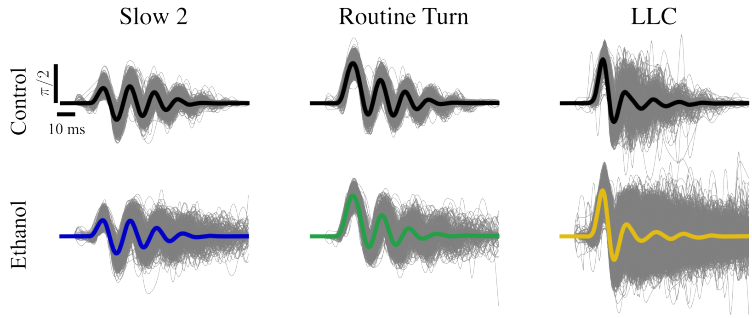


Figure 4.7. **Comparison of bout types across fish pre-treated and post-treated with ethanol drug.** Three bout categories (slow 2, routine turns and long-latency C-starts) were pulled from the pre-treated (top row) and post-treated (bottom row) datasets. The corresponding average 7^{th} segment tail angle overlaid on the individual traces. After 1% ethanol drug treatment, bout types appear to last longer than the pre-treatment control bouts.

drives a behavior. This can be used as a means to see how drug-induced neural perturbations alter the control of behavior.

We used a dataset collected by S. Lackner that recorded 20 fish during spontaneous behavior under acute 1% ethanol exposure (Lackner, 2018). As bout classification mostly depends on kinematic features from the first half beat, bouts could be reliably labeled at pre- and post-treatment despite kinematic changes occurring to the overall movement. As shown in Figure 4.7, within bout types, ethanol-treated bouts tended to extend with higher tail activity and were less stereotyped.

Due to the distinct differences in locomotion across bout types, we investigated if the underlying dynamics across treated and untreated fish differed. We fitted two separate linear models with 120 latent dimensions and 10 control inputs. One model was trained on 15,000 bouts pre-ethanol treatment (control cohort) and the other on 15,000 bouts post-ethanol treat-

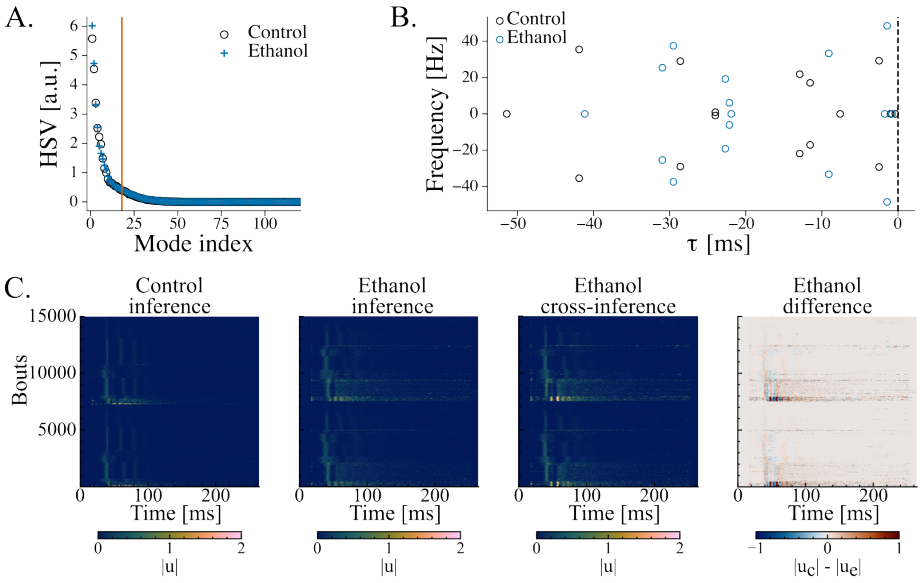


Figure 4.8. **The reduced model eigen spectrum of the ethanol-treated cohort shares similar dynamical modes with the untreated cohort.** (A) The HSVs for both ethanol and control indicate a comparable decay of significance and similar minimal dimensionality size. The rank of truncation was set at 18, as indicated by the red vertical line. (B) The reduced eigen spectrum of the untreated control (black) and ethanol-treated (blue) groups showed similar frequency characteristics. (C) From left to right: inferred control energy to generate bouts in the entire datasets for the untreated control, ethanol-treated group, ethanol-treated group using the untreated dynamics, and the difference in energy between the two inferences. Bouts were sorted by their classified right-left bout types. Inputs to generate the control dataset are sparser than the ethanol-treated data.

ment. Both successfully reconstructed bouts with R^2-score$= 0.98 \pm 0.05$ and R^2-score$= 0.98 \pm 0.07$, respectively. The full-rank models exhibited normal dynamics, with Henrici's departure metrics of 0.12 (untreated) and 0.08 (treated). Due to their similar HSV spectra, both were reduced to 18 modes; see Figure 4.8. The reduced models' spectral decomposition in Figure 4.8b displayed similar dynamical modes, indicating that similar dynamics were shared across the control and drug-induced models. Therefore, we performed inferences on the ethanol-induced dataset but now used the control model dynamics. Surprisingly, the control model was able to reconstruct the ethanol-induced bouts with R^2-score $= 0.97 \pm 0.08$. For the dataset, similar inferred input energy was recruited to generate the same bout under the two different dynamics; see Figure 4.8c.

As the treated and untreated models shared similar underlying dynamics, we compared how the control signals attribute to differences in bout types illustrated in Figure 4.7. Using the control model, we inferred the inputs to generate slow 2, routine turns, and long-latency C-starts (LLC)s; see Figure 4.9. The inferred control used to generate slow 2 and LLCs had a more persistent input compared to the control dataset.

4.6 Discussion

In this chapter we demonstrated that a linear dynamical system with sparse driving inputs could be trained to model the tail dynamics of the larval zebrafish behavioral repertoire. Furthermore, we decomposed the LDS and compressed a 120 state network to 18 dimensions using balanced model reduction. This reduced subspace was found to capture most of the variance in both the control input and the tail output spaces, highlighting its effectiveness in preserving key dynamical features of the original much larger system.

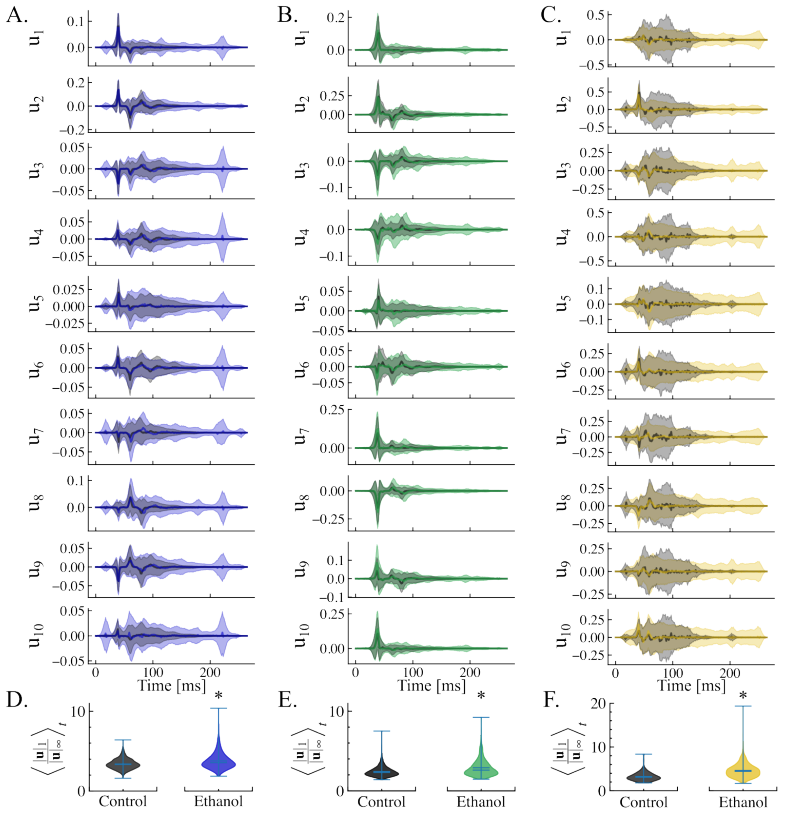


Figure 4.9. **Persistent control required to generate ethanol-treated bout types.** (A-C) The average activity of each input component used to generate Slow 2 (blue), Routine turns (green) and Long-latency C-starts (yellow), respectively, for ethanol-treated fish which is compared with the untreated cohort (black). All input components for slow 2 and long-latency C-start persist longer in the ethanol treated-fish. (D-F) The distribution of input sparsity, a two-tailed Mann-Whitney U test showed significant differences in the sparsity in control between pre- and post-treated ethanol fish. (D) For S2 swims $U = 969830.0$, $p = 3.0 \times 10^{-22}$ ($n_{pre} = 2179$; $n_{post} = 1119$). (E) For RT swims $U = 969830.0$, $p = 3.0 \times 10^{-22}$ ($n_{pre} = 1519$; $n_{post} = 1184$). (F) For LLCs swims $U = 51897.0$, $p = 1.1 \times 10^{-42}$ ($n_{pre} = 219$; $n_{post} = 1133$).

Furthermore, using spectral decomposition in the balanced subspace, we identified a set of oscillatory modes recruited within the neural network. The frequencies of these modes reflected a frequency range similar to that observed in the swimming behavior of the larvae (Severi et al., 2014; Marques et al., 2018). We illustrated how various control signals corresponding to different bout types recruited modes within the reduced model by different amounts to produce bouts. Similarly to ablation experiments from Huang et al. (2013), inferred control for routine turn swims recruited more decaying flip modes compared to Slow 1 forward swims. In reflection, the spectral decomposition of the reduced model is equivalent to selecting kinematic features of movement. In this case, the inferred control recruits the spectral modes, with different frequencies and timescales, by a different order depending on the movement. However, instead of previous methods that choose the kinematic features, such as tail beat frequency, these features are extracted in an unsupervised manner, based on the training dataset and the learned model.

Pharmacological manipulations alter the function of neural processes, which can lead to changes in outputted behavior. Bout categorization can help identify shifts in phenotypes after drug treatment (Rihel et al., 2010; Lackner, 2018). An advantage of our method is that it separately models the dynamics and control. Therefore, we have a way of grading the bouts depending on the driving control sequence. Training models individually with datasets of fish pre- and post- ethanol treatment, we showed that both models shared similar underlying dynamics. This suggested that the differences in behavior stemmed from the control inputs rather than the learned dynamics. This was validated when we compared the inferred control using the dynamics of the control model across the two cohorts for different bout types. Ethanol-induced movements required persistent control and there was greater variability in control patterns to generate the same bout types. Our method of inferring control, constrained by the

control cohort dynamics, provides a systematic framework for relating how drugs might act on locomotor output. This application could be extended to datasets with photo-stimulation and ablation of hindbrain neurons (Kimura et al., 2013; Jia & Wyart, 2024).

4.7 Acknowledgements

The data was collected by P. T. Silve and A. Jouary. The data was labelled using bout classification developed from Marques et al. (2018). The ethanol-treated dataset was collected and shared by S. Lackner and M. B. Orger. The idea of balanced model reduction and Schur decomposition originated from discussions with G. Hennequin and M. Schimel. The iLQR-VAE toolbox was developed and adapted by Schimel et al. (2022). This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service. I am grateful for the discussions with G. Hennequin, A. Jouary, M., C. Machens and M. B. Orger.

Chapter 5

Neural correlates to dynamical motifs of Larval Zebrafish

5.1 Summary

Optical techniques enable non-invasive, brain-wide neural recordings in larval zebrafish. However, this preparation often requires the fish's head to be immobilized, which limits the range of naturalistic behaviors that can be studied. A common approach to relate head-fixed to free-swimming assays is to identify characteristic parameters of movement that describe behavior. We present an alternative approach that categorizes head-restrained movements into bout types using a preliminary dataset. We make the assumption that head-restrained movements share similar dynamics to free-swimming movements. We perform inferences on head-restrained movements subject to freely swimming dynamical system, then use the head-restrained control to assign to a bout type.

In this chapter, we outline the head-fixed neural dataset and the variety of movements in the head-fixed preparation in Sections 5.3.1 and 5.3.2. In Section 5.3.3, we define a metric that characterizes the ballisticity of movements, which was used to filter movements generated by sparse control. Then, we demonstrate two approaches to bridge head-fixed movements to the free-swimming behavioral space. In Section 5.3.4, we first project the initial control state of head-fixed movements into the embedding space of the free-swimming. There, we could classify them to specific free-swimming bout types. In Section 5.3.5 we produced neural activity maps conditioned on labeled bout types. The other approach, in Section 5.4, uses the similarity of the control space to hierarchically cluster head-restrained movements into specific movements.

5.2 Introduction

Movement is generated through the influence of descending Reticulospinal (RS) neurons in the hindbrain (Grillner & El Manira, 2020), which, in zebrafish, form a bottleneck for information flow from the brain to the spinal cord. These neurons in larval zebrafish form a set of stereotyped nuclei in the hindbrain and mainly project to CPGs in the spinal cord responsible for driving oscillatory movements. Transgenic tools have allowed targeted labeling of these neurons, and together with optical techniques, specific RS neurons can be manipulated or activated to understand their role in controlling movements (Kimura et al., 2013).

Functional neural recordings have some challenges and trade-offs in larval zebrafish. Recently, Kim et al. (2017); Cong et al. (2017) have demonstrated wholebrain imaging for free-swimming larvae, using a tracking microscope. This has enabled free-swimming behavior while simultaneously acquiring neural recordings (Marques et al., 2019). However, these microscopes use single-photon illumination, which could produce responses directly through light-sensitive areas in the brain (Kokel et al., 2013), or visible light could interfere with the visual response from the arena (Portugues et al., 2014). A more common approach requires tethering the head with agarose, therefore restricting the ability to move freely while acquiring wholebrain neural recording. This was first demonstrated during escape response to a tactile stimulus. Calcium imaging, together with optical ablations of the Mauthner cells and descending homolog pairs of MiD2 and MiD3, revealed their role in shaping features of escape responses, such as latency and sensitivity (O'Malley et al., 1996; Liu & Fetcho, 1999).

A challenge with head-restrained preparations is that mapping movements to the bout types displayed in free-swimming fish is not straightforward, as their swims are more varied, as their are swims more variable than the ballistic swims displayed in free-swimming fish. This issue is addressed by

finding the relevant kinematic parameters of movement for the behavioral task. These parameters of movement (Carbo-Tano et al., 2023), or the initiation of movement (Orger et al., 2008; Berg et al., 2023; Severi et al., 2018), can be measured in both the head-restrained and free-swimming assay, bridging the two preparations and allowing for neural recordings. This approach has deepened our understanding on RS neurons control of kinematic characteristics of swims. We understand that ventromedial V2a hindbrain neurons are involved in the turn amplitude (Orger et al., 2008; Huang et al., 2013; Kimura et al., 2013). They are also important for maintaining forward swimming and receive upstream excitatory inputs from the MLR (Carbo-Tano et al., 2023). Studies have identified that the nucleus of the Medial Longitudinal Fasciculus (nMLF) regulates swim frequency (Severi et al., 2014; Thiele et al., 2014; Berg et al., 2023). In addition, neurons in the nMLF are activated during the onset of forward swimming (Severi et al., 2018). The rapid development of the larvae has revealed the layered organization of V2a neurons, which develop laterally in the hindbrain. This highlighted that functional circuit develop later on to allow for more flexible behaviors (Pujala & Koyama, 2019). Taken together, these observations suggest a complex distributed network throughout the hindbrain is employed to generate different bout types (Carbo-Tano et al., 2023; Gahtan et al., 2002).

The model in Chapter 3 had two key insights that relates to previous work on neural control of locomotion. First, the external driving input used to generate the bout was concentrated at the onset of the bout (Severi et al., 2018; Xu et al., 2021). Secondly, a distributed combination of control was required to generate different types of movement (Carbo-Tano et al., 2023). This aligns with the idea that populations of hindbrain neurons are responsible for generating different bout types, and different control inputs store information on kinematic features of movement, e.g., turn amplitude or tail beat frequency. Based on Chapter 3, we hypothesize that

the dynamics of head-restrained movements are captured within the same dynamic subspace that governs free-moving behaviors. This addresses an overarching challenge in the field of bridging free-swimming behaviors with head-restrained studies. However, instead of linking behavior to the kinematic features of movement, we linked behavior to the inferred control under the same underlying dynamics.

In this study, our aim is to show that inferred control can be used to embed head-restrained bouts into a free-moving latent space. In preliminary analysis of a small dataset we found that head-restrained bouts can be classified as specific movement types. Furthermore, we demonstrate that by labeling these bout types, we can generate neural maps of the hindbrain neurons responsible for driving these movements. This approach provides an alternative framing of behavior in head-restrained conditions that relates movements to the underlying dynamics of free-swimming behaviors via the low-dimensional control space.

5.3 Results

5.3.1 Head-fixed behavioral and neuronal dataset

For this study, we took a dataset collected in the Bianco lab that is part of the thesis of Lau (2019). This dataset simultaneously recorded tail-tracking and hindbrain neural activity. The neural data was collected using a two-photon (2P) microscope, spanning 13 planes at a frequency of 4.8 Hz per plane. Each plane was separated by 5 μ m, with the reticulospinal population spanning a depth of 100 μm.

The head-restrained dataset was composed of 2237 bouts, lasting from 100-1500 ms, of a single fish. The 400 fps tail tracking was upsampled to 700 fps to match the frame rate of our free-swimming datasets.

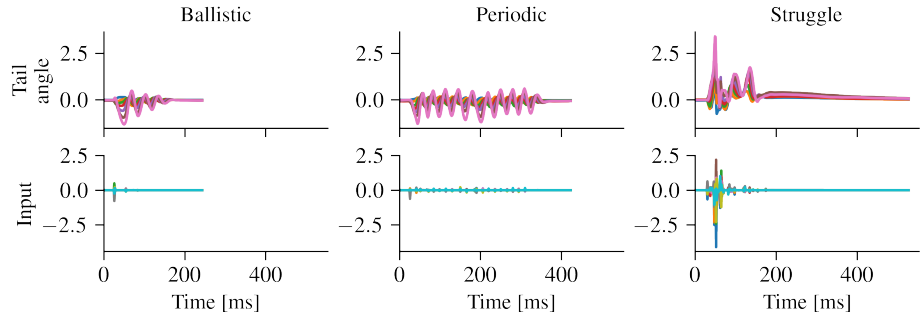


Figure 5.1. **Examples of the variety of head-fixed bouts.** Head-fixed behavioral assay changes the behavioral conditions, so not just ballistic swims (left) are exhibited but also prolonged swims (middle) and bouts with struggle swims (right). Below show the inferred control used to generate the corresponding bout above using the free-swimming model in Chapter 3.

5.3.2 Inferring control of head-fixed movement under free-swimming dynamics

The head-fixed preparation alters the behavior of the zebrafish, in that more types of swims are observed beyond the stereotyped set of swims observed in free-swimming assays; see Figure 5.1. The most common observations are that swims tend to be sustained for prolonged periods, and that occasionally we observe struggle-like swims; illustrated in Figure 5.1. Struggles are produced mainly in head-restrained conditions, as the fish tries to escape from the agarose gel. The agarose gel also restricts the ability to produce O-bend swims, a common swim driven by sudden light-dark transitions (Beppi et al., 2021; Marques et al., 2018).

We used the pre-trained model in Chapter 3 that could reconstruct free-swimming movements to infer the control of head-restrained bouts. The iLQR algorithm (details in Section 2.7) is used to infer the best sequence of control inputs to reconstruct head-restrained; see Figure 5.1. All head-

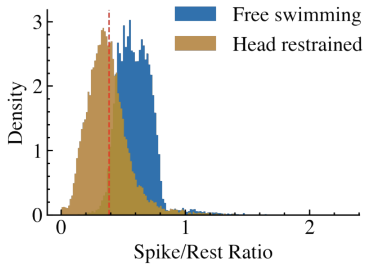


Figure 5.2. **Distribution of control sparsity for freely swimming and head-fixed data.** The distribution of input sparsity to generate bouts with the head-restrained (orange) and free-swimming (blue) data.

restrained bouts were reconstructed with a R^2-score > 0.94. However, unlike free-swimming movements, head-restrained bouts required more persistent control to drive the dynamics. As shown in Figure 5.1, head-restrained movements have more extended bouts compared to the ballistic free-swimming dataset.

5.3.3 Sparse control identifies naturalistic swims in head-restrained condition

Bouts in free-swimming conditions were typically ballistic and, thus, were generated from a single control peak. This provided a criteria to extract head-restrained movements that were similar to free-swimming movements. We quantified the degree of ballisticity of movements, by calculating the ratio of the initial control peak to the total control input. The heavy tail in the ballisticity distribution for the head-restrained dataset, Figure 5.2, indicated the presence of naturalistic swims that coincide with both datasets. We used the lower bound of the free-swimming distribution to define a threshold of 0.4 to filter the head-restricted dataset, reducing the dataset from 2237 to 456 bouts.

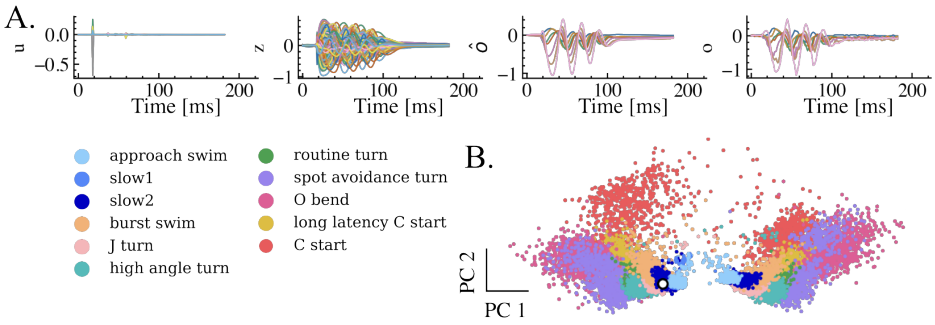


Figure 5.3. **Head-restrained swim embedded in free-swimming latent space**. (A) An example of a ballistic head-restrained bout generated from sparse control, generating latent dynamics that project onto the tail, reconstructing the tail's behavior. (B) The low-dimensional embedding space of the network for free-swimming behavior. The colored points are the initial state of free-swimming movements after the initial control. The The white point is the projection of the head-restrained movement in (A) into the free-swimming embedding space.

The filtered dataset only contained ballistic bouts that were largely generated from the initial control peak; an example is shown in Figure 5.3A. We could then obtain the initial network state set by the control peak. Therefore, the initial state of the head-restrained bout could be projected in the free-swimming PC space, Figure 5.3B. As this embedding preserves bout types, see Section 3.4.2, we could assign head-restrained swims to the most likely bout type in the embedding space.

5.3.4 Classifying head-restrained bouts to free-swimming bout types

We can describe the similarity across initial control patterns in generating movement by projecting the initial state onto the free-swimming PC embedding. We used nearest-neighbor clustering of the first 3 PCs of the initial state to assign a bout type to head-restrained bouts; see

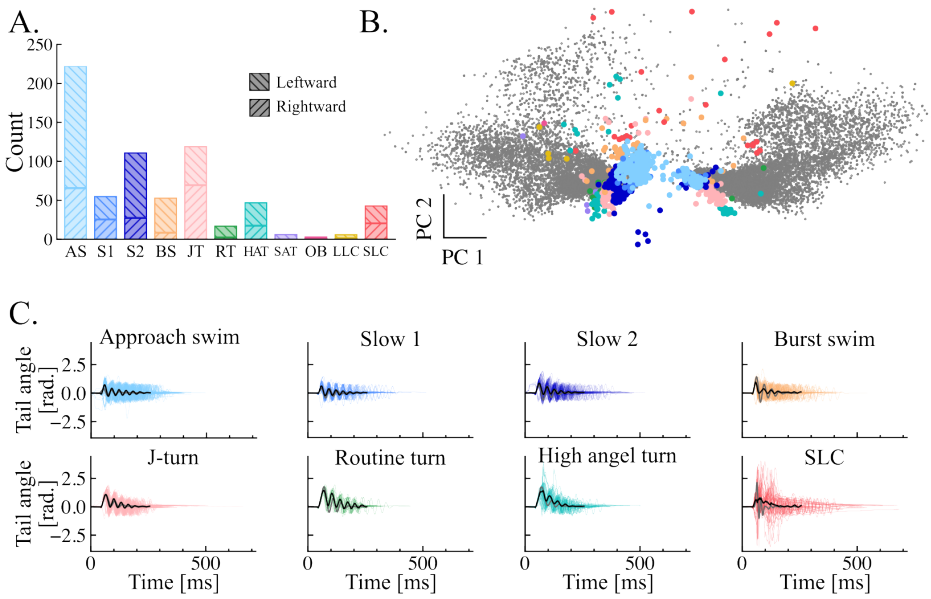


Figure 5.4. **Free-swimming bout classification on head-restrained movements**. (A) Bar plot showing the count of different right and left bout types from the head-restrained data. (B) Head-restrained bouts were classified in the free-swimming embedding space (gray data points show free swimming distribution). Each head-restrained movement's initial state were projected into the PC-space (color dot). We used nearest-neighbor clustering on the first 3 PCs of the initial state to assign a bout type to head-restrained bouts. The color corresponds to the free-swimming bout type that was closest to the head-restrained bout. (C) Plots show the 7^{th} segment tail angle of the clustered head-restrained movements. We excluded long-latency C-starts and O-bends due to the limited data points.

Figure 5.4b. The space of the head-restrained bouts with ballistic inputs was tightly constrained to regions of typical forward swims (bursts and slow swims) and turns (routine, J, high-angle turns); see Figure 5.4a-b. There was little overlap of control used to initiate O-bends, spot avoidance turns, and long-latency C-starts, which was anticipated due to the physical constraints of the head-fixed setup.

The assignment of bout type in Figure 5.4c showed that the classification could distinguish between turns and forward swims, and also within bout types. Slow 1 have a smaller first tail beat amplitude than the Slow 2 but similar tail beat frequency; however, burst swims have higher tail beat frequency, all reflected in the clustered head-restrained movements. There is also a growth in the amplitude of the first tail beat from the J turn to the routine turn to the high-angle turns. Most struggle swims were filtered out because the controls to reconstruct bouts were not ballistic. However, the remaining struggle swims in Figure 5.4 were assigned to areas where SLCs were located in the free-swimming dataset. This was likely due to the fact that they were not captured in the free-swimming dataset.

5.3.5 Neural activity maps for head-restrained swims reveal distinct populations recruited for different swim types

We have demonstrated that ballistic head-restrained movements can be assigned bout type labels using their inferred control signal. Rather than previous methods that sought neural activity patterns for specific kinematic features, we could identify neural activity patterns for specific bout types.

We extracted typical types of forward swim: slow 1, slow 2, and burst swims. Slow 1 and slow 2 swims have similar dynamics except for the amplitude of the tail beat. However, burst swims are distinctly different

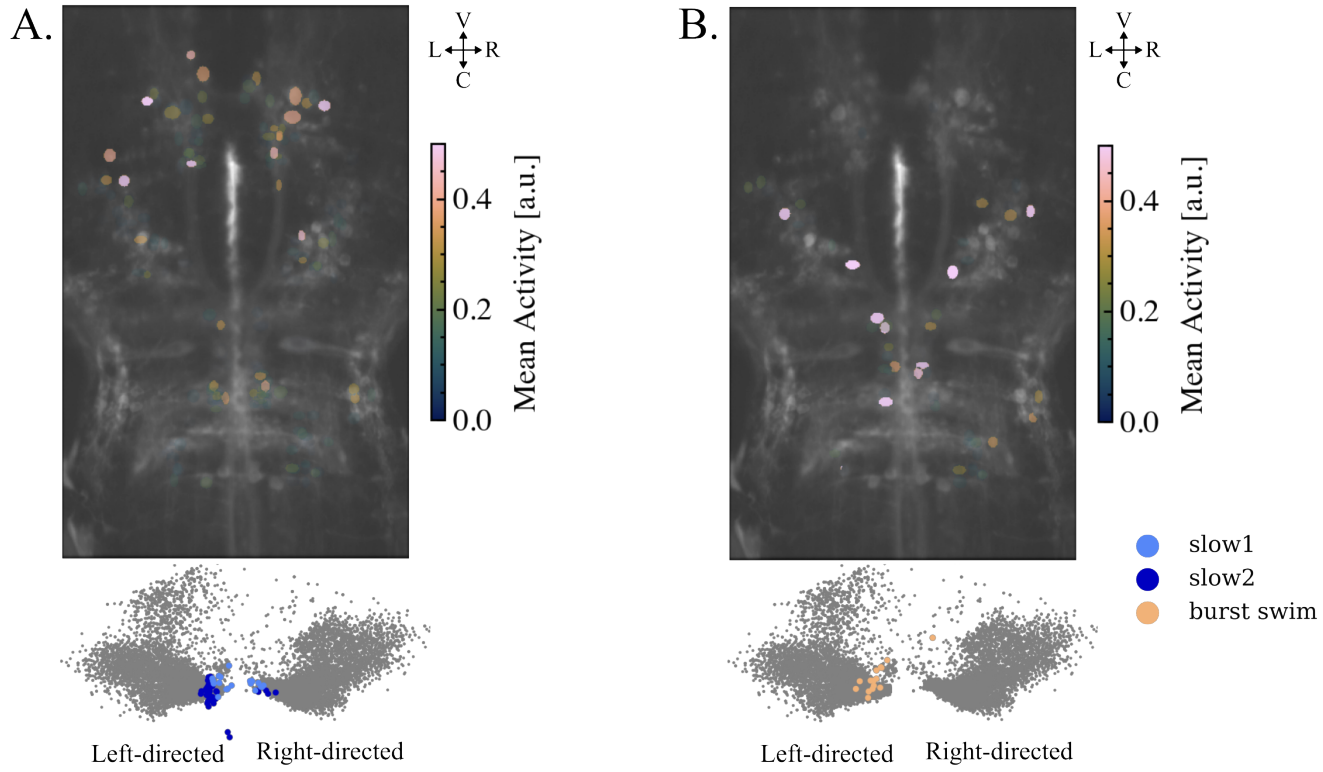


Figure 5.5. **Comparison of neural activity pattern for slow vs burst forward swims**. (A) The average neural activity map for slow 1 and slow 2 swims, with symmetric activity observed in the nMLF. Below are the projections of the initial slow 1 and slow 2 swims in the free-swimming latent space. (B) The average neural activity map for burst swims and (below) the projection of bouts in the free-swimming latent space (n=1 fish).

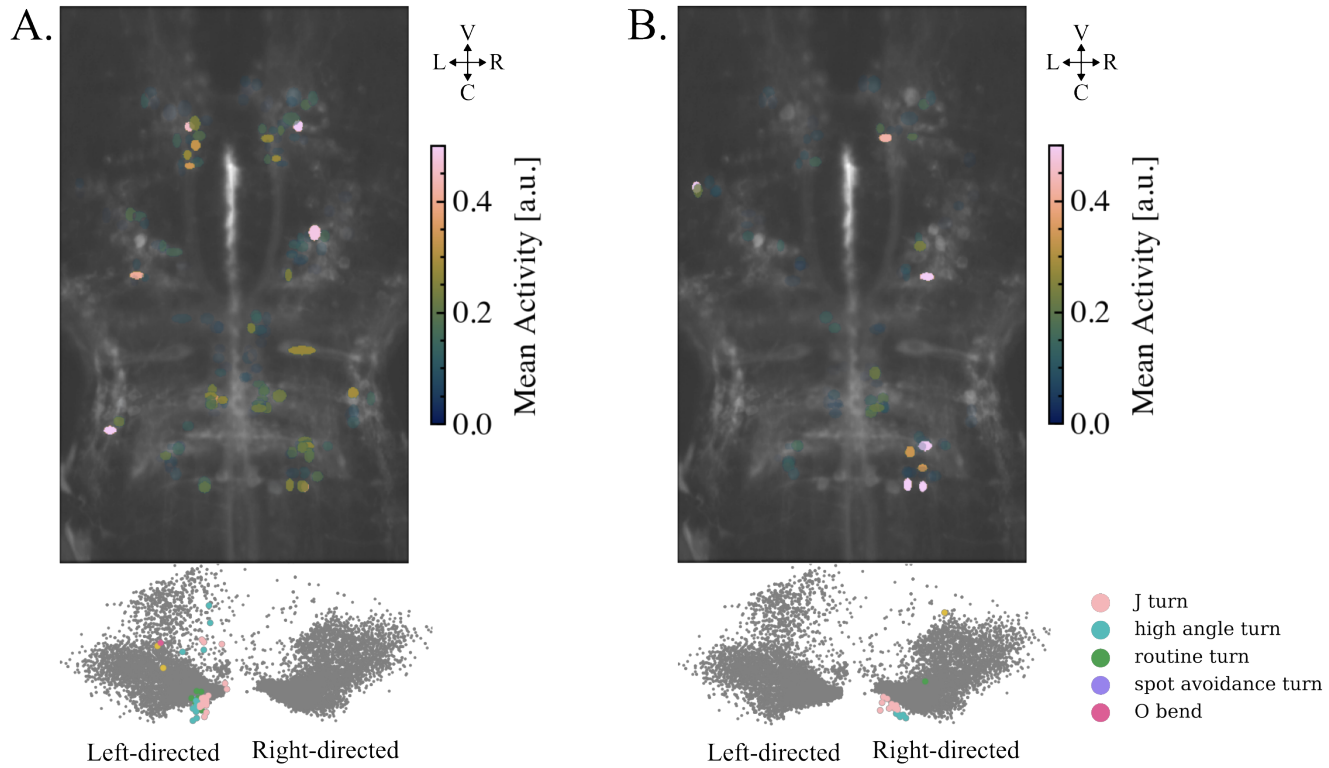


Figure 5.6. **Comparison of neural activity pattern for right vs left turns**. (A) The average neural activity map for leftward J-turns, high-angle turns, routine turns, and spot avoidance. (B) The average neural activity map for rightward J-turns, high-angle turns, routine turns, and spot avoidance. Below are the projections of the turns in the free-swimming latent space (n=1 fish).

to slow swims in their tail beat frequency as well as tail amplitude. As a proof of concept, we compared the average neural activity patterns in the hindbrain for slow and burst swims, Figure 5.5. The mean neural activity maps in Figure 5.5 showed that slow swims recruit a distinct population of neurons that appear to be located in the nMLF. Whereas, burst swims recruit a different population of neurons that appear to be situated in the ventromedial region. Both patterns show symmetric activity on the left and right sides of the hindbrain. We compared left-right turns consisting of bout types routine, J-turn, and high angle turns, Figure 5.6. The average neural activity maps show lateralization in the nMLF and symmetric activity in the ventromedial section of the hindbrain.

5.4 Neuronal population structure differs from inferred control structure of movements

Here, instead of using our prior knowledge of free-swimming bout categories to cluster head-restrained movements, we used the similarity of the control space to cluster head-restrained movements. Chapter 3 highlighted that the inferred control separated different dynamical properties of movement. We used this finding as a method to analyze neural population activity in the hindbrain. As a proof of concept, rather than using the entire dataset, we explored this approach using plane 8, as it was a plane with the most number of bouts. This plane had 143 swims during acquisition and 40 segmented ROIs that have structured patterns of neural activity across groups of bouts; see Figure 5.7E.

We examined the neuronal population similarity, Figure 5.7 D - F, and the inferred control similarity matrices, Figure 5.7 A - C, to identify the bouts encoded at the population level. Both similarity matrices displayed a block-diagonal structure, suggesting that distinct patterns were active for

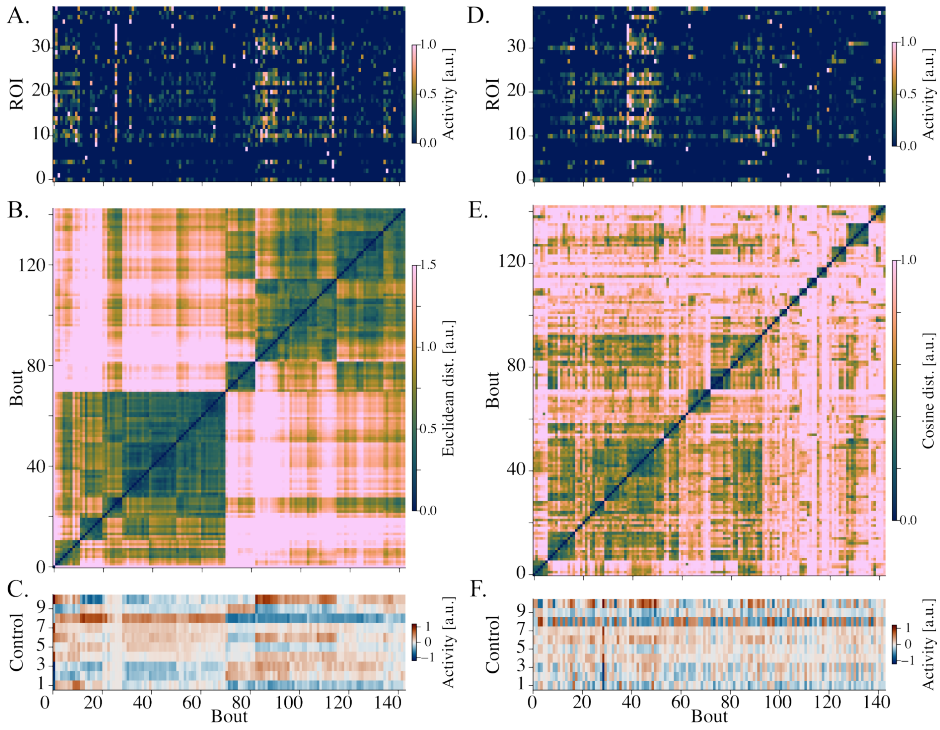


Figure 5.7. **Population-level similarity of neural ROI space and inferred control space**. (A-C) Bouts in Plane 8 were sorted by the Euclidean distance similarity of the initial inferred control peak. (A) Shows raster plot of neural ROI activity across bouts, (B) pairwise similarity matrix of initial inferred control, and (C) is the initial control peak for each component to generate each bout. (D-F) Is the same as (A-C) but instead sorted Cosine distance of neural ROI activity. (D) ROI activity across bout, (E) pairwise similarity matrix of ROIs, and (F) initial control component for bout generation.

groups of movements. In particular, the sorted control space in Figure 5.7 B separated the movements of the right-left into two quadrants. The left-right direction was previously identified as being encoded by the 8th control component. Within these two main clusters there were sub clusters with more detailed kinematic similarities. However, in the neural ROI similarity matrix, while clusters of bouts were observed in ROI activity, there was no clear distinction between right and left movements.

We exploited the inferred control space to identify the dynamical structure of HR movements by performing hierarchical clustering, Figure 5.8, on the similarity matrix in Figure 5.7B. We selected 15 clusters based on the elbow point of the cluster distance across the total range of clusters available; see Figure 5.8A. The extracting bouts of each group revealed that groups 2 to 8 contained swims initiated by the left half beat. Specifically, fast forward swims were identified in clusters 2 to 4, slow forward swims in clusters 7 and 8, and turns in clusters 5 and 6. Clusters 9–15 represented rightward swims, with clusters 11 and 12 displaying fast forward swims, cluster 13 slow forward swims, and clusters 10, 14, and 15 turns.

To discover common structures in the control and neuronal spaces, we reordered the neuronal similarity matrix based on the sorting of the control similarity matrix, Figure 5.8D. While similarity between the two spaces was not entirely consistent, clusters 2–4 and 11–13 appeared mutually similar. However, due to the limited number of bouts, it is not possible to confirm the consistency of these clusters. For completeness, we projected the total activity of each ROI for each cluster into the neural space, shown in Figure 5.9, to highlight the distribution of activity for different types of movements.

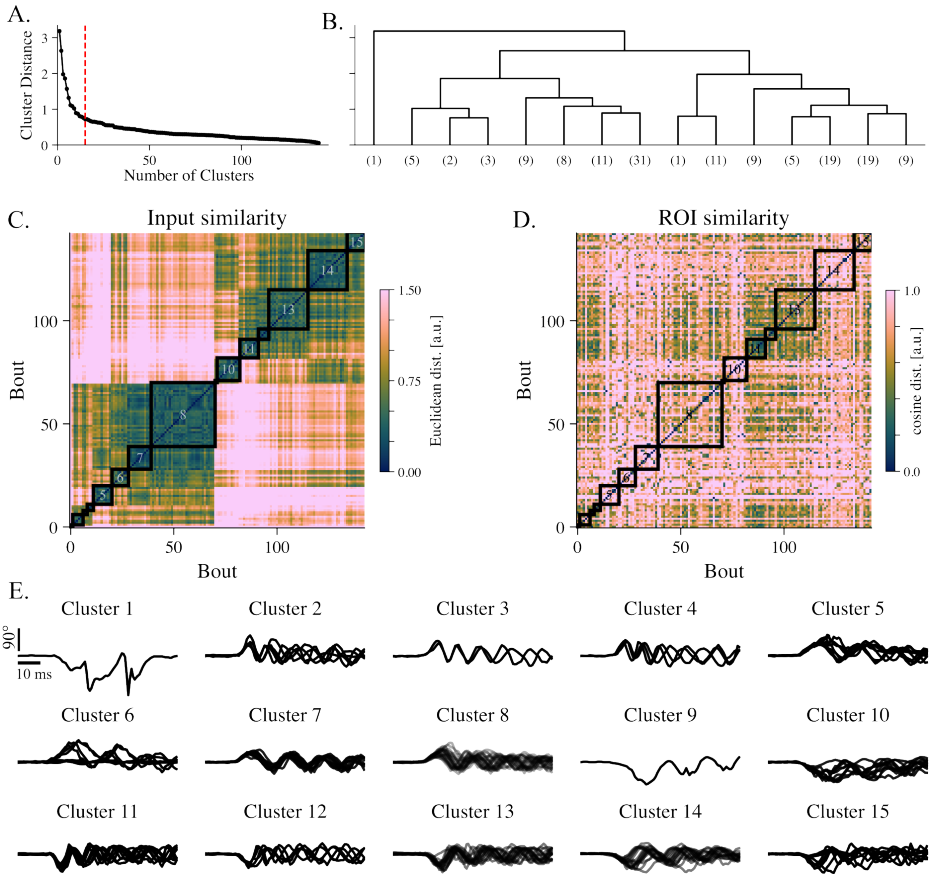


Figure 5.8. **Hierarchical clustering of head-restrained movements in the control space**. (A) The number of clusters were selected based on the elbow of cluster distance. 15 clusters were chosen, indicated by red vertical line. (B) Dendrogram of the bouts grouped based on the Euclidean distance of the initially inferred control, truncated after 15 branches. X-axis indicate the number of bouts labeled for each cluster. (C) Inferred control pairwise Euclidean distance with hierarchical clusters from (B) overlaid and labeled. (D) The neural ROI pairwise cosine distance sorted using inferred control ordering to identify whether there are clusters in common. (E) The 7th segment tail angle for bouts from each cluster in (C) is shown. Clusters 2–8 correspond to leftward bouts, and clusters 9–15 to rightward bouts. Cluster 1 appears to extract bouts with tracking errors or struggle swims.

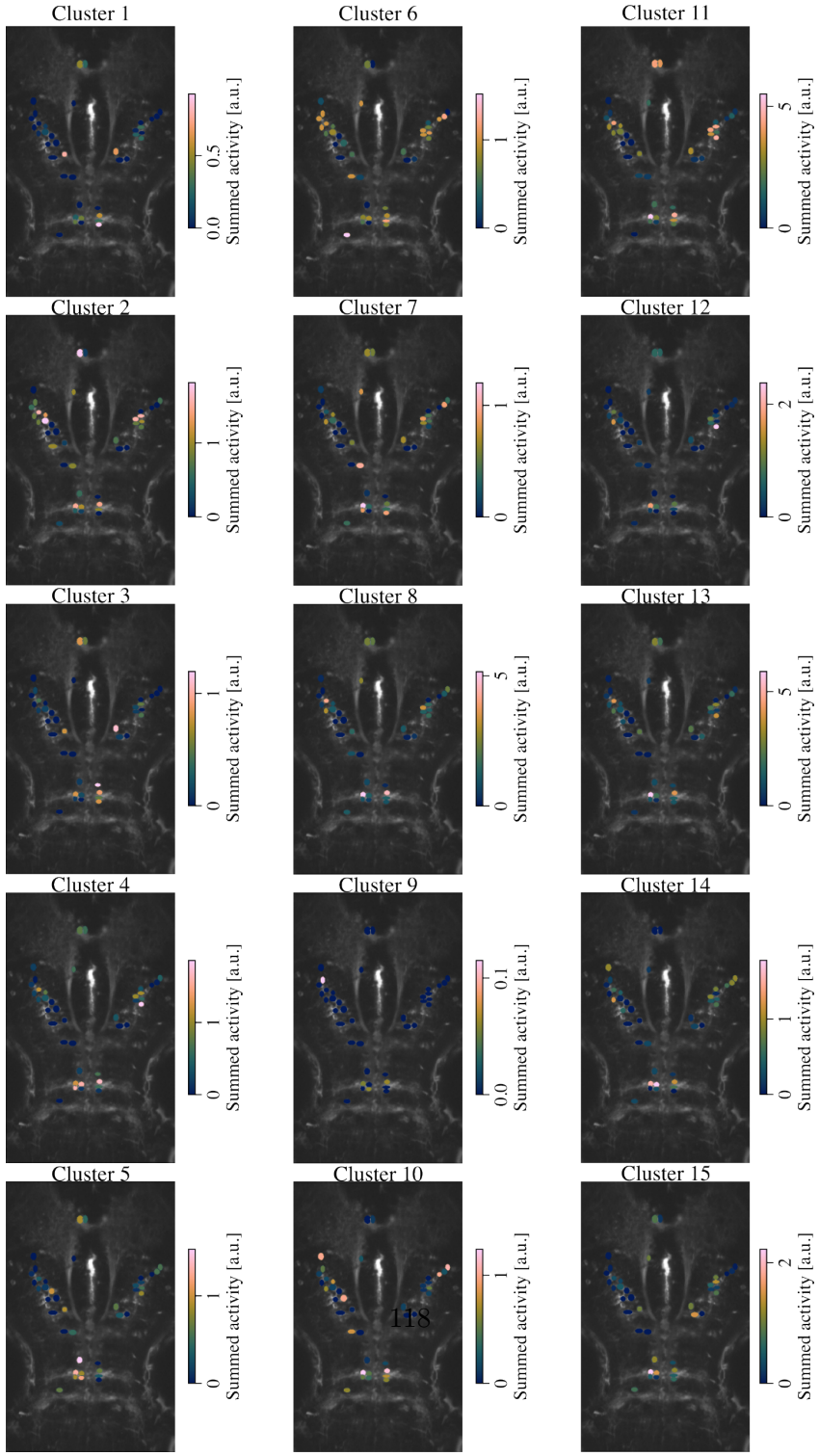


Figure 5.9. **ROI cumulative activity for each input cluster**. The ROI map corresponds to a cluster extracted from the similarity matrix of input bout types.

5.5 Discussion

As a first attempt, we demonstrated a possible avenue for bridging head-restrained to free-swimming behavior using a model trained on free-swimming locomotion. By filtering head-restrained bouts of a ballistic nature similar to free-swimming movements, we successfully classified bouts with common initiating control of bout types relevant to free-swimming behaviors (Marques et al., 2018). This sets a path for future studies in head-fixed preparations to label more complex tasks, such as sequences or adaptations of bout types in head-fixed assays.

The main limitation of this analysis was that it was performed using a single fish, and the neural imaging was acquired over single planes. Consequently, in addition to relying solely on ballistic bouts, the bout inferences could only be analyzed for single planes, reducing the number of bouts for a given ROI by 13-fold. This could potentially explain the lack of ipsilateral activity in RS neurons during right-left turns, which have been previously identified (Orger et al., 2008; Huang et al., 2013).

With a larger dataset, neural maps could be pooled for individual bouts rather than agglomerating turns, as Lau (2019) has shown that J-turns recruit a distinct population of neurons compared to routine turns and high-angle turns. Although slow forward swims recruit the nMLF, aligning with studies from Carbo-Tano et al. (2023) and (Severi et al., 2014), this result was not consistent for burst swims, creating a contradiction. In addition to acquiring datasets collected across multiple fish, two-photon light sheet microscopy would provide fast volumetric imaging without visible light disturbing the fish's behavior (Wolf et al., 2015; de Vito et al., 2022).

We demonstrated the low-dimensional input space encoded using the dynamical structure of head-fixed behavior, could be used to cluster bouts

based on similarity of control inputs. However, this input structure lacked similarity to clusters identified in the neural space. This suggested that the model found a solution to capture movement independent of the hindbrain neurons. While this again could be an artifact of limited data, an alternative explanation might be that the non-linear mapping from control space to tail observation in the model was too flexible which led to a different solution to hindbrain neural recordings. A future direction would be to constrain the model to be biologically inspired. For example, studies have constrained the dynamical system to be a set of descending coupled oscillators (Kuo & Eliasmith, 2005; Ijspeert et al., 2007). This way the inferred control would be constrained and generate oscillatory dynamics similar to our understanding of CPGs.

5.6 Acknowledgements

The head-restrained neural data was collected and preprocessed by J. Y. N. Lau and was shared by I. Bianco. The forward inference was performed using a toolbox developed by M. Schimel. Discussions on the population analysis were held with E. Dumon, A. Jouary, I. Bianco, J. Fitzgerald, and M. B. Orger.

Chapter 6

General Discussion

6.1 Summary

We used optimal control theory to model behavior in larval zebrafish as a controlled dynamical system. The underlying input-driven dynamics were identified using the iLQR-VAE method (Schimel et al., 2022). This framework allowed us to compartmentalize the model into two components: a reservoir of dynamics that captures the broad space of movements and a low-dimensional control space that selects the dynamics for movement production (Mullen et al., 2024). The sparsity constraint in the latent input was critical to ensure that the dynamics were learned in the network rather than being driven by the control. Therefore, this helped separate the control space from the dynamics space. The low-dimensional control space reflects a biological concept similar to the descending control from the brainstem to the spinal cord. Throughout this thesis, we explored parallels of this unique feature in our model to understand how inferred control mapped to different aspects of behavior.

First, we demonstrated the link between dynamics and behavioral representation by showing that the initial control for movement production correlated with global kinematic features of movement. This bridge has been made in the dynamical system view that models the response of M1 to behavioral variables (Churchland et al., 2012; Pandarinath, O'Shea, et al., 2018). We showed that the tail beat frequency and tail amplitude could decode from the initial control. Although we did not explicitly relate the control to stimulus features, experimental studies have shown that tail beat frequency and bout speed are correlated with grating speed (Severi et al., 2014; Naumann et al., 2016), and that turn angle is coupled to the stimulus angle (Orger et al., 2008). We also found that the initial control was predictive of the bout type (Marques et al., 2018). The classification of bout types were defined by clustering on the kinematic features of the behavior (Marques et al., 2018; Johnson et al., 2020; Mearns et al., 2020).

A key distinction of our method was that bouts could be generated by smoothly sampling through the 10-dimensional control space. As most of the control is encoded in the initial control peak, movements could be represented as a control state. This addresses issues with pure discrete classifications which do not allow for continuous variation (Datta et al., 2019). Furthermore, this space is compatible with different bout classification methods, as it was trained on a diverse set of behaviors (Johnson et al., 2020; Mearns et al., 2020).

Second, having a single model fitted to the behavioral repertoire of zebrafish larvae allowed us to make inferences and find embeddings in the control space for other datasets. We demonstrated this advantage by inferring latent dynamics from movements with pectoral fin and tail tracking. There has been a discussion about the control of pectoral fins and if control is shared with the tail (Budick & O'Malley, 2000; Green et al., 2011; Danos & Lauder, 2007). We highlighted that the latent dynamics could not predict fin movements for slow swims, suggesting that there were independent control modes, whereas faster burst swims shared latent dynamics. Furthermore, we aligned behavioral datasets with neural recordings using the pre-trained model. The behavioral datasets with neural recordings were acquired under head-fixed conditions. We made the assumption that the dynamics of head-fixed movements constrained to a subspace of the free-swimming dynamics, as the set-up prevented the ability to elicit O-bends and extended bouts were an artifact of the head being tether (Portugues & Engert, 2011; Jouary et al., 2024). We used the inferred control as a method to exclude bouts that were extended and not like the more ballistic free-swimming bouts. By embedding sparse head-restrained bouts in the free-swimming PC space we were able to classify head-fixed movements into free-swimming bout types based on the similarity of their initial control. Despite the preliminary nature of our findings and the limited dataset available, we introduced a novel approach for examining the neural control

of naturalistic behaviors, by enabling the classification of head-fixed movements according to free-swimming bout types (Cong et al., 2017; Marques et al., 2019).

Moving away from understanding the inferred control space, we demonstrated an approach to interpret the learned dynamics by fitting the behavior to a linear network. This allowed us to reduce the dimensionality of the network using balanced model reduction (Padoan et al., 2021), and find a minimal model that could recapture behavior. This reduced model provided a compressed representation of the underlying dynamics, so we could identify important dynamical modes recruited in the construction of different movements (Stephens et al., 2008; Ahamed et al., 2020). Zebrafish larvae have become a popular model for drug discovery due to their rapid development and genetic tractability (Patton et al., 2021). We used balanced model reduction to show that the underlying dynamics of behavior were invariant when manipulated with ethanol drug. However, the control in driving different bout types were altered. We demonstrated that while kinematic analysis identified phenotypical changes under drug screening, the inferred control can provide insights into what aspects of behavioral control are modulated due to drugs.

Throughout this work, we developed an open source Python software package, `DiffiLQRAX` , which implemented the iLQR algorithm (Mullen & Schimel, 2025). The package contains `ReadtheDocs` documentation and used case examples to encourage usability. While there existed quadratic programming solvers for optimal control (Amos & Kolter, 2017), the JAX library had functionality for associative scanning. We explored a parallel version of the LQR problem using associative programming that could parallelize linear dynamics and the recursive Value function (Sarkka & Garcia-Fernandez, 2023). This was advantageous for GPU scalability and reducing the time complexity of tradition LQR solvers (Sarkka & Garcia-

Fernandez, 2023). Associative scan algorithms have gained popularity in the machine learning community for their ability to parallelize operations in linear systems (Sarkka & Garcia-Fernandez, 2021), and some non-linear problems (Feng et al., 2024; Gonzalez et al., 2024). In addition, `DiffiLQRAX` is a differentiable solver, so it could be incorporated into other JAX-based machine learning frameworks (Bradbury et al., 2018).

6.2 Future outlook

The application of dynamical systems approaches to model behavior was largely inspired by work in the motor cortex community (Shenoy et al., 2013). Optimal feedback control has been used to describe inter-areal communication between motor areas (Sauerbrei et al., 2019; Athalye et al., 2023). Where studies with optogenetics and multi-regional recordings have shown that the motor cortex is not entirely autonomous but receives thalamic input for accurate movement execution (Sauerbrei et al., 2019; O'Shea et al., 2022). Our model frames motor control at a different level that focuses on control as a feedforward process, whereby inputs descend from the hindbrain to the spinal cord. But there is a caveat that the control in our model does not incorporate feedback, despite experiments showing that interneurons have recurrent connections between RS neurons(Vishwanathan et al., 2024). One way that feedback might be represented in our sparse control space could in be the additional peaks used to generate bouts. An approach to test this would be to use a dataset with optogenetic stimulation that could perturb RS neurons (Kimura et al., 2013; Jia & Wyart, 2024), or have a closed-loop stimulus and manipulate the gain of the stimulus feedback (Markov et al., 2021).

The iLQR-VAE method learns a non-linear mapping between the latent control and the dynamics of behavior. This bypasses the need to fit neural activity to behavior, which can be a challenging problem (Pandarinath,

O'Shea, et al., 2018). We expected that the inferred latent control should map to populations of neurons in the hindbrain. A challenge with this approach was that the RNN was too flexible, so it could not easily reflect a similar solution to the hindbrain neural activity. One approach to address this would be to learn a joint behavior and neural model (Hurwitz et al., 2021; Kudryashova et al., 2023; Gondur et al., 2024). Neural dynamics "stitching" has been proposed as a way to improve generalization by fitting the latent dynamics to multiple datasets during learning Turaga et al. (2013); Pandarinath, O'Shea, et al. (2018). In this case, the neural dataset would have a readout to the behavior coupled with ROI reconstruction. An alternative would be to define the network to incorporate more biologically plausible structures, such as coupled descending oscillators. Studies have demonstrated these oscillator-inspired models on crawling Drosophila (Pehlevan et al., 2016), walking and swimming salamanders (Ijspeert et al., 2007) and swimming zebrafish (Kuo & Eliasmith, 2005), to name a few. So, fitting behavior to this dynamical system would have biophysical constraints in the model. Incorporating these constraints into the model would learn a control space more aligned to neural control of behavior.

We only briefly touched on how the inferred control is reflected in hindbrain neural activity. This was due to two limitations; the first was that the size of the dataset was too small to apply any statistics as it was collected from a single fish and the two-photon imaging restricted neural recording to be acquired on single planes. In contrast, light-sheet microscopy can acquire volumetric images of neural activity (Keller & Ahrens, 2015; Hoffmann et al., 2023). This would allow all neural ROIs to be recorded during each tracked bout, increasing the number of bouts for a given ROI. The other drawback was the temporal resolution of the neural dataset. Although light-sheet microscopy would additionally enhance the rate of acquisition, calcium indicators have a slow decay time,

and so the temporal resolution would still be limited. Advancements in genetic tools have developed genetically encoded voltage indicators with faster kinetics (Miyazawa et al., 2018; Wang et al., 2023). This would make it possible to capture the fast dynamics of neural activity in the hindbrain and relate it to the inferred control space.

The larval zebrafish was an ideal model organism for studying behavior as their swims are naturally segmented into bouts. The iLQR-VAE method has also proven to be a flexible tool to learn controlled dynamics of continuous locomotion such as that of C. Elegans (Mullen et al., 2024). There has been extensive work quantifying the behavioral dynamics of C. Elegans (Stephens et al., 2008; Ahamed et al., 2020). An advantage with C. Elegans is that all of their neurons can be labeled and recorded from, so behavior can be mapped to brain states (Kato et al., 2015). Another interesting avenue would be to apply the method to other model organisms that are very closely related to the zebrafish, such as the Danionella, which have a similar reticulospinal organization, but with different locomotor patterns. For example, the locomotion of Danionella is does not occur in discrete bouts, but is more continuous (Rajan et al., 2022).

With larger computational resources and exploiting the associative scanning of the `DiffiLQRAX` solver, it would be interesting to train the behavior on longer sequences of data, not just individual bouts. This would allow the inferred control to capture information about the previous bouts and interbout interval (Johnson et al., 2020; Dunn et al., 2016). We could then explore how the control sequences change over long timescales and see if there are temporal correlations in control for sequences of bout types (Sridhar et al., 2024; Costa et al., 2024).

On the software side, there are several extensions that could be made to `DiffiLQRAX` . The first would be to integrate the software with other physics simulators, such as Mujoco and Open-Sim (Todorov et al., 2012;

Delp et al., 2007). This would enhance usability and contributions, as there is a large community of researchers who use these simulators. Secondly, incorporate inequality constraints into the solver (Howell et al., 2019). This would allow optimal control to drive dynamical in closed environments and is important for robotics applications. Finally, it would be useful to add a Kalman smoother to `DiffiLQRAX` for state estimation. This functionality would be parallel to iLQR, but would allow for optimizing the state space rather than the control from partial observations.

References

Aaqaoui, A., & Mohammed, Y. M. E. (2024). *Application of iterative lqr on a mobile robot with simple dynamics.* arXiv. Retrieved from https://arxiv.org/abs/2404.18312 doi: 10.48550/ARXIV.2404.18312

Abbaspourazad, H., Erturk, E., Pesaran, B., & Shanechi, M. M. (2023). Dynamical flexible inference of nonlinear latent factors and structures in neural population activity. *Nature Biomedical Engineering*, *8*(1), 85–108. Retrieved from http://dx.doi.org/10.1038/s41551-023-01106-1 doi: 10.1038/s41551-023-01106-1

Agha, M. A., Kishore, S., & McLean, D. L. (2024). Cell-type-specific origins of locomotor rhythmicity at different speeds in larval zebrafish. *eLife*, *13*. Retrieved from http://dx.doi.org/10.7554/eLife.94349 doi: 10.7554/elife.94349

Ahamed, T., Costa, A. C., & Stephens, G. J. (2020). Capturing the continuous complexity of behaviour in caenorhabditis elegans. *Nature Physics*, *17*(2), 275–283. Retrieved from http://dx.doi.org/10.1038/s41567-020-01036-8 doi: 10.1038/s41567-020-01036-8

Ahrens, M. B., Orger, M. B., Robson, D. N., Li, J. M., & Keller, P. J. (2013). Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nature Methods*, *10*(5), 413–420. Retrieved from http://dx.doi.org/10.1038/nmeth.2434 doi: 10.1038/nmeth.2434

Amos, B., & Kolter, J. Z. (2017). *Optnet: Differentiable optimization as a layer in neural networks.* arXiv. Retrieved from https://arxiv.org/abs/1703.00443 doi: 10.48550/ARXIV.1703.00443

Anderson, B., & Moore, J. (2007). *Optimal control: Linear quadratic methods*. Dover Publications. Retrieved from https://books.google.pt/books?id=fW6TAwAAQBAJ

Anderson, D. J., & Perona, P. (2014). Toward a science of computational ethology. *Neuron*, *84*(1), 18–31. Retrieved from http://dx.doi.org/10.1016/j.neuron.2014.09.005 doi: 10.1016/j.neuron.2014.09.005

Athalye, V. R., Khanna, P., Gowda, S., Orsborn, A. L., Costa, R. M., & Carmena, J. M. (2023). Invariant neural dynamics drive commands to control different movements. *Current Biology*, *33*(14), 2962–2976.e15. Retrieved from http://dx.doi.org/10.1016/j.cub.2023.06.027 doi: 10.1016/j.cub.2023.06.027

Batty, E., Whiteway, M., Saxena, S., Biderman, D., Abe, T., Musall, S., ... others (2019). BehaveNet: nonlinear embedding and Bayesian neural decoding of behavioral videos. *Advances in Neural Information Processing Systems*, *32*.

Beppi, C., Beringer, G., Straumann, D., & Bögli, S. Y. (2021). Light-stimulus intensity modulates startle reflex habituation in larval zebrafish. *Scientific Reports*, *11*(1). Retrieved from http://dx.doi.org/10.1038/s41598-021-00535-9 doi: 10.1038/s41598-021-00535-9

Berg, E. M., Mrowka, L., Bertuzzi, M., Madrid, D., Picton, L. D., & El Manira, A. (2023). Brainstem circuits encoding start, speed, and duration of swimming in adult zebrafish. *Neuron*, *111*(3), 372–386.e4. Retrieved from http://dx.doi.org/10.1016/j.neuron.2022.10.034 doi: 10.1016/j.neuron.2022.10.034

Bergeron, S. A., Carrier, N., Li, G. H., Ahn, S., & Burgess, H. A. (2014). Gsx1 expression defines neurons required for prepulse inhibition. *Molecular Psychiatry*, *20*(8), 974–985. Retrieved from http://dx.doi.org/10.1038/mp.2014.106 doi: 10.1038/mp.2014.106

Berkowitz, A., & Laurent, G. (1996). Central generation of grooming motor patterns and interlimb coordination in locusts. *The Journal of Neuroscience*, *16*(24), 8079–8091. Retrieved from http://dx.doi.org/10.1523/JNEUROSCI.16-24-08079.1996 doi: 10.1523/jneurosci.16-24-08079.1996

Berman, G. J., Choi, D. M., Bialek, W., & Shaevitz, J. W. (2014).

Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of The Royal Society Interface*, *11*(99), 20140672. Retrieved from http://dx.doi.org/10.1098/rsif.2014.0672 doi: 10.1098/rsif.2014.0672

Bhattacharjee, S., Kashyap, R., Abualait, T., Annabel Chen, S.-H., Yoo, W.-K., & Bashir, S. (2020). The role of primary motor cortex: More than movement execution. *Journal of Motor Behavior*, *53*(2), 258–274. Retrieved from http://dx.doi.org/10.1080/00222895.2020.1738992 doi: 10.1080/00222895.2020.1738992

Bian, T., Jiang, Y., & Jiang, Z.-P. (2014). Adaptive dynamic programming and optimal control of nonlinear nonaffine systems. *Automatica*, *50*(10), 2624–2632. Retrieved from http://dx.doi.org/10.1016/j.automatica.2014.08.023 doi: 10.1016/j.automatica.2014.08.023

Blelloch, G. (1989). Scans as primitive parallel operations. *IEEE Transactions on Computers*, *38*(11), 1526–1538. Retrieved from http://dx.doi.org/10.1109/12.42122 doi: 10.1109/12.42122

Blelloch, G. E. (1990). *Prefix sums and their applications* (Tech. Rep. No. CMU-CS-90-190). School of Computer Science, Carnegie Mellon University.

Blondel, M., Berthet, Q., Cuturi, M., Frostig, R., Hoyer, S., Llinares-López, F., ... Vert, J.-P. (2021). *Efficient and modular implicit differentiation.* arXiv. Retrieved from https://arxiv.org/abs/2105.15183 doi: 10.48550/ARXIV.2105.15183

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., ... Zhang, Q. (2018). *JAX: composable transformations of Python+NumPy programs.* Retrieved from http://github.com/google/jax

Brown, T. G. (1911). The intrinsic factors in the act of progression in the mammal. *Proceedings of the Royal Society of London. Series B, containing papers of a biological character*, *84*(572), 308–319.

Brunton, S. L., Brunton, B. W., Proctor, J. L., Kaiser, E., & Kutz, J. N. (2017). Chaos as an intermittently forced linear system. *Nature Communications*, *8*(1). Retrieved from http://dx.doi.org/10.1038/s41467-017-00030-8 doi: 10.1038/s41467-017-00030-8

Budick, S. A., & O'Malley, D. M. (2000). Locomotor repertoire of the

larval zebrafish: Swimming, turning and prey capture. *Journal of Experimental Biology*, *203*(17), 2565–2579. Retrieved from http://dx.doi.org/10.1242/jeb.203.17.2565 doi: 10.1242/jeb.203.17.2565

Burgess, H. A., & Granato, M. (2007). Modulation of locomotor activity in larval zebrafish during light adaptation. *Journal of Experimental Biology*, *210*(14), 2526–2539. Retrieved from http://dx.doi.org/10.1242/jeb.003939 doi: 10.1242/jeb.003939

Capelli, P., Pivetta, C., Soledad Esposito, M., & Arber, S. (2017). Locomotor speed control circuits in the caudal brainstem. *Nature*, *551*(7680), 373–377. Retrieved from http://dx.doi.org/10.1038/nature24064 doi: 10.1038/nature24064

Carbo-Tano, M., Lapoix, M., Jia, X., Thouvenin, O., Pascucci, M., Auclair, F., ... Wyart, C. (2023). The mesencephalic locomotor region recruits v2a reticulospinal neurons to drive forward locomotion in larval zebrafish. *Nature Neuroscience*, *26*(10), 1775–1790. Retrieved from http://dx.doi.org/10.1038/s41593-023-01418-0 doi: 10.1038/s41593-023-01418-0

Castiñeiras, J. R., & Renart, A. (2022). Control limited perceptual decision making. *bioRxiv*. Retrieved from http://dx.doi.org/10.1101/2022.06.24.497481 doi: 10.1101/2022.06.24.497481

Churchland, M. M., Cunningham, J. P., Kaufman, M. T., Foster, J. D., Nuyujukian, P., Ryu, S. I., & Shenoy, K. V. (2012). Neural population dynamics during reaching. *Nature*, *487*(7405), 51–56. Retrieved from http://dx.doi.org/10.1038/nature11129 doi: 10.1038/nature11129

Churchland, M. M., Santhanam, G., & Shenoy, K. V. (2006). Preparatory activity in premotor and motor cortex reflects the speed of the upcoming reach. *Journal of Neurophysiology*, *96*(6), 3130–3146. Retrieved from http://dx.doi.org/10.1152/jn.00307.2006 doi: 10.1152/jn.00307.2006

Churchland, M. M., & Shenoy, K. V. (2007). Temporal complexity and heterogeneity of single-neuron activity in premotor and motor cortex. *Journal of Neurophysiology*, *97*(6), 4235–4257. Retrieved from http://dx.doi.org/10.1152/jn.00095.2007 doi: 10.1152/jn.00095.2007

Churchland, M. M., Yu, B. M., Ryu, S. I., Santhanam, G., & Shenoy, K. V. (2006). Neural variability in premotor cortex provides a signature of motor preparation. *The Journal of Neuroscience*, *26*(14), 3697–3712. Retrieved from http://dx.doi.org/10.1523/JNEUROSCI.3762-05.2006 doi: 10.1523/jneurosci.3762-05.2006

Cohen, A., & Wallén, P. (1980). The neuronal correlate of locomotion in fish: Fictive swimming induced in an in vitro preparation of the lamprey spinal cord. *Experimental Brain Research*, *41*(1). Retrieved from http://dx.doi.org/10.1007/BF00236674 doi: 10.1007/bf00236674

Cong, L., Wang, Z., Chai, Y., Hang, W., Shang, C., Yang, W., ... Wen, Q. (2017). Rapid whole brain imaging of neural activity in freely behaving larval zebrafish (danio rerio). *eLife*, *6*. Retrieved from http://dx.doi.org/10.7554/eLife.28158 doi: 10.7554/elife.28158

Costa, A. C., Ahamed, T., Jordan, D., & Stephens, G. J. (2024). A markovian dynamics for caenorhabditis elegans behavior across scales. *Proceedings of the National Academy of Sciences*, *121*(32). Retrieved from http://dx.doi.org/10.1073/pnas.2318805121 doi: 10.1073/pnas.2318805121

Costa, A. C., Ahamed, T., & Stephens, G. J. (2019). Adaptive, locally linear models of complex dynamics. *Proceedings of the National Academy of Sciences*, *116*(5), 1501–1510.

Crutcher, M. D., & Alexander, G. E. (1990). Movement-related neuronal activity selectively coding either direction or muscle pattern in three motor areas of the monkey. *Journal of Neurophysiology*, *64*(1), 151–163. Retrieved from http://dx.doi.org/10.1152/jn.1990.64.1.151 doi: 10.1152/jn.1990.64.1.151

Cunningham, J. P., & Yu, B. M. (2014). Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, *17*(11), 1500–1509. Retrieved from http://dx.doi.org/10.1038/nn.3776 doi: 10.1038/nn.3776

Danos, N., & Lauder, G. V. (2007). The ontogeny of fin function during routine turns in zebrafishdanio rerio. *Journal of Experimental Biology*, *210*(19), 3374–3386. Retrieved from http://dx.doi.org/10.1242/jeb.007484 doi: 10.1242/jeb.007484

Datta, S. R., Anderson, D. J., Branson, K., Perona, P., & Leifer, A.

(2019). Computational neuroethology: A call to action. *Neuron*, *104*(1), 11–24. Retrieved from http://dx.doi.org/10.1016/j.neuron.2019.09.038 doi: 10.1016/j.neuron.2019.09.038

Dayan, P., Hinton, G. E., Neal, R. M., & Zemel, R. S. (1995). The helmholtz machine. *Neural Computation*, *7*(5), 889–904. Retrieved from http://dx.doi.org/10.1162/neco.1995.7.5.889 doi: 10.1162/neco.1995.7.5.889

DeepMind, Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., ... Viola, F. (2020). *The DeepMind JAX Ecosystem.* Retrieved from http://github.com/google-deepmind

Delp, S. L., Anderson, F. C., Arnold, A. S., Loan, P., Habib, A., John, C. T., ... Thelen, D. G. (2007). Opensim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, *54*(11), 1940–1950. Retrieved from http://dx.doi.org/10.1109/TBME.2007.901024 doi: 10.1109/tbme.2007.901024

Delvolvé, I., Branchereau, P., Dubuc, R., & Cabelguen, J.-M. (1999). Fictive rhythmic motor patterns induced by nmda in an in vitro brain stem–spinal cord preparation from an adult urodele. *Journal of Neurophysiology*, *82*(2), 1074–1077. Retrieved from http://dx.doi.org/10.1152/jn.1999.82.2.1074 doi: 10.1152/jn.1999.82.2.1074

de Vito, G., Turrini, L., Müllenbroich, C., Ricci, P., Sancataldo, G., Mazzamuto, G., ... Pavone, F. S. (2022). Fast whole-brain imaging of seizures in zebrafish larvae by two-photon light-sheet microscopy. *Biomedical Optics Express*, *13*(3), 1516. Retrieved from http://dx.doi.org/10.1364/BOE.434146 doi: 10.1364/boe.434146

Dreosti, E., Lopes, G., Kampff, A. R., & Wilson, S. W. (2015). Development of social behavior in young zebrafish. *Frontiers in Neural Circuits*, *9*. Retrieved from http://dx.doi.org/10.3389/fncir.2015.00039 doi: 10.3389/fncir.2015.00039

Dubreuil, A., Valente, A., Beiran, M., Mastrogiuseppe, F., & Ostojic, S. (2022). The role of population structure in computations through neural dynamics. *Nature Neuroscience*, *25*(6), 783–794. Retrieved from http://dx.doi.org/10.1038/s41593-022-01088-4 doi: 10.1038/s41593-022-01088-4

Dubuc, R., Cabelguen, J.-M., & Ryczko, D. (2023). Locomotor pattern generation and descending control: a historical perspective. *Journal of Neurophysiology*, *130*(2), 401–416. Retrieved from http://dx.doi.org/10.1152/jn.00204.2023 doi: 10.1152/jn.00204.2023

Duncker, L., & Sahani, M. (2021). Dynamics on the manifold: Identifying computational dynamical activity from neural population recordings. *Current Opinion in Neurobiology*, *70*, 163–170. Retrieved from http://dx.doi.org/10.1016/j.conb.2021.10.014 doi: 10.1016/j.conb.2021.10.014

Dunn, T. W., Gebhardt, C., Naumann, E. A., Riegler, C., Ahrens, M. B., Engert, F., & Del Bene, F. (2016). Neural circuits underlying visually evoked escapes in larval zebrafish. *Neuron*, *89*(3), 613–628. Retrieved from http://dx.doi.org/10.1016/j.neuron.2015.12.021 doi: 10.1016/j.neuron.2015.12.021

Durstewitz, D., Koppe, G., & Thurm, M. I. (2023). Reconstructing computational system dynamics from neural data with recurrent neural networks. *Nature Reviews Neuroscience*, *24*(11), 693–710. Retrieved from http://dx.doi.org/10.1038/s41583-023-00740-7 doi: 10.1038/s41583-023-00740-7

Ehrlich, D. E., & Schoppik, D. (2019). A primal role for the vestibular sense in the development of coordinated locomotion. *eLife*, *8*. Retrieved from http://dx.doi.org/10.7554/eLife.45839 doi: 10.7554/elife.45839

Feng, L., Tung, F., Ahmed, M. O., Bengio, Y., & Hajimirsadeghi, H. (2024). *Were rnns all we needed?* arXiv. Retrieved from https://arxiv.org/abs/2410.01201 doi: 10.48550/ARXIV.2410.01201

Fieseler, C., Zimmer, M., & Kutz, J. N. (2020). Unsupervised learning of control signals and their encodings in caenorhabditis elegans whole-brain recordings. *Journal of The Royal Society Interface*, *17*(173), 20200459. Retrieved from http://dx.doi.org/10.1098/rsif.2020.0459 doi: 10.1098/rsif.2020.0459

Fotoohi Bafghi, M. H., Effati, S., & Solaymani Fard, O. (2022). A numerical method for solving stochastic linear quadratic problem with a finance application. *Journal of Mathematical Modeling*(Online First). Retrieved from https://doi.org/10.22124/jmm.2022.20887.1826 doi: 10.22124/jmm.2022.20887.1826

Fu, Q. G., Flament, D., Coltz, J. D., & Ebner, T. J. (1995). Temporal encoding of movement kinematics in the discharge of primate primary motor and premotor neurons. *Journal of Neurophysiology*, *73*(2), 836–854. Retrieved from http://dx.doi.org/10.1152/jn.1995.73.2.836 doi: 10.1152/jn.1995.73.2.836

Funahashi, S., & Takeda, K. (2002). *Journal of Biological Physics*, *28*(3), 527–537. Retrieved from http://dx.doi.org/10.1023/A:1020309916014 doi: 10.1023/a:1020309916014

Gahtan, E., Sankrithi, N., Campos, J. B., & O'Malley, D. M. (2002). Evidence for a widespread brain stem escape network in larval zebrafish. *Journal of Neurophysiology*, *87*(1), 608–614. Retrieved from http://dx.doi.org/10.1152/jn.00596.2001 doi: 10.1152/jn.00596.2001

Georgopoulos, A., Crutcher, M., & Schwartz, A. (1989). Cognitive spatial-motor processes: 3. motor cortical prediction of movement direction during an instructed delay period. *Experimental Brain Research*, *75*(1). Retrieved from http://dx.doi.org/10.1007/BF00248541 doi: 10.1007/bf00248541

Georgopoulos, A., Kalaska, J., Caminiti, R., & Massey, J. (1982). On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *The Journal of Neuroscience*, *2*(11), 1527–1537. Retrieved from http://dx.doi.org/10.1523/JNEUROSCI.02-11-01527.1982 doi: 10.1523/jneurosci.02-11-01527.1982

Georgopoulos, A. P., Caminiti, R., Kalaska, J. F., & Massey, J. T. (1983). Spatial coding of movement: A hypothesis concerning the coding of movement direction by motor cortical populations. In *Neural coding of motor performance* (p. 327–336). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-68915-4_34 doi: 10.1007/978-3-642-68915-4_34

Georgopoulos, A. P., Schwartz, A. B., & Kettner, R. E. (1986). Neuronal population coding of movement direction. *Science*, *233*(4771), 1416–1419. Retrieved from http://dx.doi.org/10.1126/science.3749885 doi: 10.1126/science.3749885

Goldberg, J. M., & Brown, P. B. (1969). Response of binaural neurons of dog superior olivary complex to dichotic tonal stimuli: some physiological mechanisms of sound localization. *Journal of Neurophysiology*, *32*(4), 613–636. Retrieved from http://dx.doi.org/10.1152/jn.1969.32.4.613 doi: 10.1152/jn.1969.32.4.613

Golub, M., & Sussillo, D. (2018). Fixedpointfinder: A tensorflow toolbox for identifying and characterizing fixed points in recurrent neural networks. *Journal of Open Source Software*, *3*(31), 1003. Retrieved from http://dx.doi.org/10.21105/joss.01003 doi: 10.21105/joss.01003

Gondur, R., Sikandar, U. B., Schaffer, E., Aoi, M. C., & Keeley, S. L. (2024). Multi-modal gaussian process variational autoencoders for neural and behavioral data. In *The twelfth international conference on learning representations*. Retrieved from https://openreview.net/forum?id=aGH43rjoe4

Gonzalez, X., Warrington, A., Smith, J. T. H., & Linderman, S. W. (2024). *Towards scalable and stable parallelization of nonlinear rnns.* arXiv. Retrieved from https://arxiv.org/abs/2407.19115 doi: 10.48550/ARXIV.2407.19115

Graziano, M. S. A., Cooke, D. F., Taylor, C. S. R., & Moore, T. (2004). Distribution of hand location in monkeys during spontaneous behavior. *Experimental Brain Research*, *155*(1), 30–36. Retrieved from http://dx.doi.org/10.1007/s00221-003-1701-4 doi: 10.1007/s00221-003-1701-4

Green, M., & Limebeer, D. (2012). *Linear robust control*. Dover Publications, Incorporated. Retrieved from https://books.google.pt/books?id=CI-DyLffACcC

Green, M. H., & Hale, M. E. (2012). Activity of pectoral fin motoneurons during two swimming gaits in the larval zebrafish (danio rerio) and localization of upstream circuit elements. *Journal of Neurophysiology*, *108*(12), 3393–3402. Retrieved from http://dx.doi.org/10.1152/jn.00623.2012 doi: 10.1152/jn.00623.2012

Green, M. H., Ho, R. K., & Hale, M. E. (2011). Movement and function of the pectoral fins of the larval zebrafish (danio rerio) during slow swimming. *Journal of Experimental Biology*, *214*(18), 3111–3123. Retrieved from http://dx.doi.org/10.1242/jeb.057497 doi: 10

.1242/jeb.057497

Grillner, S. (1981). *Control of locomotion in bipeds, tetrapods, and fish.* Wiley. Retrieved from http://dx.doi.org/10.1002/cphy.cp010226 doi: 10.1002/cphy.cp010226

Grillner, S. (1985). Neural control of vertebrate locomotion - central mechanisms and reflex interaction with special reference to the cat. In *Feedback and motor control in invertebrates and vertebrates* (p. 35–56). Springer Netherlands. Retrieved from http://dx.doi.org/10.1007/978-94-011-7084-0_3 doi: 10.1007/978-94-011-7084-0_3

Grillner, S. (2003). The motor infrastructure: from ion channels to neuronal networks. *Nature Reviews Neuroscience*, *4*(7), 573–586. Retrieved from http://dx.doi.org/10.1038/nrn1137 doi: 10.1038/nrn1137

Grillner, S. (2008). The neural control of movement. *Journal of Men's Health*, *5*(3), A5–A6. Retrieved from http://dx.doi.org/10.1016/j.jomh.2008.06.010 doi: 10.1016/j.jomh.2008.06.010

Grillner, S., & El Manira, A. (2020). Current principles of motor control, with special reference to vertebrate locomotion. *Physiological Reviews*, *100*(1), 271–320. Retrieved from http://dx.doi.org/10.1152/physrev.00015.2019 doi: 10.1152/physrev.00015.2019

Grillner, S., & Zangger, P. (1975). How detailed is the central pattern generation for locomotion? *Brain Research*, *88*(2), 367–371. Retrieved from http://dx.doi.org/10.1016/0006-8993(75)90401-1 doi: 10.1016/0006-8993(75)90401-1

Groneberg, A. H., Marques, J. C., Martins, A. L., Diez del Corral, R., de Polavieja, G. G., & Orger, M. B. (2020). Early-life social experience shapes social avoidance reactions in larval zebrafish. *Current Biology*, *30*(20), 4009–4021.e4. Retrieved from http://dx.doi.org/10.1016/j.cub.2020.07.088 doi: 10.1016/j.cub.2020.07.088

Guo, N., Lin, J., Peng, X., Chen, H., Zhang, Y., Liu, X., & Li, Q. (2015). Influences of acute ethanol exposure on locomotor activities of zebrafish larvae under different illumination. *Alcohol*, *49*(7), 727–737. Retrieved from http://dx.doi.org/10.1016/j.alcohol.2015.08.003 doi: 10.1016/j.alcohol.2015.08.003

Hale, M. E. (2014). Developmental change in the function of movement systems: Transition of the pectoral fins between respiratory and locomotor roles in zebrafish. *Integrative and Comparative Biology*, *54*(2), 238–249. Retrieved from http://dx.doi.org/10.1093/icb/icu014 doi: 10.1093/icb/icu014

Hennequin, G., Vogels, T. P., & Gerstner, W. (2012). Non-normal amplification in random balanced neuronal networks. *Physical Review E*, *86*(1). Retrieved from http://dx.doi.org/10.1103/PhysRevE.86.011909 doi: 10.1103/physreve.86.011909

Hennequin, G., Vogels, T. P., & Gerstner, W. (2014). Optimal control of transient dynamics in balanced networks supports generation of complex movements. *Neuron*, *82*(6), 1394–1406. Retrieved from http://dx.doi.org/10.1016/j.neuron.2014.04.045 doi: 10.1016/j.neuron.2014.04.045

Hoffmann, M., Henninger, J., Veith, J., Richter, L., & Judkewitz, B. (2023). Blazed oblique plane microscopy reveals scale-invariant inference of brain-wide population activity. *Nature Communications*, *14*(1). Retrieved from http://dx.doi.org/10.1038/s41467-023-43741-x doi: 10.1038/s41467-023-43741-x

Howell, T. A., Jackson, B. E., & Manchester, Z. (2019). Altro: A fast solver for constrained trajectory optimization. In *2019 ieee/rsj international conference on intelligent robots and systems (iros)*. IEEE. Retrieved from http://dx.doi.org/10.1109/IROS40897.2019.8967788 doi: 10.1109/iros40897.2019.8967788

Huang, K.-H., Ahrens, M., Dunn, T., & Engert, F. (2013). Spinal projection neurons control turning behaviors in zebrafish. *Current Biology*, *23*(16), 1566–1573. Retrieved from http://dx.doi.org/10.1016/j.cub.2013.06.044 doi: 10.1016/j.cub.2013.06.044

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, *148*(3), 574–591. Retrieved from http://dx.doi.org/10.1113/jphysiol.1959.sp006308 doi: 10.1113/jphysiol.1959.sp006308

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, *160*(1), 106–154. Retrieved from http://dx.doi.org/10.1113/jphysiol.1962.sp006837 doi: 10.1113/

jphysiol.1962.sp006837

Hubel, D. H., & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, *195*(1), 215–243. Retrieved from http://dx.doi.org/10.1113/jphysiol.1968.sp008455 doi: 10.1113/jphysiol.1968.sp008455

Hurwitz, C., Srivastava, A., Xu, K., Jude, J., Perich, M., Miller, L., & Hennig, M. (2021). Targeted neural dynamical modeling. *Advances in Neural Information Processing Systems*, *34*, 29379–29392.

Ijspeert, A. J., Crespi, A., Ryczko, D., & Cabelguen, J.-M. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, *315*(5817), 1416–1420. Retrieved from http://dx.doi.org/10.1126/science.1138353 doi: 10.1126/science.1138353

Ikeda, H., Delargy, A. H., Yokogawa, T., Urban, J. M., Burgess, H. A., & Ono, F. (2013). Intrinsic properties of larval zebrafish neurons in ethanol. *PLoS ONE*, *8*(5), e63318. Retrieved from http://dx.doi.org/10.1371/journal.pone.0063318 doi: 10.1371/journal.pone.0063318

Jia, X., & Wyart, C. (2024). Holographic optogenetic activation of neurons eliciting locomotion in head-embedded larval zebrafish. In J. F. Amatruda, C. Houart, K. Kawakami, & K. D. Poss (Eds.), *Zebrafish: Methods and protocols* (pp. 125–140). New York, NY: Springer US. Retrieved from https://doi.org/10.1007/978-1-0716-3401-1_8 doi: 10.1007/978-1-0716-3401-1_8

Johnson, R. E., Linderman, S., Panier, T., Wee, C. L., Song, E., Herrera, K. J., ... Engert, F. (2020). Probabilistic models of larval zebrafish behavior reveal structure on many scales. *Current Biology*, *30*(1), 70–82.e4. Retrieved from https://doi.org/10.1016/j.cub.2019.11.026 doi: 10.1016/j.cub.2019.11.026

Jouary, A., Silva, P. T., Laborde, A., Mata, J. M., Marques, J. C., Collins, E., ... Orger, M. B. (2024). Megabouts: a flexible pipeline for zebrafish locomotion analysis. *bioRxiv*. Retrieved from https://www.biorxiv.org/content/early/2024/11/28/2024.09.14.613078 doi: 10.1101/2024.09.14.613078

Kalidindi, H. T., Cross, K. P., Lillicrap, T. P., Omrani, M., Falotico, E., Sabes, P. N., & Scott, S. H. (2021). Rotational dynamics in

motor cortex are consistent with a feedback controller. *eLife*, *10*. Retrieved from http://dx.doi.org/10.7554/eLife.67256 doi: 10.7554/elife.67256

Kao, T.-C., & Hennequin, G. (2019). Neuroscience out of control: control-theoretic perspectives on neural circuit dynamics. *Current Opinion in Neurobiology*, *58*, 122–129. Retrieved from https://doi.org/10.1016/j.conb.2019.09.001 doi: 10.1016/j.conb.2019.09.001

Kato, S., Kaplan, H. S., Schrödel, T., Skora, S., Lindsay, T. H., Yemini, E., ... Zimmer, M. (2015). Global brain dynamics embed the motor command sequence of caenorhabditis elegans. *Cell*, *163*(3), 656–669. Retrieved from http://dx.doi.org/10.1016/j.cell.2015.09.034 doi: 10.1016/j.cell.2015.09.034

Katz, P. S. (1996). Neurons, networks, and motor behavior. *Neuron*, *16*(2), 245–253. Retrieved from http://dx.doi.org/10.1016/S0896-6273(00)80043-4 doi: 10.1016/s0896-6273(00)80043-4

Kaufman, M. T., Churchland, M. M., Ryu, S. I., & Shenoy, K. V. (2014). Cortical activity in the null space: permitting preparation without movement. *Nature Neuroscience*, *17*(3), 440–448. Retrieved from http://dx.doi.org/10.1038/nn.3643 doi: 10.1038/nn.3643

Kaufman, M. T., Seely, J. S., Sussillo, D., Ryu, S. I., Shenoy, K. V., & Churchland, M. M. (2016). The largest response component in the motor cortex reflects movement timing but not movement type. *eneuro*, *3*(4), ENEURO.0085–16.2016. Retrieved from http://dx.doi.org/10.1523/ENEURO.0085-16.2016 doi: 10.1523/eneuro.0085-16.2016

Kawato, M. (1999). Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, *9*(6), 718–727. Retrieved from http://dx.doi.org/10.1016/s0959-4388(99)00028-8 doi: 10.1016/s0959-4388(99)00028-8

Keller, P. J., & Ahrens, M. B. (2015). Visualizing whole-brain activity and development at the single-cell level using light-sheet microscopy. *Neuron*, *85*(3), 462–483. Retrieved from http://dx.doi.org/10.1016/j.neuron.2014.12.039 doi: 10.1016/j.neuron.2014.12.039

Keshtkaran, M. R., Sedler, A. R., Chowdhury, R. H., Tandon, R., Basrai, D., Nguyen, S. L., ... Pandarinath, C. (2022). A large-scale neural network training framework for general-

ized estimation of single-trial population dynamics. *bioRxiv*. Retrieved from https://www.biorxiv.org/content/early/2022/06/07/2021.01.13.426570 doi: 10.1101/2021.01.13.426570

Kim, D. H., Kim, J., Marques, J. C., Grama, A., Hildebrand, D. G. C., Gu, W., ... Robson, D. N. (2017). Pan-neuronal calcium imaging with cellular resolution in freely swimming zebrafish. *Nature Methods*, *14*(11), 1107–1114. Retrieved from http://dx.doi.org/10.1038/NMETH.4429 doi: 10.1038/nmeth.4429

Kimura, Y., Satou, C., Fujioka, S., Shoji, W., Umeda, K., Ishizuka, T., ... Higashijima, S.-i. (2013). Hindbrain v2a neurons in the excitation of spinal locomotor circuits during zebrafish swimming. *Current Biology*, *23*(10), 843–849. Retrieved from http://dx.doi.org/10.1016/j.cub.2013.03.066 doi: 10.1016/j.cub.2013.03.066

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.

Kirk, D. (2004). *Optimal control theory: An introduction*. Dover Publications. Retrieved from https://books.google.pt/books?id=fCh2SAtWIdwC

Kirk, D. E. (2012). Optimal control theory: An introduction. In (p. 50-96). Dover Publications. Retrieved from https://books.google.pt/books?id=onuH0PnZwV4C

Knüsel, J., Crespi, A., Cabelguen, J.-M., Ijspeert, A. J., & Ryczko, D. (2020). Reproducing five motor behaviors in a salamander robot with virtual muscles and a distributed cpg controller regulated by drive signals and proprioceptive feedback. *Frontiers in Neurorobotics*, *14*. Retrieved from http://dx.doi.org/10.3389/fnbot.2020.604426 doi: 10.3389/fnbot.2020.604426

Kokel, D., Dunn, T. W., Ahrens, M. B., Alshut, R., Cheung, C. Y. J., Saint-Amant, L., ... Peterson, R. T. (2013). Identification of nonvisual photomotor response cells in the vertebrate hindbrain. *The Journal of Neuroscience*, *33*(9), 3834–3843. Retrieved from http://dx.doi.org/10.1523/JNEUROSCI.3689-12.2013 doi: 10.1523/jneurosci.3689-12.2013

Kudryashova, N., Perich, M. G., Miller, L. E., & Hennig, M. H. (2023). Ctrl-tndm: Decoding feedback-driven movement corrections from motor cortex neurons. In *Computational and systems neuroscience*

(cosyne) 2023 (pp. 207–216). Montréal, Canada: Cosyne 2023.

Kuo, P. D., & Eliasmith, C. (2005). Integrating behavioral and neural data in a model of zebrafish network interaction. *Biological Cybernetics*, *93*(3), 178–187. Retrieved from http://dx.doi.org/10.1007/s00422-005-0576-9 doi: 10.1007/s00422-005-0576-9

Kutz, N. J., Brunton, S. L., Brunton, B. W., & Proctor, J. L. (2016). In *Dynamic mode decomposition* (p. 185–193). Society for Industrial and Applied Mathematics. Retrieved from http://dx.doi.org/10.1137/1.9781611974508.ch12 doi: 10.1137/1.9781611974508.ch12

Lackner, S. (2018). *Demonstration of a bout categorization algorithm to study effects of pharmcological manipulations on locomotor behaviour in response to illumination changes* (Unpublished doctoral dissertation). Oeiras Portugal.

Lau, J. Y. N. (2019). *Reticulospinal recruitment during locomotion* (Unpublished doctoral dissertation). London UK.

Laub, A., Heath, M., Paige, C., & Ward, R. (1987). Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms. *IEEE Transactions on Automatic Control*, *32*(2), 115–122. Retrieved from https://doi.org/10.1109/tac.1987.1104549 doi: 10.1109/tac.1987.1104549

Lemieux, M., & Bretzner, F. (2019). Glutamatergic neurons of the gigantocellular reticular nucleus shape locomotor pattern and rhythm in the freely behaving mouse. *PLOS Biology*, *17*(4), e2003880. Retrieved from http://dx.doi.org/10.1371/journal.pbio.2003880 doi: 10.1371/journal.pbio.2003880

Li, W., & Todorov, E. (2004). Iterative linear quadratic regulator design for nonlinear biological movement systems. In *International conference on informatics in control, automation and robotics.* Retrieved from https://api.semanticscholar.org/CorpusID:19300

Linderman, S. W., Miller, A. C., Adams, R. P., Blei, D. M., Paninski, L., & Johnson, M. J. (2016). *Recurrent switching linear dynamical systems.* arXiv. Retrieved from https://arxiv.org/abs/1610.08466 doi: 10.48550/ARXIV.1610.08466

Litwin-Kumar, A., Harris, K. D., Axel, R., Sompolinsky, H., & Abbott, L. (2017). Optimal degrees of synaptic connectivity. *Neu-*

ron, *93*(5), 1153–1164.e7. Retrieved from http://dx.doi.org/10.1016/j.neuron.2017.01.030 doi: 10.1016/j.neuron.2017.01.030

Liu, K. S., & Fetcho, J. R. (1999). Laser ablations reveal functional relationships of segmental hindbrain neurons in zebrafish. *Neuron*, *23*(2), 325–335. Retrieved from http://dx.doi.org/10.1016/S0896-6273(00)80783-7 doi: 10.1016/s0896-6273(00)80783-7

Lurito, J., Georgakopoulos, T., & Georgopoulos, A. (1991). Cognitive spatial-motor processes: 7. the making of movements at an angle from a stimulus direction: studies of motor cortical activity at the single cell and population levels. *Experimental Brain Research*, *87*(3). Retrieved from http://dx.doi.org/10.1007/BF00227082 doi: 10.1007/bf00227082

MacKay-Lyons, M. (2002). Central pattern generation of locomotion: A review of the evidence. *Physical Therapy*, *82*(1), 69–83. Retrieved from http://dx.doi.org/10.1093/ptj/82.1.69 doi: 10.1093/ptj/82.1.69

Mante, V., Sussillo, D., Shenoy, K. V., & Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, *503*(7474), 78–84. Retrieved from http://dx.doi.org/10.1038/nature12742 doi: 10.1038/nature12742

Marder, E., & Bucher, D. (2001). Central pattern generators and the control of rhythmic movements. *Current Biology*, *11*(23), R986–R996. Retrieved from http://dx.doi.org/10.1016/S0960-9822(01)00581-4 doi: 10.1016/s0960-9822(01)00581-4

Marder, E., Bucher, D., Schulz, D. J., & Taylor, A. L. (2005). Invertebrate central pattern generation moves along. *Current Biology*, *15*(17), R685–R699. Retrieved from http://dx.doi.org/10.1016/j.cub.2005.08.022 doi: 10.1016/j.cub.2005.08.022

Markov, D. A., Petrucco, L., Kist, A. M., & Portugues, R. (2021). A cerebellar internal model calibrates a feedback controller involved in sensorimotor control. *Nature Communications*, *12*(1). Retrieved from http://dx.doi.org/10.1038/s41467-021-26988-0 doi: 10.1038/s41467-021-26988-0

Markowitz, J. E., Gillis, W. F., Jay, M., Wood, J., Harris, R. W., Cieszkowski, R., ... others (2023). Spontaneous behaviour is structured by reinforcement without explicit reward. *Nature*, *614*(7946),

108–117.

Marques, J. C., Lackner, S., Félix, R., & Orger, M. B. (2018). Structure of the zebrafish locomotor repertoire revealed with unsupervised behavioral clustering. *Current Biology*, *28*(2), 181–195.e5. Retrieved from http://dx.doi.org/10.1016/j.cub.2017.12.002 doi: 10.1016/j.cub.2017.12.002

Marques, J. C., Li, M., Schaak, D., Robson, D. N., & Li, J. M. (2019). Internal state dynamics shape brainwide activity and foraging behaviour. *Nature*, *577*(7789), 239–243. Retrieved from http://dx.doi.org/10.1038/s41586-019-1858-z doi: 10.1038/s41586-019-1858-z

Mastrogiuseppe, F., & Ostojic, S. (2018). Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, *99*(3), 609–623.e29. Retrieved from http://dx.doi.org/10.1016/j.neuron.2018.07.003 doi: 10.1016/j.neuron.2018.07.003

Maunsell, J. H., & Van Essen, D. C. (1983). Functional properties of neurons in middle temporal visual area of the macaque monkey. i. selectivity for stimulus direction, speed, and orientation. *Journal of Neurophysiology*, *49*(5), 1127–1147. Retrieved from http://dx.doi.org/10.1152/jn.1983.49.5.1127 doi: 10.1152/jn.1983.49.5.1127

McInnes, L., Healy, J., & Melville, J. (2018). *Umap: Uniform manifold approximation and projection for dimension reduction.* arXiv. Retrieved from https://arxiv.org/abs/1802.03426 doi: 10.48550/ARXIV.1802.03426

Mearns, D. S., Donovan, J. C., Fernandes, A. M., Semmelhack, J. L., & Baier, H. (2020). Deconstructing hunting behavior reveals a tightly coupled stimulus-response loop. *Current Biology*, *30*(1), 54–69.e9. Retrieved from http://dx.doi.org/10.1016/j.cub.2019.11.022 doi: 10.1016/j.cub.2019.11.022

Michaels, J. A., Dann, B., & Scherberger, H. (2016). Neural population dynamics during reaching are better explained by a dynamical system than representational tuning. *PLOS Computational Biology*, *12*(11), e1005175. Retrieved from http://dx.doi.org/10.1371/journal.pcbi.1005175 doi: 10.1371/journal.pcbi.1005175

Minchala-Avila, L. I., Garza-Castañón, L. E., Vargas-Martínez, A., & Zhang, Y. (2015). A review of optimal control techniques applied to the energy management and control of microgrids. *Procedia Computer Science*, *52*, 780–787. Retrieved from http://dx.doi.org/10.1016/j.procs.2015.05.133 doi: 10.1016/j.procs.2015.05.133

Miyazawa, H., Okumura, K., Hiyoshi, K., Maruyama, K., Kakinuma, H., Amo, R., ... Tsuda, S. (2018). Optical interrogation of neuronal circuitry in zebrafish using genetically encoded voltage indicators. *Scientific Reports*, *8*(1). Retrieved from http://dx.doi.org/10.1038/s41598-018-23906-1 doi: 10.1038/s41598-018-23906-1

Ml, S., Fv, S., & Gn, O. (1966). Control of walking and running by means of electric stimulation of the midbrain. *Biofizika*, *11*, 659. Retrieved from https://api.semanticscholar.org/CorpusID:82096385

Moore, B. (1981). Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, *26*(1), 17–32. Retrieved from https://doi.org/10.1109/tac.1981.1102568 doi: 10.1109/tac.1981.1102568

Moran, D. W., & Schwartz, A. B. (1999). Motor cortical representation of speed and direction during reaching. *Journal of Neurophysiology*, *82*(5), 2676–2692. Retrieved from http://dx.doi.org/10.1152/jn.1999.82.5.2676 doi: 10.1152/jn.1999.82.5.2676

Mori, S. (1987). Integration of posture and locomotion in acute decerebrate cats and in awake, freely moving cats. *Progress in Neurobiology*, *28*(2), 161–195. Retrieved from http://dx.doi.org/10.1016/0301-0082(87)90010-4 doi: 10.1016/0301-0082(87)90010-4

Morrow, M. M., Jordan, L. R., & Miller, L. E. (2007). Direct comparison of the task-dependent discharge of m1 in hand space and muscle space. *Journal of Neurophysiology*, *97*(2), 1786–1798. Retrieved from http://dx.doi.org/10.1152/jn.00150.2006 doi: 10.1152/jn.00150.2006

Mu, Y., Narayan, S., Mensh, B. D., & Ahrens, M. B. (2020). Brain-wide, scale-wide physiology underlying behavioral flexibility in zebrafish. *Current Opinion in Neurobiology*, *64*, 151–160. Retrieved from http://dx.doi.org/10.1016/j.conb.2020.08.013 doi: 10.1016/j.conb.2020.08.013

Mullen, T. S. (2024). *Fish behaviour simulator.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.11406468 doi: 10.5281/zenodo.11406468

Mullen, T. S., & Schimel, M. (2025). *DiffiLQRAX: Differentiable iLQR in JAX.* Retrieved from https://zenodo.org/doi/10.5281/zenodo.14861292 doi: 10.5281/ZENODO.14861292

Mullen, T. S., Schimel, M., Hennequin, G., Machens, C. K., Orger, M., & Jouary, A. (2024). Learning interpretable control inputs and dynamics underlying animal locomotion. In *The twelfth international conference on learning representations.* Retrieved from https://openreview.net/forum?id=MFCjgEOLJT

Nath, T., Mathis, A., Chen, A. C., Patel, A., Bethge, M., & Mathis, M. W. (2019). Using deeplabcut for 3d markerless pose estimation across species and behaviors. *Nature Protocols*, *14*(7), 2152–2176. Retrieved from http://dx.doi.org/10.1038/s41596-019-0176-0 doi: 10.1038/s41596-019-0176-0

Naumann, E. A., Fitzgerald, J. E., Dunn, T. W., Rihel, J., Sompolinsky, H., & Engert, F. (2016). From whole-brain data to functional circuit models: The zebrafish optomotor response. *Cell*, *167*(4), 947–960.e20. Retrieved from http://dx.doi.org/10.1016/j.cell.2016.10.019 doi: 10.1016/j.cell.2016.10.019

Orger, M. B., Kampff, A. R., Severi, K. E., Bollmann, J. H., & Engert, F. (2008). Control of visually guided behavior by distinct populations of spinal projection neurons. *Nature Neuroscience*, *11*(3), 327–333. Retrieved from http://dx.doi.org/10.1038/nn2048 doi: 10.1038/nn2048

Orger, M. B., Smear, M. C., Anstis, S. M., & Baier, H. (2000). Perception of fourier and non-fourier motion by larval zebrafish. *Nature Neuroscience*, *3*(11), 1128–1133. Retrieved from http://dx.doi.org/10.1038/80649 doi: 10.1038/80649

Orlovskiĭ, G., Severin, F., & Shik, M. (1966). Locomotion induced by stimulation of the mesencephalon. *Doklady Akademii nauk SSSR*, *169*(5), 1223–1226.

O'Shea, D. J., Duncker, L., Goo, W., Sun, X., Vyas, S., Trautmann, E. M., ... Shenoy, K. V. (2022). Direct neural perturbations reveal a dynamical mechanism for robust computation. *bioRxiv*. Retrieved

from http://dx.doi.org/10.1101/2022.12.16.520768 doi: 10.1101/2022.12.16.520768

O'Malley, D. M., Kao, Y.-H., & Fetcho, J. R. (1996). Imaging the functional organization of zebrafish hindbrain segments during escape behaviors. *Neuron*, *17*(6), 1145–1155. Retrieved from http://dx.doi.org/10.1016/s0896-6273(00)80246-9 doi: 10.1016/s0896-6273(00)80246-9

Padoan, A., Forni, F., & Sepulchre, R. (2021). Balanced truncation for model reduction of biological oscillators. *Biological Cybernetics*, *115*(4), 383–395. Retrieved from http://dx.doi.org/10.1007/s00422-021-00888-4 doi: 10.1007/s00422-021-00888-4

Pandarinath, C., Ames, K. C., Russo, A. A., Farshchian, A., Miller, L. E., Dyer, E. L., & Kao, J. C. (2018). Latent factors and dynamics in motor cortex and their application to brain–machine interfaces. *The Journal of Neuroscience*, *38*(44), 9390–9401. Retrieved from http://dx.doi.org/10.1523/JNEUROSCI.1669-18.2018 doi: 10.1523/jneurosci.1669-18.2018

Pandarinath, C., O'Shea, D. J., Collins, J., Jozefowicz, R., Stavisky, S. D., Kao, J. C., ... Sussillo, D. (2018). Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, *15*(10), 805–815. Retrieved from http://dx.doi.org/10.1038/s41592-018-0109-9 doi: 10.1038/s41592-018-0109-9

Park, Y., Rossi, R. A., Wen, Z., Wu, G., & Zhao, H. (2020). *Structured policy iteration for linear quadratic regulator.* arXiv. Retrieved from https://arxiv.org/abs/2007.06202 doi: 10.48550/ARXIV.2007.06202

Patton, E. E., Zon, L. I., & Langenau, D. M. (2021). Zebrafish disease models in drug discovery: from preclinical modelling to clinical trials. *Nature Reviews Drug Discovery*, *20*(8), 611–628. Retrieved from http://dx.doi.org/10.1038/s41573-021-00210-8 doi: 10.1038/s41573-021-00210-8

Pearson, K. (2000). Motor systems. *Current Opinion in Neurobiology*, *10*(5), 649–654. Retrieved from http://dx.doi.org/10.1016/S0959-4388(00)00130-6 doi: 10.1016/s0959-4388(00)00130-6

Pehlevan, C., Paoletti, P., & Mahadevan, L. (2016). Integrative neuromechanics of crawling in d. melanogaster larvae. *eLife*, *5*. Retrieved

from http://dx.doi.org/10.7554/eLife.11031 doi: 10.7554/elife.11031

Pereira, T. D., Shaevitz, J. W., & Murthy, M. (2020). Quantifying behavior to understand the brain. *Nature neuroscience*, *23*(12), 1537–1549.

Pereira, T. D., Tabris, N., Matsliah, A., Turner, D. M., Li, J., Ravindranath, S., ... Murthy, M. (2022). Sleap: A deep learning system for multi-animal pose tracking. *Nature Methods*, *19*(4), 486–495. Retrieved from http://dx.doi.org/10.1038/s41592-022-01426-1 doi: 10.1038/s41592-022-01426-1

Petrucco, L., Lavian, H., Wu, Y. K., Svara, F., Štih, V., & Portugues, R. (2023). Neural dynamics and architecture of the heading direction circuit in zebrafish. *Nature Neuroscience*, *26*(5), 765–773. Retrieved from http://dx.doi.org/10.1038/s41593-023-01308-5 doi: 10.1038/s41593-023-01308-5

Portugues, R., & Engert, F. (2011). Adaptive locomotor behavior in larval zebrafish. *Frontiers in Systems Neuroscience*, *5*. Retrieved from http://dx.doi.org/10.3389/fnsys.2011.00072 doi: 10.3389/fnsys.2011.00072

Portugues, R., Feierstein, C. E., Engert, F., & Orger, M. B. (2014). Whole-brain activity maps reveal stereotyped, distributed networks for visuomotor behavior. *Neuron*, *81*(6), 1328–1343. Retrieved from http://dx.doi.org/10.1016/j.neuron.2014.01.019 doi: 10.1016/j.neuron.2014.01.019

Privat, M., & Sumbre, G. (2020). Naturalistic behavior: The zebrafish larva strikes back. *Current Biology*, *30*(1), R27–R29. Retrieved from http://dx.doi.org/10.1016/j.cub.2019.11.014 doi: 10.1016/j.cub.2019.11.014

Pujala, A., & Koyama, M. (2019). Chronology-based architecture of descending circuits that underlie the development of locomotor repertoire after birth. *eLife*, *8*. Retrieved from http://dx.doi.org/10.7554/eLife.42135 doi: 10.7554/elife.42135

Raghu, M., Gilmer, J., Yosinski, J., & Sohl-Dickstein, J. (2017). *Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability.* arXiv. Retrieved from https://arxiv.org/abs/1706.05806 doi: 10.48550/ARXIV.1706.05806

Rajan, G., Lafaye, J., Faini, G., Carbo-Tano, M., Duroure, K., Tanese, D., ... Del Bene, F. (2022). Evolutionary divergence of locomotion in two related vertebrate species. *Cell Reports*, *38*(13), 110585. Retrieved from http://dx.doi.org/10.1016/j.celrep.2022.110585 doi: 10.1016/j.celrep.2022.110585

Recht, B. (2019). A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, *2*(1), 253–279. Retrieved from http://dx.doi.org/10.1146/annurev-control-053018-023825 doi: 10.1146/annurev-control-053018-023825

Riehle, A., MacKay, W., & Requin, J. (1994). Are extent and force independent movement parameters? preparation- and movement-related neuronal activity in the monkey cortex. *Experimental Brain Research*, *99*(1). Retrieved from http://dx.doi.org/10.1007/BF00241412 doi: 10.1007/bf00241412

Rihel, J., Prober, D. A., Arvanites, A., Lam, K., Zimmerman, S., Jang, S., ... Schier, A. F. (2010). Zebrafish behavioral profiling links drugs to biological targets and rest/wake regulation. *Science*, *327*(5963), 348–351. Retrieved from http://dx.doi.org/10.1126/science.1183090 doi: 10.1126/science.1183090

Rouse, A. G., Schieber, M. H., & Sarma, S. V. (2022). Cyclic, condition-independent activity in primary motor cortex predicts corrective movement behavior. *eneuro*, *9*(2), ENEURO.0354–21.2022. Retrieved from http://dx.doi.org/10.1523/ENEURO.0354-21.2022 doi: 10.1523/eneuro.0354-21.2022

Russo, A. A., Bittner, S. R., Perkins, S. M., Seely, J. S., London, B. M., Lara, A. H., ... Churchland, M. M. (2018). Motor cortex embeds muscle-like commands in an untangled population response. *Neuron*, *97*(4), 953–966.e8. Retrieved from http://dx.doi.org/10.1016/j.neuron.2018.01.004 doi: 10.1016/j.neuron.2018.01.004

Russo, A. A., Khajeh, R., Bittner, S. R., Perkins, S. M., Cunningham, J. P., Abbott, L., & Churchland, M. M. (2020). Neural trajectories in the supplementary motor area and motor cortex exhibit distinct geometries, compatible with different classes of computation. *Neuron*, *107*(4), 745–758.e6. Retrieved from http://dx.doi.org/10.1016/j.neuron.2020.05.020 doi: 10.1016/j.neuron.2020.05.020

Ryczko, D. (2022). The mesencephalic locomotor region: Multiple cell types, multiple behavioral roles, and multiple implications for disease. *The Neuroscientist*, *30*(3), 347–366. Retrieved from http://dx.doi.org/10.1177/10738584221139136 doi: 10.1177/10738584221139136

Ryczko, D., Auclair, F., Cabelguen, J., & Dubuc, R. (2015). The mesencephalic locomotor region sends a bilateral glutamatergic drive to hindbrain reticulospinal neurons in a tetrapod. *Journal of Comparative Neurology*, *524*(7), 1361–1383. Retrieved from http://dx.doi.org/10.1002/cne.23911 doi: 10.1002/cne.23911

Sarkka, S., & Garcia-Fernandez, A. F. (2021). Temporal parallelization of bayesian smoothers. *IEEE Transactions on Automatic Control*, *66*(1), 299–306. Retrieved from http://dx.doi.org/10.1109/TAC.2020.2976316 doi: 10.1109/tac.2020.2976316

Sarkka, S., & Garcia-Fernandez, A. F. (2023). Temporal parallelization of dynamic programming and linear quadratic control. *IEEE Transactions on Automatic Control*, *68*(2), 851–866. Retrieved from http://dx.doi.org/10.1109/TAC.2022.3147017 doi: 10.1109/tac.2022.3147017

Sauerbrei, B. A., Guo, J.-Z., Cohen, J. D., Mischiati, M., Guo, W., Kabra, M., ... Hantman, A. W. (2019). Cortical pattern generation during dexterous movement is input-driven. *Nature*, *577*(7790), 386–391. Retrieved from http://dx.doi.org/10.1038/s41586-019-1869-9 doi: 10.1038/s41586-019-1869-9

Saxena, S., Russo, A. A., Cunningham, J., & Churchland, M. M. (2022). Motor cortex activity across movement speeds is predicted by network-level strategies for generating muscle activity. *eLife*, *11*. Retrieved from http://dx.doi.org/10.7554/eLife.67620 doi: 10.7554/elife.67620

Scherpen, J. (2011). Balanced realizations, model order reduction, and the hankel operator. In W. Levine (Ed.), *The control handbook. control system advanced methods* (pp. 4–1–4–24). CRC Press. (Relation: http://www.rug.nl/tbk/onderzoek/onderzoeksinstituten/itm/index Rights: University of Groningen, Research Institute of Technology and Management)

Schimel, M., Kao, T.-C., Jensen, K. T., & Hennequin, G. (2022). iLQR-VAE : control-based learning of input-driven dynamics with applications to neural data. In *International conference on learning representations.* Retrieved from https://openreview.net/forum?id=wRODLDHaAiW

Schmid, P. J. (2022). Dynamic mode decomposition and its variants. *Annual Review of Fluid Mechanics*, *54*(1), 225–254. Retrieved from http://dx.doi.org/10.1146/annurev-fluid-030121-015835 doi: 10.1146/annurev-fluid-030121-015835

Scott, S. H. (2004). Optimal feedback control and the neural basis of volitional motor control. *Nature Reviews Neuroscience*, *5*(7), 532–545. Retrieved from http://dx.doi.org/10.1038/nrn1427 doi: 10.1038/nrn1427

Selverston, A. I., & Moulins, M. (1985). Oscillatory neural networks. *Annual Review of Physiology*, *47*(1), 29–48. Retrieved from http://dx.doi.org/10.1146/annurev.ph.47.030185.000333 doi: 10.1146/annurev.ph.47.030185.000333

Sengupta, S., Lefohn, A. E., & Owens, J. D. (2006). A work-efficient step-efficient prefix sum algorithm. In *In proceedings of the workshop on edge computing using new commodity architectures* (p. 27-27).

Severi, K., Portugues, R., Marques, J., O'Malley, D., Orger, M., & Engert, F. (2014). Neural control and modulation of swimming speed in the larval zebrafish. *Neuron*, *83*(3), 692–707. Retrieved from http://dx.doi.org/10.1016/j.neuron.2014.06.032 doi: 10.1016/j.neuron.2014.06.032

Severi, K. E., Böhm, U. L., & Wyart, C. (2018). Investigation of hindbrain activity during active locomotion reveals inhibitory neurons involved in sensorimotor processing. *Scientific Reports*, *8*(1). Retrieved from http://dx.doi.org/10.1038/s41598-018-31968-4 doi: 10.1038/s41598-018-31968-4

Shenoy, K. V., & Kao, J. C. (2021). Measurement, manipulation and modeling of brain-wide neural population dynamics. *Nature Communications*, *12*(1). Retrieved from http://dx.doi.org/10.1038/s41467-020-20371-1 doi: 10.1038/s41467-020-20371-1

Shenoy, K. V., Sahani, M., & Churchland, M. M. (2013). Cortical control of arm movements: A dynamical systems perspective. *Annual Review of Neuroscience*, *36*(1), 337–359. Retrieved from http://dx.doi.org/10.1146/annurev-neuro-062111-150509 doi: 10.1146/annurev-neuro-062111-150509

Smith, J. T., Linderman, S., & Sussillo, D. (2021). Reverse engineering recurrent neural networks with jacobian switching linear dynamical systems. In A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems.* Retrieved from https://openreview.net/forum?id=od-00q5T2vB

Smith, J. T., Warrington, A., & Linderman, S. (2023). Simplified state space layers for sequence modeling. In *The eleventh international conference on learning representations.* Retrieved from https://openreview.net/forum?id=Ai8Hw3AXqks

Smyrnis, N., Taira, M., Ashe, J., & Georgopoulos, A. (1992). Motor cortical activity in a memorized delay task. *Experimental Brain Research*, *92*(1). Retrieved from http://dx.doi.org/10.1007/BF00230390 doi: 10.1007/bf00230390

Soldatenko, S., & Yusupov, R. (2021). An optimal control perspective on weather and climate modification. *Mathematics*, *9*(4), 305. Retrieved from http://dx.doi.org/10.3390/math9040305 doi: 10.3390/math9040305

Soner, H. M. (2004). *Stochastic optimal control in finance.* Scuola normale superiore. Retrieved from https://link.springer.com/content/pdf/10.1007/0-387-23570-1_1.pdf

Squire, L., Berg, D., Bloom, F., du Lac, S., Ghosh, A., Spitzer, N., & Squire, L. (2008). *Fundamental neuroscience.* Academic Press. Retrieved from https://books.google.pt/books?id=GOxrtYzmixcC

Sridhar, G., Vergassola, M., Marques, J. C., Orger, M. B., Costa, A. C., & Wyart, C. (2024). Uncovering multiscale structure in the variability of larval zebrafish navigation. *Proceedings of the National Academy of Sciences*, *121*(47). Retrieved from http://dx.doi.org/10.1073/pnas.2410254121 doi: 10.1073/pnas.2410254121

Stein, P. S. G. (2004). Neuronal control of turtle hindlimb motor rhythms. *Journal of Comparative Physiology A*, *191*(3), 213–229. Retrieved

from http://dx.doi.org/10.1007/s00359-004-0568-6 doi: 10.1007/s00359-004-0568-6

Steinmetz, N. A., Koch, C., Harris, K. D., & Carandini, M. (2018). Challenges and opportunities for large-scale electrophysiology with neuropixels probes. *Current Opinion in Neurobiology*, *50*, 92–100. Retrieved from http://dx.doi.org/10.1016/j.conb.2018.01.009 doi: 10.1016/j.conb.2018.01.009

Stephens, G. J., Johnson-Kerner, B., Bialek, W., & Ryu, W. S. (2008). Dimensionality and dynamics in the behavior of c. elegans. *PLoS Computational Biology*, *4*(4), e1000028. Retrieved from http://dx.doi.org/10.1371/journal.pcbi.1000028 doi: 10.1371/journal.pcbi.1000028

Stih, V., Petrucco, L., Kist, A. M., & Portugues, R. (2019). Stytra: An open-source, integrated system for stimulation, tracking and closed-loop behavioral experiments. *PLOS Computational Biology*, *15*(4). Retrieved from https://doi.org/10.1371/journal.pcbi.1006699 doi: 10.1371/journal.pcbi.1006699

Stringer, C., Zhong, L., Syeda, A., Du, F., Kesa, M., & Pachitariu, M. (2024). Rastermap: a discovery method for neural population recordings. *Nature Neuroscience*, *28*(1), 201–212. Retrieved from http://dx.doi.org/10.1038/s41593-024-01783-4 doi: 10.1038/s41593-024-01783-4

Sussillo, D., & Barak, O. (2013). Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, *25*(3), 626–649. Retrieved from http://dx.doi.org/10.1162/NECO_a_00409 doi: 10.1162/neco_a_00409

Sussillo, D., Churchland, M. M., Kaufman, M. T., & Shenoy, K. V. (2015). A neural network that finds a naturalistic solution for the production of muscle activity. *Nature Neuroscience*, *18*(7), 1025–1033. Retrieved from http://dx.doi.org/10.1038/nn.4042 doi: 10.1038/nn.4042

Takens, F. (1981). Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, warwick 1980* (p. 366–381). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/BFb0091924 doi: 10.1007/bfb0091924

Takens, F. (2006). Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, warwick 1980: proceedings of a symposium held at the university of warwick 1979/80* (pp. 366–381).

Tan, E., Algar, S., Corrêa, D., Small, M., Stemler, T., & Walker, D. (2023). Selecting embedding delays: An overview of embedding techniques and a new method using persistent homology. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *33*(3).

Tankus, A., Yeshurun, Y., Flash, T., & Fried, I. (2009). Encoding of speed and direction of movement in the human supplementary motor area: Laboratory investigation. *Journal of Neurosurgery*, *110*(6), 1304–1316. Retrieved from http://dx.doi.org/10.3171/2008.10.JNS08466 doi: 10.3171/2008.10.jns08466

Tassa, Y., Erez, T., & Todorov, E. (2012). Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 ieee/rsj international conference on intelligent robots and systems*. IEEE. Retrieved from http://dx.doi.org/10.1109/IROS.2012.6386025 doi: 10.1109/iros.2012.6386025

Tedrake, R. (2023). *Underactuated robotics*. Course Notes for MIT 6.832. Retrieved from https://underactuated.csail.mit.edu

Thiele, T., Donovan, J., & Baier, H. (2014). Descending control of swim posture by a midbrain nucleus in zebrafish. *Neuron*, *83*(3), 679–691. Retrieved from http://dx.doi.org/10.1016/j.neuron.2014.04.018 doi: 10.1016/j.neuron.2014.04.018

Thorsen, D. H., Cassidy, J. J., & Hale, M. E. (2004). Swimming of larval zebrafish: fin–axis coordination and implications for function and neural control. *Journal of Experimental Biology*, *207*(24), 4175–4183. Retrieved from http://dx.doi.org/10.1242/jeb.01285 doi: 10.1242/jeb.01285

Todorov, E. (2006). Optimal control theory. In *Bayesian brain* (p. 269–298). The MIT Press. Retrieved from http://dx.doi.org/10.7551/mitpress/1535.003.0018 doi: 10.7551/mitpress/1535.003.0018

Todorov, E. (2008). General duality between optimal control and estimation. In *2008 47th ieee conference on decision and control*. IEEE. Retrieved from http://dx.doi.org/10.1109/CDC.2008.4739438 doi: 10.1109/cdc.2008.4739438

Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 ieee/rsj international conference on intelligent robots and systems*. IEEE. Retrieved from http://dx.doi.org/10.1109/IROS.2012.6386109 doi: 10.1109/iros.2012.6386109

Todorov, E., & Jordan, M. I. (2002). Optimal feedback control as a theory of motor coordination. *Nature Reviews Neuroscience*, *5*(11), 1226–1235. Retrieved from https://doi.org/10.1038/nn963 doi: 10.1038/nn963

Turaga, S., Buesing, L., Packer, A. M., Dalgleish, H., Pettit, N., Hausser, M., & Macke, J. H. (2013). Inferring neural population dynamics from multiple partial recordings of the same neural circuit. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 26). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2013/file/01386bd6d8e091c2ab4c7c7de644d37b-Paper.pdf

Varma, B., Swamy, N., & Mukherjee, S. (2020). Trajectory tracking of autonomous vehicles using different control techniques(pid vs lqr vs mpc). In *2020 international conference on smart technologies in computing, electrical and electronics (icstcee)*. IEEE. Retrieved from http://dx.doi.org/10.1109/ICSTCEE49637.2020.9276986 doi: 10.1109/icstcee49637.2020.9276986

Vishwanathan, A., Sood, A., Wu, J., Ramirez, A. D., Yang, R., Kemnitz, N., ... Williams, S. (2024). Predicting modular functions and neural coding of behavior from a synaptic wiring diagram. *Nature Neuroscience*, *27*(12), 2443–2454. Retrieved from http://dx.doi.org/10.1038/s41593-024-01784-3 doi: 10.1038/s41593-024-01784-3

Vogels, T. P., Rajan, K., & Abbott, L. (2005). Neural network dynamics. *Annual Review of Neuroscience*, *28*(1), 357–376. Retrieved from http://dx.doi.org/10.1146/annurev.neuro.28.061604.135637 doi: 10.1146/annurev.neuro.28.061604.135637

Vyas, S., Golub, M. D., Sussillo, D., & Shenoy, K. V. (2020). Computation through neural population dynamics. *Annual Review of Neuroscience*, *43*(1), 249–275. Retrieved from http://dx.doi.org/10.1146/annurev-neuro-092619-094115 doi: 10.1146/annurev

-neuro-092619-094115

Wang, Z., Zhang, J., Symvoulidis, P., Guo, W., Zhang, L., Wilson, M. A., & Boyden, E. S. (2023). Imaging the voltage of neurons distributed across entire brains of larval zebrafish. Cold Spring Harbor Laboratory. Retrieved from http://dx.doi.org/10.1101/2023.12.15.571964 doi: 10.1101/2023.12.15.571964

Wiltschko, A. B., Johnson, M. J., Iurilli, G., Peterson, R. E., Katon, J. M., Pashkovski, S. L., ... Datta, S. R. (2015). Mapping sub-second structure in mouse behavior. *Neuron*, *88*(6), 1121–1135.

Wiltschko, A. B., Tsukahara, T., Zeine, A., Anyoha, R., Gillis, W. F., Markowitz, J. E., ... Datta, S. R. (2020). Revealing the structure of pharmacobehavioral space through motion sequencing. *Nature Neuroscience*, *23*(11), 1433–1443. Retrieved from http://dx.doi.org/10.1038/s41593-020-00706-3 doi: 10.1038/s41593-020-00706-3

Wimalasena, L. N., Braun, J. F., Keshtkaran, M. R., Hofmann, D., Gallego, J. Á., Alessandro, C., ... Pandarinath, C. (2022). Estimating muscle activation from emg using deep learning-based dynamical systems models. *Journal of neural engineering*, *19*(3), 036013.

Wolf, S., Supatto, W., Debrégeas, G., Mahou, P., Kruglik, S. G., Sintes, J.-M., ... Candelier, R. (2015). Whole-brain functional imaging with two-photon light-sheet microscopy. *Nature Methods*, *12*(5), 379–380. Retrieved from http://dx.doi.org/10.1038/nmeth.3371 doi: 10.1038/nmeth.3371

Wolpert, D. M., Ghahramani, Z., & Jordan, M. I. (1995). An internal model for sensorimotor integration. *Science*, *269*(5232), 1880–1882. Retrieved from http://dx.doi.org/10.1126/science.7569931 doi: 10.1126/science.7569931

Wolpert, D. M., Miall, R., & Kawato, M. (1998). Internal models in the cerebellum. *Trends in Cognitive Sciences*, *2*(9), 338–347. Retrieved from http://dx.doi.org/10.1016/s1364-6613(98)01221-2 doi: 10.1016/s1364-6613(98)01221-2

Wright, S. J. (1996). *Applying new optimization algorithms to more predictive control* (Tech. Rep.). Argonne National Lab.(ANL), Argonne, IL (United States).

Xie, Z., & Gang, T. (2021). *Discrete lqr and ilqr methods based on high order runge-kutta methods.* arXiv. Retrieved from https://arxiv

.org/abs/2112.15261 doi: 10.48550/ARXIV.2112.15261

Xu, L., Guan, N. N., Huang, C.-X., Hua, Y., & Song, J. (2021). A neuronal circuit that generates the temporal motor sequence for the defensive response in zebrafish larvae. *Current Biology*, *31*(15), 3343–3357. Retrieved from http://dx.doi.org/10.1016/j.cub.2021.06.054 doi: 10.1016/j.cub.2021.06.054

Yang, C., Mammen, L., Kim, B., Li, M., Robson, D. N., & Li, J. M. (2024). A population code for spatial representation in the zebrafish telencephalon. *Nature*, *634*(8033), 397–406. Retrieved from http://dx.doi.org/10.1038/s41586-024-07867-2 doi: 10.1038/s41586-024-07867-2

Zhou, G.-B., Wu, J., Zhang, C.-L., & Zhou, Z.-H. (2016). *Minimal gated unit for recurrent neural networks.* arXiv. Retrieved from https://arxiv.org/abs/1603.09420 doi: 10.48550/ARXIV.1603.09420

Appendix A

Further derivations iLQR

A.1 Detailed derivation of the LQR problem solved by dynamic program

There exists a particular class of control problems that minimizes a quadratic cost that is constrained by linear dynamics, known as a linear quadratic program. This is formally written as

$$\underset{\{u_0,\ldots,u_{T-1}\}}{\text{minimize}} \quad \mathcal{J}(x,u) = l_f(x_T) + \sum_{t=0}^{T-1} l_t(x_t,u_t) \tag{A.1}$$

$$\text{subject to} \quad x_{t+1} = A_t x_t + B_t u_t, \quad t = 0,\ldots,T-1 \tag{A.2}$$

$$x_0 = x_{\text{init}} \tag{A.3}$$

l_t denotes the momentary cost function at time t, the quadratic terms are defined as,

$$l_t(x_t,u_t) = \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} Q_t & S_t \\ S_t^T & R_t \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} q_t \\ r_t \end{bmatrix} \tag{A.4}$$

and

$$l_T(x_T) = x_T^T Q_T x_T + x_T^T q_T \tag{A.5}$$

The optimization is constrained by the forward linear dynamics, where the successive state $x_{t+1} \in \mathbb{R}^n$ is determined by the state space matrix $A \in \mathbb{R}^{n \times n}$ and the input matrix $B \in \mathbb{R}^{n \times m}$, which act on the current state $x_t \in \mathbb{R}^n$ and control $u_t \in \mathbb{R}^m$, respectively. The state space matrix A and the input matrix B can vary over time.To ensure uniqueness and convergence, the cost matrices are constrained with the state cost $Q_t = Q_t^T \succcurlyeq 0$ and input cost $R_t = R_t^T \succ 0$.

The LQR can be solved using DP using the Riccati equation (Kirk, 2004), provided that the Value function at each time point has the same structure. The Value function is defined as the minimum cost-to-go from time t to T given the state x_t and the control u_t at time t. The Value function at time t is defined by,

$$\mathcal{V}(x_t, t) = \min_{u_t} \left(\begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} Q_t & S_t \\ S_t^T & R_t \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} q_t \\ r_t \end{bmatrix} + \mathcal{V}(x_{t+1}, t+1) \right) \tag{A.6}$$

which is bounded by the terminal cost state in Equation (A.5),

$$\mathcal{V}(x_T, T) = \frac{1}{2} x_T^T V_T v_T + x_T^T v_T \tag{A.7}$$

where $V_T = Q_T$ and $v_T = q_T$. Here, we shall outline the proof that the value function has the same structure using proof by induction. That is, given the Value function at time T, has the quadratic structure as in Equation (A.7), and we show that the Value function at time $T-1$ has the same structure too, then the Value function holds for all time points. Therefore, substituting Equation (A.7) into Equation (A.6) and using the linear dynamics constraint to evaluate all variables at t, yields,

$$\mathcal{V}(x_{T-1}, T-1) = \min_{u_{T-1}} \Bigg(\begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T \begin{bmatrix} Q_{T-1} & S_{T-1} \\ S_{T-1}^T & R_{T-1} \end{bmatrix} \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix} + \begin{bmatrix} q_{T-1} \\ r_{T-1} \end{bmatrix}^T \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}$$
$$+ \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T \begin{bmatrix} A_{T-1}^T V_T A_{T-1} & A_{T-1}^T V_T B_{T-1} \\ B_{T-1}^T V_T A_{T-1} & B_{T-1}^T V_T B_{T-1} \end{bmatrix} \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}$$
$$+ \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T \begin{bmatrix} A_{T-1}^T v_T \\ B_{T-1}^T v_T \end{bmatrix} \Bigg) \tag{A.8}$$

$$= \min_{u_{T-1}} \Bigg(\begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T \begin{bmatrix} Q_{T-1} + A_{T-1}^T V_T A_{T-1} & S_{T-1} + A_{T-1}^T V_T B_{T-1} \\ S_{T-1}^T + B_{T-1}^T V_T A_{T-1} & R_{T-1} + B_{T-1}^T V_T B_{T-1} \end{bmatrix} \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix} +$$
$$\begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T \begin{bmatrix} q_{T-1} + A_{T-1}^T v_T \\ r_{T-1} + B_{T-1}^T v_T \end{bmatrix} \Bigg) \tag{A.9}$$

$$= \min_{u_t} \Bigg(\begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T \begin{bmatrix} Q_{T-1}^{xx} & Q_{T-1}^{xu} \\ Q_{T-1}^{ux} & Q_{T-1}^{uu} \end{bmatrix} \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix} + \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T \begin{bmatrix} q_{T-1}^x \\ q_{T-1}^u \end{bmatrix} \Bigg) \tag{A.10}$$

where $Q_{T-1}^{xx}, Q_{T-1}^{xu}, Q_{T-1}^{ux}, Q_{T-1}^{uu}, q_{T-1}^x, q_{T-1}^u$ aggregates the terms in the matrix form. Then minimizing the Value function with respect to u_{T-1}, that is, $\frac{\partial \mathcal{V}}{\partial u_{T-1}} = 0$, gives,

$$0 = \frac{\partial \mathcal{V}}{\partial u_{T-1}} = Q_{T-1}^{uu} u_{T-1} + (Q_{T-1}^{ux} x_{T-1} + q_{T-1}^u) \tag{A.11}$$

which can be re-arranged to give the input state feedback law,

$$u_{T-1} = K_{T-1} x_{T-1} + k_{T-1} \tag{A.12}$$

defined by,

$$\begin{aligned} K_{T-1} &= - Q_{T-1}^{uu^{-1}} Q_{T-1}^{ux} \\ &= - (R_{T-1} + B_{T-1}^T V_T B_{T-1})^{-1} (S_{T-1}^T + B_{T-1}^T V_T A_{T-1}) \end{aligned} \tag{A.13}$$

$$\begin{aligned} k_{T-1} &= - Q_{T-1}^{uu^{-1}} q_{T-1}^u \\ &= - (R_{T-1} + B_{T-1}^T V_T B_{T-1})^{-1} (r_{T-1}^T + B_{T-1}^T v_T) \end{aligned} \tag{A.14}$$

With the state feedback law, for completeness, the Value function can be expressed only in terms of the state x_{T-1} as by substituting Equation (A.12) into Equation (A.8) to give,

$$\begin{aligned}\mathcal{V}(x_{T-1}, T-1) = &\begin{bmatrix} x_{T-1} \\ K_{T-1}x_{T-1} + k_{T-1} \end{bmatrix}^T \begin{bmatrix} Q^{xx}_{T-1} & Q^{xu}_{T-1} \\ Q^{ux}_{T-1} & Q^{uu}_{T-1} \end{bmatrix} \begin{bmatrix} x_{T-1} \\ K_{T-1}x_{T-1} + k_{T-1} \end{bmatrix} \\ &+ \begin{bmatrix} x_{T-1} \\ K_{T-1}x_{T-1} + k_{T-1} \end{bmatrix}^T \begin{bmatrix} q^{x}_{T-1} \\ q^{u}_{T-1} \end{bmatrix} \\ =&\frac{1}{2}x^T_{T-1}V_{T-1}x_{T-1} + x^T_{T-1}v_{T-1} + \text{const.}\end{aligned} \tag{A.15}$$

where,

$$V_{T-1} = Q^{xx}_{T-1} + K^T_{T-1}Q^{ux}_{T-1} + Q^{xu}_{T-1}K_{T-1} + K^T_{T-1}Q^{uu}_{T-1}K^T_{T-1} \tag{A.16}$$

$$v_{T-1} = q^{x}_{T-1} + K^T_{T-1}Q^{uu}_{T-1}k_{T-1} + k^T_{T-1}q^{u}_{T-1} \tag{A.17}$$

which returns the Value function at time $T-1$ with the same structure as in Equation (A.7). Therefore, the Value function holds for all time points with quadratic structure,

$$\mathcal{V}(x, t) = \frac{1}{2}x^T_t V_t x_t + x^T_t v_t. \tag{A.18}$$

Using the terminal Value function boundary condition, we can obtain all the gain terms, $\{K\}$ and $\{k\}$, for the optimal control law, following the same procedure as Equations (A.12), (A.13), (A.15) and (A.16) at each time point t starting from T; illustrated in Algorithm 3. With the optimal gains, the optimal trajectory can then be computed using the optimal feedback law in Equation (A.12).

Algorithm 3 Riccati Recursion for Solving LQR with KKT Conditions

procedure BACKWARD PASS
  $V_N \leftarrow Q_f$
  $v_N \leftarrow q_f$
  for $n = N-1 \rightarrow 0$ **do**
    $\tilde{R}_n \leftarrow R_n + B_n^T V_{n+1} B_n$
    $K_n \leftarrow -\tilde{R}_n^{-1}(S_n + B_n^T V_{n+1} A_n)$
    $V_n \leftarrow Q_n + A_n^T V_{n+1} A_n - K_n^T \tilde{R}_n K_n$
    $k_n \leftarrow -\tilde{R}_n^{-1}(s_n + B_n^T (V_{n+1} a_n + v_{n+1}))$
    $v_n \leftarrow q_n + A_n^T (V_{n+1} a_n + v_{n+1}) - K_n^T \tilde{R}_n k_n$
  end for
  $\lambda_0 \leftarrow V_0 x_0 + v_0$
end procedure

procedure FORWARD PASS
  for $n = N-1 \rightarrow 0$ **do**
    $u_n \leftarrow K_n x_n + k_n$
    $x_{n+1} \leftarrow A_n x_n + B_n u_n + a_n$
    $\lambda_{n+1} \leftarrow V_{n+1} x_{n+1} + v_{n+1}$
  end for
end procedure

A.2 Lagrangian method to solve LQR

The quadratic and linear terms in Equation (A.1) can be collected and redefined in block matrix form as,

$$\begin{aligned} \underset{\boldsymbol{z}}{\text{minimize}} \quad & \phi = \frac{1}{2}\boldsymbol{z}^T \boldsymbol{H} \boldsymbol{z} + \boldsymbol{g}^T \boldsymbol{z} \\ \text{subject to} \quad & \boldsymbol{F}\boldsymbol{z} - \boldsymbol{b} = 0 \end{aligned} \tag{A.19}$$

where $\boldsymbol{H}$ and $\boldsymbol{F}$ are sparse block diagonal matrices, and $\boldsymbol{g}$, $\boldsymbol{b}$, $\boldsymbol{z} = [\boldsymbol{x}\ \boldsymbol{u}]$ are the linear terms concatenated through time, explicitly,

$$\boldsymbol{z} = \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ x_{T-1} \\ u_{T-1} \\ x_T \end{bmatrix}, \quad \boldsymbol{H} = \begin{bmatrix} Q_0 & S_0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ S_0^T & R_0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & Q_1 & S_1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & S_1^T & R_1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & Q_{T-1} & S_{T-1} & 0 \\ 0 & 0 & 0 & 0 & \cdots & S_{T-1}^T & R_{T-1} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & Q_T \end{bmatrix}, \quad \boldsymbol{g} = \begin{bmatrix} q_0 \\ r_0 \\ q_1 \\ r_1 \\ \vdots \\ q_{T-1} \\ r_{T-1} \\ q_T \end{bmatrix}$$

$$\boldsymbol{F} = \begin{bmatrix} A_0 & B_0 & -I & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & A_1 & B_1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & A_{T-1} & B_{T-1} & -I \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & A_T \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 0 \\ a_0 \\ a_1 \\ \vdots \\ a_{T-1} \\ a_T \end{bmatrix}$$

The Lagrangian, $\mathcal{L}$, of Equation (A.19) is given as,

$$\mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} + \boldsymbol{z}^T\boldsymbol{g} + \boldsymbol{\lambda}^T(\boldsymbol{F}\boldsymbol{z} - \boldsymbol{b}) \tag{A.20}$$

Where the Lagrange multiplier, $\boldsymbol{\lambda}$, is a vector of the same dimension as the equality constraints, and characterizes the costate dynamics of the system (Amos & Kolter, 2017).

The quadratic optimization in Equation (A.19) can be directly minimized, by solving the Lagrangian in Equation (A.20) with the Karush-Kuhn-Tucker (KKT) condition that satisfies Sufficient Optimal Conditions (SOC). That is, the partial derivatives of Equation (A.20) satisfies the KKT conditions of stationarity and primal feasibility, as,

$$\nabla_{\boldsymbol{z}}\mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) = \boldsymbol{H}\boldsymbol{z}^* + \boldsymbol{g} + \boldsymbol{F}^T\boldsymbol{\lambda}^* = 0 \tag{A.21}$$

$$\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) = \boldsymbol{F}\boldsymbol{z}^* - \boldsymbol{b} = 0 \tag{A.22}$$

yielding,

$$\begin{bmatrix} \boldsymbol{H} & -\boldsymbol{F}^T \\ -\boldsymbol{F} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{\lambda} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{b} \end{bmatrix} \tag{A.23}$$

The KKT condition states that if solution $\boldsymbol{z}^*$ exists, then there also exists a vector $\boldsymbol{\lambda}^*$ that satisfies the KKT conditions in Equation (A.23) (Amos & Kolter, 2017; Schimel et al., 2022). Equation (A.23) can be solved by matrix inversion,

$$\begin{bmatrix} \boldsymbol{z}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = - \begin{bmatrix} \boldsymbol{W} & \boldsymbol{F}^T \\ \boldsymbol{F} & \boldsymbol{0} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{b} \end{bmatrix} \tag{A.24}$$

to obtain the optimal trajectory and inputs given the costs and dynamics.

The time duality of the system, expressed via state (forward) and costate (reverse) equations, emerges when Equation (A.20) rewritten in terms of time-dependent components from Equations (A.4) and (A.5). The partial derivatives of the Lagrangian subjected to conditions for optimality yields,

$$\nabla_{x_t} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) = Q_t x_t + q_t + S_t u_t + A_t^T \lambda_{t+1} - \lambda_t = 0 \tag{A.25}$$
$$\nabla_{x_T} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) = Q_f x_T + q_T - \lambda_T = 0 \tag{A.26}$$
$$\nabla_{\lambda_0} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) = x_0 - x_0 = 0 \tag{A.27}$$
$$\nabla_{\lambda_{t+1}} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) = A_t x_t + B_t u_t + a_t - x_{t+1} = 0 \tag{A.28}$$
$$\nabla_{u_t} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) = S_t^T x_t + R_t u_t + r_t + B_t^T \lambda_{t+1} = 0 \tag{A.29}$$

a set of equations that can be solved to obtain the optimal trajectory and inputs (Amos & Kolter, 2017; Schimel et al., 2022). Hence, Equations (A.27) and (A.28) reveals the state equations,

$$x_{t+1} = A_t x_t + B_t u_t + a_t \tag{A.30}$$

subject to initial value boundary condition $x_0 = x_0$. Contrary, Equations (A.25) and (A.26) describe reverse time dynamics, or the costate equations,

$$\lambda_t = Q_t x_t + q_T + S_t u_t + A_t^T \lambda_{t+1} \tag{A.31}$$

with terminal boundary condition, $\lambda_T = Q_f x_T + q_T$. The partial derivatives of the Lagrangian provide necessary (but not sufficient) conditions for optimality.

One caveat with the method that uses the Lagrangian and conditions of optimality is that inverting the matrix in Equation (A.24) scale insufficiently, the complexity of this method is $\mathcal{O}[T^3(m+n)^3]$. In contrast, Equation (A.24) can be solved using DP with complexity $\mathcal{O}[T(m+n)^3]$. Wright (1996) showed the KKT system can be written as a band-diagonal matrix, which can be solved using a linear complexity algorithm, using Riccati recursion to factorize the KKT system.

A.3 LQR tracking

Currently, we have discussed the case when LQR solver is used to find the optimal trajectory that minimizes quadratic state and input cost. The LQR algorithm can also be used to track a target trajectory with linear dynamics,

$$\tilde{x}_{t+1} = A_t \tilde{x}_t + B_t \tilde{u}_t \tag{A.32}$$

where $\tilde{x}_t$ is the target trajectory and $\tilde{u}_t$ is the target input. The dynamics can be transformed to project the perturbed dynamics from the target trajectory,

$$\delta x_{t+1} = A_t \delta x_t + B_t \delta u_t \tag{A.33}$$

that is, the target trajectory and inputs are replaced with deviations from the actualized trajectory, $\delta x_t = x_t - \tilde{x}_t$, and inputs, $\delta u_t = u_t - \tilde{u}_t$ (Anderson & Moore, 2007). The momentary cost function, in Equation (A.4), is also transformed to the cost around deviations,

$$l_t(x_t, u_t) = \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix}^T \begin{bmatrix} Q_t & S_t \\ S_t^T & R_t \end{bmatrix} \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix} + \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix}^T \begin{bmatrix} q_t \\ r_t \end{bmatrix} \tag{A.34}$$

and

$$l_T(x_T) = \delta x_t^T Q_T \delta x_t + \delta x_t^T q_T \tag{A.35}$$

The LQR algorithm finds an optimal policy that minimizes the state deviation based on the state deviation feedback, δu_t, which is derived from the state deviation, δx_t, and the gain in the state deviation feedback, K_t, as,

$$\delta u_t^* = K_t \delta x_t + k_t \tag{A.36}$$

similar to Equation (A.12). But the additional step of updating the target input using the optimal state-deviation feedback, as,

$$u_t^* = \tilde{u}_t + \delta u_t^* \tag{A.37}$$

The LQR problem characterised by deviations, Equations (A.33) to (A.35), is known as a *local* LQR problem. This is because the cost and dynamics are centered about the target trajectory. In other words, we are finding optimal state feedback perturbation around the target trajectory. This paradigm can be extended to non-linear dynamics where small perturbations around the trajectory can be approximated using Taylor expansion. With a tracked trajectory $\tilde{\boldsymbol{x}} = \{\tilde{x}_0, \tilde{x}_1 \dots \tilde{x}_T\}$ generated through the nonlinear dynamics, such that,

$$\tilde{x}_{t+1} = f(\tilde{x}_t, \tilde{u}_t) \tag{A.38}$$

we can approximate the dynamics around the target trajectory using a first order Taylor expansion, with respect to x_t and u_t, yielding,

This case requires a first order Taylor expansion about the target trajectory, with respect to x_t and u_t, yielding,

$$f(x_t, u_t) \approx f(\tilde{x}_t, \tilde{u}_t) + \underbrace{\frac{\partial f}{\partial x_t}\bigg|_{\tilde{x}_t, \tilde{u}_t}}_{\tilde{A}} \delta x_t + \underbrace{\frac{\partial f}{\partial u_t}\bigg|_{\tilde{x}_t, \tilde{u}_t}}_{\tilde{B}} \delta u_t \tag{A.39}$$

and rearranging the terms and substituting Equation (A.38), we arrive at the same tracking problem in Equation (A.33),

$$\delta x_{t+1} = \tilde{A}_t \delta x_t, \tilde{B}_t \delta u_t \tag{A.40}$$

here, the Jacobian is approximated as the state transition matrix $\tilde{A}_t$ and input matrix $\tilde{B}_t$ (Tedrake, 2023).

For completeness, in the case when we have a *global* LQR problem, the dynamics are linearized about the actual state and input values. Therefore, we need to factor the offsets of the target trajectory from the actual state and input values, and so, the linearization of the non-linear dynamics is given by,

$$f(x_t, u_t) \approx f(\tilde{x}_t, \tilde{u}_t) + \frac{\partial f}{\partial x_t}\bigg|_{\tilde{x}_t, \tilde{u}_t} (x_t - \tilde{x}_t) + \frac{\partial f}{\partial u_t}\bigg|_{\tilde{x}_t, \tilde{u}_t} (u_t - \tilde{u}_t) \tag{A.41}$$

$$= \underbrace{\left(f(\tilde{x}_t, \tilde{u}_t) - \frac{\partial f}{\partial x_t}\bigg|_{\tilde{x}_t, \tilde{u}_t} \tilde{x}_t - \frac{\partial f}{\partial u_t}\bigg|_{\tilde{x}_t, \tilde{u}_t} \tilde{u}_t \right)}_{\tilde{a}} + \underbrace{\frac{\partial f}{\partial x_t}\bigg|_{\tilde{x}_t, \tilde{u}_t}}_{\tilde{A}} x_t + \underbrace{\frac{\partial f}{\partial u_t}\bigg|_{\tilde{x}_t, \tilde{u}_t}}_{\tilde{B}} u_t \tag{A.42}$$

$$= \tilde{a} + \tilde{A} x_t + \tilde{B} u_t. \tag{A.43}$$

The offset correction holds for higher order terms in the Taylor expansion. The linear cost terms q_t and r_t are also corrected for the offsets in the quadratic terms of the Taylor expansion in the state and input trajectories, for example, $\tilde{q} = \frac{\partial c}{\partial x_t}\Big|_{\tilde{x}_t, \tilde{u}_t} - \tilde{x}_t^T \frac{\partial^2 c}{\partial x_t^2}\Big|_{\tilde{x}_t, \tilde{u}_t} - \tilde{u}_t^T \frac{\partial^2 c}{\partial u_t \partial x_t}\Big|_{\tilde{x}_t, \tilde{u}_t}$, and $\tilde{r} = \frac{\partial c}{\partial u_t}\Big|_{\tilde{x}_t, \tilde{u}_t} - \tilde{u}_t^T \frac{\partial^2 c}{\partial u_t^2}\Big|_{\tilde{x}_t, \tilde{u}_t} - \tilde{x}_t^T \frac{\partial^2 c}{\partial x_t \partial u_t}\Big|_{\tilde{x}_t, \tilde{u}_t}$.

***Local* and *Global* LQR problems**

The subtle difference between the *local* and *global* LQR problems lies in where the optimal values are evaluated. For example, in iterative methods like iLQR, each iteration solves a LQR problem around a nonlinear input and state trajectory. After convergence, when evaluating the adjoints and approximated LQR parameters, it is important to account for the offsets in the state and input trajectories, and a final *global* LQR problem should be solved. Moreover, when handling gradients of the LQR problem, the actual gradients should be derived from the *global* LQR problem, which accounts for these offsets (Schimel et al., 2022).

A.4 Iterative LQR: non-linear optimal control

An extension from the optimal trajectory tracking with non-linear dynamics, is adapting the framework to handle non-linear cost functions. This time, the optimization problem is given as

$$\begin{aligned} \underset{u,x}{\text{minimize}} \quad & \mathcal{J}(x,u) = \mathcal{C}_f(x_T) + \sum_{t=0}^{T-1} \mathcal{C}_t(x_t,u_t) \\ \text{subject to} \quad & x_{t+1} = f(x_t,u_t), \quad t \in [0,T] \end{aligned} \tag{A.44}$$

The iLQR algorithm locally approximates the cost and the dynamics function around the current state x and u. The non-linear dynamics can be linearly approximated as

$$\begin{aligned} \tilde{x}_{t+1} &= f(\tilde{x}_t,\tilde{u}_t) \\ f(x_t,u_t) &\approx f(\tilde{x}_t,\tilde{u}_t) + \underbrace{\left.\frac{\partial f}{\partial x_t}\right|_{\tilde{x}_t,\tilde{u}_t}}_{\tilde{A}}(x_t-\tilde{x}_t) + \underbrace{\left.\frac{\partial f}{\partial u_t}\right|_{\tilde{x}_t,\tilde{u}_t}}_{\tilde{B}}(u_t-\tilde{u}_t) \end{aligned} \tag{A.45}$$

after rearranging, yields,

$$f(\delta x_t, \delta u_t) \approx \tilde{A}\delta x_t + \tilde{B}\delta u_t \tag{A.46}$$

which approximates perturbations with linear dynamics, the state and input matrices correspond with the Jacobians of the non-linear dynamics (Li & Todorov, 2004). Equally, the cost function is approximated to a quadratic function via a second-order Taylor expansion,

$$\mathcal{J}(\delta x_t, \delta u_t) = \mathcal{J}(x_t^{(i)}, \delta u_t^{(i)}) + \sum_{t=0}^{T-1} \frac{1}{2} \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} \mathcal{C}_t^{xx} & \mathcal{C}_t^{xu} \\ \mathcal{C}_t^{ux} & \mathcal{C}_t^{uu} \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} \mathcal{C}_t^{x} \\ \mathcal{C}_t^{u} \end{bmatrix} \tag{A.47}$$

where $\mathcal{C}^{xx} = \nabla_x^2 \mathcal{C}$, $\mathcal{C}^{uu} = \nabla_u^2 \mathcal{C}$, $\mathcal{C}^{ux} = \nabla_{ux} \mathcal{C}$, $\mathcal{C}^{x} = \nabla_x \mathcal{C}$, $\mathcal{C}^{u} = \nabla_u \mathcal{C}$, and $\mathcal{C}^{xx} = \nabla_x^2 \mathcal{C}$. Now, Equations (A.46) and (A.47) form a local LQR problem. Therefore, from an initialised control sequence and current iteration i, an optimal input perturbation, δu^*, can be used to update the control sequence for the next iteration $i+1$, as

$$u_t^{(i+1)} = u_t^{(i)} + \delta u_t^* \tag{A.48}$$

This can generate another trajectory with a forward rollout, and the process of local approximations can be repeated until the input sequence converges to a local optimum.

Similar to LQR, the non-linear iLQR control problem can also be solved using dynamic programming by recursively solving the Value function backwards. The terminal boundary condition is initialized as

$$\mathcal{V}(\delta x_T, T) = \frac{1}{2} \delta x_T^T V_T \delta x_T + \delta x_T^T v_T \tag{A.49}$$

$$= \frac{1}{2} \delta x_T^T \mathcal{C}_T^{xx} \delta x_T + \delta x_T^T \mathcal{C}_T^{x} \tag{A.50}$$

in general, the Value function is described as,

$$\mathcal{V}(\delta x_t, t) = \frac{1}{2}\delta x_t^T V_{t+1}\delta x_t + \delta x_t^T v_{t+1} \tag{A.51}$$

which is the minimization of δu from the current to the terminal input in the sequence. This is expressed in recursive form as

$$\begin{aligned}\mathcal{V}(\delta x_t, t) = \min_{\delta u_t}\Bigg\{ &\frac{1}{2}\begin{bmatrix}\delta x_t \\ \delta u_t\end{bmatrix}^T \begin{bmatrix}\mathcal{C}_t^{xx} & \mathcal{C}_t^{xu} \\ \mathcal{C}_t^{ux} & \mathcal{C}_t^{uu}\end{bmatrix}\begin{bmatrix}\delta x_t \\ \delta u_t\end{bmatrix} + \begin{bmatrix}x_t \\ u_t\end{bmatrix}^T \begin{bmatrix}\mathcal{C}_t^{x} \\ \mathcal{C}_t^{u}\end{bmatrix} + \\ &\mathcal{V}(f(x_t + \delta x_t, u_t + \delta u_t), t+1)\Bigg\}\end{aligned} \tag{A.52}$$

substituting Equation (A.51) in Equation (A.52), the Value function can be re-arranged in terms of δx_t and δu_t,

$$\begin{aligned}\mathcal{V}(\delta x_t, t) \approx \min_{\delta u_t}\Bigg\{ &\frac{1}{2}\begin{bmatrix}\delta x_t \\ \delta u_t\end{bmatrix}^T \begin{bmatrix}\mathcal{C}_t^{xx} & \mathcal{C}_t^{xu} \\ \mathcal{C}_t^{ux} & \mathcal{C}_t^{uu}\end{bmatrix}\begin{bmatrix}\delta x_t \\ \delta u_t\end{bmatrix} + \begin{bmatrix}\delta x_t \\ \delta u_t\end{bmatrix}^T \begin{bmatrix}\mathcal{C}_t^{x} \\ \mathcal{C}_t^{u}\end{bmatrix} + \\ &\frac{1}{2}\begin{bmatrix}\delta x_t \\ \delta u_t\end{bmatrix}^T \begin{bmatrix}\tilde{A}_t^T V_{t+1}\tilde{A}_t & \tilde{A}_t^T V_{t+1}\tilde{B}_t \\ \tilde{B}_t^T V_{t+1}\tilde{A}_t & \tilde{B}_t^T V_{t+1}\tilde{B}_t\end{bmatrix}\begin{bmatrix}\delta x_t \\ \delta u_t\end{bmatrix} + \begin{bmatrix}\delta x_t \\ \delta u_t\end{bmatrix}^T \begin{bmatrix}\tilde{A}_t^T v_{t+1} \\ \tilde{B}_t^T v_{t+1}\end{bmatrix}\Bigg\}\end{aligned} \tag{A.53}$$

thus collecting the terms yield,

$$\mathcal{V}(\delta x_t, t) \approx \min_{\delta u_t}\left\{\frac{1}{2}\begin{bmatrix}\delta x_t \\ \delta u_t\end{bmatrix}^T \begin{bmatrix}Q_t^{xx} & Q_t^{xu} \\ Q_t^{ux} & Q_t^{uu}\end{bmatrix}\begin{bmatrix}\delta x_t \\ \delta u_t\end{bmatrix} + \begin{bmatrix}\delta x_t \\ \delta u_t\end{bmatrix}^T \begin{bmatrix}q_t^{x} \\ q_t^{u}\end{bmatrix}\right\} \tag{A.54}$$

where,

$$\begin{aligned} &Q_t^{xx} = \mathcal{C}_t^{xx} + \tilde{A}_t^T V_{t+1}\tilde{A}_t \qquad Q_t^{uu} = \mathcal{C}_t^{uu} + \tilde{B}_t^T V_{t+1}\tilde{B}_t \qquad Q_t^{xu} = \mathcal{C}_t^{xu} + \tilde{A}_t^T V_{t+1}\tilde{B}_t \\ &q_t^{x} = \mathcal{C}_t^{x} + \tilde{A}_t^T v_{t+1} \qquad\qquad\qquad q_t^{u} = \mathcal{C}_t^{u} + \tilde{B}_t^T v_{t+1}\end{aligned}$$

The argmin of δu_t, i.e. the optimal input perturbation minimizes the Value function via the feedback law,

$$\delta u_t^* = -(Q_t^{uu})^{-1} \left(Q_t^{ux}\delta x_t + q_t^u\right) \tag{A.55}$$
$$\delta u_t^* = K_t \delta x_t + k_t \tag{A.56}$$

where the gains are denoted as $K = -(Q_t^{uu})^{-1}Q_t^{ux}$ and $k = -(Q_t^{uu})^{-1}q_t^u$. Substituting the optimal input perturbations in Equation (A.54) and re-arranging in the form of the Value function of,

$$\mathcal{V}(\delta x_t, t) = \frac{1}{2}\delta x_t^T V_{t+1} \delta x_t + \delta x_t^T v_{t+1} \tag{A.57}$$

gives,

$$\delta V_t = \frac{1}{2} K_t^T Q_t^{uu} K_t \tag{A.58}$$
$$V_t = Q_t^{xx} + K_t^T Q_t^{uu} K_t + K_t^T Q_t^{ux} + Q_t^{xu} K_t \tag{A.59}$$
$$v_t = q_t^x + K_t^T Q_t^{uu} k_t + k_t^T q_t^u + Q_t^{xu} k_t. \tag{A.60}$$

The updated state trajectory $x^{(i+1)}$ and the control input sequence $u^{(i+1)}$ can be acquired once all the gain terms have been collected. The control gains can update the input sequence with the the optimal input perturbations,

$$\delta x_t = x_t^{(i+1)} - x_t^{(i)} \tag{A.61}$$
$$u_t^* = u_t^{(i)} + \underbrace{K_t(x^{(i+1)} - x^{(i)}) + k_t}_{\delta u_t^*} \tag{A.62}$$
$$x_{t+1} = f(x_t, u_t^*(x_t)). \tag{A.63}$$

The non-linear cost value can be evaluated using the new $i+1$ iteration of the state and input trajectories.

An additional feature to reduce the number of iteration until convergence to a local optimum in the iLQR algorithm is to apply a 1D line search on the linear gain term of Equation (A.56). The linesearch backtracking algorithm finds the optimal step size in the gain update to prevent overshooting in the perturbation update (Tassa et al., 2012). This becomes useful when the control problem is highly non-convex. Equation (A.56) is adapted to have an addition hyper parameter, α, such that,

$$\delta u_t^* = K_t \delta x_t + \alpha k_t \tag{A.64}$$

Backtrack line search performs as a series of forward rollouts using Equation (A.64) with the function $f(x, u^*(x, \alpha))$ and the cost $\mathcal{J}(x, u; \alpha)$ evaluated (Park et al., 2020). Initially set with $\alpha = 1.$, that is the largest step size, then geometrically decrements until the local minimum is found.

A.5 Details of Associative scan

We highlight further results from the work of Blelloch (1990) on associative algorithms and recurrence equations, focussed on Section 1.4.

An associative algorithm is possible providing:

1. the function is a binary operator, e.g., $f : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$,
2. and the operator is associative, e.g., $(a \oplus b) \oplus c = a \oplus (b \oplus c)$.

For example, given an array length N,

$$[a_1, a_2, \ldots, a_{N-1}] \tag{A.65}$$

and a binary operator $\bullet$ sequentially scan through the array, yielding

$$[a_1, (a_1 \bullet a_2), \ldots, (a_1 \bullet a_2 \bullet \cdots \bullet a_{N-1})] \tag{A.66}$$

providing the operator $\bullet$ satisfies the conditions above, this output sequence can be computed in parallel. The implementation requires:

1. defining the initial value,
2. defining the associative elements,
3. defining the binary associative operator.

A.5.1 Recurrence equation

For a sequence of length N, a recurrence is a set of equations of form,

$$x_i = f_i(x_{i-1}, \ldots, x_{i-m}) \quad m \leq i < N \tag{A.67}$$

with an initial set of known values $x_0 \ldots x_{m-1}$.

The scan operation, common in functional programming, is a special case of recursion. The scan operation is defined as

$$x_i = \begin{cases} a_0 & i = 0 \\ x_{i-1} \circ a_i & 0 < i < N \end{cases} \tag{A.68}$$

with the binary operator $\circ$, and the initial value a_0. Typically, the binary operator is a summation $\oplus$, or a multiplication $\otimes$. Blelloch (1989) demonstrated parallelization of Equation (A.68) as the all-prefix summation, which returned a cumulative sum sequence using the $\oplus$ binary associative operator on a fixed length array.

First order recurrence

A more advanced case of recursion is the first order linear recurrence. This has important applications in integration and linear dynamical systems (Smith et al., 2023). The equation set of first-order recurrence is defined as

$$x_i = \begin{cases} b_0 & i = 0 \\ (x_{i-1} \otimes a_i) \oplus b_i & 0 < i < n \end{cases} \tag{A.69}$$

where a_i and b_i are a set of constants. In order to parallelize this equation, we need to rewrite Equation (A.69) in a form like Equation (A.68). The operators in Equation (A.69) have important properties to help reformulate the first order recurrences into an associative algorithm that are:

1. $\oplus$ is associative, $(a \oplus b) \oplus c = a \oplus (b \oplus c)$;
2. $\otimes$ is semi-associative, $(a \otimes b) \otimes c = a \otimes (b \odot c)$;
3. $\otimes$ is distributive across, $\oplus$, $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$.

The constants a_i and b_i inEquation (A.69) are grouped as tuple, c_i, that is, $c_i := [c_{i,a}, c_{i,b}] = [a_i, b_i]$. Blelloch (1990) defined the associative operator, $\bullet$, to act on each component of the tuple and store an intermediate component. The new binary operator $\bullet$ acts on c_i tuple as

$$c_i \bullet c_j = [c_{i,a} \odot c_{j,a}, (c_{i,b} \otimes c_{j,a}) \oplus c_{j,b}] \tag{A.70}$$

where the second component reflects the first order recurrence in Equation (A.69) computation and the first component operator $\odot$ is associative.

The proof of associativity for operator $\bullet$ is as follows,

$$\begin{aligned}
(c_i \bullet c_j) \bullet c_k &= [c_{i,a} \odot c_{j,a}, (c_{i,b} \otimes c_{j,a}) \oplus c_{j,b}] \bullet c_k \\
&= [(c_{i,a} \odot c_{j,a}) \odot c_{k,a}, \\
&\quad (((c_{i,b} \otimes c_{j,a}) \oplus c_{j,b}) \otimes c_{k,a}) \oplus c_{k,b}] \\
&= [c_{i,a} \odot (c_{j,a} \odot c_{k,a}), && \text{Associative } \odot \\
&\quad (((c_{i,b} \otimes c_{j,a}) \oplus c_{j,b}) \otimes c_{k,a}) \oplus c_{k,b}] \\
&= [c_{i,a} \odot (c_{j,a} \odot c_{k,a}), \\
&\quad ((c_{i,b} \otimes c_{j,a}) \otimes c_{k,a}) \oplus (c_{j,b} \otimes c_{k,a}) \oplus c_{k,b}] && \text{Property 3} \\
&= [c_{i,a} \odot (c_{j,a} \odot c_{k,a}), \\
&\quad ((c_{i,b} \otimes c_{j,a}) \odot c_{k,a}) \oplus (c_{j,b} \otimes c_{k,a}) \oplus c_{k,b}] && \text{Property 2} \\
&= c_k \bullet [c_{j,a} \odot c_{k,a}, (c_{j,b} \otimes c_{k,a}) \oplus c_{k,b}] \\
&= c_i \bullet (c_j \bullet c_k)
\end{aligned}$$

If we define an ordered set as $c_i = [y_i, x_i]$, y_i already satifies recurrence condition in Equation (A.68),

$$y_i = \begin{cases} a_0 & i = 0 \\ y_{i-1} \odot a_i & 0 < i < n \end{cases} \tag{A.71}$$

Blelloch (1990) showed that c_i more generally obeys the recurrence condition in Equation (A.68). If s_0 is initialized as

$$c_0 = [y_0, x_0] = [a_0, b_0] = c_0 \tag{A.72}$$

Using the proof of associativity of the first order recurrence equation can show how the binary operator can be parallelized,

$$\begin{aligned} c_i &= [y_i, x_i] && \text{(A.73)} \\ &= [y_{i-1} \odot a_i, (x_{i-1} \otimes a_i) \oplus b_i] && \text{(A.74)} \\ &= [y_{i-1} \odot c_{i,a}, (x_{i-1} \otimes a_i) \oplus c_{i,b}] && \text{(A.75)} \\ &= [y_{i-1}, x_{i-1}] \bullet c_i && \text{(A.76)} \\ &= c_{i-1} \bullet c_i && \text{(A.77)} \end{aligned}$$

thus matching the form of Equation (A.68).

Associative forward integration of linear dynamics

This principle applied to the forward integration of a linear dynamical system, $x_{i+1} = Ax_i + Bu_i$ with $n = 4$ time steps. At each timepoint the state integrates the dynamics and external input, starting from an initial

state x_0,

$$
\begin{aligned}
x_1 &= Ax_0 + Bu_0 \\
x_2 &= Ax_1 + Bu_1 = A^2x_0 + ABu_0 + Bu_1 \\
x_3 &= Ax_2 + Bu_2 = A^3x_0 + A^2Bu_0 + ABu_1 + Bu_2 \\
x_4 &= Ax_3 + Bu_3 = A^4x_0 + A^3Bu_0 + A^2Bu_1 + ABu_2 + Bu_3
\end{aligned}
$$

Substituting $a_i = A$ and $b_i = B_iu_i$ and applying the conditions of Equations (A.70) and (A.71) we arrive at

$$
\begin{aligned}
c_0 = [y_0, x_0] &= [\mathbb{I}x_0, x_0] \\
c_1 = [y_1, x_1] &= c_0 \bullet c_1 = [\mathbb{I}x_0, x_0] \bullet [A, Bu_0] \\
&= [Ax_0, \quad Ax_0 + Bu_0] \\
c_2 = [y_2, x_2] &= c_1 \bullet c_2 = [Ax_0, Ax_0 + Bu_0] \bullet c_2 \\
&= [Ax_0, Ax_0 + Bu_0] \bullet c_2 \\
&= [A^2x_0, \quad (A^2x_0 + ABu_0) + Bu_1] \\
c_3 = [y_3, x_3] &= c_2 \bullet c_3 = [A^2x_0, A^2x_0 + ABu_0 + Bu_1] \bullet c_3 \\
&= [A^2x_0, A^2x_0 + ABu_0 + Bu_1] \bullet [A, Bu_2] \\
&= [A^3x_0, \quad A^3x_0 + A^2Bu_0 + ABu_1 + Bu_2]
\end{aligned}
$$

Where the second element of the set s encodes the integration of the state dynamics with external input u. In summary, the parallel implementation of forward dynamics of integration requires the following steps:

1. Initialization - compute in parallel all a_i and b_i terms;
2. Parallel scan - call forward scan routine passing the binary operator rule and initialized set;
3. Extraction - return the second term from scan routine.

A.5.2 Associative reverse integration of value function

The optimal control sequence is shaped by the gains which are computed by solving the Value functions. Value functions obey the principal of optimality that follows a first order recurrence in reverse time; see Equations (A.6) and (A.18). Appendix A.1 demonstrate how the Value function can be solved and is summarized by their self-consistent structure,

$$\mathcal{V}(x_t, t) = x_t V_t x_t + v_t^T x_t \tag{A.78}$$

and, optimal state feedback law,

$$u_k = -K_k^x x_k + k_k^v v_{k+1} + k_k^c a_k \tag{A.79}$$

where,

$$K_k^x = (B_k^T V_{k+1}^{-1} B_k + R_k)^{-1} B_k^T V_{k+1} A \tag{A.80}$$
$$k_k^v = (B_k^T V_{k+1}^{-1} B_k + R_k)^{-1} B_k^T \tag{A.81}$$
$$k_k^c = (B_k^T V_{k+1}^{-1} B_k + R_k)^{-1} B_k^T V_{k+1}. \tag{A.82}$$

The principle of optimality in Equation (A.18), frames the solution to the value function as a first-order recurrence in reverse time, similar to Equation (A.69). Sarkka and Garcia-Fernandez (2023) proposed solving partial value functions independently and described the combination method to solve for the full value function. Defining the associative elements and combinatorics enabled the prefix summation algorithm to parallelize the backward scan through the Riccati recursion, similar to Equation (A.71). Provided the Riccati recursion is associative, we can generate a set of sub-problems to find the Conditional Value Function (CVF).

Conditional Value function The Conditional Value Function is the cost function of the optimal trajectory from x_i to x_k, and is denoted as $\mathcal{V}_{i \to k}(x_i, x_k)$. It is defined as

$$\begin{aligned}\mathcal{V}_{i\to k}(x_i, x_k) = \underset{u_{k:i-1}}{\text{minimize}} \quad & \sum_{t=k}^{i-1} l_t(x_t, u_t) \\ \text{subject to} \quad & x_{t+1} = A_t x_t + B_t u_t + a_t, \quad t \in [k, i-1]\end{aligned} \tag{A.83}$$

The Conditional Value Function can be solved recursively using the Bellman equation, similar to the value function. Due to the principal of optimality, that any subsequence of an optimal trajectory is optimal, means that any segment of that path must also be optimal for the corresponding sub-problem. Therefore, we can partition the Value function into conditional value functions, minimized over given states.

Therefore, combination of the CVF can be written as

$$\mathcal{V}_{i\to k}(x_i, x_k) = \min_{x_j}\{\mathcal{V}_{i\to j}(x_i, x_j) + \mathcal{V}_{j\to k}(x_j, x_k)\} \tag{A.84}$$

where $i < j < k \leq T$ and so the Value function can be obtained via the CVFs as,

$$\mathcal{V}(x_k, k) = \min_{x_i}\{\mathcal{V}_{k\to i}(x_k, x_i) + \mathcal{V}(x_i, i)\} \tag{A.85}$$

where, $k < i \leq T$. The minimization in Equation (A.85) also returns the minimizing state x_i. According to the principle of optimality, this value represents the state at time step i, situated on the optimal trajectory from x_k to time T. Likewise for Equation (A.84), minimizing over x_j yields the optimal trajectory from x_i to x_k.

Associative operator of conditional Value function and value function The associative elements, s, of the parallel scan are defined as CVF, that is,

$$s_i = \mathcal{V}_{s_i}(x, x') \tag{A.86}$$

For two elements, e.g. $s_i = \mathcal{V}_{s_i}(x, x')$ and $s_j = \mathcal{V}_{s_j}(x', x'')$, the binary associative operator for DP is defined as,

$$s_i \circ s_j \hat{=} \min_{x'} \{\mathcal{V}_{s_i}(x, x') + \mathcal{V}_{s_j}(x', x'')\} \tag{A.87}$$

given the min operator is associative, the associativity holds for $\circ$ in Equation (A.87). Below, outlines a quick proof,

$$\begin{aligned}
(s_i \circ s_j) \circ s_k &= \min_{x'} \{\mathcal{V}_{s_i}(x, x') + \mathcal{V}_{s_j}(x', x'')\} \circ s_k && \text{(A.88)} \\
&= \min_{x''} \{\min_{x'} \{\mathcal{V}_{s_i}(x, x') + \mathcal{V}_{s_j}(x', x'')\} + \mathcal{V}_{s_k}(x'', x''')\} && \text{(A.89)} \\
&= \min_{x'} \{\mathcal{V}_{s_i}(x, x') + \min_{x''} \{\mathcal{V}_{s_j}(x', x'')\} + \mathcal{V}_{s_k}(x'', x''')\} && \text{(A.90)} \\
&= s_i \circ (s_j \circ s_k) && \text{(A.91)}
\end{aligned}$$

The set of elements s_i can be initialized for $k \in [S, \ldots, T]$ in parallel as,

$$s_k = \mathcal{V}_{k \to k+1}(x_k, x_{k+1}) \tag{A.92}$$

with a terminal element $\mathcal{V}_{T \to T+1}(x_T, x_{T+1}) := \mathcal{V}_T(x_T)$. Then, as shown in the prefix summation section Algorithm 1, the forward sweep is combined as,

$$s_S \circ s_{S+1} \circ \cdots \circ s_{k-1} = \mathcal{V}_{S \to k}(x_S, x_k) \tag{A.93}$$

reflecting the recursion of the CVF in Equation (A.84), and the backward sweep as,

$$s_k \circ s_{k+1} \circ \cdots \circ s_T = \mathcal{V}(x_k, k) \tag{A.94}$$

corresponding to the recursion of the value function in Equation (A.85).

Appendix B

Linear systems and matrix decomposition

B.1 Matrix decomposition in linear systems

A common technique to understand the computations of the state matrix is to identify the structure using types of matrix decomposition. There are two main types of matrix decomposition: *spectral* and *Schur* decomposition. Spectral decomposition finds a set of oscillations that independently describe the dynamics of the system. Schur decomposition finds a set of modes but also these mode have feedforward dynamics. While there modes are not independent, they are orthogonal which can be a more efficient way to describe dynamics. Therefore, it avoids the overlap of modes in generating trajectories in the state space.

B.1.1 Spectral decomposition

One of the most common matrix decompositions for square matrices is the spectral decomposition. This decomposition finds a solution to $A \in \mathbb{R}^{n \times n}$ matrix defined as,

$$Av_i = \lambda_i v_i \tag{B.1}$$

where the direction, v_i, is invariant under the transformation of A and the magnitude is scaled by the eigenvalue λ_i. The spectral decomposition finds a complete set of n solutions for A,

$$V\Lambda = \begin{bmatrix} | & & | \\ v_1 & \cdots & v_n \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix} \tag{B.2}$$

thus, each column vector of $V \in \mathbb{R}^{n\times n}$ is an eigenvector of A and an element of $\text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is the corresponding eigenvalue which describes decay rate and the oscillation. This transformation is also known as diagonalization,

$$A = V\Lambda V^{-1} \tag{B.3}$$

which reduces the dimensionality of interpretation from $n \times n$ to n as collective modes along the diagonal. Moreover, spectral decomposition eigenvectors can be sorted by the magnitude of the eigenvalues to identify the most dominant modes of the system. If A is symmetric, that is, $A = A^T$, the eigenvectors are orthogonal, and thus $V^TV = I$, hence it is unitary $V^T = V^{-1}$.

For completeness, Singular Value Decomposition (SVD) is an extension of spectral decomposition to non-square matrices. This decomposes a matrix, $M \in \mathbb{R}^{n\times m}$, into two unitary matrices U and V,

$$M = U\Sigma V^T \tag{B.4}$$

where, $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{m\times m}$ are the left and right singular vectors, respectively, and $\Sigma \in \mathbb{R}^{n\times m}$ is a diagonal matrix of singular values. SVD can translate as a generalization of Equation (B.3) by forcing M to be square and symmetric so that $A = M^TM$ the eigenvalues $\Lambda = \Sigma^2$.

B.1.2 Schur decomposition

In the case where A is not symmetric, $AA^T \neq A^TA$, this informs us that the eigenvectors are not orthogonal and so dynamics are non-normal. Non-normal eigenvectors with different magnitudes can drive transient growths and decays, even in the stable dynamical regime. Sometime, using spectral decomposition can be misleading and inefficient in capturing the dynamics of the system, as there is competition amongst similar modes to capture the state dynamics. In this case, we can use the Schur decomposition to identify feedforward modes in the system.

$$A = UTU^T \tag{B.5}$$

where U is a unitary matrix, i.e., $UU^T = I$, and T is an upper triangular matrix (Hennequin et al., 2012). The Schur decomposition forces the vector basis U to be orthogonal, and the diagonal elements of T are the eigenvalues of A. At the expense of the an orthogonal basis set, the off-diagonal elements of T are non-zero, which can be used to identify the feedforward modes of the system.

B.2 State controllability and observability

The controllability of the LDS is determined by, both, the coupling of the state dynamics A, and, the number of control inputs in the LDS m. The controllability matrix $\mathcal{C}$ measures the behavior in which the input matrix propagates through the state dynamics A, defined as,

$$\mathcal{C}(\mathbf{A}\ \mathbf{B}) = \begin{bmatrix}\mathbf{B} & \mathbf{AB} & \cdots & \mathbf{A}^{n-1}\mathbf{B}\end{bmatrix} \tag{B.6}$$

where $\mathcal{C} \in \mathbb{R}^{n\times(m*n)}$. If the control can reach all directions in the A subspace, the LDS is completely controllable, thus the $\mathcal{C}$ will have column rank n. In such case, any state in the network from an initial state can eventually be reached by an arbitrary input signal $\{u\}$.

In contrast, the observability matrix $\mathcal{O}$ assesses whether all states within the LDS can be realized from its measured observable o. $\mathcal{O}$ measures how

much information can be retrieved from the readout matrix C about the state dynamics A, defined by,

$$\mathcal{O}(\mathbf{C};\mathbf{A}) = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix} \tag{B.7}$$

where $\mathcal{O} \in \mathbb{R}^{(n_o * n) \times n}$. The LDS is observable providing $\mathcal{O}$ has full rank n, otherwise regions in the state space cannot be reached via observations o. Observability is important in order to control the LDS.

B.3 Identification of balanced model transformation

In order to find balanced basis transform, T for the LDS, we employed the framework from (Laub et al., 1987). Due to the LDS state dynamics being constrained to be asympotically stable, both, W_c and W_o are positive definite. Therefore, we can compute the Cholesky factors of the Gramians,

$$W_c = L_c L_c^T, \tag{B.8}$$

$$W_o = L_o L_o^T \tag{B.9}$$

where L_c and L_o are the lower triangles of the respective factorization. Using the product of the two Cholesky factors, $L_c^T L_o$, we computed the singular value composition,

$$L_c^T L_o = U \Lambda V^* \tag{B.10}$$

where $U \in \mathbb{R}^{n \times n}$ and $V^* \in \mathbb{R}^{n \times n}$ are right and left basis, respectively, and $\Lambda \in \mathbb{R}^{n \times n}$ is a diagonal matrix where $\Lambda = \Sigma^2$ corresponding to the Hankel Singular Values (HSV)s. From this decomposition, the balancing transformation can be obtained, yielding,

$$T = L_c V \Lambda^{\frac{1}{2}} \tag{B.11}$$

$$T^{-1} = \Lambda^{-\frac{1}{2}} U^* L_o^T. \tag{B.12}$$

This basis is then projected on the LDS, yielding a system to that of Equation (4.8).

Appendix C

Background on variational auto-encoder

C.1 Variational auto-encoder

In these sections we shall hightlight on the background and basic theory of the Variational Autoencoder (VAE) model (Kingma & Welling, 2013). VAEs are a family of models that are used to approximate the true distribution of observed data $p(o)$ via a latent variable, x. Unlike traditional auto-encoder models, VAE encode a set of distributions as the latent variables, such that, smooth, continuous sampling from the latent space generate observations belonging to observed data, classifying VAEs as a type of generative model. The approximated distribution of the observed data is known as the marginal likelihood or evidence, defined as

$$p_\theta(o) = \int p_\theta(o|z)p_\theta(x)dx \tag{C.1}$$

here, $p_\theta(x)$ represents the latent variable priors, and $p_\theta(o|x)$ is the likelihood. However, marginalizing the joint distribution $p(x, o)$ in eq. (C.1) quickly becomes intractable due to the need to integrate over all latent variables x (Dayan et al., 1995). The VAE can be split into two modules.

First, the generative model, which generates samples of the evidence using the prior belief of the latent variables $p_\theta(x)$ and the likelihood $p_\theta(o|x)$. The other, is the recognition model that infers posterior $p_\theta(x|o)$, that is, the latent state given the observations. The posterior is defined as,

$$p_\theta(x|o) = \frac{p_\theta(o|x)p_\theta(x)}{p_\theta(x)} \tag{C.2}$$

where the normalization term is obtained by $p_\theta(o) = \int p_\theta(o, x)dx$, but become intractable as the dimension of x scales.

VAEs learn the distribution of observed data $p(o)$ through a distribution of latent variables. Samples can then be generated from the learned latent distributions, and the model can be used to evaluate the likelihood of observed samples. Rather than directly finding the posterior $p_\theta(x|o)$, as in eq. (C.2), a variational method approximates the posterior using a family of exponential distributions $q_\phi(x|o)$, parametrized by ϕ. The posterior approximation results in maximizing a lower bound of the marginal likelihood, as a proxy for maximizing the marginal likelihood, which is known as the Evidence Lower Bound (ELBO), Ł. This is maximized to fit the evidence and match the approximate posterior with the ground truth.

C.2 Evidence lower bound

Here, demonstrates that maximizing the ELBO implicitly optimizes the posterior matching term of our variational distributions. Note, Jensen's inequality states, $\psi\left(\mathbb{E}[X]\right) \leq \mathbb{E}[\psi\left(X\right)]$. As shown in the derivation,

$$\log p_\theta(o) = \log p_\theta(o) \underbrace{\int q_\phi(x|o)dx}_{=1} \tag{C.3}$$

$$= \log \int p_\theta(o) q_\phi(x|o)dx \tag{C.4}$$

$$= \mathbb{E}_{q_\phi(x|o)} \log[p_\theta(o)] \tag{C.5}$$

$$= \mathbb{E}_{q_\phi(x|o)} \log \left[\frac{p_\theta(o,x)}{p_\theta(x|o)} \underbrace{\frac{q_\phi(x|o)}{q_\phi(x|o)}}_{=1} \right] \tag{C.6}$$

$$= \mathbb{E}_{q_\phi(x|o)} \log \left[\frac{p_\theta(o,x)}{q_\phi(x|o)} \right] + \mathbb{E}_{q_\phi(x|o)} \log \left[\frac{q_\phi(x|o)}{p_\theta(x|o)} \right] \tag{C.7}$$

$$= \mathbb{E}_{q_\phi(x|o)} \log \left[\frac{p_\theta(o,x)}{q_\phi(x|o)} \right] + \mathcal{D}_{kl} \left(q_\phi(x|o) || p(x|o) \right) \tag{C.8}$$

$$\log p_\theta(o) \geq \mathbb{E}_{q_\phi(x|o)} \log \left[\frac{p_\theta(o,x)}{q_\phi(x|o)} \right] = Ł(\phi, x|o) \tag{C.9}$$

the first term in eq. (C.8) is the approximated log marginal likelihood sampled from the approximated posterior. The second term is the mode-matching Kullback-Leibler (KL) divergence of the true and approximated posterior. This is a non-negative, asymmetric score of the similarity of two probability distributions.

The ELBO can be interpreted as maximizing likelihood $p_\theta(o|x)$ while matching prior beliefs of the latent distributions. It can be split into two components,

$$Ł(\phi, x|o) = \mathbb{E}_{q_\phi(x|o)} \left[\log p_\theta(o|x) + \log p_\theta(x) - \log q_\phi(x|o) \right] \tag{C.10}$$

$$= \underbrace{\mathbb{E}_{q_\phi(x|o)} \left[\log p_\theta(o|x) \right]}_{\text{Reconstruction}} - \underbrace{\mathcal{D}_{KL}(q_\phi(x|o)||p_\theta(x))}_{\text{Prior matching regularization}} \tag{C.11}$$

the first term characterizes the expected log-likelihood which ensures the learned distributions can reconstruct the original data, via the latent bottleneck. The second term measures the similarity of the variational distributions to prior belief are preserved over the latent variables (Kingma & Welling, 2013). This term encourages the model to learn the distribution based on the prior, and prevents prior collapsing to a Dirac function, therefore, samples are generated through a larger manifold in the latent space.

C.3 Variational inference

Amortized variational inference trains a model to predict the ϕ parameters that generalizes across all observations. Amortized variational inference finds a mapping function, that is the recognition model, from the observed sample to the latent state. We define our variational class as $q_\phi(x_n|o_n) = q(x_n|f_\phi^{rec}(o_n))$. Therefore, the ELBO is adapted as,

$$Ł(\theta, \phi|\mathcal{D}) = \sum_{n=1}^{N} \mathbb{E}_{q_\phi(x_n|o_n)} \left[\log p_\theta(o_n, x_n) - \log q_\phi(x_n|o_n)\right] \tag{C.12}$$

here, the subscript n denotes individual samples, and the joint distribution is equivalent to the prior and likelihood in eq. (C.10).

C.4 Gradients through the VAE

Gradients of the ELBO in eq. (C.10) need to be back-propagated through the generative and recognition model in order to find the optimal parameters ϕ and θ. The gradient of Łwith respect to θ can be moved inside the expectation and evaluated from Monte Carlo samples of the variational posterior, $x^s \sim q_\phi(x|o)$, thus,

$$\nabla_\theta Ł(\theta, \phi|o) = \mathbb{E}_{q_\phi(x|o)} \left[\nabla_\theta \{\log p_\theta(o, x) - \log q_\phi(x|o)\}\right] \tag{C.13}$$

$$\approx \mathbb{E}_{q_\phi(x|o)} \left[\nabla_\theta \log p_\theta(o, x^s)\right] \tag{C.14}$$

where the second term can be ignored there is no dependency

However, the gradients of Ł, with respect to the inference parameters ϕ, cannot be moved inside the expectation as x is stochastic and dependent on ϕ. This requires the reparametrization trick where the recognition model defines a new function, $g(.)$, which is differentiable, and takes an additional stochastic parameter $\boldsymbol{\epsilon}$. As an example, $x \sim \mathcal{N}(\mu, \boldsymbol{\Sigma})$ is transformed to $x = \mu + \boldsymbol{\Sigma}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0,\nVdash)$. This yields, $\mathbb{E}_{q_\phi(x|o)}[f(x)] = \mathbb{E}_{p(\epsilon)}[f(x)]$. Now, the gradient with respect to ϕ can be moved inside the expectation but with a change in variables,

$$\log q_\phi(x|o) = \log p(\boldsymbol{\epsilon}) - \log \left[\det \left(\frac{\partial x}{\partial \boldsymbol{\epsilon}} \right) \right] \tag{C.15}$$

where $\frac{\partial x}{\partial \boldsymbol{\epsilon}}$ is the Jacobian (Kingma & Welling, 2013).

ITQB-UNL | Av. da República, 2780-157 Oeiras, Portugal
Tel (+351) 214 469 100
Fax (+351) 214 411 277

www.itqb.unl.pt