Observe, Predict, Adapt: A Neural model of Adaptive Motor Control

by

Natarajan Vaidyanathan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis.

Author's Declaration

Abstract

Biological control systems have evolved to perform efficiently in an environment characterized by high uncertainty and unexpected disturbances, while relying on noisy sensors and unreliable actuators. Despite these challenges, biological control systems remain superior to engineered control systems in many respects. This edge in performance can be attributed to the exceptional ability of the brain to predict adaptively, and continuously update its control strategies in the face of uncertainties. Consequently, to harness these control abilities, it is crucial to delve into the study and modeling of cortical functioning.

This thesis presents a novel and comprehensive approach to elucidate the underlying mechanisms governing motor control. Specifically, we propose a biologically plausible spiking neural network model of the primate sensory-motor control system. The core of the model lies in its effective handling of noisy observations, integration of different sensory modalities, and the ability to learn to control the arm in the presence of perturbations. This is accomplished through the introduction of the Neural Adaptive Filter, a mechanism that dynamically predicts sensory consequences based on control inputs and observations.

The developed functional model of the sensory-motor control system exhibits complex behaviours observed in primates' reaching and demonstrates neural activities comparable to experimental findings. By adopting a spiking architecture, and connecting the lower-level synaptic dynamics and higher-level behaviours, such as visuomotor rotation, the model offers valuable insights into underlying mechanisms. Furthermore, the incorporation of anatomical structure and neural constraints enhances the biological plausibility and explanatory power of the model.

Moreover, the realization of a functional spiking model of the sensory-motor control system holds broader implications, particularly in control theory and its applications. This spiking model preserves the brain's inherent sparse coding, optimal performance, and energy efficiency, all of which are highly advantageous for engineering solutions. The model's operation can be generalized to that of a filter-controller framework capable of adapting to unknown nonlinear systems. This enhances robustness and plasticity derived from biological inspiration. Ultimately, by integrating the principles of a biological control system into modern control theory, our model not only offers insights into the sensory-motor control system and proposes potential advancements in modern control methodologies.

Acknowledgements

As I reflect on my PhD journey, I am struck by the profound experiences and invaluable memories that I encountered along the way. This journey would have been markedly different without the support of several amazing individuals and I want to raise a toast to them for being an incredible part of it.

Foremost, I would like to express my gratitude to my advisor, Chris Eliasmith. He has been a great mentor and a friend throughout my years in Waterloo. His patience, depth of knowledge and both "What did you expect? That is what research is like" and "Ah! Those are some nice plots" has always been a source of inspiration to me.

Special thanks are due to Terry Stewart, Travis DeWolf, and Aaron Voelker. Their willingness to entertain my relentless email spams and provide assistance at various stages of my research was invaluable.

I extend my appreciation to both the past and present members of the CNRG lab. It has been a special pleasure to pick the minds of Andreas Stöckel, Ivana Kajić, Pete Duggins, Nicole Dumont and Michael Furlong. They have been a cheerful bunch and a pleasure to share ideas with. Will definitely miss the board game nights and summer school musings.

My journey would have been incomplete without my Waterloo kin. Kenan Ulveea, for being a pillar of support, pushing and accompanying me in this journey (and the relentless one to glory). Ruta Patel, ever so kind and compassionate, has been a constant through the rollercoaster of highs, lows and despicable memes. Snehaa Suryanarayanan for being a patient listener, understanding and providing relaxing/chaotic company. Lakshmanan Arumugam, always there assisting me with much needed reason and more heated debates. I'm lucky to have them and my heartfelt greetings for you're stuck with me.

Finally, I would like to thank my family for, well, everything. My deepest gratitude is reserved for my mum, Bala Vaidyanathan, and my dad, Vaidyanathan Natarajan - who instilled in me the love for learning, whose sacrifices have paved the way for my achievements and for putting up with me. And to my brother, Akshy Kumar, for his faith, boundless support and an occasional "Isn't your thesis over already?".

Table of Contents

List of Figures

Chapter 1

Introduction

Modern control theory and artificial neural network researchers have developed innovative algorithms to control various systems, ranging from robotic arms to spacecrafts. In the process of solving these diverse problems, better sensors, precise actuators and novel optimization techniques have emerged, but so has the demand for faster computations, energy efficiency and more sophisticated algorithms. In order to meet these demands, a controller is typically tailor-made for a particular problem. As a result, such controllers are typically highly specific, designed using a variety of particular control methods, and highly tuned. This often results in controllers with very little ability to adapt to dynamic situations.

When compared to modern technologies, biological systems outperform them at core control tasks with remarkable efficiency and speed. This superior performance is primarily due to biology's unique ability to process and integrate vast amounts of information simultaneously to adapt to changing conditions, and learn and evolve over time. For instance, consider the problem of autonomous navigation, one of the difficult challenges faced by engineers. Biology has not only solved this problem but has also demonstrated remarkable proficiency, even in the most primitive of brains. *Megalopta genalis*, a variety of bee, achieves its feat of rainforest navigation in the dead of night by estimating its location with fewer than five photons [Warrant, 2004]. Similarly, an essential element in the behavioural repertoire of a dragonfly is its swift flight. Dragonflies can skillfully maneuver, anticipate the trajectories of their prey, and intercept at exceptionally high velocities relative to its body length. Its small brain of only a million neurons is able to achieve this with a success rate as high as 97% [Olberg et al., 2005].

When considering neurobiological systems as complex as humans and other non-human primates, we find extraordinary learning, adaptability and dexterity in performing complex, context-dependant control tasks. What is more remarkable is the ability of the brain to exert control, and simultaneously adapt to the changes in its uncertain environment on-the-fly. The brain's ability to gracefully handle constantly changing internal goals, different external rewards, dynamic and kinematic transformations, and varying biological costs, further bolsters the claim that the brain remains superior in control to our current technologies. The advancement of neural sensory and motor systems through years of evolution has rendered them highly intricate and notoriously resilient. Thus, the sensorimotor control system has the potential to help us improve our engineering control methodologies.

Understanding the brain's working and creating a model of its mechanisms is not a novel idea. Years of studying the cortex has provided a rich literature describing how different brain circuits and their mechanisms work. In fact, experiments from multiple domains of neuroscience has increased our breadth of understanding of the brain's working at varying functional levels [Sejnowski et al., 2014]. At a behavioural level, experiments have investigated how the brain chooses between different courses of action and how different functional areas of the cortex influence these decisions. Behavioural studies allow us to investigate the abilities of the brain to learns and adapt to a variety of tasks, but these studies are often limited in the information they provide about individual mechanisms that give rise to these behaviours. Experiments on a systems level, are better equipped to give insights into the relevant cortical pathways. Observations from studying injuries, disorders, and clinical experiments, can help us deduce the system states, and the functions of specific regions, and the relevant cortical circuits at a mechanistic level. On the other hand, with the help of multi-electrode arrays and optogenetic techniques, it is possible to dive further into the firing patterns, and how the cortical mechanisms manifest at a cellular level. Interestingly, experiments at multiple levels of the hierarchy, solve different parts of the same puzzle of how does the brain works. Hence, it is difficult for us to restrict ourselves to a single methodology to get a comprehensive understanding of the working of the brain. Rather, there is a need to combine findings to have a unifying model that is capable of explaining the workings from the synaptic level to higher level behaviour.

To fuse the breadth of experimental findings to produce a unified computational brain model, we need to address some significant challenges. Figure 1 depicts how different experimental techniques can probes the brain at varying spatial and temporal resolutions [Sejnowski et al., 2014]. Recordings range from patch clamps and EEGs to large scaled fMRI and lesion studies. To approximate the neural mechanisms, it is necessary for our model to capture the details of synaptic level dynamics that happens within a few milliseconds as well as the brain's behaviours that spans across minutes or even longer timescales. It is important to note that many of the higher level behaviours are often dictated by the lower level circuitry. For instance, consider long-term potentiation (LTP) and long-term depression (LTD), [RobertC et al., 1999] forms of synaptic plasticity referring to the ability of synapses to change their strength in response to patterns of neural activity. Studies have shown how LTP in the hippocampus is critical for encoding and storing spatial information [Bliss and Collingridge,

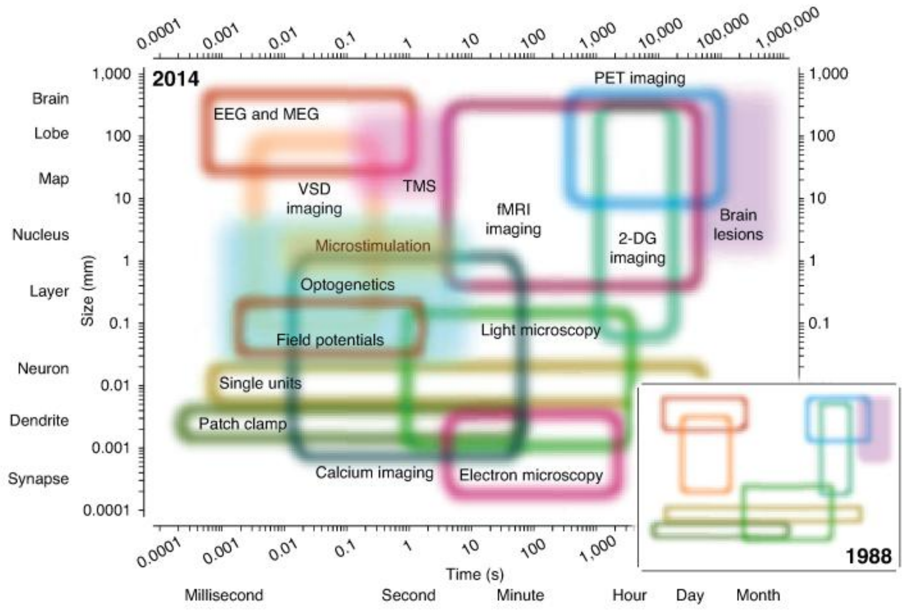


Figure 1.1: **The spatio-temporal domain of neuroscience methods available for the study of the nervous system**. Each colored region represents the useful domain of spatial and temporal resolution for one method available for the study of the brain. Open regions represent measurement techniques; filled regions, perturbation techniques (as described in [Sejnowski et al., 2014]).

1993]. Also, particularly for motor control, LTD in the Purkinjee cells of cerebellum has shown to be is involved in motor learning and coordination [Masetty et al., 1989]. Changes in the connectivity between the amygdala and the prefrontal cortex can lead to changes in emotional regulation and decision-making, showing how a lower level change can dictate a higher level behaviour. This evidence collectively supports the idea that it is necessary to model the brain at different scales to encapsulate the overall working of the brain, with comparable firing patterns, neural structures and synaptic connectivity.

While modelling the brain to exhibit organizational details captured by various levels of analysis, we should also remember to attend to the inherent biological constraints of the relevant mechanisms. For instance, we can model a neural network to perform a task of image classification, and use a very common optimization algorithm in Artificial Neural Networks (ANN's), such as back propagation. This neural network model, while capable of performing classification with remarkable accuracy [Lu and Weng, 2007], may not not guarantee biological realism. As shown by many, back propagation is not biologically realistic [Scellier and Bengio, 2017], hence makes the algorithm difficult to realize biologically and may not mimic the actual working of the brain. Likewise, it is possible to design a number of computationally adept models to solve a particular task, but it is the models with biological realism and comparable structure and function that are the ones most relevant for understanding the mechanisms of the brain. The model must be explainable and comparable with observed neuroscientific results, since the goal is to match the computational function of the cortex and not the mere performance at a given task. At first, the above additional constraints might appear as an unnecessary overhead to an already difficult problem of engineering the nuances of motor control. Interestingly, biological mimicry in modelling sometimes offers its own advantages. Constraining biological structure in encoding and information processing can help us incorporate effective computational strategies into our models. For example, in models that process images, V1 like cells emerge in convolutional models of vision, where the convolution is inspired by neural observations. Similarly,in [Cueva and Wei, 2018], the authors show grid-like representations, similar to those observed in the entorhinal cortex of rodents, emerge in artificial neural networks trained to perform spatial localization tasks. These ideas highlights a particular challenge in creating a unified model of the cortex. It is crucial to not merely reduce the brain's functions into to a black box, but to incorporate explainable structures with added neuro-anatomical constraints to not only have a comparable and comprehensive understanding of the brain, but also to enhance those solutions that we seek for our control problems.

While tackling the challenges in the modelling process, it is also important to select the system to model wisely – after all modelling the entire brain is currently an impossible task. The sensorimotor control system is an exceptional candidate for modelling, however, not only for the control engineering implications but from a neurobiological perspective as well. Motor control involves the execution of coordinated movements and is relatively well understood compared to other complex cognitive processes such as memory or decision-making. Movements are observable and quantifiable behaviour and makes it easier to model and compare against experimental findings. Furthermore, it is through the sensory and motor systems we interact with our reality. We experience sensory observations, anticipate changes and model our body as well as the environment and constantly adapt to environmental changes. In fact, the goals of the sensorimotor control system such as handling uncertainties, predicting action outcomes and adapting to changes, are not objectives exclusive to motor control alone, but common to many other cortical systems. Given that there are strong parallels between other cortical systems and the motor control system, modelling the sensorimotor system is an excellent target for improving our understanding of the brain. Furthermore, the engineering and control implications, generalizability to other perceptual systems, and applications in robotics and medical interventions all emphasize the necessity of such modeling endeavors. Consequently, modelling the sensorimotor control is a useful approach not only to enhance our engineering methodologies but also for building a more comprehensive model of the brain.

Currently, there are very few models of sensorimotor control that tie lower level dynamics with higher level behaviour. One pioneering model that addresses this issue is the REACH model (Recurrent Error-driven Adaptive Control Hierarchy ([DeWolf et al., 2016]). The REACH model establishes connections between spiking neural networks and higher-level behaviours, incorporating experimental findings. To achieve this, the REACH model utilizes the Neural Engineering Framework (NEF), a tool for implementing neurobiological circuitry and processing information through spike trains. By leveraging the NEF, the REACH model not only organizes and structures the motor control system more effectively but also provides insights into the functionalities of this complex system. In subsequent sections, we will delve into a more detailed examination of the REACH model and the NEF, exploring their respective implementations and contributions to motor control. This discussion helps to situate the current research and shows how we can leverage the NEF for building a new model of sensorimotor control.

In this thesis, we introduce a biologically plausible spiking neuron model of the sensorimotor control system, which integrates lower-level dynamics with higher-level behaviour. We begin by identifying the challenges present in control and estimation methodologies and contrast them with biological observations that shed light on the brain's approaches to control, estimation, and adaptation. These challenges include handling noisy observations, integrating multiple sources of sensory information, and effectively controlling the arm for various tasks. Based on the analysis of these challenges, we construct a system that aims to address them effectively. Initially, we present a linear version of the model for estimation, followed by a more accurate spiking nonlinear version. The model's performance is evaluated by testing it in a well-studied experimental paradigm and comparing the results against experimental data. The proposed model effectively handles noisy observations, integrates multiple sources of sensory information to make predictions, and successfully controls the arm for various tasks. By adopting a spiking architecture with anatomical and neurobiological constraints, the model replicates the functioning of the elements sensorimotor system, allowing for comparisons across different spatio-temporal resolutions, from synaptic to behavioural levels. The incorporation of anatomical structure and neural constraints ensures the model's biological plausibility and explanatory power. Furthermore, translating these mechanisms into an engineering context not only addresses control challenges but also harnesses advantages from bio-mimicry, such as optimality, robustness, and energy efficiency. The presentation of this model aims to foster the integration of neuroscience and engineering, offering inspiration for novel, sophisticated control methodologies. We believe this model will contribute to the development of a comprehensive, explainable, comparable, and accurate modeling framework, thereby hopefully furthering our understanding of the mammalian brains.

Chapter 2

Engineering Background

Dynamic systems are systems that evolve with time. The mathematical modelling of dynamic systems is ubiquitous in engineering. Such modelling has diverse applications ranging from controlling the temperature of a room to predicting the weather of a city. Regardless of the application, it is of great importance to study the interaction of the system variables and how the inputs can be controlled to produce desired behaviour in the system. The fundamental principle of control engineering is to characterize these processes, and design tools for studying and manipulating the behaviour of a system to drive the system to the desired state. In this section we describe different control theoretical concepts and leverage this a foundation towards the building of our model. In control engineering, the main components typically involved in designing and implementing control systems are (see Figure 2.1):

- **Plant or dynamical system**: The plant refers to the physical system or process being controlled. In general, this could be a mechanical system, an electrical circuit, a chemical process, or any other system that exhibits dynamic behaviour.
- **Sensors**: Sensors are devices that measure the relevant system signals of the plant. They provide feedback to the control system by converting physical quantities (e.g., position, temperature, pressure) into electrical signals that can be processed by the controller. In reality, it is often common to see noises added to the sensory measurements.
- **Estimator**: The estimator gathers the noisy measurements of the relevant system states from the sensors. Often combining multiple sensory information, the estimator provides an accurate estimate the current system state. This estimate is then compared with the desired system states to generate an error signal.
- **Controller**: The controller is responsible for determining the command signal. It takes the error signal as input and generates the control signal as output. The control signal is then sent to the actuator.

- **Actuators**: The calculated control signal influences the plant with the help of an actuator. An actuator converts the control signal into a physical action which could be a motor, a heater, a valve, or any other device that can manipulate the plant. Actuators, commonly physical devices, are subject to noises and disturbances that are capable of adding uncertainties into the system states.

Drawing parallels between the panels (a) and (b) of Figure 2.1, we can see how the classical control formulation can be helpful in modelling the sensorimotor control system. Essentially, in sensory-motor control, the system we aim at controlling is the arm interacting with the environment. We observe the system states using our vision (position and velocity information of the arm in Cartesian space) and proprioception (angular position and velocity information from the joint tension). It is our perception system that takes the role of an estimator in our control system. It uses the efference copy of the command signal and the two streams of noisy sensory information, to produce estimates in the two modalities and then the combined estimate of the hand location. This estimate is then compared with the desired trajectory of the arm to produce an error signal to generate the necessary command signal to follow the desired trajectory. Our muscles convert the neuronal activation into joint torques, completing the control loop that helps us move our hand to the desired trajectory. The added noise in neural transmission, physical disturbances such as gravity, external forces, etc. adds noise to the system as shown in the model in the Figure.2.1, (b). It is also important to note that there are details omitted here: for e.g., the system identification or learning that occurs in biological estimation or adaptive control while moving the actuators. We shall discuss such subtleties and the dynamic abilities of the individual components in the later chapters. For now, we delineate the individual elements that define the working process of the control system we aim at modelling, and importantly, how these elements can be realized within the framework of control theory.

In the remaining sections of the chapter, we provide brief descriptions of control and estimation – moving from linear to nonlinear methodologies – and also discuss the current state of system identification to provide an engineering foundation for our later neurobiological modelling.

2.1 Linear control

Building control systems, historically started with linear controllers and estimators for linear systems. Linear Time Invariant (or LTI) systems are a class of systems whose outputs for a linear combination of inputs are the same as a linear combination of individual responses to those inputs. When the input/output relation does not change with changes in time, it refers to the time invariant quality of the system. When the system under control is an LTI system, we can employ a corresponding linear controller to move the system to the desired target optimally. An LTI system can be expressed as a set of n coupled first-order ordinary differential equations, known as the *state equations* (eq. 2.1). For

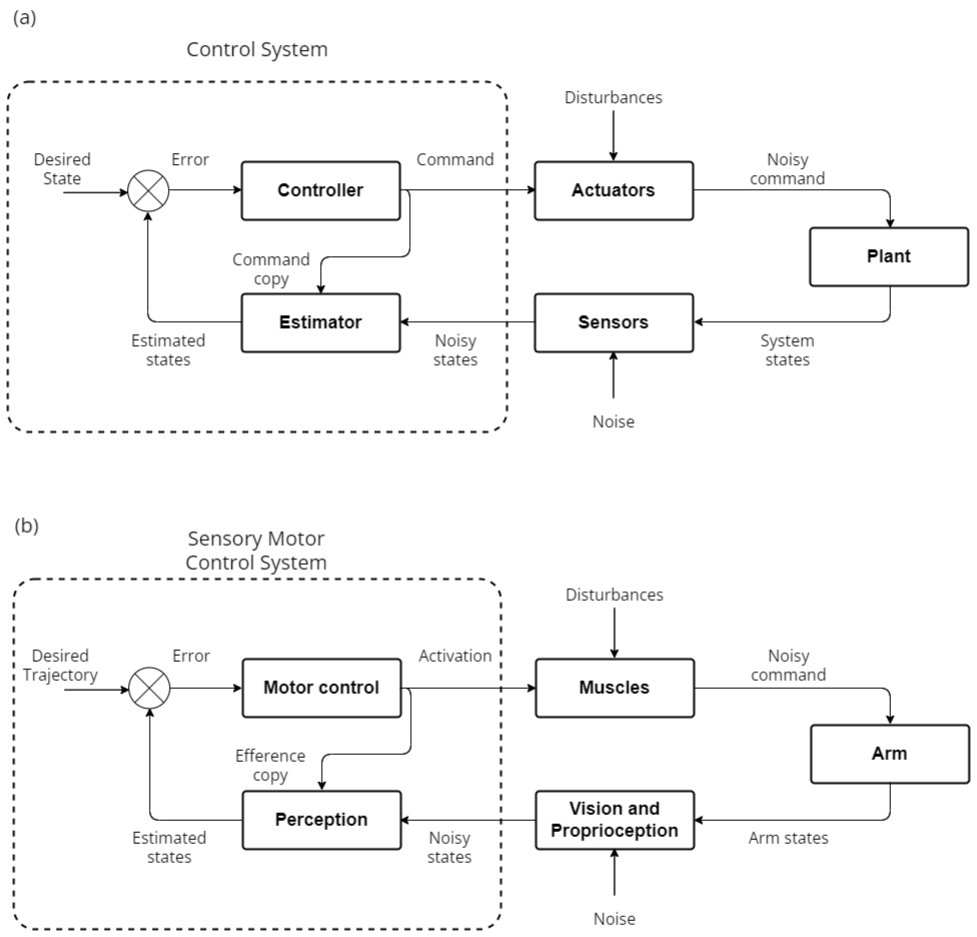


Figure 2.1: **State flow diagram of control feedback in classical control and sensorimotor control.** The top panel (a) shows the flow of control states in a classical feedback control loop from the plant enabled by the sensors and actuators. The bottom panel (b) shows the comparison of control flow in a sensory motor feedback loop states flowing from the arm to the cortex through vision and proprioception and the muscle actuating the arm.

describing a linear system, the states are combined into a vector form $x \in \mathbb{R}^n$, propagated by the dynamics matrix $A \in \mathbb{R}^{n\times n}$ and the input matrix $B \in \mathbb{R}^{n\times m}$ for the control inputs $u \in \mathbb{R}^m$:

$$\dot{x}(t) = A\,x(t) + B\,u(t) \tag{2.1}$$

The above equation characterizes the evolution of the linear system. For the time invariant case, the A and B matrices do not change through time, and we have dropped the time notation for simplicity. Linear control theory revolves around optimally estimating the evolving system and driving the system states to the desired target.

The optimal linear control problem is solved by a quadratic cost for driving the states to zero [Stengel, 1994], [Kirk, 2004]. The cost function J is defined as:

$$J = \int_o^{t_f} \left(\frac{1}{2}\boldsymbol{x}^T Q x\, dx \;+\; \frac{1}{2} u^T R\, u\, du \right) \tag{2.2}$$

where Q and R are state and control penalties respectively. Solving for the above cost function using the *Algebraic Riccati equation*, we get the optimal gain for u to drive the $x \to 0$ as $t \to t_f$. Adding the optimal feedback control gain $K_c \in \mathbb{R}^{n\times m}$ we get the control law:

$$u^* = -K_c \varepsilon \tag{2.3}$$

$$\boxed{\dot{x} = (A - BK_c)\varepsilon} \tag{2.4}$$

By employing the above control, we ensure the system is driven to ($\varepsilon \to 0$). For controlling the system to a desired state x_{des}, the optimal control law becomes eq. 2.3, where the error is given by $\varepsilon = x_{des} - x$. Substituting it into 2.1 gives us the new system dynamics 2.4, and with the gain parameter K_c, where $K_c > 0$, we can change the eigen values of the combined matrix $A - BK_c$, making it negative, converging the system to be asymptotically stable at zero. The Figure 2.2 (a) shows the control diagram of the linear state space control problem. Since this setup regulates the system states to a desired target by optimally reducing a quadratic cost, this control structure is known as a *Linear Quadratic Regulator*. This method, if incorporated with an integral term for the error, becomes similar to the working of a control strategy often known as the *PID control*, and is a common controlling strategy in implementing a linear control system with a simple feedback loop.

2.2 Linear filter

The above control scenario, with a simple feedback gain, works only when we have access to the actual state of the system. Often we have limited access to the full state of the system, either due to uncertainty or system restrictions.

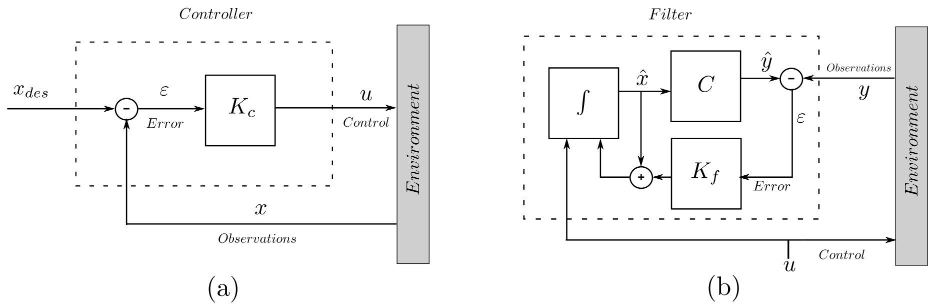


Figure 2.2: **Individual components of a Linear controller and a Linear filter**: (a) Shows the flow diagram of a linear controller giving the control signal u when the desired state x_{des} is given where K_c is the control gain. (b) Shows the flow diagram of a Linear filter or a Kalman filter, estimating $\hat{x}$ when given y, the observations are made from the environment. K_c is the controller gain and K_f is the filter gain

For example, we have access to the states of our arm through the noisy sensors of vision and proprioception. In such cases, it is necessary that we estimate the system states using our observations. An optimal linear filter or estimator provides a state estimate $\hat{x}$ by observing the noisy sensory information y and can be described as a form of linear observer:

$$\begin{aligned} \hat{y} &= C\hat{x} \\ \dot{\hat{x}} &= A\hat{x} + B\,u + K_f(y - \hat{y}) \\ \dot{\hat{x}} &= (A - K_fC)\hat{x} + \begin{bmatrix} B & K_f \end{bmatrix} \begin{bmatrix} u \\ y \end{bmatrix} \end{aligned} \tag{2.5}$$

The K_f is known as the optimal filter gain. We can determine how the error (ε) between the estimate and actual state is driven to zero. Let us combine our system equation 2.1 and the estimate equation 2.5 to get the propagation of our error.

$$\begin{aligned} \dot{\varepsilon} = \dot{x} - \dot{\hat{x}} &= (A\,x + B\,u) - (A - K_fC)\hat{x} + \begin{bmatrix} B & K_f \end{bmatrix} \begin{bmatrix} u \\ y \end{bmatrix} \\ &= A\,x - A\hat{x} + \cancel{B\,u} + K_fC\hat{x} - K_f\,C\hat{x} - \cancel{B\,u} \end{aligned} \tag{2.6}$$

$$\boxed{\dot{\varepsilon} = (A - K_fC)\varepsilon} \tag{2.7}$$

From the above equation, we can choose an appropriate K_f to make the combined $A - K_fC$ matrix negative, thereby ensuring the error converges to zero. This implies that the estimate $\hat{x}$ would eventually converge to the actual state x. Figure 2.2 (b) shows the flow diagram of the implementation of this linear filter formulation.

The optimal filter gain can be seen as reducing the quadratic cost function:

$$J = \int_{o}^{t_f} \left(\frac{1}{2} (x - \hat{x})^T Q (x - \hat{x}) \, dx \right) \tag{2.8}$$

where Q is the penalty term for the difference between the actual state and the estimated state. It is because of solving for the quadratic cost function, this formulation of the filter eq. 2.7 is called as *Linear Quadratic Estimator*.

Comparing the previous eq. 2.4 and eq. 2.7 shows the duality of the control and the estimation problems. In both cases we are solving for a quadratic cost, to give a gain that guarantees convergence. Employing these two together, gives us the *Linear Quadratic Gaussian* controller that together estimates the system states and regulates the system towards the desired trajectory.

2.2.1 The Kalman filter

The *Kalman Filter* is a reformulation of the Linear Quadratic Estimator that helps us in dealing with process and measurement noise. In addition to the linear assumptions we made for the LQR, we also assume zero-mean and Gaussian distributions for the noises added to the system. This gives us the Kalman filter, a common, optimal methodology for linear state estimation that is widely used. With the added noise, the linear system propagation is slightly modified as below:

$$\dot{x} = A\,x + B\,u + Gw \tag{2.9}$$

$$y = Cx + v \tag{2.10}$$

where A is the dynamics matrix, B is the control matrix, G is the dynamics noise matrix and C is the observation matrix. y corresponds to the observations of the system states x. w_d is the process noise, sampled from a Gaussian, zero-mean and of variance Q_n is the measurement noise, similarly sampled from a zero-mean Gaussian distribution with covariance R_n.

The combined system equation with a linear controller and Kalman gains becomes:

$$\begin{bmatrix} \dot{x} \\ \dot{\varepsilon} \end{bmatrix} = \begin{bmatrix} (A - BK_c) & BK_c \\ 0 & (A - K_f C) \end{bmatrix} \begin{bmatrix} x \\ \varepsilon \end{bmatrix} + \begin{bmatrix} I & 0 \\ I & -K_f \end{bmatrix} \begin{bmatrix} w_d \\ w_n \end{bmatrix} \tag{2.11}$$

It is important to note that in the combined system, the state convergence is still dictated by the control and filter gains. The knowledge of noise and uncertainty variance help in identifying how far the estimate is from the ground truth.

The Kalman filter technique is often expressed as an iterative approach as described below. With the knowledge of dynamics and measurement covariance Q and R and an estimate covariance, P, we can deduce the steps as:

$$\dot{P} = AP + PA^T + GQG^T - PC^T R^{-1} CP^T \tag{2.12}$$

$$K = P(t)C^T R^{-1} \tag{2.13}$$

$$\dot{\hat{x}} = (A\hat{x} + Bu) + K(y - \hat{x}) \tag{2.14}$$

The estimation covariance is often of interest during estimation to know how certain we are of the estimates from the Q_n and R_n. This gives more information on the certainty especially if Q_n and R_n becomes time varying. Also, during times when the observations is not available or has a large noise, the measurement uncertainty R is increased implying, the current $\hat{x}$ should be a function of our prediction system rather than the observation.

The preceding formulations have provided a framework for understanding linear dynamical systems and have facilitated the design of optimal controller and estimator pairs for such cases. However, real-world systems, including the biological control of our arm, often exhibit nonlinear behaviour. Nonlinear systems involve intricate relationships and nonlinearities that cannot be adequately described by linear models. Consequently, linear control and estimation techniques fall short in capturing the rich dynamics needed to accurately model nonlinear systems. Therefore, alternative approaches specifically designed to handle nonlinear systems are essential for effective control and estimation. In the following section, we will explore the setup of nonlinear systems and delve into the corresponding techniques for control and estimation. By doing so, we aim to tackle the challenges posed by nonlinear dynamics and provide insights into effective strategies for addressing them.

2.3 Nonlinear control

The behaviour of linear systems through time is well defined. Change to the input to the system produces a proportional change in the system states. For an LTI, there are several ways to understand the system response to a given control input and its further evolution through time, using techniques including the impulse response, Laplace transforms, root locus, Nyquist stability criterion, etc. [Bhattacharyya et al., 2018]. As we have seen, there exist well-established control and estimation techniques that work with great generality. The prescription of linear controllers and estimation works for all classes of problems that can be fully defined in a LTI framework. However, linear formulations cannot address the complexities of nonlinear systems due to their inherent limitations in capturing nonlinear dynamics and interactions. Fundamentally, nonlinear systems are not characterized by superposition principle, which means that the relationships between the system variables are neither proportional nor additive. Furthermore, nonlinear systems are highly dissimilar from one another, and are difficult to categorize. Hence, a control regime that works for one system might not work for another. Similar to systems in nature, this class of problems is characterized by inherently nonlinear interaction of states. Nonlinear systems include those systems that are characterized by chaos, which means the system evolution is highly sensitive to minute changes in the initial conditions bringing drastically different state trajectories [Slotine and Li, 1987]. The system becomes more complex if it is also time variant. These differences from typical linear systems makes nonlinear systems often impossible to solve analytically and hence it is difficult to devise generalized systemic procedures to such systems. Hence, there is a need to develop new methods to test for stability and novel tools to design controllers.

A nonlinear model can be represented by an equation of the form eq. 2.15, where f is a nonlinear function and $x \in \mathbb{R}^{1\times n}$ state vector. The number of states n is called the order of the system. The input u is the control signal that drives the system:

$$\dot{x} = f(x(t), u(t)) \tag{2.15}$$

It is necessary to emphasize that nonlinearity is inevitable when modelling real world control systems. Many of the forces present in natural systems, including coriolis forces, drag, damping and friction, behaves nonlinearly. Furthermore, during the linear control formulation we assumed that the dynamic system parameters are known, time invariant constants. In practice the system parameters vary with time and some times present us with no means to measure quantitatively. For instance, consider the system of an inverted pendulum. The system parameters such as mass, length, damping from the air resistance, and friction can be measured with great precision. However, these physical systems are susceptible to wear, tear and fatigue, and modelling dynamic friction, damping, and corialis forces becomes impractical. In these cases of nonlinearities, applying linear control solutions to nonlinear systems, even with linearization, comes with severe limitations [Morgan, 2015]. Overall, when dynamic systems are characterized by non-linearities, a separate formulation for characterizing and controlling these systems is required, which we shall discuss in the following sections.

2.3.1 Lyapunov's direct method

Before we start devising control methodologies, let us understand the stability of nonlinear systems. Essentially, understanding the stability and the equilibrium of the system can help us study the evolution of the states and can be used as precursors to estimating and controlling the plant of interest. Lyapunov's theory provides us with tools for determining the stability of an equilibrium point or a trajectory of a dynamical system. An equilibrium point, also known as a fixed point, is a state of the system where the state variables do not change over time. Stability describes whether the system returns to the equilibrium point or diverges away from it over time. Consider the dynamical system described before:

$$\dot{x} = f(x(t), u(t)) \tag{2.15}$$

where x represents the state variables of the system, t is time, and $f(x)$ is a function that determines the system's behaviour. An equilibrium point is defined as a state x^* such that $f(x^*) = 0$.

If there exists a function $V(x)$ (referred to as a Lyapunov function) that satisfies the following conditions, then Lyapunov's direct method prescribes that there are three types of stability that can be inferred: local stability, local asymptotic stability, and global asymptotic stability. Informally,:

1. If the candidate function $V(x)$ is locally positive definite and the derivative is locally negative **semi-definite**, then the system is stable inside a given radius.

   $$V(x) > 0 \; \forall x \in \mathcal{B} \tag{2.16}$$
   $$\dot{V}(x) \leq 0 \forall x \in \mathcal{B} \tag{2.17}$$

   for some neighborhood $\mathcal{B}$ of the equilibrium, the system is proven to be **stable**.

2. If the candidate function is locally positive definite and the derivative is locally negative **definite**, then the system will always converge to the equilibrium state starting inside a given radius.

   $$V(x) > 0 \; \forall x \in \mathcal{B} \tag{2.18}$$
   $$\dot{V}(x) < 0 \forall x \in \mathcal{B} \tag{2.19}$$

   for some neighborhood $\mathcal{B}$ of the equilibrium, the system is proven to be **locally asymptotically stable**.

3. If the candidate function is globally positive definite and the derivative is globally negative definite, then regardless of initial state the system will always converge to equilibrium, and the system is proven to be **globally asymptotically stable**.

   $$V(x) > 0 \; \forall x \in \mathcal{B} \tag{2.20}$$
   $$|x| \rightarrow \inf \implies V(x) \rightarrow \inf \tag{2.21}$$
   $$\dot{V}(x) < 0 \; \forall x \in \mathcal{B} \tag{2.22}$$

The Lyapunov's candidate function $V(x)$ can be seen as analogous to the energy of a system, and help us in finding points of equilibrium where the system converges upon dissipating energy. The idea here is: "if the system continues to dissipate energy, then it will eventually settle down to a state of minimal energy or an equilibrium point". Asymptotic stability is a very desirable feature of a system, that points us in the direction of formulating control as we have seen before in linear systems – driving the system to a "zero" state. But this can be difficult to prove with the above theorems, as often $V(x)$ is only negative semi-definite, rather than negative definite, and finding a Lyapunov function candidate with a negative definite derivative can be very difficult. To alleviate the difficulty of finding a candidate function with a strictly negative derivative, the invariant set theorem can be used. An invariant set is any set of states of a dynamical system where once the set has been entered, the system remains in that set. To put these theories into practice, let us implement an example using the above prescriptions.

2.3.2 Adaptive nonlinear control

It is necessary to reiterate the fact that nonlinear systems are quite different from one another, and hence to manipulate the system, generalized control frameworks are not available. Instead, each system requires its own stability consideration and control formulation. Therefore, in the following sections, we set up specific examples of nonlinear dynamics, identify system stability, and also go a step further by manipulating the system to follow a desired state trajectory.

Dampled pendulum dynamics

Let us analyze the stability of a nonlinear system of a damped pendulum dynamics [Slotine and Li, 1987] (see Figure 2.3. It has a simple non-linearity. The governing equation of the damped pendulum is given by:

$$J\,\ddot{q} + b\,\dot{q}|\dot{q}| + mgl\,sin(q) = u \tag{2.23}$$

where m, l, J are the mass, length and inertia of the pendulum respectively, b is the damping coefficient and the g is the acceleration due to gravity.

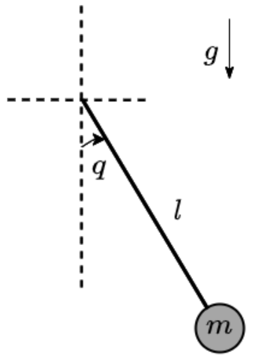


Figure 2.3: Damped pendulum model

Here we consider a trajectory control problem, where the desired trajectory to maintain is q_{des} and the tracking error is $\tilde{q}$:

$$\tilde{q} = q(t) - q_{des}(t) \tag{2.24}$$

For the ease of writing let us also introduce an intermediate variable q_r and a sliding variable s, given by :

$$s = \dot{\tilde{q}} + \lambda \tilde{q} \; = \dot{q} - \dot{q_r} \tag{2.25}$$

$$\implies \dot{s} = \ddot{q} - \ddot{q_r} \tag{2.26}$$

where we define $\dot{q_r} = \dot{q}_{des} - \lambda \tilde{q}$. In having the prescriptions stated before, consider a Lyapunov candidate function $V(s(x))$ given by,

$$V(s) = \frac{1}{2} J \, s^2 \tag{2.27}$$

whose derivative now becomes:

$$\dot{V}(s) = s \, J \, \dot{s} \; = s(J\ddot{q} - J \, \ddot{q_r}) \tag{2.28}$$

$$= s \, (u - b\dot{q}|\dot{q}| - mgl \, sin(q) - J\ddot{q_r}) \tag{2.29}$$

Structuring this further, let us rearrange the system state variable separately from the system parameters θ_d. Then the equation 2.28 becomes:

$$\dot{V}(s) = s \, (u - Y_d(q, \dot{q}, \ddot{q_r}) \, \theta) \tag{2.30}$$

where $Y_d(q, \dot{q}, \ddot{q_r}) = [\ddot{q_r} \;\; \dot{q}|\dot{q}| \;\; sin(q)]$ and $\theta = [m \;\; b \;\; mgl]^T$. The Y_d is a set of known functions of the considered system states and the θ_d is the constant encompassing the system parameters.

If we were to apply the control law:

$$\boxed{u^*(t) = -K_s s + Y_d(q, \dot{q}, \ddot{q_r}) \, \theta} \tag{2.31}$$

Plugging it back into eq (2.28), then the function V becomes,

$$\dot{V}(s) = -K_s s^2 \leq 0 \tag{2.32}$$

According to Lyapunov's theory, the system has essentially become asymptotically stable, and since $u^*(t)$ penalizes the tracking error as a part of s, the system also follows the trajectory $q_{des}(t)$. As a result, we have successfully devised a control regime for a nonlinear system, with the help of Lyapunov's direct method.

It is important to draw a contrast between the classic linear control we have seen before and the new control regime implemented here. The choice of control here in eq 2.31, is not only a function of the system states, but now a function of the system parameters as well. This $u^*(t)$ driving the system on the desired trajectory hinges on the assumption that the system parameters, are known with certainty. However, there is uncertainty in the measurement of the system parameters and as a result this control choice can introduce noise in the system, the convergence of the system states cannot be guaranteed. To tackle this problem, we adopt the algorithms developed by Dr. Slotine ([Slotine and Li, 1987]). They provide an adaptive control algorithm to manage the system parameter uncertainties, thereby guaranteeing system convergence for a nonlinear system.

We shall demonstrate the new algorithm by continuing the previous example and its control regime. Let us consider an offset in system parameter measurements from the actual parameters $\tilde{\theta}$:

$$\tilde{\theta}(t) = \hat{\theta}(t) - \theta(t). \tag{2.33}$$

Given this new term, we are going to add this to the cost function.

$$V(s) = \frac{1}{2}Js^2 + \frac{1}{2}\tilde{\theta}^T P^{-1}\tilde{\theta} \tag{2.34}$$

where P^{-1} is a symmetric, positive semi definite matrix. The derivative can be calculated as:

$$\frac{d}{dt}\left(\frac{1}{2}\tilde{\theta}^T P^{-1}\right) = \frac{1}{2}\left(\dot{\theta} - \dot{\hat{\theta}}\right)^T P^{-1}\tilde{\theta} + \frac{1}{2}\tilde{\theta}^T P^{-1}\left(\dot{\theta} - \dot{\hat{\theta}}\right) \tag{2.35}$$

Given that $\dot{\theta} = 0$ indicating that the system parameters are constants, and that each constant term is equal to it's own transpose, the two terms become equivalent and the derivative can be rewritten as:

$$\frac{d}{dt}\left(\frac{1}{2}\tilde{\theta}^T P^{-1}\right) = -\dot{\hat{\theta}}^T P^{-1}\dot{\hat{\theta}} \tag{2.36}$$

Here, $\dot{\hat{\theta}}$ specifies how $\hat{\theta}$ changes over time. Hence it is sufficient for us to specify $\dot{\hat{\theta}}$ so that the newly added term cancels out, making the the system stable again. So, the reformulated adaptation law is given by:

$$\boxed{\dot{\hat{\theta}} = PY^T(q, \dot{q}, \ddot{q}_r)\, s} \tag{2.37}$$

We can see from the above eq.2.37, the sliding error of system states s also drives the system parameter learning too. Now the control law suggested from the previous section 2.31 combined with our eq. 2.37 provides an *Adaptive control* algorithm to control a plant to follow a desired state trajectory q_{des} while taking into account the uncertainty in the system parameters.

Two-link arm dynamics

This technique is useful in tackling a larger family of similar nonlinear dynamical systems. Next we consider an example of two-link arm dynamics (or a double pendulum dynamics) that is more relevant from a motor control perspective. We discuss how we can translate the working of the adaptive controller to this new dynamical system. The dynamics of the two-link model can be described by (see Figure 2.4):

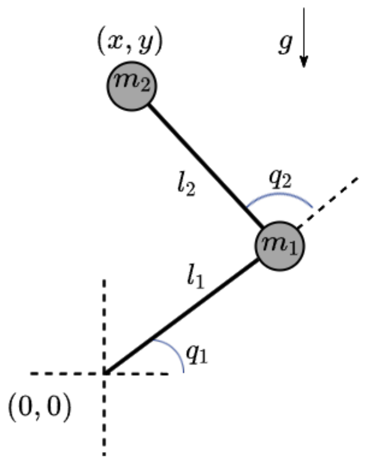


Figure 2.4: Two-link arm dynamical system

$$M\ddot{q} + C(q, \dot{q}) + g(q) = u \tag{2.38}$$

where M is the inertial matrix of the system in joint space, $C(q, \dot{q})$ is the matrix describing the Coriolis and the centrifugal component and the $g(q)$ is the term that describes gravity in joint space. Following a similar strategy as before, we can also define the sliding error and the reference trajectories. Now the $q(t)$ will essentially be the state vector containing both the joint's angles:

$$\tilde{q} = q(t) - q_{des}(t) \tag{2.39}$$

$$s = \dot{\tilde{q}} + \lambda\tilde{q} \ = \dot{q} - \dot{q}_r \tag{2.40}$$

$$\implies \dot{s} = \ddot{q} - \ddot{q}_r \tag{2.41}$$

The candidate value function is:

$$V(s) = \frac{1}{2} s^T M(q)\, s, \tag{2.42}$$

and the derivative now becomes,

$$\dot{V}(s) = s^T M(q)\, \dot{s} + \frac{1}{2} s^T \dot{M}(q) s \tag{2.43}$$

$$= s^T M(q)\, \dot{s} + s^T C(q, \dot{q}) s \tag{2.44}$$

$$= \dot{s}\Big(M(q)\ddot{q} - M(q)\ddot{q}_r + C(q, \dot{q})\dot{q} - C(q, \dot{q})\dot{q}_r\Big) \tag{2.45}$$

$$\dot{V}(s) = \dot{s}\Big(u - g(q) - M(q)\ddot{q}_r - C(q, \dot{q})\dot{q}_r\Big) \tag{2.46}$$

Setting the control signal equal to:

$$\boxed{u = -K_s s + g(q) + M(q) q_r + C(q, \dot{q}) \dot{q}_r} \tag{2.47}$$

where K_s is a symmetric positive definite matrix, and substituting into eq. 2.46 gives:

$$\dot{V}(s) = -s^T K_s s \leq 0. \tag{2.48}$$

The system is globally asymptotically stable, and now it is guaranteed to have zero position and velocity error when it enters the steady-state because $\dot{V}$ contains $s = \dot{q} - \dot{q}_r$, which must be zero when the system is at equilibrium.

Adaptation to unknown dynamics

The given control design above compensates for the gravity $g(q)$, inertial $M(q)$ and Corialis and centripetal forces $C(q, \dot{q})$ assuming that a perfect model of the system dynamics is known. As discussed earlier, this is often unrealistic and we need to adapt to the uncertainties to avoid adding error or otherwise the system becomes unstable. Similar to our previous system, we can split forcing function into unknown system parameters θ and the function of observable system states Y:

$$u = s^T K_s + Y(q, \dot{q}, \ddot{q}_r) \hat{\theta}. \tag{2.49}$$

The external forces that are not accounted for are thus counter-acted by learning the weights for a set of basis functions, $Y \in \mathbb{R}^{n \times d}$, defined over system states q and $\dot{q}$. The output of the basis functions is weighted by a set of learned parameters $\hat{\theta} \in \mathbb{R}^{d \times 1}$ consisting of estimates of $[\hat{M}(q) \big| \hat{C}(q, \dot{q}) \big| \; g(q)]^T$. Following the same method as before, choosing the candidate function becomes:

$$V(s) = \frac{1}{2} s^T M(q) \, s + \frac{1}{2} \tilde{\theta}^T L^{-1} \tilde{\theta} \tag{2.50}$$

The first term in the eq. 2.50, is similar to our previous equation 2.43. Hence let us calculate the derivative of the second term:

$$\frac{d}{dt}\Big(\frac{1}{2}\tilde{\theta}^T L^{-1} \tilde{\theta}\Big) = \frac{1}{2}(\dot{\theta} - \dot{\hat{\theta}})^T L^{-1} \tilde{\theta} + \frac{1}{2} \tilde{\theta}^T L^{-1} (\dot{\theta} - \dot{\hat{\theta}}) \tag{2.51}$$

$$\frac{d}{dt}\Big(\frac{1}{2}\tilde{\theta}^T L^{-1} \tilde{\theta}\Big) = -\dot{\hat{\theta}}^T L^{-1} \tilde{\theta} \tag{2.52}$$

To bring about stability in the system, consider the adaptation law:

$$\dot{\hat{\theta}} = L Y^T (q, \dot{q}, \ddot{q}_r) s, \tag{2.53}$$

Now, substituting this in the candidate function 2.51 the derivative becomes:

$$\begin{aligned}\dot{V}(s) &= -K_s s^2 + sY(q,\dot{q},\ddot{q}_r)\tilde{\theta} - \dot{\hat{\theta}}^T L^{-1}\tilde{\theta}\\ &= -K_s s^2 + sY(q,\dot{q},\ddot{q}_r)\tilde{\theta} - (LY^T(q,\dot{q},\ddot{q}_r)s)^T L^{-1}\tilde{\theta},\\ &= -K_s s^2 + sY(q,\dot{q},\ddot{q}_r)\tilde{\theta} - sY(q,\dot{q},\ddot{q}_r)LL^{-1}\tilde{\theta},\\ &= -K_s s^2 + sY(q,\dot{q},\ddot{q}_r)\tilde{\theta} - sY(q,\dot{q},\ddot{q}_r)\tilde{\theta},\\ &= -K_s s^2 \leq 0.\end{aligned}$$

and the system is proven to be globally asymptotically stable. With the choice of the control law eq. 2.49, and the adaptation law in eq. 2.53, it it now guaranteed that the system will follow the trajectory given by $q_{des}, \dot{q}_{des}$, and more importantly converge to this trajectory even with imperfect measurement of the system states initially.

The above methods of Lyapunov and the derived adaptive control mechanisms help us address the limitations of traditional linear control methods in dealing with complex and highly nonlinear systems. In short, the Lyapunov method can help us analyze a nonlinear dynamical system and with careful choice of control we can drive the systems to the desired state trajectory.

2.4 Nonlinear filter

With a foundation in nonlinear control established, the subsequent sections will delve into the methods of nonlinear filter (or nonlinear estimators). Nonlinear estimation is as an integral component within a control loop, providing the essential state information necessary for guiding control actions by processing the noisy state observations. In the following sections we briefly examine the principles underlying two different nonlinear state estimation techniques, namely the Extended Kalman filters, and the particle filters.

2.4.1 Extended Kalman filter

The Extended Kalman filter (EKF) is a nonlinear state estimation technique that extends the traditional, previously seen linear Kalman Filter to handle systems with nonlinear dynamics. It is widely used in control systems, and signal processing, where accurate state estimation is crucial. Let us understand the process of state estimation of EKF.

The system equations are given by :

$$x = f(x,u) + w_d \tag{2.54}$$

$$y = h(x) + w_n \tag{2.55}$$

where $w_d \sim \mathcal{N}(0, Q(t))$ and $w_n \sim \mathcal{N}(0, R(t))$ are process and measurement noise, respectively. We initialize the estimated state $(\hat{x})$ and covariance (P) as,

$$\hat{x}(t_0) = E[x(t_0)],\ P(t_0) = Var[x(t_0)] \tag{2.56}$$

To update the prediction, we propagate both the state estimate and covariance:

$$\dot{\hat{x}} = f(\hat{x}, u) + K(y - h(\hat{x})) \tag{2.57}$$

$$\dot{P} = FP + PF^T - KHP + Q \tag{2.58}$$

$$K = PHR^{-1} \tag{2.59}$$

$$F = \frac{\partial f}{\partial x}|_{\hat{x},u} \tag{2.60}$$

$$H = \frac{\partial h}{\partial x}|_{\hat{x}} \tag{2.61}$$

where K is the *Kalman gain*, and the state transition and observation matrices are the Jacobians F and H. Note here that x, h, y, P, K, F and H are all functions of time, but the time notation is dropped for readability.

Here we handle the nonlinear state transition matrix by linearizing the system dynamics function and observation function. To propagate the estimated state and finding derivatives of the state transition function, we assume that we have perfect knowledge of the system dynamics and they do not change over time.

2.4.2 Particle filter

Another approach to estimating state propagation of a nonlinear system is the particle filter. A particle filter is a Bayessian state estimator that uses discrete particles to approximate the state estimate [Lundquist et al., 2015; Thrun, 2002]. The EKF, being a Gaussian filter, assumes a unimodal Gaussian distribution for the state estimate. In contrast, particle filters are capable of capturing and representing multiple modes in the state estimate.

The particle filter formulation, often described in discrete space, is given by:

$$x_k = f(x_{k-1}, u_k, v_{k-1}) \tag{2.62}$$

$$z_k = h(x_k, u_k, n_k) \tag{2.63}$$

Drawing parallels from eq. 2.15, the f is the dynamic function that maps the state x_{k-1} at time step $k-1$ to the next time step k and h is the observation function. A particle is an individual state estimate defined by its location in the state space $\hat{x}$ and a probability p, that indicates the likelihood of the estimation. The samples of a posterior distribution are called *particles* and are denoted by:

$$\chi_t = \{x_t^{[1]}, x_t^{[2]}, x_t^{[3]}, ..., x_t^{[M]}\} \tag{2.64}$$

Now the state estimation problem is addressed as a Bayessian estimation problem. The posterior distribution at the previous time step $p(x_{k-1}|z_{1:k-1})$, is combined with the process model that describes how the state evolves over time in the prediction step. The result is referred to as the prior state:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{2.65}$$

The *prior* represents the best guess at a given time, given measurements upto time $k-1$. The algorithm for implementing a particle filter is:

1. Sample $x_t^{[m]} \sim p(x_t|z_{1:t}, u_{1:t})$
2. Calculate $w_t^{[m]} = p(z_t|x_t^{[m]})$
3. Update $\chi_t^+ = \chi_t + < x_t^{[m]}, w_t^{[m]} >$
4. For all partcles, draw i with probability with $\propto w_t^{[m]}$
5. Add $x_t^{[i]}$ to χ_t

The denser a subregion of the state space is populated by samples, the more likely it is that the true state falls into this region.

Although the particle filter is a powerful estimation technique for nonlinear systems, especially effective at capturing noises from a multimodal distribution, they come with limitations. Firstly, this technique for solving for estimation using multiple particles is computationally expensive. It becomes difficult to scale as soon as we have many states in the system. Another limitation of a particle filter is that the algorithm is essentially nondeterministic. The fact that the particles are drawn with a sampling technique, could give rise to different state estimates depending upon the collective sample obtained.

Both the EKF and the particle filters help us control known nonlinear systems, estimate system states, and leverage system propagation to adapt to the varying unknown parameters of the system. This controller/estimator pair is also found in an approach called Model Predictive Control (MPC) where we observe the systems, predict the plant's system states for a time horizon, and produce the appropriate command to control the system of interest. Depending upon the system, the required control and the noise involved, a good choice for filtering can be made.

However, in designing MPC's, irrespective of the individual computational advantages with each estimation methods (eg. the EKF, the particle filter etc.), they collectively suffer from a common challenge. In propagating the system state estimates, it is often assumed that the dynamic system is constant, and more importantly, that it is known with certainty. In reality, system interactions change with time. From a motor control perspective, we have to manipulate different objects with varying dynamic and kinematic nonlinearities. Our brain is capable of handling new dynamics with different state interactions, and learns to control the new system. The current estimator techniques are not sufficient to handle unknown and changing nonlinear dynamics. This is where we employ the methods of system identification, which we shall discuss in the following section.

2.5 System identification methods

Here we discuss a scenario where we need to identify the interaction of the system states with one another, to employ predictive control methods. Given a continuous stream of observations, the problem is to deduce the dynamics of the plant. Techniques that solve this problem are often known as system identification or dynamics system discovery methods. These methods use statistical techniques to obtain mathematical models of the dynamics of interest, necessary both to predict the future, and if possible, drive the system in a desired trajectory. Such methods are employed in numerous applications across several disciplines. It is useful in modelling climate [Lim and Zohren, 2021], biology [MacNeil et al., 1992], finance [Lee et al., 2006]. Unsurprisingly, such methods also apply to modelling the motor system. Our brain tries to model the surroundings all the time. We observe different states spaces, across different time spans to model changes that happens around us – from dropping a ball to the change of seasons. While some predictions are more abstract, we here delve into system dynamics frameworks for a control system. It should be noted that these methods are not simply estimation techniques, but a system identification for a plant of unknown dynamics.

Historically, there have been several techniques that are explored for dealing with data driven system identification, including Dynamic Mode Decomposition (DMD) [Schmid, 2010], Koopman theory [Budišić et al., 2012] [Williams et al., 2015], nonlinear autoregressive models [Akaike, 1969], and neural network models [Wehmeyer and Noé, 2018] [Lu et al., 2021], [Raissi et al., 2019]. Effectively, system identification techniques can be classified into two different approaches:

1. **Black box modelling**: These methods often work without assuming any prior mathematical structure to the observed data. They try to fit the observation configuration given as much data as possible to constrain the model. Most system identification methods fall under this category. While system evolution can be predicted with some certainty, it is often difficult to explain how the system states actually interact to give rise to these dynamics.

2. **White box modelling**: These methods are inspired by the style of modelling that uses the first principles to impose structure on the observed data and fit a prioiri constraints to estimate the system model. Given these constraints, it is more explainable at multiple levels as to how the system states interact to propagate the dynamics

In the next section, we take a detailed look at one recent example of white box modelling that captures system dynamics called Sindy-C. This method provides a state-of-the-art benchmark for a new, biologically plausible method that we discuss later in the thesis.

2.5.1 Sparse Identification of Nonlinear Dynamics with control - SINDy-c

The sparse identification of nonlinear dynamics with control (SINDy-c) [Budišić et al., 2012; Kaiser et al., 2018] approach uses sparse regression methods to approximate the equations governing the system. SINDy-c trains on state observations to give a concise description of interactions between variables to describe the system. SINDy-C also enables us to sweep through the parameters to choose the Pareto-optimal model from a family of models by balancing accuracy and efficiency.

Let us derive the working of the SINDy-c algorithm. Consider a non linear system described as:

$$\frac{d}{dt}x = f(x, u),\ x(0) = x_0. \tag{2.66}$$

Now give m measured snapshots of the state $x \in \mathbb{R}_{[p\times 1]}$ and the input signal $u \in \mathbb{R}_{[q\times 1]}$ in time, we arrange the observations into two matrices:

$$X = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & ... & x_m \\ | & | & & | \end{bmatrix}_{p\times m} \quad U = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & ... & u_m \\ | & | & & | \end{bmatrix}_{q\times m}$$

The library of the candidate nonlinear functions can be chosen using the observed measurements X and input U, for example:

$$\Theta(X, U) = \left[1^T\, X^T\, U^T\, (X \otimes X)^T\, (X \otimes U)^T ...\, sin(X)^T\, sin(U)^T\, sin(X \otimes U)^T\, ...\right]$$

where $x \otimes y$ defines the vector of all product combinations of the components in X and U. A suitable library of candidate terms is crucial for the working of SINDy-c algorithm. One potential strategy is to start with a common choice of polynomials, and increase the complexity of the library by including other nonlinear terms (trigonometric functions, exponential etc). A prior knowledge of the physics of the system of interest can also help us in making a smarter choices of functions too.

The system equation can thus be written as:

$$\dot{X} = \Xi\Theta^T(X, U) \tag{2.67}$$

The time derivative $\dot{X}$, if not measured directly can be computed by numerical differentiation. With the new formulation, we employ sparse regression to identify a Ξ. This regression is done by balancing sparsity, which corresponding to fewest nonlinearities to describe the system, with the accuracy to fit the given measurements. The components ξ of Ξ are obtained using:

$$\xi_k = \arg\min_{\hat{\xi}_k} \frac{1}{2}\|\dot{X}_k - \hat{\xi}_k\theta^T(X, U)\|_2^2 + \lambda\|\hat{\xi}_k\|_1. \tag{2.68}$$

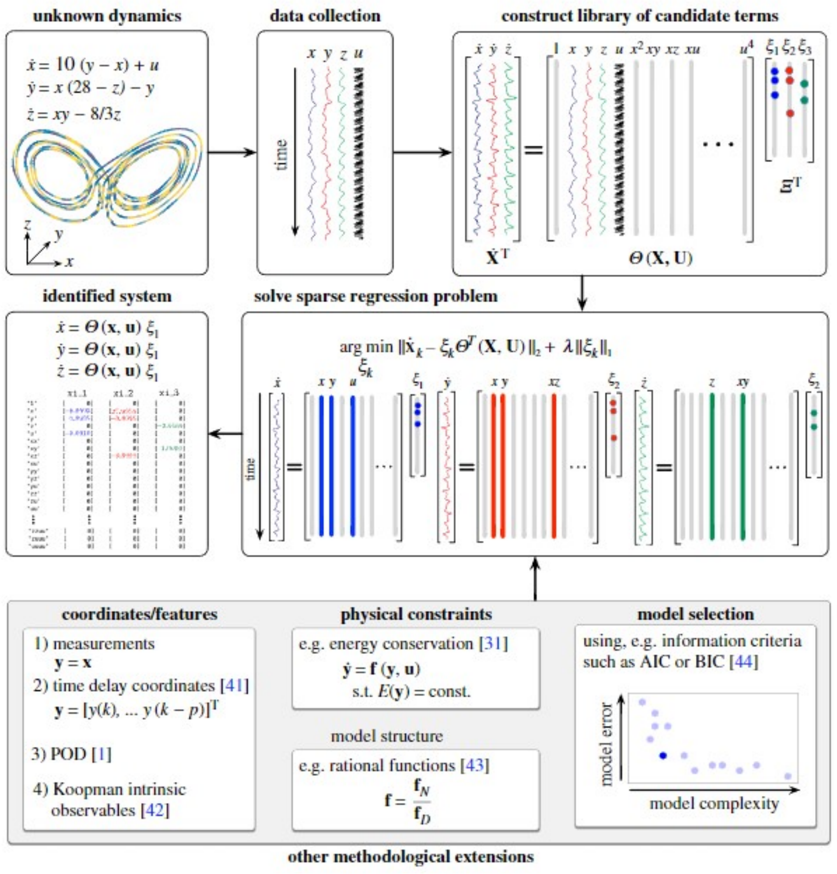

unknown dynamics
$\dot{x} = 10(y - x) + u$
$\dot{y} = x(28 - z) - y$
$\dot{z} = xy - 8/3z$
data collection
x y z u
time
construct library of candidate terms
$\dot{\mathbf{X}}^T$
$\Theta(\mathbf{X}, \mathbf{U})$
Ξ^T
identified system
$\dot{x} = \Theta(\mathbf{x}, \mathbf{u})\,\xi_1$
$\dot{y} = \Theta(\mathbf{x}, \mathbf{u})\,\xi_1$
$\dot{z} = \Theta(\mathbf{x}, \mathbf{u})\,\xi_1$
solve sparse regression problem
$\underset{\xi_k}{\arg\min} \|\dot{\mathbf{x}}_k - \xi_k \Theta^T(\mathbf{X}, \mathbf{U})\|_2 + \lambda \|\xi_k\|_1$
coordinates/features
1) measurements $\mathbf{y} = \mathbf{x}$
2) time delay coordinates [41] $\mathbf{y} = [y(k), \dots y(k-p)]^T$
3) POD [1]
4) Koopman intrinsic observables [42]
physical constraints
e.g. energy conservation [31]
$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, \mathbf{u})$ s.t. $E(\mathbf{y}) = \text{const.}$
model structure
e.g. rational functions [43]
$\mathbf{f} = \frac{\mathbf{f}_N}{\mathbf{f}_D}$
model selection
using, e.g. information criteria such as AIC or BIC [44]
model error
model complexity
other methodological extensions

Figure 2.5: **SINDy-c model**: Schematic of the SINDy-c algorithm and extensions.Active terms in a library of candidate nonlinearities are selected via sparse regression.Illustration of the modular nature of the SINDy-c with control framework (bottom row) and its ability to handle high-dimensional systems, limited measurements, known physical constraints and model selection.

where $\dot{X}_k$ represents the k^{th} row of $\dot{X}$ and ξ_k is the k^{th} row of Ξ. Upon regression, the Ξ can be used to describe the system equations as a linear combination of polynomials.

This method of sparse regression has many advantages. With very little data, we can approximate the system interaction and it is comparatively robust against noise [Budišić et al., 2012]. Unlike many blackbox methods, SINDy-c is also explainable. Despite these advantages, there are a few drawbacks to address in implementing this technique for modelling the sensory-motor control system of our brain.

- Choice of functional basis: SINDy-c is highly sensitive to the choice of functional bases we choose in Θ. Consequently, with a poor choice of the library of functions, an accurate system dynamics may not be arrived.
- Cannot do feedback control: If the input u corresponds to feedback control, so that $u = f(x)$, it becomes impossible to disambiguate the effect of the feedback control u from the interactions within the dynamical system; In other words, the SINDY-C regression becomes ill-conditioned.
- No online learning: SINDy-c is not a recurrent algorithm, but relies on a separate training and testing phases. Hence it is impossible to correct for errors online as we obtain new observations. This effectively prevents us from estimating of evolving systems or perform dynamic or kinematic learning, which are essential for model our biological control system.
- Incompatible biological framework: This framework inherently works with a process that cannot be reproduced using a biological system (library of polynomials, etc). Hence SINDY-C methodology is not apt for the purposes of neuro-scientifically realistic model of the brain.
- Numerical solving: After system identification, the system is solved for the state propagation using a numeric solver. This state trajectory is only as good as the choice of the numerical solver. Even when system is identified, since the dynamics are predominately chaotic, this method of system propagation can causes the system states to diverge from the actual states of the system.

2.5.2 Feedback-based Online Local Learning Of Weights (FOLLOW)

One comparable estimation technique is proposed by [Gilra and Gerstner, 2017]. The authors present an estimation technique that is capable of learning the system dynamics online. This method is described as a specific learning rule called the Feedback-based Online Local Learning Of Weights (FOLLOW). This method is notable for its local and online nature. The method in their work demonstrates learning of linear, non-linear, and chaotic dynamics.

From an online estimation perspective, the FOLLOW approach stands out for several reasons. Firstly, it is adaptable and works online, constantly incorporating new data, which is a significant advantage for online systems that need to respond dynamically to changing inputs. Let us outline the working of the FOLLOW method that helps in capturing the dynamics. Given a nonlinear system described by:

$$\dot{x}_\alpha(t) = h_\alpha(x(t), u(t)), \tag{2.69}$$

where x with components x_α (where $\alpha = 1, \dots\ N_d$) is the vector of observable state variables and h is a vector whose components are arbitrary non-linear functions h_α. The forward predicitve model then learns to make the error,

$$\epsilon \equiv x_\alpha(t) - \hat{x}_\alpha(t) \tag{2.70}$$

They assume that the predicted state is linearly decoded from the activity of the recurrent neural network of integrate and fire neurons. The input current to a neuron in the network learning the dynamics is given by:

$$J_i = \sum_l w_{il}^{ff}(S_l^{ff} * k)(t) + \sum_j w_{ij}(S_j * k)(t) + \sum_a k_{ia}^e(\epsilon_a * k)(t) + b_i \tag{2.71}$$

where the w_{il}^{ff} are the feedforward weights that computes the influence of the control input $u(t)$ to the system, S_l^{ff} are the spike trains from the feedforward path, w_{ij} are the recurrent weights and S_j are the spike trains within the recurrent network.The k_{ia}^e are fixed error feedback weights. The $k(t)$ is an exponential filter kernel and b_i is a neuron-specific bias.

The learning rule of the FOLLOW mechanism is given by:

$$\Delta w_{ij} = \eta\, \delta(t - t_j^{pre})\, (\epsilon_i * k)(t) \tag{2.72}$$

This equation represents the weight change Δw_{ij} in the learning rule. Here, η is the learning rate, $\delta(t - t_j^{pre})$ is the Dirac delta function centered at the presynaptic spike time t_j^{pre}, and $(\epsilon_i * k)(t)$ is the filtered error signal at the postsynaptic neuron. The equation is a key part of the learning mechanism described in the paper, specifically in the context of synaptic weight adjustment based on timing and error signals. The FOLLOW method is highly similar to what we aim at designing and hence an ideal candidate to compare to. At the end of our model performance evaluation (see Section 6.4) we discuss how the technique we propose differs in its working and compare against this mechanism.

This chapter has provided an overview of various control theory concepts, encompassing linear control, linear filters, nonlinear controls, nonlinear filters, and system identification methods. While these techniques serve as valuable foundations for control engineering, it is important to recognize their limitations and challenges when applied to model real-world systems, particularly when the goal is to biologically realize perception and control systems. Real-world systems often involve nonlinearity and unfamiliar dynamics that cannot be adequately addressed by linear approaches alone. To address these challenges, it is necessary to build upon these fundamental concepts and leverage their strengths in developing more sophisticated models and techniques. Bridging the gap between theory and practice requires considering the biological background of motor control systems within our brain. The upcoming section explores the existing body of knowledge and research, providing insights from relevant studies that inform our understanding and guide the construction of our model. By capturing the mechanisms and abilities of the motor control system and framing the problem from an engineering perspective, we can incorporate the wisdom gained from this broader context. This approach enables the development of more robust and accurate models for the complex motor control systems found in biological systems.

Chapter 3

Neuroscience background

Understanding the intricacies of primate sensorimotor control systems is essential in developing accurate models capable of effectively mimicking and manipulating such systems. By delving into the behavioural and neuroscientific aspects of motor control, we aim to obtain insights to enhance our understanding of mechanisms that drive rich motor behaviours. Recent research has shed light on the vital roles played by various brain regions, such as the motor cortex, basal ganglia, and the cerebellum, in orchestrating and coordinating complex movements. Furthermore, the interplay between sensory input, motor commands, and feedback loops collectively helps the brain to achieve movement goals. By reviewing relevant studies and findings, we can uncover the critical components required for execution of motor tasks, and for modelling the biological control system.

3.1 The brain: a powerful controller

Biological systems often deal with highly nonlinear and unreliable systems by handling observations through often noisy sensors with delay and uncertainty. It is interesting how the brain integrates the sensory information to estimate the relevant system states and solve for the control required. In many ways, nonlinear control and estimation is a solved problem in biology, and our brains have surpassed the performance of many of the current engineering solutions. In the following sections, we discuss a variety of experimental findings and how they describe specific qualities of the cortex that are critical for the efficient control of limbs.

3.1.1 Prediction

The delays in most sensorimotor loops are large. Efferent sensory signals are delayed as a result of neural conduction latency, low-pass filtering and interaction between neurons and muscles. It can take 100-300 ms to send a command, move the arms and receive visual feedback to perceive a change in hand location [Miall et al., 1993; Saunders and Knill, 2003; Miall et al., 2007]. The delays present in the sensory feedback loops are thus long enough to cause stability issues in the control system. Counter-intuitively, the stable control of the limbs indicates the presence of an internal mechanism to handle these delays for regulating movements, even before the controller receives a sensory feedback. In fact, Colby et al. found evidence suggesting that prediction of the sensory consequence in a saccade movement study, can be seen explicitly in monkeys. A group of cells in posterior parietal cortex (PPC), fires in anticipation of a light stimulus falling on the receptive field as the animal moves the eye from one location to another. In another example, [Haarmeier et al., 1997] conducted a study related to the perception of motion on a patient 'RW'. The patient, who suffered a stroke in the parietal and occipital cortex, reported seeing moving dots while the dots were actually stationary. 'RW' was unable to cancel the velocity of the eyes' movements from the image that fell on the retina, rendering the dots motionless only when the dots moved at the same speed as the eyes.

When considering the motor plant itself, parameters often change dynamically, both internally and externally with respect to the plant. This change happens at a shorter timescales (e.g., addition of masses, tool use, external forces like gravity, or damping) and in a longer timescale (e.g. muscular and neuronal fatigue, growing of limbs and the strengthening of muscles). To counteract such changes, it is reasonable to assume that the nervous system monitors external changes and forms an internal model of the plant and the environment [Shadmehr and Mussa-Ivaldi, 1994]. Numerous studies show the role of the cerebellum in creating a forward model [Manto et al., 2012], getting the efference copy, understand the timing of tasks and also generating errors comparing the action outcome and the estimated outcome [Ivry and Keele, 1989; Wolpert et al., 1995; Ohyama et al., 2003]. In a step tracking task performed by trained monkeys, researchers presented evidence supporting the prediction hypothesis [Fishbach et al., 2007]. Different analyses of the velocity profiles also made it evident that the corrective sub movements were initiated when the probability distribution of the predicted end-point, from the continuously accumulated information, is statistically different from the target location. Moreover, a latency in onset of the following sub-movements proportional to the amplitude of the previous primitive and the 'extent to go' of the reach shows that there is a continuous feedforward controller. These studies in addition to many others, strongly point towards the existence of an internal model in addition to the feedback system, that continuously predicts the plant's states during reaching.

The above suggests that delays in sensorimotor loops, as well as dynamic changes in the motor plant itself, necessitate the existence of internal mechanisms to handle these delays and regulate movements even before receiving sensory feedback. Hence, to accommodate for these behaviours our sensorimotor control model should include a predictive system that anticipates the consequence of a motor command.

3.1.2 Perception

Adapting to external perturbations and providing the appropriate control, demands knowledge of the relevant state variables. In the context of reaching, this includes the location and velocity information of the hand. With the available noisy observations from vision and proprioception, the brain estimates the state variable with considerable accuracy. Rushworth et al. found neural correlates of the independent contributions of both vision and proprioception in the posterior parietal cortex of monkeys. Lesions to lateral intraparietal area (LIP) and area 7 in the inferior parietal lobe (IPL) did not affect reaches where the goals were defined in the proprioceptive coordinates (in the dark), but produced mis-reaching, when in light [Rushworth et al., 1997]. In contrast, monkeys with lesions to Area 5 in the superior parietal lobule (SPL) could reach accurately in the light but not in the dark. Hence it is appropriate to suggest that the SPL may be involved in proprioceptive estimates of limb position in space, while the IPL is involved in visual estimates of limb position. An example of this in humans comes from the patient 'PJ' who had an extra-axial cyst encroaching on her left SPL [Wolpert et al., 1998a].Without vision of her right arm, PJ's perception of arm position became increasingly uncertain until she reported that the arm disappeared altogether. Thus, PJ was unable to store a proprioceptively derived estimate of limb state but, presumably because of the intact IPL, could use vision to maintain a sense of the limb's position in space. Similar to Rushworth's monkeys with SPL lesion, PJ made accurate reaches when vision was present.

Furthermore, in addition to individual perception, it is interesting to study how multiple sources of information are used to have a combined perception of the arm states. In adaptation studies (see Figure 3.1) [van Beers et al., 2002], [Haith et al., 2008], after adaptation subjects estimated the hand location somewhere in between the actual location and the observed visual feedback. This was a clear indication of a weighted summation of sensory information. In fact, the model proposed by [Reuschel et al., 2010] shows that the sensory integration is close to the *maximum likelihood estimate* – the most probable prediction of the state given the prior knowledge and current observations. Further, in the process of obtaining the estimate, the brain seems to perform kinematic learning, as these two sources of sensory information are observed in different representational spaces – proprioception in angular reference frame and the visual feedback in a Cartesian reference frame. The estimate of the final hand state is obtained by learnt combination of the two streams of information. [Körding and Wolpert, 2004; Vaziri et al., 2006; Izawa and Shadmehr, 2008].

Overall, it evident that an effective motor control model should not only anticipate the state propagation, but it also gathers multiple streams of sensory information to obtain a combined perception to estimate the limb states, to generate errors and learn to move the limbs in a variety of situations.

3.1.3 Generalization

Another important aspect of the motor control system is its ability to generalize, or apply what has been learned in one context to another unfamiliar context. This aspect of reaching goes beyond rote memory - not simply accessing a built-in look-up table of labelled sequences of muscle activation, but learning to reach under modified contexts. Examples of this generalization have been shown in many context such as temporal and amplitude generalization in motor learning in [Goodbody and Wolpert, 1998]. Here, upon learning to move in a novel velocity dependant force-field, subjects successfully generalized to a new state space while being asked to make movements of either half the duration or twice the amplitude. Generalization has also been described between movements with movement direction [Bedford, 1993; Huang and Shadmehr, 2007], angle [Krakauer et al., 2000] and even varying movement paths [Conditt et al., 1997]. Overall, generalization is a favorable trait that helps the brain to use prior knowledge to reach in a different context.

3.1.4 Adaptation

Given an internal model of the state estimate and constant feedback to correct the model and the consequence of movements, the brain demonstrates plethora of interesting adaptations. There are numerous instances showing compensation for dynamic changes, including force field studies where the hand is perturbed by external forces [Shadmehr and Mussa-Ivaldi, 1994; Li et al., 2001]. The brain can also learn to make successful reaches while in a rotating room with centripetal forces [Lackner and Dizio, 1994] and when other manipulations to the properties of the arm by attaching additional masses and other devices [Wang and Sainburg, 2004]). Studies have shown examples of reaching adaptation for kinematic changes, as seen in: (i) visuomotor gains [Heuer and Hegele, 2008], where the extent of reaching only altered visually; and (ii) visuomotor rotations, where the visual feedback is implicitly decorrelated and rotated with respect to the actual arm [Krakauer, 2009]. There are also more complex prism tasks[Von Helmholtz, 1867; Inoue and Kitazawa, 2018] and mirror tasks [Telgen et al., 2014], where the visual feedback is altered with a complex transformation and does not represent arm location explicitly. The brain is shown to adapt to making reaches when the hand visual feedback is provided with varying visual cues such as bars or rings indicating the distance away from targets [Vaidyanathan et al., 2020]. Adaptation happens even when the cursor is mapped on to highly complex hand gestures [Danziger and Mussa-Ivaldi, 2012]. The brain can adapt to reach successfully even in the presence of sensory decorrelation [Krakauer, 2009]. This evidence collectively showcase the extent of complexity the brain can handle when it comes to inferring limb states and performing a variety of unfamiliar tasks.

What is interesting in all of the adaptation studies, is that the cortex learns to compensate for changes to the environment, sensory misalignment and the relevant task simultaneously online. In some studies, the changes in the environment or the arm is made explicit during the experiment, and in other the changes are implicit. One involves a deliberate change of plan while the other is unconsciously driven by task errors. This further highlights the fact that error generation, planning and adaptations occur at multiple levels from observation, prediction, perception, and task goals. Additionally, the brain learns to compensate on the fly, not only when the perturbations are introduced, but also when removed. In sum, it is clear that adaptation is a staggeringly essential trait that demonstrates the versatility of sensorimotor control and capturing this adaptation is vital in modelling the biological control system.

Among the various adaptation studies, two experimental paradigms are most common– force field studies and visuomotor adaptation studies (see Figure 3.1). Each study reveals a unique property of the motor adaptive system. In force field studies, both the proprioceptive as well as the visual consequence of the hand are perturbed. Numerous studies have found effortless adaptation to external forces applied to the hand. Some of the most direct evidence comes from [Gandolfo et al., 2000]. They show a gradual recruitment of previously silent cortical neurons in M1, which begin responding to the sensory error. In this experiment the estimate from vision and from proprioception coincide with each other, giving one source of error – an unambiguous difference between the observed and perceived hand location. This error drives the learning process. After adaptation, the system regains its original performance of reaching to targets by learning to compensate for the external forces. The adaptation to force-fields, provides evidence that the brain accommodates for uncertainties in dynamics by correcting for errors in the task space.

The second common type of adaptation to visuomotor rotation [Shadmehr and Mussa-Ivaldi, 1994; Krakauer et al., 2000; Krakauer, 2009; Taylor and Ivry, 2011]. This experimental setup consists of a tabletop reaching task with the goal of moving a cursor, that represents the subject's hand, to targets that appear on the screen. The screen is placed directly above the subject's hand thereby blocking the direct view of the hand, while the cursor moves in accordance to the hand's movement. The subjects reach for the targets with the help of the visual cues on the screen and their proprioceptive feedback. After a period of successful reaches, visuomotor rotation is introduced. In the visuomotor rotation paradigm, a systematic directional bias is introduced around the hand thereby mismatching the visual feedback and the proprioceptive information of the hand location. For instance, if the subject were to move at $0°$, the cursor would move $30°$ away with respect to the angle of reach. It is logical to infer that this inconsistency affects the estimate of the hand location we have in our brain since proprioception and vision indicate different estimate for the hand location in such a scenario. It is observed in these experiments, that subjects aim and reach for targets without compensating for the rotational bias initially. But after a few trials, subjects gradually adapt to this new mapping and move their actual hand in the opposite direction to the transformation, adapting to the change. This learning also happens when the experiment is made more difficult by providing the cursor only during the beginning and the end of a

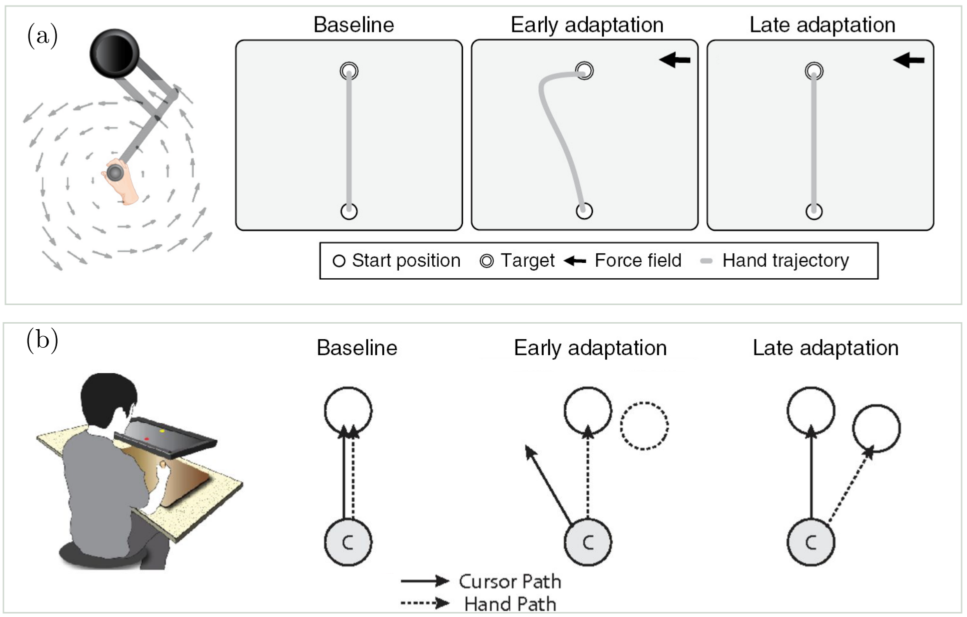


Figure 3.1: **Two adaptive behaviour paradigms**: The top panel shows the set up for force field adaptation and the bottom panel describes adaptation to visuomotor rotation. The baselines show perfect straight reaches in both conditions. (*a*) Subjects in early force-field adaptation have curved reaches in a counterclockwise curl-field applied by the robotic manipulator, and they learn to reach straight by the end of few trials. (*b*) Subjects in visuomotor rotation experiments learn the rotational transformation and learn to aim differently to reach for the target when the view of their hand is blocked. These two experiments provide complementory perspectives on primate motor control

single reaching movement.

The adaptation to visuomotor rotation is interesting for a number of reasons. Firstly, the behaviour reinforces the robustness and adaptability of the biological motor control. In addition to previous evidence of forward models which comes from the period during the reach where the visual feedback is unavailable, this task sheds light into the details of how the brain predicts the kinematics of the hand, while also dealing with unfamiliar changes in observations. In this case of sensory inconsistency, it seems that the brain assigns the observed error to both the visual estimate and the proprioceptive estimate, even though the actual bias is introduced only in the visual feedback. In particular, studies have shown that the brain creates a sensory illusion and perceives the hand somewhere in between the proprioceptive and visual estimate [van Beers et al., 2002]. An estimate of the hand location different from the visual and the proprioceptive observation, hints that the brain attributes the sensory error to sensory uncertainties. The model proposed by [Haith et al., 2008] describes how the adaptation in the brain is driven by associating the visual error partially with the internal model (i.e., accounting for the unreliability of the plant) and partially to the uncertainty in the sensory information. There is rich evidence suggesting that there is continuous learning and kinematic mapping to account for these uncertainties. Schaeffer et al. shows that the cerebellum plays a major role in learning an internal model during reaching movements. Studies in monkeys, [Wise et al., 1998], also show the changes in M1, M2, and PMd reflect the learning of visuomotor transformations. This adaptation is also known to have (a) retrograde, anterograde after effects (when the learning of new information interferes with the recall of old information) (b) structured recall (immediate adaptation strategy can be employed when given cues) (c) consolidation (adaptation becomes better with time and committed to memory). These features of the adaptation exemplify the richness of this learning ability and the efficiency in handling uncertainties and dynamic changes. Collectively, these evidence supports the potential involvement of prediction, perception, generalization and adaptation in a single experimental paradigm of visuomotor rotation.

Overall, the sensorimotor control system is able to efficiently control a nonlinear and unreliable plant, and is robust against a variety of perturbations. As a solution to this complex problem, the system has incorporated several mechanisms that are constantly updated to conform with dynamic and kinematic changes in both the environment and the plant. Furthermore, the biological controller remains superior to engineered controls by its ability to accommodate generalizations, adaptation depending upon contexts and prior information all while continuously learning from predictive error. Modelling these features of biological learning, could not only be a remarkable tool in describing the essential components that are crucial for accurately modeling the sensorimotor control system but also in replicating adaptable control mechanisms in engineered control systems.

Chapter 4

Motor control modelling

In this chapter, we consider attempts to bridge the gap between neuro-behavioural observations and mathematical tools to construct a models of the motor control system. Building upon the foundations laid in the previous chapters, which discussed the control engineering and neuroscience background, we discuss the history of motor control models. We begin by describing the varieties of models in motor control, the current state of models, and finally describe the modelling framework that we intend to leverage in building our model.

4.1 Current state of motor control models

Over the past decades, our understanding of motor control has undergone significant improvements through vast number of experiments and employment of theoretical tools to model the pathway of voluntary movements. While most computational models rely on empirical data generated by observation and experiments, they often capture different levels of abstraction. Models span from describing the cellular and molecular mechanisms of neural activities, to higher level behaviours of complex movements in humans and primates. Depending upon the experimental paradigm and the objective being explored, the current computational models in motor control can be largely classified into three groups: (a) cost strategy models (b) dynamic systems models (c) neural population models.

Cost strategy models are motivated by the theory that brains should have evolved to optimize the control strategy. Given that movement itself is an energy expensive process – ranging from firing of neurons to using energy to produce joint torques. The cost strategy approach seeks to find the optimal set of control inputs that minimizes an objective function while satisfying system constraints often minimizing mechanical precursors of metabolic costs, such as joint torque rates, energy ([Uno et al., 1989; Alexander, 1997; Kang et al., 2005; Flash and Sejnowski, 2001]. [Flash and Hogan, 1985] proposed a model of reaching movements that incorporates the minimization of torque as a cost function. The authors in [Todorov and Jordan, 2002] showed that their model can accurately predict the trajectory of reaching movements by providing a theoretical framework of *'minimal intervention principle'*. This principle states that an optimal controller, upon providing the optimal command for controlling the system, does not have to correct for task-irrelevant deviations. Relatedly, [Prilutsky and Zatsiorsky, 2002], which compared different choices of cost functions that explained the observed muscle activation pattern during reaching movements. There are several other models that look into the working of the motor control system by explaining its behaviour as an optimizing strategy. These mathematical models typically ground their findings in behaviours to a higher level optimizing control.

Dynamic systems models move their focus from a higher level variables such as movement cost, to a lower level movement states such as instantaneous joint torques, velocity, position, etc. These models try to characterize movement system as a controller concerned with task objectives, state variables, and the errors that drive adaptation. They also help in studying adaptation (or the lack of, in e.g. lesion studies) by attributing them to system level mechanisms to understand the distribution of the movement goal and contributing towards its respective anatomically distributed structures. Through this perspective, the movement system is perceived as a controller, balancing task objectives with state variables and the inevitable errors that guide adaptation. This view aligns well with Scott, who evidences the harmonization of movement cost optimization in reaching movements with internalized models of both the individual's physiology and the surrounding environment. Such an understanding is further advanced by Wolpert et al., suggesting an internal model driven by a predictive Kalman filter feedback system, emphasizing the brain's anticipatory nature in movement regulation. The dynamic systems approach also encapsulates the interaction of external forces on movement, a perspective supported by Shadmehr and Mussa-Ivaldi. Their research underscores how adaptations to these external forces can be mapped through the intrinsic coordinate systems of sensors and actuators. Moreover, when aiming to discern sources of motor errors, models like those of [Berniker and Kording, 2008] point to intricate neural mechanisms through probabilistic frameworks. Some models describe the evolution of specifics such as the timeline of learning, adaptations, and generalization. All of dynamic system models aim at replicating the behaviour aspect of the motor cortex.

The third approach of *neural population* modelling shifts focus to a more cellular level. Such models enable us to connect the firing patterns of neurons, or the activity of brain regions, to provide understanding of the neural mechanisms underlying motor behaviour. One of the earliest studies to inspire models is [Georgopoulos et al., 1989], which demonstrated that the activity of individual neurons in the motor cortex are highly correlated with the direction of movement. This study proposed a 'population code' to predict movement direction by observing neurons from dorsal premotor cortex and the primary motor cortex. Subsequent research has bolstered our capability to correlate various movement aspects with their corresponding neural activities. Cellular-level models can also capture complex interactions between different brain regions, as demonstrated by Churchland et al., who proposed a model of neural population dynamics during reaching that accounted for both feedforward and feedback control. These models can also be employed to make predictions about the effects of neural manipulations, as shown by [Sadtler et al., 2014], who used optogenetic techniques to examine how learning is constrained by the neural representation of the task. In [Mante et al., 2013], they monitored an area of PFC involved in the selection and execution of saccadic eye movements.

Different approaches to modeling motor control operate at different functional levels and offer varying levels of detail. The cost strategy model, based on optimization principles, focuses on efficient control by considering the cost of movement. While it can explain higher-level constraints in motion selection, it may not fully capture the underlying low-level mechanisms of neural control. On the other hand, the dynamics systems approach can mimic individual brain structures, providing an area-level understanding of the neural circuitry involved. However, caution is necessary as not all system-level descriptions are biologically plausible, limiting their ability to accurately represent the motor pathways. Cellular-level approaches offer comprehensive and detailed insights into synaptic mechanisms, but can be challenging to interpret and scale up to capture higher-level behaviours. It is important to acknowledge that each approach has its strengths and aids in testing different hypotheses about the brain's function. However, it is evident that none of these approaches single-handedly can provide a comprehensive explanation connecting overall behaviours to neural mechanisms. To achieve a more holistic understanding, a new approach is required, one that integrates cellular activities with higher-level behaviours, organizes cortical structures, and incorporates biological plausibility to better approximate the workings of the brain.

Recently, there has been a few models that tried connecting some of the aspects we discuss here. In Denève et al., they proposed a computational framework where the brain can be thought of as a hierarchical system of processing layers, each layer containing a set of computational units that learn to encode increasingly complex and abstract representations of sensory information. The authors provided evidence to support this framework, including simulations of artificial neural networks that exhibit similar properties to the brain, and reproduce neurophysiological data from studies of learning and adaptation in humans and animals. Although this work is an appropriate candidate for unifying high-level behaviour and cellular mechanisms, the proposed model assumes that each layer in the hierarchy is composed of the same type of computational units and oversimplified processing units and connectivity patterns, especially when it comes to adaptivity and learning. Additionally, the model's treatment of noise and uncertainty—-fundamental elements in any biological system, and especially in neural processing-—is fairly limited. While the model may be computationally effective, it could not accurately represent the actual biological processes occurring in the brain. In its current state, Denève et al.'s model might offer a mechanistic and theoretical viewpoint, but it falls short in truly capturing the multifaceted and nuanced processes intrinsic to the brain's operations. A genuine unification of high-level behaviour and cellular mechanisms mandates a model that respects the complexity, adaptability, and intricacies of the biological substrate it seeks to emulate

In contrast to the previous deterministic models, [Berniker and Penny, 2019] provided a probabilistic approach that focused on the goals and the constraints of limb movement. They proposed a network to implement an LQR controller for a reaching task. However, there are areas where the model could see enhancements. For instance, the training of the proposed network presumes flawless state encoding. The model's foundation on the inherent stochastic nature of neurons, from a probabilistic viewpoint, resulted in a void in neuronal dynamics. The environmental parameters used for the model were also strictly Cartesian and purely observation-based. Enriching the model with predictive feedforward control and a more nuanced grasp of arm dynamics would undoubtedly fortify its modeling prowess. It is helpful to have the ability to capture biological details in encoding, learning and capturing neuroscietific details when describing the behaviour. While replicating a computational description of the causal relationship between neurons and the motor behaviour, it is equally imperative that the framework be rooted in biological realism.

Many of the deterministic frameworks and probabilistic approaches have notable limitations that hinder their ability to fully capture the intricacies of the brain's biological processes. A comprehensive model should incorporate biologically plausible elements, such as realistic processing units, adaptivity, robustness to noise, and uncertainty, at the same time scalable to encompass the complexity of the brain. Developing such a model can provide a more deeper and realistic understanding of the brain's functioning, enabling accurate representations of biological processes. In the following section we briefly discuss one such framework, that not only connects the lower level activity to the higher level behaviour, but also capturing biological constraints.

4.2 The Neural Engineering Framework

The Neural Engineering Framework [Eliasmith and Anderson, 2003] leverages techniques from dynamical systems theory that help us model neuronal and synaptic dynamics using a spiking neural network. Biological constraints can be included by specifying the details of neurons, such as their dynamics, firing rate, the time constants of the neurotransmitters and many other features. From an engineering standpoint, NEF also allows smooth transition of these neural models to be implemented in spiking neuromorphic hardwares making it possible to also have robotic implementations and have sophisticated controllers. In the following sections, we shall discuss the core principles of this framework and the tools available for us to construct our cortical structures within this framework.

4.2.1 Representation

To perceive the information about the environment, to cognize them as internal states and to use the states to solve problems, it is important to construct representations of the physical quantities into system states. Here, the NEF describes a mathematical framework for representation, by encoding the physical quantities into neural activities. In other words, if $x(t) \in R^q$ denotes a continuous time varying system state, it is represented in a higher dimensional vector space in the population of neuronal activity, $a(t)$ given by:

$$J_i(t) = \alpha < e_i, x(t) > +\beta_i \tag{4.1}$$

$$a_i(t) = G_i[J_i(t)] \tag{4.2}$$

where $s_i(t)$ is the input current to the i^{th} neuron, $< .,. >$ denotes a dot product, $a_i(t)$ is the neural activity generated by the i^{th} neuron encoding the vector $x(t)$ at time t, and $G_i[.]$ is the nonlinear transfer function modelling the spike-generation of a single neuron.

The function G depends upon the choice of neuron model used. A common choice is a *Leaky Integrate and Fire* or the *LIF* neuron model [Koch and Segev, 1998]. For a given neuron, this model maintains a one dimensional voltage trace $v_i(t)$ by continuously integrating the current source J_i over time with some leakage governed by τ_{RC}. It is equivalent to a resistor-capacitor network or a low pass filter dynamics. After scaling the voltage to the interval [0,1], the governing equation is:

$$\tau_{RC}\dot{v}_i(t) = -v_i(t) + J_i(t),\ 0 \leq v_i(t) < 1 \tag{4.3}$$

where $v_i(t) \rightarrow 1$ generates a spike at time t and the voltage resets to zero for a τ_{ref} time period. Although this neuron model is an approximation to neuronal dynamics, the NEF allows computation with a choice of more biologically detailed neuron models as well.

Along with the neuronal dynamics, we include the post-synaptic filter that models the post synaptic current triggered by the action potential arriving at the synaptic cleft. This is modeled to be an exponentially decaying post-synaptic current with a time constant τ:

$$h(t) = \frac{1}{\tau}e^{-(\frac{t}{\tau})},\ t \geq 0 \tag{4.4}$$

As we represent the state variable in this fashion, we obtain a neuronal spike train that encodes the given value through time. These neuron spike trains are nonlinear encoding of vector spaces that can be linearly decoded to obtain the state representations. We can characterize the decoding of the neural response as follows:

$$(x * h)(t) \approx \sum_{i=1}^{n} (a_i * h)(t) d_i \tag{4.5}$$

$$= \sum_{i=1}^{n} \sum_{m} h(t - t_{i,m}) d_i \tag{4.6}$$

where $D = [d_1, d_2, ..., d_n]^T \in R^{n \times q}$ is the decoding matrix, used to decode the filtered version of $x(t)$ from the population activity. The $*$ operator is the convolution operator representing the synaptic filter. The collective decoders D can be found optimally to decode the appropriate state from the spike train, which is described in the following section.

4.2.2 Transformation

While the representation principle helps us encode state variables into neuron activities $a(x)$, we now describe how we not only decode the state variable but also realize new functions of the state variable. Here we leverage different linear decoding to estimate any arbitrary vector functions of the given input. In order to realize a function f we can compute it in the connection between two neuronal populations. We can then approximate function f using a function decoder D^f. This decoder D^f should have the property:

$$y = f(x) \approx \hat{y} = D^f a(x) \tag{4.7}$$

These decoders can be found by solving for a least squares problem by reducing the error in:

$$E = \frac{1}{vol(X)} \int \int_X |f(x) - D^f(a(x) + v)|_2^2 dx, \text{ where } v \ \mathcal{N}(0, \sigma). \tag{4.8}$$

This decoding error is both a function of the number of neurons in the pre population as well as the smoothness of the function calculated. With the help of the calculated decoders, arbitrary nonlinear functions can be realized. Combining the encoding and decoding equations, we can interpret the ‘neuronal weights’ required to compute the function, that is $W = ED$.

4.2.3 Dynamics

We have shown how to represent a state variable and devised a method to compute nonlinear functions of the state variable as well. In this section, we add temporal elements to describe dynamics. Here, we implement arbitrary dynamics of the form:

$$\dot{x} = Ax(t) + Bu(t) \tag{4.9}$$

where x is the time varying system state, $\dot{x}$ is the time derivative. Similar to how the feedforward connection weights were computed, we can find a weight

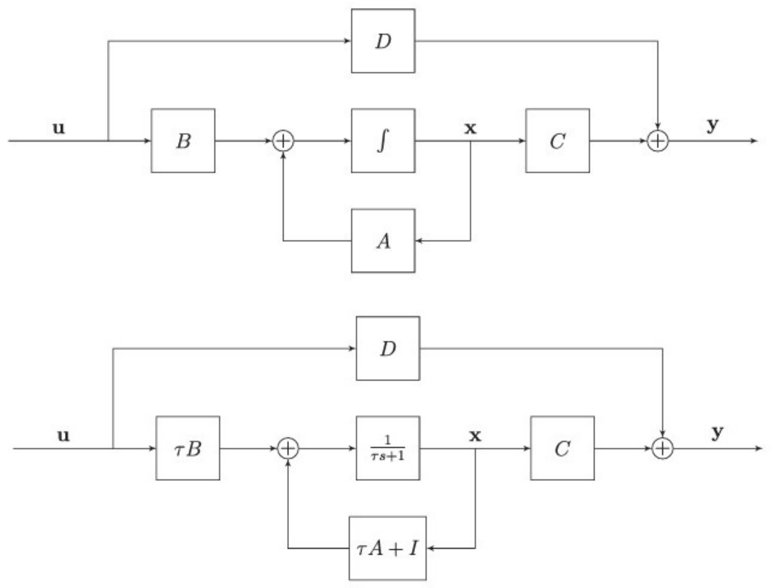


Figure 4.1: **LTI system vs Neural implementation**: NEF dynamics Comparison between an LTI system being evaluated using an integrator, compared to the neural implementation. The $\tau A + I$ and the τB is equivalent to the A, B matrices of the LTI system.

W^f that approximates a certain dynamical equation. For the LTI system in 4.9, the propogation through time happens with the help of an integrator. Similarly in neurons characterized as above, the dynamics stems from the leaky integrator given by the canonical first order low pass filter modelled by the synapse.

$$h(t) = \frac{1}{\tau} e^{-(\frac{t}{\tau})} = \mathcal{L}^{-1} \frac{1}{\tau s + 1} \tag{4.10}$$

In the Laplasian domain this can be rewritten as:

$$X(s) = \frac{1}{s}(AX(s) + BU(s)) \tag{4.11}$$

$$X(s) = \frac{1}{1 + \tau s}(A'X(s) + B'U(s)) \tag{4.12}$$

Rearranging the second equation, we get,

$$X(s) = \frac{1}{s}(\frac{A' - I}{\tau} X(s) + \frac{B'}{\tau} U(s)) \tag{4.13}$$

$$\implies A' = \tau A + I, \; B' = \tau B \tag{4.14}$$

Here we have introduced the neural dynamics matrix, A' and input matrix B' which define the dynamics of the system implemented in a recurrent neural fashion, and can be related to the standard dynamics and input matrices A and B from the state eq. 4.13 (See Figure 4.1). By deducing the dynamics representation of a linear system, this principle allows us to map any arbitrary dynamical system that can represented in the format of eq.4.9. This gives raise to interesting circuitry realizations, such as oscillators, which are often an integral part of cortical mechanisms.

4.2.4 The Legendre Memory Unit (LMU)

An important dynamic system that we employ in our model one which represents a window of signal across time. The 'LMU' is mathematically derived to orthogonalize the continuous-time history in a window by mapping it into corresponding Legendre polynomials [Voelker et al., 2019]. This approximation of the signal allows us to learn functions from the memory of the signal in time. In this section we describe briefly the working of LMU and represent it as a linear dynamical system, which can then be implemented within the framework of NEF using the methods described in Sections 4.2.1-4.2.3.

Let us consider a continuous time delay of θ seconds, expressed as :

$$y(t) = u * \delta_{-\theta}(t) = u(t-\theta),\ \theta > 0 \tag{4.15}$$

where $\delta_{-\theta}$ denotes the Dirac delta function shifted backwards in time. This function helps us in taking in a time varying scalar signal $u(t)$ and outputs a purely delayed version, $u(t-\theta)$. For this the transfer function of the system is defined as:

$$F(s) := \frac{\mathcal{L}(y(t))}{\mathcal{L}(u(t))} = e^{-\theta s} \tag{4.16}$$

The pure delay (see equation 4.16) has infinite order when expressed as a ratio of polynomials. To overcome this, it is approximated the irrational transfer function $e^{-\theta s}$ as a proper ratio of finite order polynomials. We do so using Padé approximates — the coefficients of a Taylor series extended to the ratio of two polynomials — expanded about $s = 0$ [Padé, 1892]. Doing so gives:

$$[p/q]e^{-\theta s} = \frac{\mathcal{B}_p(-\theta s)}{\mathcal{B}_q(\theta s)}, \tag{4.17}$$

$$\mathcal{B}_m(s) := \sum_{i=0}^{m} \begin{pmatrix} m \\ i \end{pmatrix} \frac{(p+q-i)!}{(p+q)!} s^i. \tag{4.18}$$

Upon choosing $0 \leq p \leq q$, we may numerically find the state-space model that satisfies equation 4.17 given by:

$$\dot{m}(t) = Am(t) + Bu(t) \tag{4.19}$$

where m is the Legendre representation of the input u. The state space matrix A and B are given by:

$$A = \begin{pmatrix} -v_0 & -v_0 & \cdots & -v_0 \\ v_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & v_{q-1} & 0 \end{pmatrix}, \quad B = \begin{pmatrix} v_0 & 0 & \cdots & 0 \end{pmatrix}^\top, \tag{4.20}$$

where $v_i := (q+i)(q-i)(i+1)^{-1}, fori = 0..q-1$. The C vector helps us in decoding any instant of the signal represented in the window back in to its time dimension.

$$u(t-\theta`) \approx \sum_{i=0}^{q-1} \mathcal{P}_i \left(\frac{\theta'}{\theta} \right) m_i(t) \ , 0 \leq \theta' \leq \theta \tag{4.21}$$

$$\mathcal{P}_i(r) = (-1)^i \sum_{j=0}^{i} \begin{pmatrix} i \\ j \end{pmatrix} \begin{pmatrix} i+j \\ j \end{pmatrix} (-r)^j \tag{4.22}$$

where $\mathcal{P}_i(r)$ is the i^{th} shifted Legendre polynomial. The basic idea is that the LMU allows us to represent a sliding window of time θ and then extract different parts of the window linearly. This method of representing a signal through time has a huge significance in representing and learning function of the signal represented across time. In later chapters, we leverage this methodology to build a prediction model.

4.2.5 Learning

Synaptic plasticity is arguably one of the fundamental mechanisms in our brain that makes it possible for the cortex to change synaptic weights to adapt over time. Many synaptic plasticity experiments result in learning rules that explain how individual connection weights change. One of the methods by which plasticity is incorporated into the framework of NEF is the Prescribed Error Sensitivity (PES) learning rule [Bekolay et al., 2013]. This rule or the PES learning rule uses the NEF principles to help modify the decoders to learn a specific function. An error signal is generated comparing the present and the target value represented. Mathematically, this results in the learning rule:

$$\Delta d_i = -\frac{\kappa}{n} E a_i \tag{4.23}$$

where d_i are the decoders that are being learned, κ is the scalar learning rate, E is the error signal $E = f(\hat{x}) - f(x)$, and a is the activation of the neurons. This rule can either directly modify the decoders, d, or the connection weights $w_i j$ between neural populations. In later models, we use this rule to implement a variety of adaptation models for motor control and perception.

These core principles of NEF – representation, transformation and dynamics – give a framework for us to represent system states and calculate different functions within a spike based network. Embedding these tools in our methodology allows us to realize neural network mechanisms in a more biologically plausible fashion and to test out the working of various cortical mechanisms. In the next section, we look at a recent model of motor control that uses the NEF and connect the lower level dynamics with the motor behaviours in a spiking architecture.

4.3 Recurrent Error-driven Adaptive Control Hierarchy - REACH

The Recurrent Error-driven Adaptive Control Hierarchy (REACH model) [DeWolf et al., 2016] presents an anatomically organized spiking neuron model for the motor control system using the NEF (see figure 4.2). The neural adaptive controller presented here is also used as neuromorphic implementation in [DeWolf et al., 2020]. The REACH model proposes an implementation of motor control with the help of control feedback in the operational space for moving an arm model to reach for targets. It exhibits complex high-level behaviours while also capturing lower level neuronal dynamics.

By leveraging the anatomical organization of primary motor cortex (M1), premotor cortex (PMC), supplementary motor cortex (SCx), and the cerebellum, the model generates trajectories and controls that closely resemble human-like traces and velocity profiles (see Figure.4.3). This is made possible through the integration of the spike timing-based PES learning rule, which facilitates two crucial aspects – cerebellar adaptation and cortical adaptation. During reaching tasks in an unknown force field (the task described previously in Section 3.1.4), the REACH model showcases its cerebellar adaptation capability. Initially confronted with an unknown force field, the model adapts its movements by incorporating the dynamic changes introduced by the external forces. Through the use of the cerebellum, the model fine-tunes its motor commands, gradually aligning its movements to accurately navigate the force field. In addition to cerebellar adaptation, the REACH model also exhibits cortical adaptation, specifically when faced with incorrect information about segment lengths. In scenarios where the model starts with inaccurate segment length information, it seamlessly adapts its kinematics to ensure precise reaches within a velocity-based force field. This cortical adaptation mechanism reflects the model's capacity to rectify initial errors and optimize its motor output to achieve accurate reaching movements.

This model was one of the first realizations of such high level capabilities with built-in biological constraints. The REACH model uses spiking neural mechanisms for the neural representation and for computing dynamics to realize its control process. This made it possible to group the network into well defined anatomical structures in the brain. As a consequence of combining the bottom up and top down approaches, it was also able to realize neural firing patterns that are very similar to the ones observed in the monkey cortex (see Figure.4.3) reflecting correlations found in experimental research. REACH re-

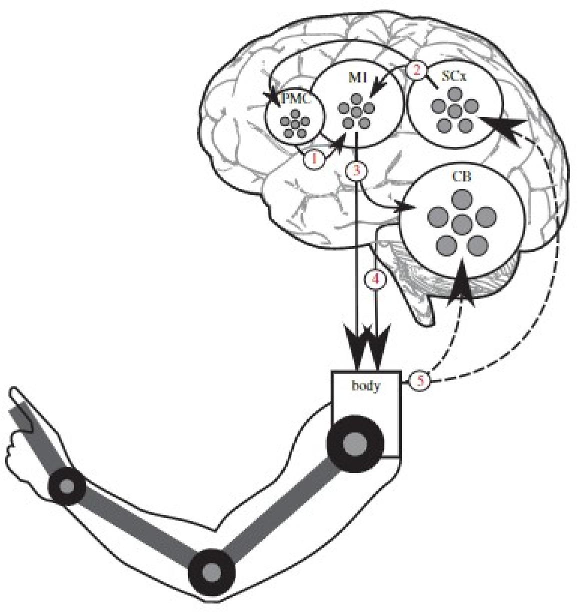


Figure 4.2: **REACH model**: An overview of the REACH model, shown controlling a three-link arm. Numbers identify major communication pathways. Dashed lines indicate closed-loop feedback signals generated from the senses. The premotor cortex(PMC) generates a trajectory for the system to follow with a sequence of (x,y) coordinates. The primary motor cortex (M1) receives these target positions (1) from the PMC and compares them with the current system state, received from the sensory cortices (SCx), through (2). M1 combines this signal with locally calculated Jacobians to transform the desired hand movement commands into a low-level signal that is sent to the arm and cerebellum (CB) along (3). The CB projects an adaptive signal to the body along (4) that compensates for velocity and movement errors. Visual and proprioceptive feedback projects from the body along (5) to the CB and SCx (taken from [DeWolf et al., 2016]).

produces population encoding of the movement magnitude, movement velocity, acceleration, and rotational dynamics that can be seen in M1. It also exhibits similar neural oscillatory patterns in low-dimensional state space analyses observed in primates. The ability to reproduce these neural dynamics with the adaptation behaviours, showed how the holistic and biologically constrained approach was useful for better explaining the working of the primate motor control system. By adding neurobiological details, the REACH model encapsulated both behavioural and neural level details of the cortex. This framework employed a modified form of the adaptive nonlinear control (discussed earlier

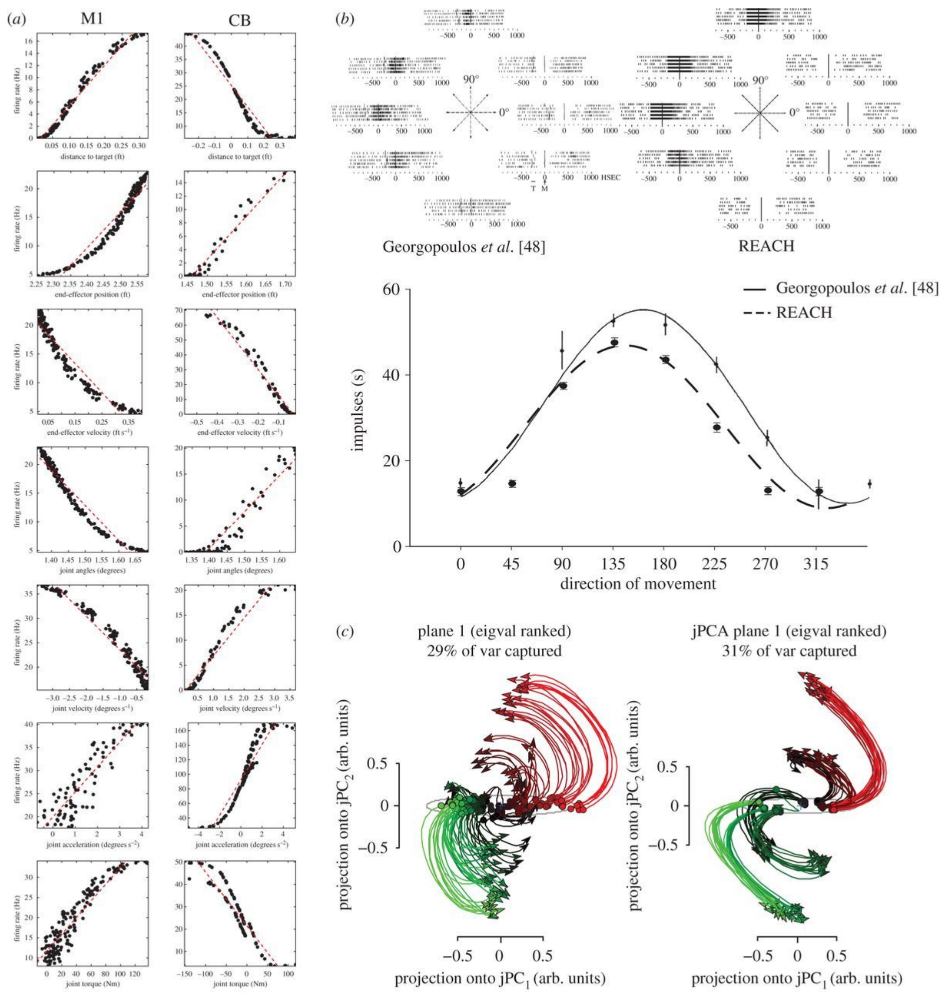


Figure 4.3: **Results of the REACH model**(a) Plots of the neurons from the primary motor cortex (M1; left column) and cerebellum (CB; right column) correlating with various high-level movement parameters. (b) Examining a neuron's activity profile in response to the arm reaching out to eight surround-centre targets over five trials. Top: original results from [48] on the left, and REACH model results on the right. Bottom: performing the same sinusoidal regression as in [48] allows a quantitative comparison. (c) jPCA analysis applied to data collected from monkeys and the model during reach trials. Results from the original analysis are on the left, and from model analysis on the right. The colour is determined by the starting position of the neural activity. Both analyses exhibit a similar rotating path through the low-dimensional state space over time (taken from [DeWolf et al., 2016]).

[Slotine and Li, 1987] and Section 2.3.2) that formed the basis of adaptive control mechanisms implemented in many neuromorphic control of robotic arms ,[DeWolf et al., 2020, 2023].

Despite the REACH model capturing many of the adaptive control aspects of the motor control system and reproducing biological observations, it has a few critical limitations. Firstly, the model lacks details in the sensory cortex. The majority of adaptations captured by the REACH model relies on the perfect instantaneous access to the system states – both end-effector position and joint angles. With the lack of any form of 'forward model' or predictive system, the model in incapable of reaching during noisy measurements or importantly in the absence of sensory observations. As we have discussed earlier (see Section 3.1.1), prediction plays an important role in the perception and control of the limbs and actively helps in deducing plant and environment dynamics. In addition, there are no separate sensory streams of visual and proprioceptive information. The characteristic features of these sensory observations give useful information about the uncertainty and sources of error that drives adaptation at multiple levels. As discussed in Section 3.1.2 the individual estimates of state values from various modalities are necessary to characterize both the sensory cortex and the motor cortex, which together orchestrate various adaptations. There is strong evidence for separate representations of visual, proprioceptive and hand perceptions, which need not align and give rise to some of the important behaviours. There is an "illusion" of the hand that drives the fulfilment of the task at hand, until any learning for de-correlation or changes in the environment happens. Behavioural adaptation in the absence of sensory perception, the abilities of the motor control model to adapt to several complex changes which are observed in biology. Hence, towards the goal of constructing a detailed realization of the cortex a combined sensorimotor perception system is an indispensable step.

Sensory perception systems can also be helpful well beyond adaptive strategies. While interacting with dynamically changing environments, we also learn contexts and build models of the word around us. This can only be made possible with a detailed perception systems generating errors between the anticipated states and the observed states. For example, error source attribution [Liepelt et al., 2008], a crucial mechanism in the brain that enables the discerning of changes and attributing the motor errors to improper prediction, sensory changes or from environmental factors. Such mechanisms hint at a larger system at play that leverages the detailed working of perceptual systems beyond just motor control, but to understand the physics of the environment that we are a part of. Hence, the collective shortcomings from REACH and other the current models of motor control creates a need for a more detailed and elaborate biological modelling of the sensorimotor control system.

If the control and actuation side of the motor system helps the brain to influence the environment, it is the sensory side of the motor system that helps us observe and create a perception of the outside world we interact with. Consequently, in a step towards more accurate modelling of the motor cortex, in the following sections, we work towards building a model of a sensory perception system to help us embody a more complete motor control system, instilling the ability to adapt to more complex environments, and hopefully pushing the model towards a more comprehensive model of the cortex that control our limbs.

Chapter 5

A Linear Model

With reference to the experimental accounts reviewed in the previous chapter, we have identified the salient components of the motor control system: prediction, perception, and adaptation mechanisms. To summarize, the predictive system plays a crucial role in constructing a forward model of the plant dynamics while continuously estimating the necessary system states for limb control. These predictions actively guide perception by inferring system states from multiple sensory inputs and contextual information. Moreover, the predictive mechanism assists in perceptual filling-in phenomena by providing insights into sensory uncertainties and discerning system states when direct observations are unavailable. The brain detects discrepancies by comparing observations, predictions, and perceptions, facilitating adaptations that lead to improved future motor commands. In the subsequent sections, we delve into the construction of a preliminary linear filter model that incorporates these components to encompass the motor control system.

Based on the analysis of biological behaviours, let us distill the necessary functional elements of the prediction system. The prediction mechanism has a running recurrent estimate that forms the basis of the forward model. To construct an effective forward prediction model, it is essential to incorporate the concept of an efference copy of the control signal. This copy of the input command is used to propagate the system states, enabling the formation of an internal prediction of the expected sensory outcomes resulting from the applied motor commands. Given the dynamic nature and noise present in sensory information, the prediction mechanism should effectively handle observation noise. Additionally, it is clear that the brain estimates system states by weighing the certainty of incoming sensory observations (see Section 3.1). In the presence of increased noise in the observations, greater trust is placed in estimates, while more accurate observations allow for adjustments of the prior beliefs. Furthermore, the intended motor control model must also account for dynamic changes in limbs that occur over multiple timescales, including variations in system parameters (e.g., mass, length, stiffness) and kinematic or dynamic alterations in the target dynamics. It is important to note that arm dynamics are often nonlinear, underscoring the complexity inherent the prediction system.

5.1 Linear prediction model

Let us draw upon the engineering methodologies established earlier in Chapter 2, to build a biologically relevant prediction mechanism. Among the various systems explored, the Kalman filter emerges as a compelling starting point for an estimation model due to its alignment with many of our prediction model requirements. The Kalman filter offers a reliable framework for real-time estimation and control, operating recursively to refine predictions with each new measurement. Notably, the filter's utility arises from its optimal estimation, with the knowledge of the noise covariance of input measurements and the dynamics. Moreover, the robustness against noise better equips the filter to handle the challenges of an online motor control system to deal with noisy sensory data, providing an accurate estimates of the system's true state. Constant calculation of system gradients is required in an Extended Kalman filter and Particle filters are stochastic in nature making these filters difficult to realize these filters in a biologically plaussible fashion. The simplicity and deterministic nature of the Kalman filter further reinforces its suitability, compared to its computationally heavy and biologically implausible counterparts. Given these merits, the Kalman filter serves as an ideal real-time estimation system and a promising starting point for our prediction model.

Indeed, the Kalman filter is a viable candidate, but it is important to consider its limitations and its impact on the prerequisites of building a biologically realistic sensory-motor prediction model. One limitation is the filter's reliance on assumptions of linearity for system state propagation. This linearity constraint results in inaccurate predictions when dealing with real-world nonlinear interactions in hand control dynamics, kinematics, and sensory transformations. Additionally, the Kalman filter assumes Gaussian distributions for the measurement and process noise, while noise sources in practice may exhibit non-Gaussian characteristics, such as heavy tails or skewness [Victor and Purpura, 1996; Johnson, 2001]. These violations of the Gaussian noise assumption can significantly affect the filter's performance and result in less accurate state estimates. Furthermore, the Kalman filter requires precise knowledge of system dynamics and the influence of control inputs on the system, which may not be fully known in the context of motor control. Moreover, the motor control plant is also known to be susceptible to change over time. In light of these inherent shortcomings, along with the overhead of tailoring the mechanism to a biologically realistic framework, using a Kalman filter model presents significant challenges.

To strike a balance between capturing complexity and ease of implementation, we consider a few alterations and assumptions while building an initial version of the model. In an effort to match the linearity assumption of the filter, our preliminary model will approximate the nonlinear arm dynamics of the plant with a unit mass system moving on a frictionless surface perpendicular to the plane of gravity. This simplification linearizes the target plant, making the system suitable for a Kalman filter framework while realizing the relevant motor control system states such as position, velocity, and control forces. In the future, this system will also be appropriate for capturing the estimation of a 'cursor' or a visual marker dynamics representing the arm's end-effector, which is inherently linear. To address the complexity of noise characteristics, we assume an independent Gaussian distribution. While this simplification allows for ease of estimation and hypothesis testing, it is important to acknowledge that real-world noise encompasses various complexities across multiple levels, including system dynamics perturbations, visual and proprioceptive sensory noises, and transmission noises. Additionally, to account for instances when observations are unavailable, such as when visual cues disappear or when the eyes are closed, we prevent system correction driven by errors by inhibiting the error population. This inhibition ensures that the estimation system remains robust during sensory unavailability. Furthermore, for this initial implementation, we relax the constraint of dynamically changing systems and instead work with a known time-invariant system. These assumptions and alterations allow us to develop an implementation of a Kalman filter-based estimation system that provides an initial step towards a more sophisticated implementation.

Despite the simplifications mentioned, Kalman filter estimation serves as a valuable starting point for the forward prediction model and is relevant to our ultimate goal of building a novel motor control model. The objective here is to establish a basic filter-controller mechanism that accurately deduces system states from noisy observations in a biologically plausible manner, and drives the system to the desired target. To achieve biological plausibility, we implement the algorithm using the NEF framework and spiking neurons. We simulate the system physics numerically and handle the representation and prediction in spiking neural ensembles. We shall evaluate the prediction system's performance in the absence of observations, adding noise, changing the noise covariance while implementing it in a motor control model to reproduce biological behaviours. It is important to note that many of these assumptions will be dropped, and a more realistic estimation framework is developed in the later parts of this thesis. For now, the realized linear prediction will be used for testing initial performance while also incorporating a neurobiologically plausible framework for realtime feedback control.

5.2 Neural implementation of a Kalman Filter

Consider a unit mass system moving on a frictionless surface, as the system to be controlled in our neural Kalman filter implementation (see Fig.5.1). The system's dynamics are described by linear state space equations:

$$\dot{x}(t) = A\,x(t) + B\,u(t) + Gw(t) \tag{5.1}$$

$$z(t) = Cx(t) + v(t) \tag{5.2}$$

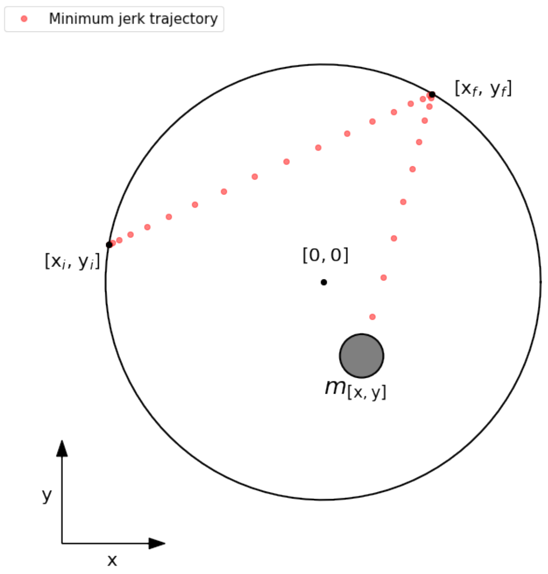


Figure 5.1: **System environment - Unit mass**: Control of a unit mass system to move to different targets on a unit circle. The grey circle represents the mass and the red dots represents the minimum jerk trajectory from one target to another, along which the mass is made to move.

where $x = \left[\mathrm{x}, \mathrm{y}, \dot{\mathrm{x}}, \dot{\mathrm{y}}\right]^T$ is the state of the system consisting of the Cartesian position and velocity, $u = \left[\mathrm{u_x}, \mathrm{u_y}\right]^T$ are the forces acting in the x and y directions and $z = \left[\mathrm{z_1}, \mathrm{z_2}, \mathrm{z_3}, \mathrm{z_4}\right]^T$ is the corresponding observation vector. The A, B, G, C matrices are the state transition, control, dynamic noise and observation matrices, respectively. $w(t)$ is the process noise, sampled from a Gaussian, zero-mean and of covariance Q_n and $v(t)$ is the measurement noise, sampled from a Gaussian, zero-mean and of covariance R_n. Furthermore, $w(t)$, $v(t)$ and $x(0)$ are chosen to be mutually uncorrelated. The A, B, G, C matrices are chosen appropriately to describe a unit mass dynamics observing all the four system states.

Given the ground state $x = \left[\mathrm{x}, \mathrm{y}, \dot{\mathrm{x}}, \dot{\mathrm{y}}\right]^T$, the objective of the controller is to move the mass, to randomly generated targets on a circle, along the shortest minimum jerk trajectory to the target. The implemented controller is a linear proportional derivative (PD) controller, given by:

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = K_p \left(\begin{bmatrix} x_r \\ y_r \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \right) + K_d \left(\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \end{bmatrix} - \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \right) \tag{5.3}$$

where the x_r , y_r and $\dot{x}_r$, $\dot{y}_r$ are the desired minimum jerk positions and velocities. The trajectory and the control gains K_p and K_d are chosen so as to drive the mass to the target within 1 second. When the mass hits a position threshold and a velocity threshold, a trial is considered complete and a new target is generated.

The observed state z from the simulation is represented in an ensemble of neurons. The estimation mechanism does not have access to the ground truth, but rather the noisy observations only. The goal of the prediction system is to provide a state estimate $\hat{x}$, given the noisy observation z, and the forces u applied on the mass to move in the desired trajectory to the target. The estimate process noise covariance Q_{est} and the estimate measurement noise covariance R_{est} (in eq. 5.4) are variables that represent the uncertainties, and can be varied for different trials to evaluate the performance of the filter. It is to be noted that the earlier Q_n and R_n are the actual noise covariances that gave raise to the simulation measurements, whereas the Q_{est} and R_{est} are perceived covariances that are used for our estimation. The covariances from the simulation and the estimation need not necessarily match with each other, since the estimation network does not have the ground truth knowledge of actual covariance even from a biological perspective. Especially it is convenient to have these as separate variables as we intend to vary the Q_{est} and R_{est} to evaluate their influence in the estimation process.

Recall the previously discussed Kalman filter estimation algorithm. The propagation of the estimate covariance is with the help of the continuous recurrent Kalman filter equation given by:

$$\dot{P}(t) = AP(t) + P(t)A^T + GQ_{est}G^T - P(t)C^T R_{est}^{-1} CP(t)^T \tag{5.4}$$

The covariance of the current belief or the estimate is given by P. From the eq. 5.4, it is clear that both the process and measurement noise covariance affect the estimate covariance propagation. Note that if Q_{est} and R_{est} are maintained at a constant value, P either increases or decreases accordingly but settles at a steady state value eventually. The rate of change of this estimate covariance is also a function of the state transition matrix A. With the help of this estimated covariance matrix, we can calculate the filter gain K using:

$$K(t) = P(t)C^T R_{est}^{-1} \tag{5.5}$$

$$\dot{\hat{x}}(t) = (A\hat{x}(t) + Bu(t)) + K(t)(z(t) - \hat{x}(t)). \tag{5.6}$$

In the neural implementation, a separate ensemble calculates the Kalman gain (K) with the help of the estimate covariance matrix (P). The neural network structure of the filter is represented in Figure 5.2. The neuron populations that represents the state estimates of position and velocity recurrently predicts the states, with the knowledge of the state transition matrix A and the efference copy of the command $u(t)$, which is sent to the plant. These two equations form the prediction part of the estimation system that propagates the system states. Given eq. 5.4 and eq. 5.5, we have a rough intuition of how the estimate covariance and the Kalman gain propagates. The estimate covariance P acts

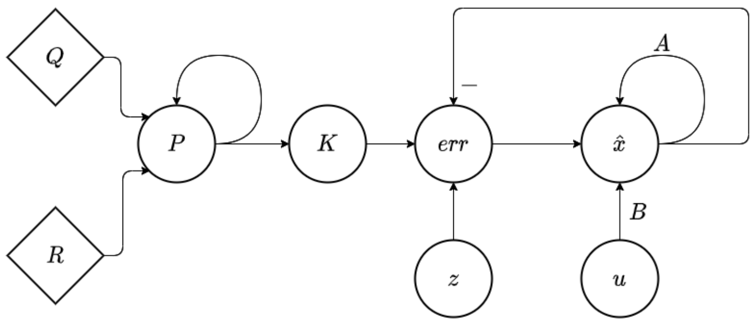


Figure 5.2: **Spiking neuron model of the Kalman Filter**: The Q and P nodes represent the the covariance of dynamics uncertainty Q_{est} and the covariance of the measurement uncertainty R_{est}. The ensemble P represents the running estimate covariance of the states and K represents the Kalman gain. The $\hat{x}$ is the estimate of the system given the calculated control u and measurement z from the simulation. The circles represents neuron ensembles while diamonds represent non-neural input nodes.

as a weight that dictates how much of the new estimate is reliant on the observation versus the estimates provided by the prediction system that propagates the previous estimate using the system model. When the process covariance Q_{est} increases (or the measurement noise covariance R_{est} decreases), P and K increase. This results in adjustment of the prediction by correcting with respect to the newly obtained observation. Conversely, when the process noise Q_{est} decreases (or the measurement noise R_{est} increases), P is reduced and so is the Kalman gain. This way, the new estimate is driven closer to the prediction than that to observation.

The measurements $z(t)$ are compared against the prediction $\hat{x}$, generating the error signal required for correction that is calculated in the *err* population. This correction is now scaled with respect to the calculated Kalman gain.

5.3 Linear Filter Results

The Figure 5.3 shows the results of the controlling of the mass on a 2D frictionless plane, with the numerical implementation on the left and the spiking neural implementation on the right. The dynamics simulation that provides the observation for both the implementations are run in python. The dynamics simulation takes in the control forces u(t) as the input and generates the ground truth position and velocity observations of the mass, with the added noise (according to Q_n and R_n). In this section, since we purely evaluate the the performance of the estimation system and its accuracy compared to the ground truth, the system dynamics are controlled by the ground truth, as op-

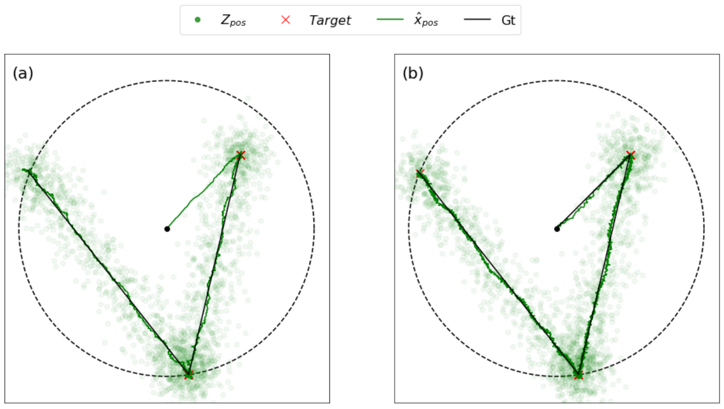


Figure 5.3: **Results of Numerical simulation vs Neural Implementation**: (a) Numerical implementation (b)Neural Implementation. The noisy observations are depicted by green dots. The Targets are represented by 'red x' and the actual position or the ground truth is depicted in solid black and the estimate is shown in green.

posed to the estimate obtained from the Kalman filter. So the system can be viewed as independent filter and controller systems, and the imperfections in the estimate do not affect the system dynamics.

The left panel in Figure 5.3(a) shows an empirically run Kalman filter model and the panel on the right 5.3(b) shows the neural implementation of the same Kalman filter. The measurement used for estimation is shown in green dots. The state estimate is initialized at origin, where $\hat{x} = [\hat{\mathrm{x}}, \hat{\mathrm{y}}, \hat{\dot{\mathrm{x}}}, \hat{\dot{\mathrm{y}}}]^T = [0,0,0,0]^T$, whereas the actual mass is randomly placed away from the origin. From the plots, it is evident that the model's estimate $\hat{x}$ (in solid green) in either simulation, falls close to the ground truth (in black). Upon observing the states of the mass, the filter quickly recovers from the offset and corrects the estimate within a few steps. This correction for error can be more clearly seen from the Figure 5.4c and the error plot in Figure 5.4. The estimates corresponds to an estimate noise covariance of $R_{est} = 0.01$ and plant dynamics noise $Q_{est} = 10$. The predictive system, despite getting a comparatively noisy observation, is able to predict the states of the system accurately. Comparing the states from both the panels, in Figure 5.4 and the error plots from Figure5.4c, the performance of the neural implementation is comparable with the empirical simulation of the Kalman filter.

There are a few differences between the results of the two implementations as seen in Figure 5.4. Although the dynamics of the state propagation is similar, it is important to note that there is additional noise in the neural implementation compared to the numerical implementation. This is due to the fact that the estimation system is running in spiking neurons, which has an inherently noisy representation due to the use of spikes. This comes from respecting the biological

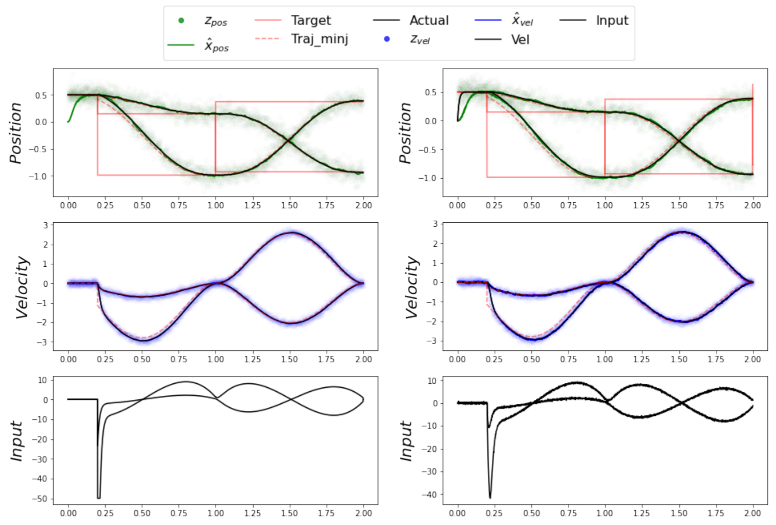


Figure 5.4: **State propagation and estimation of the filter**: The left column is the numerical simulation and the right column is the neural simulation. The figure shows the ground truth and the estimates across time. The control input is given in the bottom panel

constraints of the working of the mechanism and further helps us induce the robustness of the model against measurement noise. The slight difference in the beginning of the trial (about 0.2 sec), corresponds to the transient period in the neural simulation (initial time required for the simulation to begin), and hence the control is turned on only after the transient period is over. Within this period the initialized observation catches up with the observation comparatively quickly.

5.3.1 Performance while varying measurement uncertainties

One requirement of the model is to combine predictions with measurements. The system's estimate should consider the uncertainty in the prediction and the measurements and the Kalman gain is the optimal weighting based on the relevant noise covariances. In this section, we test the system for this property by varying the process noise covariances, and the measurement noise covariances across different runs. The resulting position estimates are shown in the Figure 5.5 and the Root Mean Square Error(RMSE) is calculated for the noise covariance sweep, given in Fig.5.6.

The top panel (both Figure 5.5 and Figure 5.6), shows how the estimate changes with an increase in Q_{est} dynamics covariance with a constant measurement covariance, and the bottom panels, shows how the model estimate changes

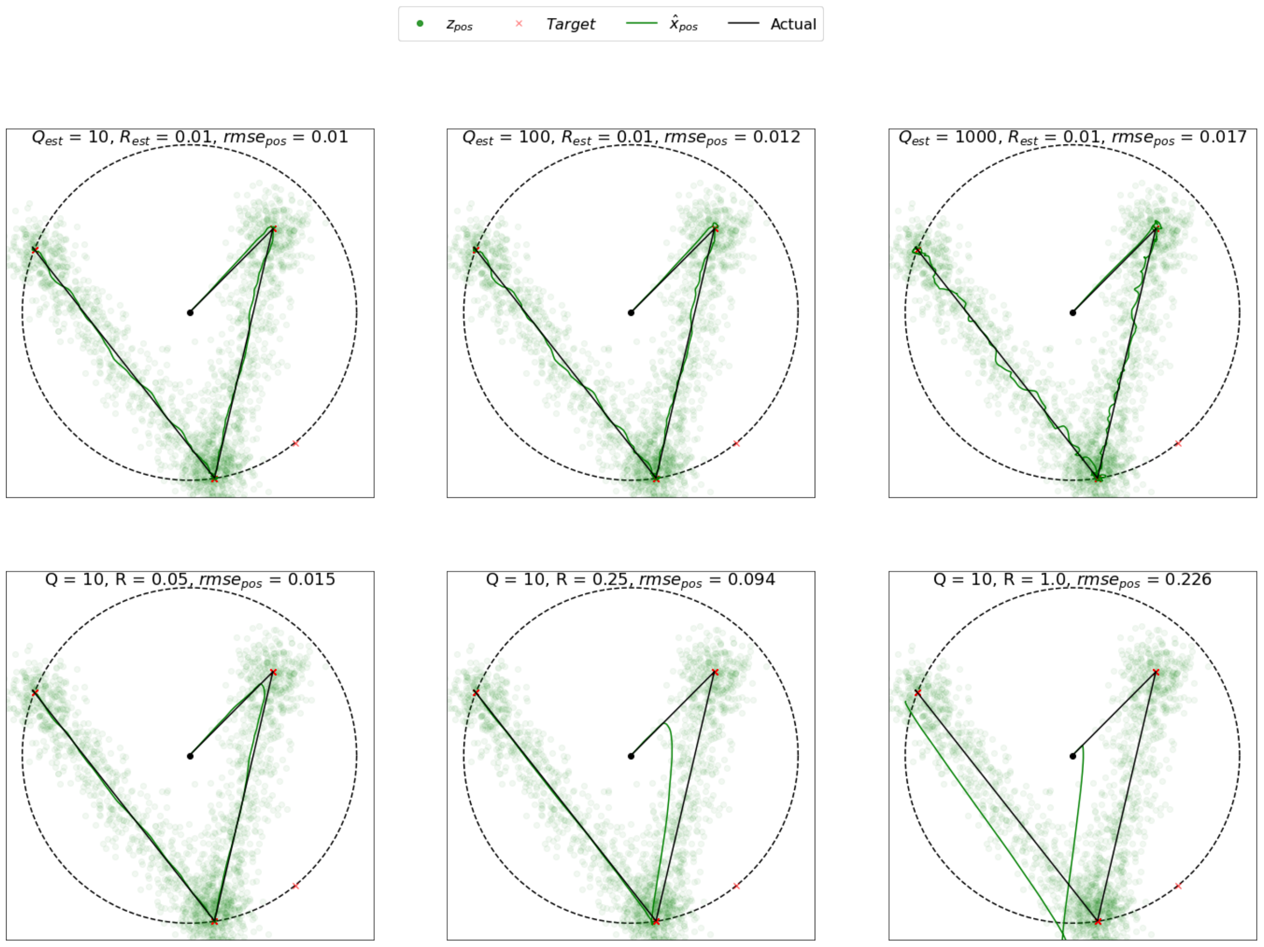


Figure 5.5: **Performance of the neural implementation of Kalman filter**: The top panel shows the performance of the model with a constant measurement noise (R_{est} = 0.01) while changing Q_{est} and the bottom panel has a constant dynamic noise (Q_{est}=10) while varying R_{est}. The state estimation is smoother and closer to ground truth when both the uncertainties are low. The filter relies on measurement more with lower R_{est} and relies more with the prediction when Q_{est} is low.

when the R_{est} measurement covariance is increased but the dynamic covariance is held constant. The increase of Q_{est} is tested for over the range of [0-1000] while the R_{est} is varied in the range of [0-1], since Q_{est} affects the derivative and the noise is scaled by a factor of dt. The RMSE is calculated for both the position and velocity after the transient period of 200ms to prevent the initial dynamics of the neurons and initiation error affecting the RMSE. Hence the RMSE error reported here is a quantitative measure of the steady state error.

The 2D position plot in Figure 5.5, with the increase in Q_{est} in the top panel, the estimate relies on the measurement more. This is also evident from the increasing trend in RMSE error in Figure 5.6. The "unit mass" nature of the dynamics is lost and the estimate looks increasingly noisy. In the bottom panel, the R_{est} is increased, indicating high measurement uncertainty. Thus the calculated gains drive the estimate to rely less on the measurements and rely more on the prediction. Especially, given that the position estimate is initialized at zero and the estimate quickly converging towards the measurements supports this inference further. As shown in the bottom panel in Figure 5.5, the system has an increased reliance on the incorrect estimate initialization, indicating a stark deviation from the ground truth. It is also interesting to point out that since the velocity estimate is closer to the ground truth, the estimated heading direction is in the same direction as the ground truth. Additionally, the error gradually diminishes, pushing the estimate towards the ground truth eventually.

The trends in the performance of the neural implementation of Kalman filter can be seen in Figure 5.6. With an increase in either noise covariance, Q_{est} and R_{est}, the RMSE error increases. There is also a considerable error difference in position compared with the RMSE error in velocity. This is caused by three different factors. Firstly, the initialization of the position is farther from the ground truth than the velocity, resulting in slow convergence of the estimate in position opposed to the velocity. Secondly, the R_n is a chosen constant of 0.01 for both position and velocity, where the magnitude of position and velocity is different (see Figure 5.4. Finally, the position vector is more sensitive to errors, since the incorrect estimate in velocity also adds into the position estimate by virtue of the system dynamics.

We now have a neural implementation of the Kalman filter that works in accordance with our requirements. The filter can effectively handle measurement noises, and optimally combines the prediction and the measurements based on the proper calculation of gains from the estimate process and measurement noise covariances. From our previous discussion of the features of our prediction and perception system, we have to implement the additional features namely:

- State estimation under absence of measurements
- Online estimates driving the control in place of the ground truth completing the feedback loop
- The prediction system actively driving the perception system, and hence facilitating adaptation

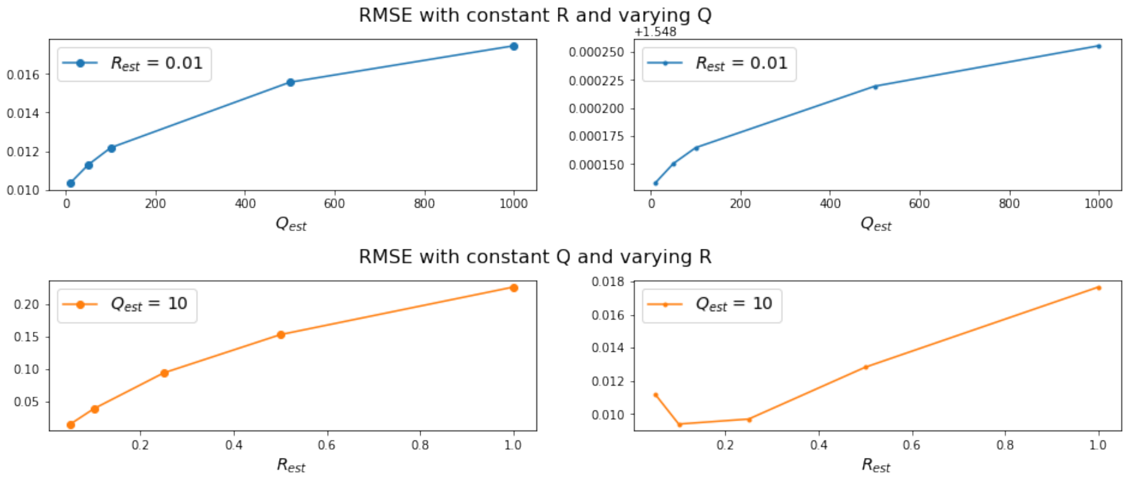


Figure 5.6: **RMSE error for different noise covariances**: The top panel shows the error change with the increase in Q_{est} with a constant measurement covariance $R_{est} = 0.01$, and the bottom panel, shows error increase with increase in R_{est} and a conostant $Q_{est} = 10$. The left panel shows the positional error and the right shows the velocity error.

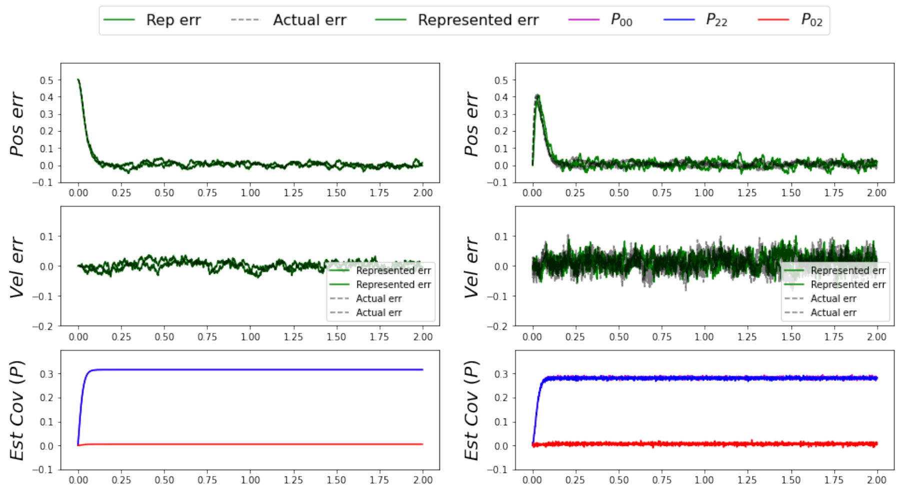


Figure 5.7: **Estimate errors and covariances**: The left column shows the error in the numerical implementation while the right column shows the error in the neural implementation. The top two panels shows the position and velocity errors in the estimate while the third shows the estimate covariance. The two implementations are comparable in their performances.

We incorporate the Kalman filter implementation in our first iteration of the sensorimotor control model. This way, we can test out the prediction system as well as incorporate the adaptation mechanisms thereby building towards a more biological plausible model.

5.4 Linear Sensorimotor control model

The sensory-motor control system deals with two streams of information to perceive the hand or the end-effector states – vision and proprioception. These two sensory inputs help the system estimate the states required for controlling the arm to accomplish tasks in the operational space. In the context of reaching, visual observation helps deduce target information, hand location, and hand velocity necessary for planning a reach. Proprioception joint angles and rotational velocities allow inference of joint locations and the transformation of end-effector forces to arm torques to follow a specified trajectory. The optimal combination of visual and proprioceptive estimates enables the brain to perceive the hand's location, which ultimately aids in control of arm reaching. To realize the entire system and incorporate biological mechanisms for prediction, perception, and adaptation, evaluating the model's behavioural implications becomes crucial. Hence, to explore the model's behavioural capability, we test its performance during the canonical behaviours of visuomotor rotation (VMR).

5.4.1 The Visuomotor Rotation paradigm

As discussed earlier (in Section 3.1.4), the task of visuomotor rotation (or VMR) is a familiar rotational perturbation experiment where a transformation is introduced in the visual feedback of the hand location. Generally, this is either induced by having the subject wear a distortion prism while reaching for different targets, or by blocking the view of the hand and providing a altered cursor feedback representing the hand (See Figure 5.8). Initially, during the baseline reaches, there is no distortion, and hence the visual and proprioceptive feedback are in agreement with each other. Hence the reaches for the targets are typical and generally head straight towards the target. After the baseline exposure, the visual consequence of the hand movement is rotated about an origin in the task space, leaving the proprioceptive feedback unperturbed. As soon the subject is exposed to this visual perturbation, there arises an incongruency between the visual and proprioceptive inference of the hand location. Subjects, while initially aiming at an offset created by directional error, within a few trials compensate for this rotational transformation by changing their motor plan. By the end of the exposure, subjects direct their hand to a different angle from the origin, thereby moving the cursor straight towards the target. What's more interesting is that when this incongruency is removed, to make the feedback to align with the actual arm location, subjects aim at the compensated angle. Now this learned compensation has to be unlearned in the washout trials, to return to the original perception and task performance.

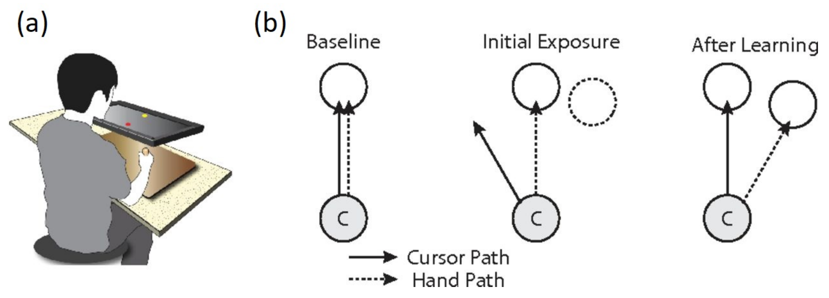


Figure 5.8: **Visuomotor rotation experiment**: (a) Experimental setup where the visual feedback is provided on the screen, blocking the actual view of the hand. (b) The baseline, initial exposure and after learning washout trials demonstrating adaptation to the visuomotor rotation of the visual feedback. The solid line shows the path of the cursor while the dashed line represents the path of the hand.

This experimental paradigm of VMR particularly aligns with our goal of capturing the intricacies of the multiple perception and adaptive mechanisms of the sensory-motor control system. Consider the architecture of the sensory motor control system as described in the Figure 5.9. Before the introduction of the rotation, the proprioceptive estimate, the visual estimate and the hand estimate are all aligned. When the combined hand estimate drives the system, it completes the feedback loop driving the arm to move to the desired location. This system on its own helps us delineate the two perceptive systems and a combined hand estimate, encapsulating the closed-loop sensory perception and control system. Once the transformation is introduced, the estimates become misaligned triggering mechanisms at multiple levels. Firstly, the visual predictive system now needs to predict differently. For the same forces applied to the arm, the visual feedback behaves differently from the previous experience and hence the predictive systems should anticipate this new change. Secondly, in many of the VMR experiments, the feedback of the cursor's position is made invisible from the start of the reach and is provided at the end of the reach. The system observes that even when the supposed hand location is on the target, the cursor is misaligned from the target. To perform the task successfully, the control system needs to learn the transformation of the movement vector such that the cursor lands on the target (and not the hand). Furthermore, in the control of the limb in real world, is a Jacobian that maps the joint angles from the proprioceptive estimate to provide the hand estimate. Since there is a misalignment, we also need to learn this kinematic transformation to perceive our hand better. These various adaptations are driven by different sources of errors, and changes in perception. This variety of modifications of prediction makes the VMR experiment a wonderful candidate to provide insights into the sensorimotor control in the brain.

While the behaviour itself is rich and informative about adaptation in the

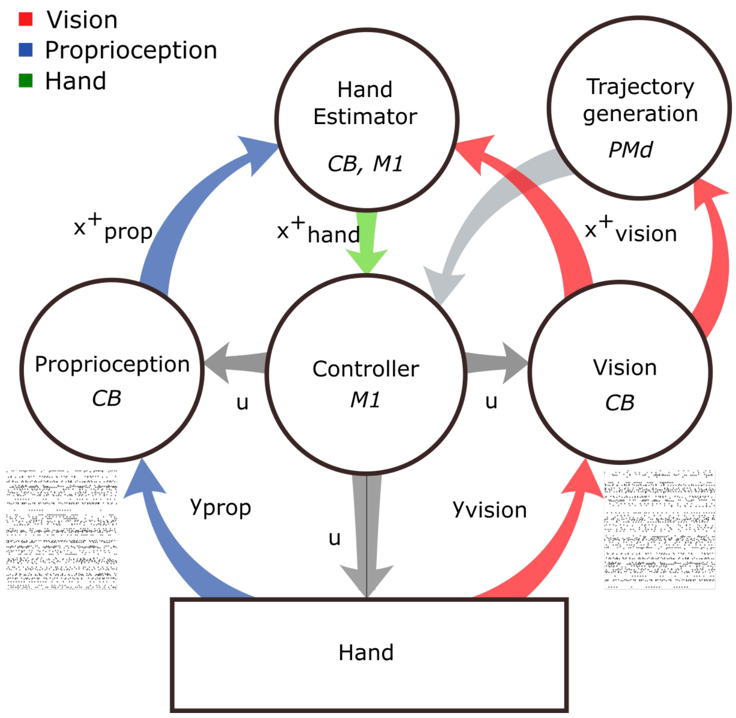


Figure 5.9: **Linear sensorimotor control model**: The hand provides the visual and proprioceptive observations. The visual predictive systems provide a visual estimate (red) and the proprioceptive predictive system provides a proprioceptive estimate (blue). CB corresponds to the cerebellum and PMd corresponds to the Pre-motor cortex. A combined movement vector is learned from the hand heading direction and correction realized from the end point error. The controller also provides rhe efference copy to the predictive systems.

sensorimotor control system, it also contributes towards the increasing complexity of the the model. This suggests that some assumptions we need to make during the building of the first iteration. In this preliminary model, we remove the complexity of working in different coordinate systems for the multiple modalities (joint and Cartesian coordinates), and hence we have approximated the arm dynamics to a unit mass linear subsystem. This relaxation is also consistent with our previous filter implementation allowing us to incorporate the Kalman filter model for the prediction of the dynamics. We also remove the adaptive prediction that learns to dynamically predict differently through the course of inaccurate observations. The prediction systems will have a constant state transition dynamics. To compensate for this, and to capture the resulting behaviour, we shall have an increased reliance on the observations, which forces the estimate to be more closer to the observations the predictions. It is necessary to emphasize, that these assumptions are for the first iteration of the model, and we remove them to realize a more accurate model in a subsequent chapter. For now, we structure the sensorimotor control system to allow the implementation of a neural realization of the prediction, perception and adaptation mechanisms to exhibit the visuomotor rotation behaviour and to have a biologically plausible working model.

5.4.2 Experimental setup

The environment in our experiment consists of a unit mass system that moves on a frictionless 2D surface, along the transverse plane and perpendicular to gravity. The mass is made to move along the surface by applying forces to reach for the targets. The simulation is run with a *dt* of 1ms. Proprioceptive observation is measured unperturbed from visual observations or rotated whenever necessary. The rotation is made at a angle of 45 degrees in a counter clockwise direction with respect to the origin. The targets appear at 8 directions around a unit circle in a pseudo-random order, and the hand is expected to make center-out reaches to these targets. A single trial is defined as the movement of the cursor, that represents the visual feedback, from the origin to the target. The reach towards the origin does not count towards the trial for the behaviour. A minimum jerk trajectory is generated so that mass lands on the target within 0.5 s. The trial ends when either the cursor lands within a position and velocity threshold or the reach time exceeds a maximum time of 1 s (reach time 0.5 s + wait time 0.5 sec). The visual feedback is provided only at the beginning and the end of the reach and is made invisible during the majority of the reach.

There are three stages to the experiment. The first 16 trials correspond to the baseline reach. During this time the visual feedback coincides with the ground truth hand location and no rotation is introduced. After the baseline stage and the initial exposure, the visuomotor rotation is introduced with a sequence of 80 trials. Following this, during the washout stage, the visual perturbation is removed constituting another batch of 80 trials. The heading direction for each of the trials is calculated when the cursor feedback appears at the end of the reach. The time course of the change of heading direction for each of the reaches is monitored across the stages to study the extent of adaptation.

The visual and proprioceptive observations are provided with the x-y position and velocity of the mass with noise as described by the Q_n and R_n for each modality. This measurement is sent to the spiking neuron model of the sensory motor control system, which gives the final movement vector that is fed back into the simulation.

5.4.3 Model description and function

To build an initial architecture of the sensory-motor control system, we implement individual prediction, perception and adaptation systems. The pictorial representation of the model is given in Figure.5.9. To incorporate visual and proprioceptive estimations, the model has two perceptual systems – vision and proprioception. The plant provides two sensory observations, y_{prop} and y_{vision}, and we implement two separate neural Kalman filters for estimating the states through each of the individual modalities. The filters generate predictions based on the built-in plant dynamics and the efference copy of the forces applied on the system u, and update the predictions based on noisy measurements. The estimate covariance and the Kalman gains are calculated based on the Q_{est} and R_{est} for each sensory modality. Each of the neural spiking Kalman filters is identical to the one described in the Section 5.1.

The estimates obtained from the two modalities are combined together to compute the combined estimate of the hand state. This combination is performed by weighting the sensory estimates with the relevant covariances of vision and proprioception systems. The resulting combined estimate of the hand is given to the controller. The controller uses a proportional derivative control to generate the movement vector that drives the arm along a minimum jerk trajectory to the target. As discussed earlier, in this simplified preliminary model of the sensorimotor control system, the only adaptation is the adaptation of the movement vector to follow a trajectory to compensate for the introduced rotation. This is learned by the PES learning rule, with hand estimates as the input feature and cursor incongruency error driving the learning. With the combined prediction, perception, and adaptation mechanisms realized in a spiking neural framework, this model describes our first iteration of a biologically plausible sensorimotor control system.

5.4.4 Results

The system is simulated for the VMR task and made to reach for the baseline, exposure and washout stages, so as to monitor the hand's heading direction and its change with the introduction of rotation.

Single reach results

Let us consider the working of the sensorimotor control model with respect to a single trial or a repeated center-out reaches. In Figure 5.9, the Kalman filter for

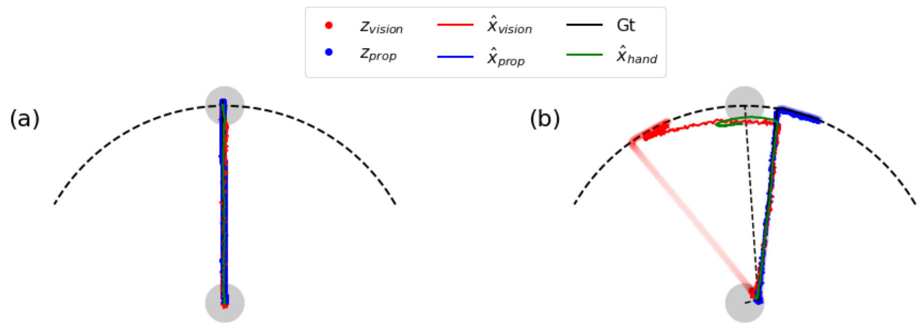


Figure 5.10: **Observations vs Estimates**: The visual (red) and proprioceptive (blue) observations and estimate are shown before and after the introduction of visuomotor rotation. (a) Before rotation all the estimates lie on top of each other during the entire trial(b) After rotation, the estimate of the vision splits after the cursor feedback. The hand estimate (green) lies in between the visual and proprioceptive estimates.

the proprioceptive system predicts the propagation of the system state based on the proprioceptive input. The Kalman filter for the visual predictive system predicts the position and velocity of the cursor visual feedback that represent the hand position and velocity. Similar to the previous results of the filter, in a single trial, accurate estimates are obtained that lie close to the ground truth of the observation (see Figure 5.10). As observed in the experimental behaviour, the proprioceptive, visual and resulting combined hand estimates of the model lies on top of each other during the baseline reaches, when there is no rotation involved. The full feedback of the cursor is available, and the estimation relies both on the prediction and the correction from the noisy state observation throughout the entire trial. During the exposure stage of the experiment, where perturbations are involved, the cursor feedback is made invisible and is provided only at the end of the reach. When the observations are not available, the error populations are inhibited so that the measurement errors are not added to the system and does not drive any of the learning mechanisms.

During the reach where visual feedback is unavailable, the estimation system completely relies on the prediction until the cursor is observed. Meanwhile the proprioceptive estimate uses both prediction and the sensory measurements, as they are always available. As shown in Figure 5.10b, we can observe that the visual estimate (in red) follows the initial observation of the cursor's movement, and the proprioceptive estimate (in blue) follows the actual hand's ground truth position. In accordance with the expected behaviour, it can also be seen that the perceived hand estimate lies in between the visual estimate and the proprioceptive estimate. This corresponds to the subject's perception of the hand location, somewhere in between the visual estimate and the proprioceptive estimate, depending upon the relative covariances of the individual modalities.

Once the cursor feedback is available, it can also be seen that the visual estimate is corrected for the new misaligned cursor observation. This in-turn changes the perceived hand's position and velocity estimate. Hence, within a single trial it is evident that the biological prediction-perception system works in harmony in generating the expected behaviour as observed in experimental results.

Across trials

The Figure 5.11 shows the trial-by-trial progression of the directional error made by the model. The dotted black line shows the rotational transformation introduced in the experiment. The first 16 trials are the baseline trials, where the visuomotor rotation is absent and the cursor feedback is available throughout the trail. During these trials, all the perceptual estimates are in congruence with each other, and the directional error is almost close to 0. From the figure, it is clear that the movement trajectories are fairly straight from the origin to the target. This describes an typical reach made with complete visual feedback.

Once the rotation is introduced in the initial exposure stage, we see a marked difference in the directional error in the experimental data. This behaviour is also reproduced by the model, during the rotation block of trials, where the initial directional error is high and the cursor hits the target by an offset. When the cursor feedback is available and the visual estimate is corrected for, we can see highly curved trajectories similar to the experimental observation. This misalignment in the cursor drives the controller to learn a new movement vector as a function of the hand estimate.

With multiple trials of rotational exposure, the model compensates for directional error by changing the trajectory and learning the additional movement vector as a function of the hand location over the course of the trials. The model exemplifies a similar rate of adaptation for the reaches similar as found in the human experimental data 5.12. Furthermore, the model's output closely aligns with the actual trajectories from the primate experiments as well 5.11. At the end of the initial exposure stage, after 80 trials, the directional error is fully compensated, and the hand's straight path is restored. This behaviour of the model is in agreement with the trajectories from the data.

After the exposure stage, a washout stage is introduced. During this stage the rotational elevation is removed and the visual feedback is now once again made to align with the proprioceptive measurements. Despite the measurements replicating the original congruence of the feedback in the baseline stage, we observe directional errors in the opposite direction to the experienced perturbation. This behaviour of the model shows that the sensory motor system has adapted to this transformation and the controller has learned to move the arm to manage the previously seen perturbations. The trend of the directional error across time is also very similar to the experimental data. By the end of the washout segment the model has completely unlearned the rotation and the trajectory is very similar to the ones that are observed during the baseline trials.

This initial implementation has now laid the foundation for a biologically

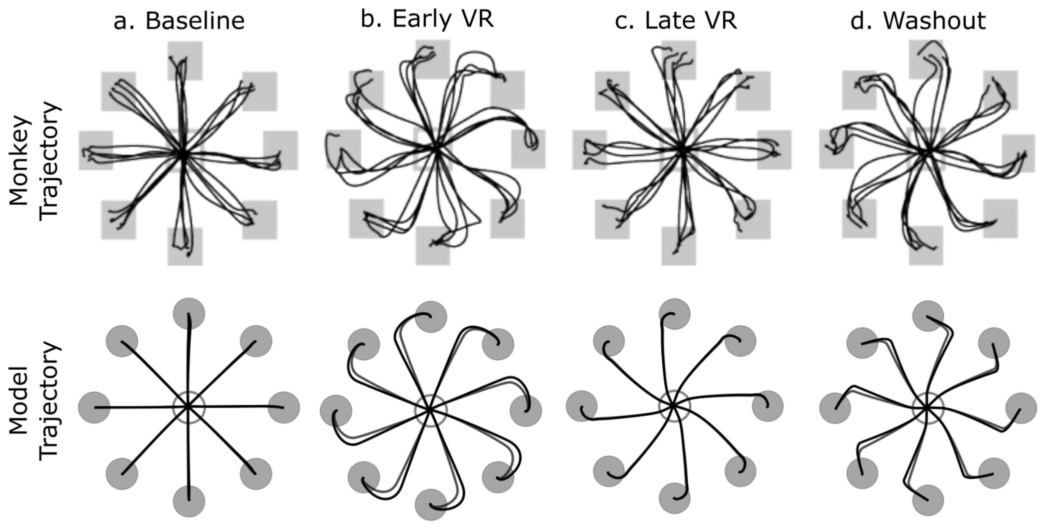


Figure 5.11: **Trajectory comparisons:** Experimental (top) and model (bottom) data during visuomotor rotation: (a) the initial baseline trials without any rotation; (b) early adaptation with onset of rotation at 30^o CCW direction; (c) later stage of adaptation to rotation (d) washout trials when the rotation is returned to 0^o.

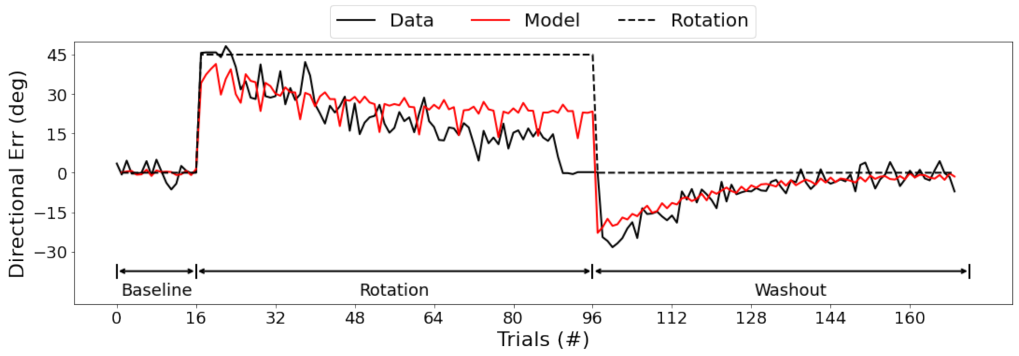


Figure 5.12: **Behaviour comparison**: Time course of directional error at threshold of the model's reaches (blue) compared against experimental data (red) (reproduced from the graphs from [Mazzoni and Krakauer, 2006]. The black line shows the angle of rotation of the cursor with respect to the hand location.

plausible framework for the prediction, perception and the adaptation systems. This model has satisfied many of the requirements that we laid out in the beginning for the sensorimotor system. Each of the prediction, perception and adaptive mechanisms are realized in spiking neural networks and each systems is modular in structure potentially helping us draw anatomical significance (see Figure 5.9). Furthermore, by reproducing the behaviour, the model is also capable of adapting to this complex sensory misalignment and is able to compensate for it in a similar fashion to that observed in the experimental data of both primates and human trials. This connection of behaviour, mechanisms, biological plausibility and anatomical significance, collectively establishes an initial, successful modelling of a biological sensory motor control system.

5.5 Limitations of a linear system model

The model developed in the previous section is able to adapt to sensory perturbation by learning to aim for a different target as the cursor appeared at a different location than predicted by the visual estimation system. Although the model is able to replicate the correction for the directional error over the trials, there are a few improvements that can be made to more realistically model the sensory motor prediction systems.

1. **Correcting for the visual change**: As soon as the rotational bias is introduced with repeated trials of incorrect visual feedback, the estimation system should predict for the change in dynamics. There is evidence for the cerebellum updating the internal model with experience of errors in predicting the consequences of the command [Imamizu and Kawato, 2009] [Wolpert et al., 1998b]. However, the visual prediction system here does not evolve over time. The system dynamics are baked into the synapses to help it predict for the given unit mass system, while the brain is known for adapting to various dynamic systems.

2. **Proper prediction of the hand estimate**: By trusting the incoming measurements more than the internal prediction we were able to get away with better visual estimates. In reality, however, we need to have a visual predictive system, which much predict the cursor's location in accordance with the previously observed dynamics of the cursor movement, in the absence of observations. This will provide a visual estimate away from the proprioceptive estimate as observed in the experimental behaviours.

3. **Accommodate for nonlinearity**: On top of continuously evolving to model the plant dynamics, the motor control system handles nonlinearities. The dynamics of the arm, any additional tool, other sensor feedback distortions often introduces nonlinearities. As a result, our prediction systems should also handle nonlinear dynamics and not merely work with linear approximation to the plant dynamics.

4. **Working with multiple modalities**: The visual and the proprioceptive streams of information are sensing different quantities, vision gives information of a virtual "end-effector" to control in the task space, while the proprioception system provides angular position and angular velocities in the joint angle space. The ideal hand states are then inferred by combining these two estimates. Hence, for proprioception we need to switch from a Cartesian space to a joint angle representation and infer the hand location by learning a Jacobian with the help of the measurements. In fact this forms the basis of another kinematic adaptation that is indeed unavoidable for this behavioural task

5. **Further anatomical parallels**: Although the above model had some anatomical significance, improvements can be made to detail the involvement of some structures. These contributions are missed out in the initial assumptions made for this first iteration. By reinstating those complexities, more cortical structures can be replicated, adding to the reproduction of the anatomical structure in the overall sensorimotor control system.

To incorporate the above improvements, in the following chapter we discuss a nonlinear adaptive filter and its performance from a perspective of the sensorimotor control system.

Chapter 6

Neural Adaptive Filter

In the previous chapter, we delved into the initial iteration of the sensory-motor control model. We modelled the system's prediction, perception, and adaptation mechanisms within a linear framework. This model offered valuable insights into the fundamental operations of the sensorimotor control system by simulating the motor pathways and identifying the states that drive adaptation. While the linear model was effective in emulating the adaptive behaviour, it struggled to capture the diverse dynamics and nonlinear characteristics found in biological systems.

In modelling the sensory motor control system, addressing nonlinearities is essential at many levels. For instance, our limbs have inherent nonlinear biomechanics. This is evident from phenomena such as, the force-length and force-velocity relationships of muscles, which deviate from simple linear predictions [Shadmehr and Wise, 2004]. Furthermore, the stretch reflex, a fundamental component of motor control, exhibits pronounced nonlinearities, especially when muscles operate at different lengths or speeds [Nichols and Houk, 1976]. The intricate interactions among muscles, tendons, and bones aside, merely the physics of controlling a multi-link arm is intrinsically nonlinear. In addition, different modalities of observations often occur in different coordinate spaces. Thus, it is clear that our system must be adept at prediction, planning, and control within the nonlinear realm. In this context, relying on linear models would be an oversimplification and can lead to significant inaccuracies in understanding the nuances of motor control. Therefore, a nonlinear approach is not just preferable, it is imperative for a comprehensive understanding of motor control dynamics.

Furthermore, the adaptability of the motor control system, a cornerstone of its functionality, was not adequately captured in the preceding model. The sensorimotor control system, at its core, is a dynamic entity that constantly interacts with an ever-evolving environment. As individuals grow and age, the system recalibrates its prediction and control strategies in response to changes in limb lengths and muscular strength. Additionally, on more immediate timescales, it must swiftly adjust to different dynamics, be it movement in water or kinematic alterations due to perceptual distortions such as mirror reflections. Sole reliance on sensory feedback during immediate distortions could easily disorient and disrupt motor actions. Yet, the system's predictive mechanism intervenes, rapidly adapting to the modified sensory feedback and fine-tuning motor commands to guarantee precise and coordinated actions. A static model, regardless of its sophistication, would become obsolete in the face of such variability. Thus, it is evident that the motor control system's intricate nature necessitates an approach that holistically embodies both nonlinearity and adaptability, highlighting the pressing need for an advanced modeling technique.

This chapter introduces the Neural Adaptive Filter (NAF), a novel nonlinear filtering method designed to address the shortcomings of the linear model, providing a more comprehensive representation of the sensory-motor control system. The NAF mechanism is tailored to predict the nonlinear system in real-time and exerts control over the plant in a manner reminiscent of biological processes. We will first delve into its workings and its integration within the NEF framework. Starting with a basic nonlinear system, we will systematically explore the filter's performance as we introduce increasing complexities, culminating in a detailed modeling of a biologically accurate sensory-motor control system.

In light of the above discussions, the filter design in this chapter adheres to specific constraints:

1. **Nonlinear Prediction and Control**: The filter is designed to predict and help control nonlinear systems.

2. **Real time stable control**: The filter processes incoming observations and controls across time. It adapts to their interactions in real-time maintaining stable control. Even when predictions are initially poor, the filter ensures consistent control until better accuracy is achieved.

3. **Quick Convergence under Uncertainty**: The filter's ability to quickly converge, especially under ambiguous conditions, is vital—since prolonged convergence might compromise control stability.

4. **Recurrent Prediction**: In instances where observations are sparse, the filter leverages its previously acquired model to generate uninterrupted predictions.

5. **Generalized Framework**: The brain's adaptability to diverse nonlinear systems necessitates a versatile filter capable of learning a range of nonlinear dynamics.

6. **Biological Plausibility**: The design should be rooted in plausible biological processes, ensuring its relevance and applicability to understanding brain mechanisms.

6.1 Method

Having established the critical constraints and requirements for our filter design, we now turn our attention to the mathematical underpinnings that will drive its design. The foundation of our approach lies in accurately modeling the nonlinear dynamics inherent to the sensory-motor control system.

Consider a non-linear dynamics of the form:

$$\dot{x} = f(x, u) \tag{6.1}$$

where x is the state of the system and u is the control input. In reality, the overall nonlinear dynamics can be a combination of any multiple nonlinear functions of the given states of the system and the control input. For example:

$$\ddot{x} = a_0 f_1(\dot{x}) + a_1 f_2(x) + a_2 f_3(u) + ... + a_n f_n(x, \dot{x}...u) \tag{6.2}$$

Here a_n are the individual weights of the different nonlinear functions f_n that describes the dynamics of the system. Assuming, the control input to the system is known, let us consider an estimator of the same form, given by:

$$\ddot{\hat{x}} = \hat{a}_0 f_1(\dot{x}) + \hat{a}_1 f_2(x) + \hat{a}_2 f_3(u) + ... + \hat{a}_n f_n(x, \dot{x}...u) \tag{6.3}$$

Similarly, the $\hat{a}_n$ are the estimated weights of the different nonlinear functions of the input states and its derivatives. Suppose, we do have access to the actual functions f_n and with the help of the observations we can calculate the different $f_n(x, \dot{x}.., u)$, then the dynamics of this nonlinear estimator can be rewritten in a state space representation. We linearly split the terms into products of a set of constant parameters and a set of known functions of the system states and the control input.

$$\begin{bmatrix} \dot{\hat{x}} \\ \ddot{\hat{x}} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & \hat{a}_0 & \hat{a}_1 & \hat{a}_2 & \cdots & \hat{a}_n \end{bmatrix} \begin{bmatrix} \dot{x} \\ f_1(\dot{x}) \\ f_2(x) \\ f_3(u) \\ \vdots \\ f_n(x, u) \end{bmatrix} \tag{6.4}$$

$$\boxed{\dot{\hat{\mathrm{x}}} = \hat{\theta} \,.\, Z(f(\mathrm{x}, \dot{\mathrm{x}}...u))} \tag{6.5}$$

where $\mathrm{x} \in \mathbb{R}^{n\times 1}$ is the state vector comprised of $x, \dot{x}$, $\theta \in \mathbb{R}^{n\times d}$ is an unknown estimated matrix of constant parameters and $Z(f(\mathrm{x}, \dot{\mathrm{x}}...u)) \in \mathbb{R}^{d\times 1}$ is a known vector containing functions of the estimator states and the control input, where n is the number of estimator states and d is the number of basis functions. Note that this formulation allows for multiple interacting x's in x. The parameters θ and the functional bases $Z(\mathrm{x}, u)$ would have corresponding entries in the columns. By comparison, the actual state dynamics can be rewritten as:

$$\dot{\mathrm{x}} = \theta \,.\, Z(f(\mathrm{x}, \dot{\mathrm{x}}...u)) \tag{6.6}$$

Now, the error ε is defined as the difference between the actual and the estimated system, given by:

$$\varepsilon = \mathrm{x} - \hat{\mathrm{x}} \tag{6.7}$$

When the estimated parameter matrix $\hat{\theta}$ is equal to the actual system's θ, then we have an ideal estimator. With this formulation, the estimation becomes a problem of regression, i.e., of finding the linear combination of the nonlinear functions of states and control. Again, it is important to note that until now the estimator is a function of the states and its nonlinear functions.

The objective is to minimize the squared error:

$$\begin{aligned} J(\hat{\theta}) &= \frac{1}{2}\varepsilon^T\varepsilon \\ &= \frac{1}{2}(x - \hat{x})^T(x - \hat{x}) \end{aligned}$$

To find the optimal *theta*, we can continue to update our parameters by:

$$\hat{\theta}^* = \arg\min_{\hat{\theta}} J(\hat{\theta})$$

This can be solved using gradient descent or any optimization algorithm on-the-fly from the observations. The update rule for gradient descent is:

$$\boxed{\hat{\theta}_{k+1} = \hat{\theta}_k - \alpha\nabla J(\hat{\theta}_k)} \tag{6.8}$$

where α is the learning rate and $\nabla J(\hat{\theta}_k)$ is the gradient of the cost function with respect to $\hat{\theta}$ at iteration 'k'. With more observations, we can continue to update our θ.

To implement the suggested optimal estimator, we face two fundamental challenges. First, our initial approach assumes the availability of observation derivatives. However, in practice we typically have access only to the state information, denoted as x, and not its derivatives. The second challenge pertains to the details of the nonlinear functions. While we might possess some prior knowledge about the choice of dynamics, the exact list of functions, f_n, that capture the system's behaviour is unknown. A potential solution might involve selecting a functional basis tailored to a specific system of choice based on prior knowledge. However, this approach is untenable for our purposes, as our estimator must possess the flexibility to adapt and learn new nonlinear systems. Given the mutable nature of the nonlinear dynamics that we aim to capture, the system could be described by a potentially infinite set of nonlinear functions of the system states and control. Consequently, to determine the parameters for optimal estimation, the function set $Z(\hat{x}, u)$ must be broad and adaptive enough to model the system dynamics faithfully. Addressing these challenges is crucial for the creation of an effective filter to predict evolving nonlinear dynamics. Solutions to these issues will be explored in the subsequent sections.

6.1.1 Moving to the Legendre space

We need to represent the observations and control inputs through time and find a workaround to obtain the state derivatives. With this goal, the Legendre representation can help us in approximate the derivatives and functions of the state. Specifically, the Legendre Delay Network (refer to section 2.4) orthogonalizes the continuous-time history of its input signal, $x(t) \in \mathbb{R}$, across a sliding window of length $\theta \in \mathbb{R} > 0$ decomposing into its q constituent Legendre polynomial coefficients m.

$$\dot{m}(t) = Am(t) + Bx(t) \tag{6.9}$$

$$A = \begin{bmatrix} -v_0 & -v_0 & \cdots & -v_0 \\ v_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & v_{q-1} & 0 \end{bmatrix}, \quad B = \begin{bmatrix} v_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{6.10}$$

The state $m \in \mathbb{R}^q$ represents more information than just the input θ seconds ago. It represents the state at every point in time up to θ seconds. In fact, m represents the function $f_{[t-\theta,t]}\left(\frac{\theta'}{\theta}\right) \approx x\left(t'-\theta'\right)$ in the Legendre basis

$$x\left(t'-\theta'\right) \approx f_{[t-\theta,t]}\left(\frac{\theta'}{\theta}\right) = \sum_{i=1}^{q} \tilde{P}_i\left(\frac{\theta'}{\theta}\right) m_i(t)\,. \tag{6.11}$$

This network implements an optimal recurrent neural network remembering a slice of the past, or a so called "reservoir". By representing the θ window of the observation and control, we split the signal into its corresponding polynomial entities, and from that can compute its time derivatives.

$$m_t = [m_t^{[1]}(x_{[t'-\theta']}), m_t^{[2]}(x_{[t'-\theta']}), ..., m_t^{[q]}(x_{[t'-\theta']})] \tag{6.12}$$

From the polynomial approximates, we can decode the state x at any given point t within the θ window using a simple linear combination:

$$x_t \approx Cm_t \tag{6.13}$$

Taking the derivative on both sides,

$$\frac{d}{dt}x_t \approx C\frac{d}{dt}m_t \tag{6.14}$$

$$\frac{d}{dt}x_t \approx C(Am_t + Bx_t) \tag{6.15}$$

$$\Delta x \approx \frac{x(t) - x(t-\Delta t)}{\Delta t} \approx \frac{C_{[\tau=t]}m - C_{[\tau=t-\Delta t]}m}{\Delta t} \tag{6.16}$$

Hence, with the help of the collective polynomial coefficients m_t we now have an approximation of the time derivatives of the input signal x. Essentially, the memory vector m_t not only has the compressed representation of the signal across the window to compute x, but also contains the approximate time derivatives that we require. Therefore, with an appropriate choice of decoders, it is possible to linearly decode $x + \Delta x$, the next state ahead in time. Hence, projecting both the states as well as the control signal into the Legendre space, we get:

$$M_t = [m_t(\mathrm{x}) \,|\, m_t(u)] \approx [\mathrm{x}, \dot{\mathrm{x}}, ..., u, \dot{u}] \tag{6.17}$$

$$\dot{\mathrm{x}} \approx \theta \,.\, Z[f(M_t(\mathrm{x}, u))] \tag{6.18}$$

By moving to the Legendre representation, we have essentially created a method to address the challenge of obtaining the derivatives. As the number of Legendre polynomials in M increases, we get closer to the true derivatives. Now the remaining challenge is to make sure $Z(M)$ is sufficiently expressive enough to capture the nonlinearities in state propagation, and can support the continuous learning of the parameters to move towards an ideal estimator.

6.1.2 Neural Implementation

The focal challenge now is to deduce a nonlinear function of the input that mirrors the dynamics of the observed system. The NEF stands out as an optimal solution for this task. Our previous forays into the NEF framework underscored its ability to represent and decode nonlinear state transformations. Moving to the neuronal space brings about a distinctive advantage. Earlier, selecting the number of basis functions necessitated specific knowledge of the dynamics being approximated. A more advantageous approach would bypass this need, and employ a diverse set of simple basis functions to comprehensively cover the state space, ensuring any relevant function can be approximated. The transition into the neuronal space equips us with a means of approximating an extensive array of functions with precision.

The goal here is to learn a set of decoders d that can accurately approximate the actual θ of our system dynamics, such that the prediction $\hat{x}$ is close to the the x, eventually driving the error ε to zero. The learning rule for updating the learned parameters $\hat{\theta}$, is responsible for minimizing the approximation error. The neuronal activation contains the nonlinear functions of the state of the system, given by $G(M)$. A set of decoders d_t can be generated to learn the parameters θ to predicts our $\hat{x}$.

$$a_t = G\left(M_t(x, \dot{x}, u)\right) \tag{6.19}$$

$$\hat{\mathrm{x}}_{t+\Delta t} = d_t a_t \tag{6.20}$$

We here use the PES learning rule (see section NEF) for learning the appropriate decoder that would give us the ideal $\dot{\hat{x}}$. This is comparable to a stochastic gradient descent, but operating iteratively online on a single sample available instantaneously. As we compare the current estimate and available observations, we then update the weights using the following learning rule:

$$\Delta d_t = \kappa \epsilon_\tau a_\tau \tag{6.21}$$

$$\epsilon_t = \mathrm{x}_t - \hat{\mathrm{x}}_t \tag{6.22}$$

Where κ is a positive constant learning rate, ϵ_τ is the error at the current instance τ. Given we use sufficient number of basis functions and we sweep though a rich enough sample space, we can see that the parameter regressor matrix for the approximate functions of the derivatives we sought in the first place:

$$\dot{\mathrm{x}} \approx \theta \,.\, Z[f(M_t(\mathrm{x}, u))] \tag{6.23}$$

which is a close approximate of the ideal estimator.

It is important to note that the estimator directly uses the observation during the learning of the parameters θ. Once these parameters are learnt, the estimated state $\hat{x}$ can be recurrently sent to the system where initially the ground truth of the system states were sent. This way the estimator can predict on its own from its previous state prediction in the absence of observations.

6.2 Architecture of the Neural Adaptive Filter (NAF)

Figure 6.1 illustrates the architecture of the filter. System state observations are denoted by z, while the system's state estimate is represented as $\hat{x}$. Both the instantaneous observation z and the previous estimate $\hat{x}$ are input into the 'ctxt' or selective context population. This population can choose to estimate the current state either from the observation or the previous estimate. When observations are available, each state z is projected into the Legendre representation by its respective Legendre Delay Network: Q_x. Concurrently, the delay network Q_u transfers the control input to the Legendre space, resulting in the combined Legendre representation M_t, which encompasses the windowed representations of both states and the control signal. The A and B matrices are chosen based on the choice of window size θ_x, θ_u and the number of Legendre polynomials q_x and q_u to represent the signal. Depending upon the system, each state x and control u gets its own Q to represent the signal.

This combined representation, M_t, is then channeled into the 'adapt' population, where the interaction between states and control takes place. Here, the signal is projected into a neural population, providing the representational bases necessary to learn the system's dynamics. The estimation process involves comparing the estimate $\hat{x}$ with the instantaneous observation z. The resulting error, represented as ε in prior equations, guides the filter's learning process to refine the estimate $\hat{x}$. It is crucial to clarify that the objective is state estimation, not merely one-step-ahead prediction. While the terms "prediction" and

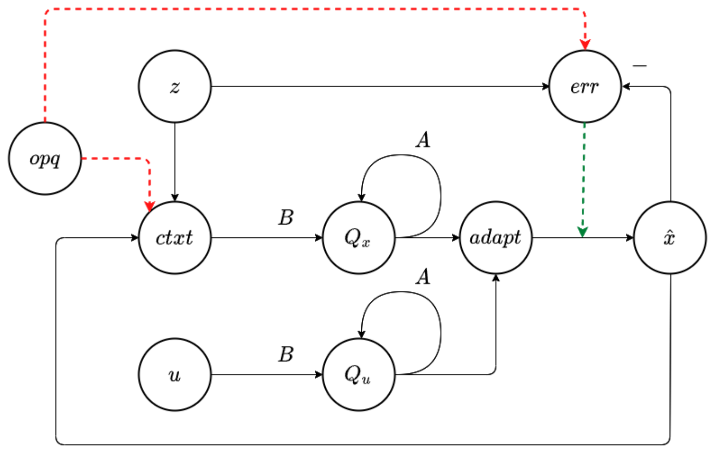


Figure 6.1: **Neural Adaptive Filter's Architecture**: A schematic representation of the filter's components and signal flow, where instantaneous observations and previous estimates guide state predictions. The z represents the observed state and the $\hat{x}$ represents the state estimate. The state and the control each has its own LDN given by Q_x and Q_u. A and B corresponds to the matrices for building the LDN for a given window size and number of polynomials. The opq lines, depicted in red, selectively control the learning process, while the learning connections from the err population are highlighted in green.

"estimation" might seem synonymous, in an ideal setup, the 'adapt' population derives the current state from the windowed observation representation. When the current estimate $\hat{x}$ aligns with the observation z, the 'err' population output is zero.

The 'opq' component modulates the operations of the 'ctxt' and 'err' populations based on observation availability. When observations are present, 'opq' is set to 1, and when absent, it is 0. The value can range anywhere between 0 and 1. This mechanism enables the filter to determine the source of prediction at any given moment. During training phases with available observations, the system learns to predict from these observations, prompting the 'ctxt' population to utilize z. Conversely, in testing phases or situations with uncertain or missing observations, the previous state estimate $\hat{x}$ is processed by Q_x for recursive prediction. The 'opq' also regulates the error population based on the context, as learning from estimates alone isn't ideal. The resulting estimate $\hat{x}$ then controls the plant in real-time.

In essence, these components and interactions constitute the Neural Adaptive Filter (NAF).

While the NAF deals with complex nonlinear systems, it is important to reiterate the choice of the working mechanism, especially towards keeping feedback delays in mind. The NAF with the inherent use of the Legendre representation, captures a windowed representation of the relevant state variables. Even with a case of a simple system working with feedback delay, when a snapshot representation through time isn't available, any method that works with an assumption of time alignment will fail to learn the dynamics. Thus, this choice of mechanism can potentially help us deal with incoming state observations that are not aligned in time. Initial tests with this scenario showed potential for handling delays upto 20ms in the damped pendulum system where conventional control failed to control the plant in a stable manner (results not shown). Although the feature of handling delay is not explored in this thesis, and is a work for the future, it is important to highlight that the NAF for now has not only added value towards nonlinear estimation through comparable performance with its predecessors but is also setup for potential improvements.

6.3 NAF Performance

In this section, we methodically evaluate the filter's performance by progressively introducing complexities in the test scenarios. Starting with the Lorenz attractor—a classic nonlinear chaotic system without control input—we progress to the damped pendulum system, where torque control comes into play. Culminating our tests, we delve into a two-link arm system, observed in Cartesian space but controlled by torque control. For each of these systems, we will discuss the construction, training, and testing of the filter. Furthermore, to position our Neural Adaptive Filter within a broader landscape of dynamical predictive systems, we will compare its performance against the state-of-the-art benchmarks at the end of the section.

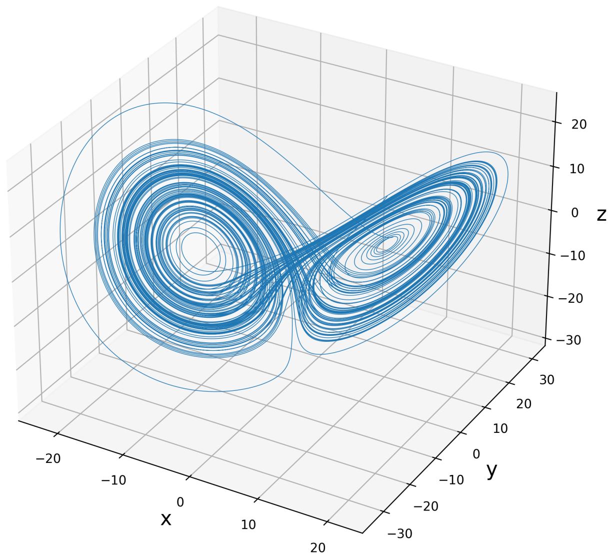


Figure 6.2: **Lorenz Attractor in 3D Space**: A visualization of the iconic chaotic system, showcasing the intricate trajectory patterns in 3D space. Chosen values: $\sigma = 10$, $\beta = \frac{8}{3}$ and $\rho = 28$

6.3.1 Lorenz Attractor

Consider the Lorenz system, a canonical model of nonlinear dynamics derived from simplified atmospheric convection models. Widely acknowledged in scientific literature for its chaotic behaviour, the Lorenz attractor is sensitive to initial conditions, leading to rapid divergence—a defining trait of chaotic nonlinear systems. Given its lack of a forcing function and this sensitivity, it serves as an apt test-bed for evaluating the Neural Adaptive Filter. The dynamics of the system is described by:

$$\dot{x} = \sigma(y - x) \tag{6.24}$$

$$\dot{y} = x(\rho - z) - y \tag{6.25}$$

$$\dot{z} = xy - \beta z \tag{6.26}$$

Here the x, y, z are the states of our system and the coefficients σ, ρ and β are fixed constants. Typical values of $\sigma = 10$, $\beta = \frac{8}{3}$ and $\rho = 28$ are chosen for the simulation. See the figure 6.2 for a visualization of this chaotic dynamics.

Method

We assessed the Neural Adaptive Filter's performance by learning the Lorenz system using a Python-developed simulation environment, with the filter implemented using NEF. The simulation runs at a 1ms time-step, and states are

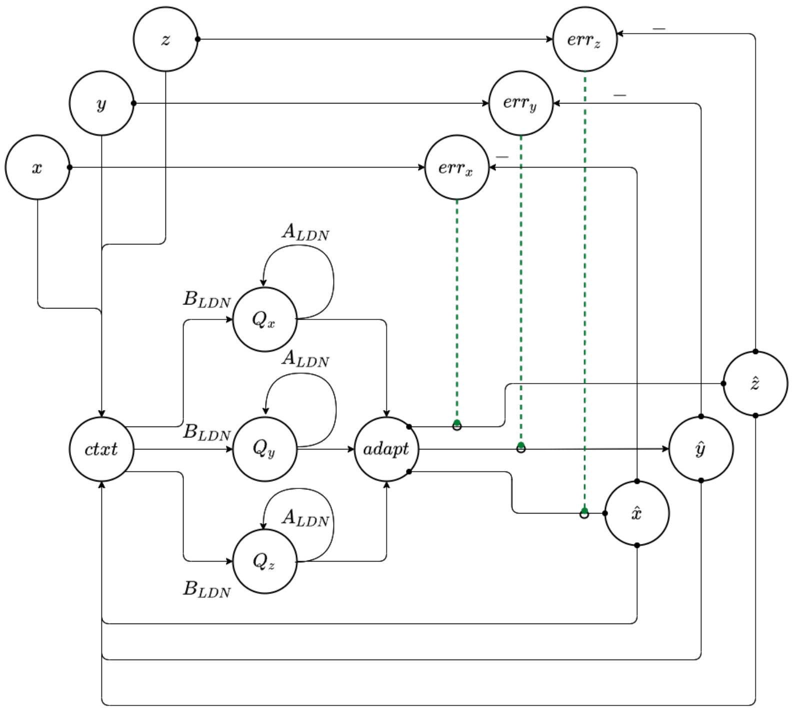


Figure 6.3: **State flow diagram illustrating the Neural Adaptive Filter (NAF) implementation for the Lorenz system:** The diagram showcases the observed system states $[x, y, z]$ and the estimates $[\hat{x},\ \hat{y}\ \hat{z}]$. Each state is processed through its respective Legendre Delay Network (LDN), resulting in three distinct LDNs for each state. The A_{LDN} and B_{LDN} are the state transition matrices for the LDN. For simplicity, the 'opq' object, which typically regulates information flow, is hidden in this representation

normalized to [-1, 1] before input. We use spiking neurons to construct the estimation network, enabling on-the-fly learning. A ‘seed’ parameter introduces variability in initial conditions of both the dynamic system and the neural population.

The Lorenz system comprises three states. Each state is fed into its respective Legendre Delay Network (LDN) to derive the Legendre polynomial coefficients M (as depicted in Figure 6.3). These coefficients, which encapsulate a windowed representation of the states, are then relayed to a neural ensemble. This ensemble is tasked with discerning the nonlinear interactions inherent to the states. The ‘ctxt’ node, governs the data relayed to the LDN node Q_x, facilitating the learning process. During the training phase, state observations $[x, y, z]$ are input into the network, allowing the ‘adapt’ population to discern the relationship between input states to estimate $[\hat{x},\ \hat{y}\ \hat{z}]$. In the testing phase, however, we shift the ‘ctxt’ node’s function to rely on the predictions from previous timestep to estimate recursively.

Each Legendre Delay Network, responsible for projecting the input into a higher-dimensional space, operates based on two hyper-parameters: the memory’s time window θ and the number of polynomials q encoding this window. Given the three distinct states, we are presented with six hyperparameters. For this study, both the window size and polynomial count were kept consistent across all states. The selection of θ and q is contingent upon the specific system under approximation. As a general guideline, dynamics with pronounced high-frequency content necessitate a greater polynomial count for accurate approximation. For this implementation, we set $\theta = 0.05$ seconds and $q = 5$ for optimal performance.

The filter’s exposure to more observations prompts iterative refinements in its estimates, drawing them closer to the actual ground truth. The overarching objective is real-time system state observation, coupled with on-the-fly system learning of the states. The training duration was set at 1500s (final 5 secs of testing). Post-training, a testing signal is introduced gradually, reducing the weight of observations and amplifying the weight of estimates, thereby facilitating recurrent estimation. This gradual transition ensures a seamless shift from training to testing, as represented by:

$$\begin{aligned} context &= (1-\alpha)\, z + \alpha\, x \\ err &= (1-\alpha)(z-x) \end{aligned}$$

Here, α denotes the testing signal value. It is 0 during training and switched to 1 during testing. This signal also modulates the error used for the learning mechanism. The onset of the testing signal corresponds to a reduction in learning. It is important to note that this signal helps in the smooth transition of the model to start using the estimations recurrently.

Results

The Figure 6.4 illustrates an instance of the Lorenz system's state propagation (dotted black) and the estimation from the NAF (in green). Initially, the error from the filter is poor but diminishes over time, aligning closely with the observation by 1500s. It can be seen that when the testing is turned on gradually and the system starts to recurrently predict using its previous estimate, it continues to stay close to the ground truth well into the training phase. Despite the inherent noise in neuron approximations, after the observation is removed, the learned filter's estimates maintain a trajectory akin to the system states and qualitatively continues to stay in a typical trajectory for a Lorenz system. This trajectory evolution underscores the filter's progressive learning of the system dynamics (see Figure 6.6). Since the system is chaotic, with a slight offset in the prediction, it eventually diverges from the actual ground truth.

In order to quantify the the error between the estimate and the ground truth different metrics were considered and the Integral of Absolute Error (IAE) was chosen to be an appropriate metric as it is widely used for evaluating nonlinear systems. Traditional error metrics such as Root Mean Squared Error(RMSE) can give misleading results because of its sensitivity to large errors and invariance to error across time. More importantly for a chaotic system, it is essential to know how far in time the estimate stayed close to the ground truth as time progresses. IAE offers a holistic view of performance over time by accumulating the absolute error. This distinction and the choice of metric will be better justified in the forthcoming section when control is involved. The IAE of the prediction is calculated by:

$$IAE = \int_0^T |e(t)|dt \tag{6.27}$$

$$IAE_{ttt} = (t >= thresh) \tag{6.28}$$

where e(t) is the error between the estimate and ground truth. IAE_{ttt} is the 'time-to-threshold' value which refers to the time in seconds required for the integral absolute value to reach the chosen threshold. The threshold is a fixed constant value that is arbitrarily chosen to indicate when the error hits a specific limit for the given dynamics. In this scenario for the Lorenz system, a threshold of 0.05 is used.

To clarify, for two signals that are highly different, this IAE_{ttt} value would be closer to 0 indicating quick accumulation of error, and for two identical signals IAE_{ttt} gets closer ∞, since the error is close to zero and hence it takes $t \to \infty$ to indicate divergence.

Across multiple runs (N=5), the IAE_{ttt} for the three states were [1.405, 1.252, 1.293]seconds with standard deviations of [0.099, 0.067, 0.076]seconds. For instance, this metric can be read as: the estimation error for state x reached a threshold of 0.05 at 1.405 seconds once testing is turned on. Notably, this metric quantifies the filter's predictive power, rather than its performance during observation. In fact, from Figure 6.4, there is very little to no error between

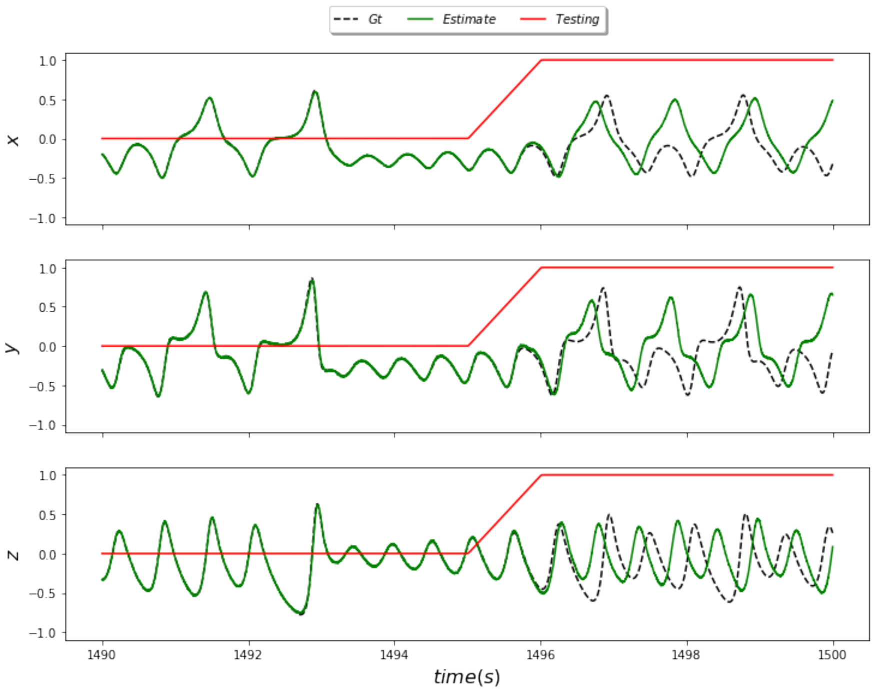


Figure 6.4: **Time series plots of the three states of the Lorenz attractor, x, y and z**: The black dashed lines represent the ground truth (GT) of the system, while the green lines depict the estimates from the Neural Adaptive Filter. The plots provide a comparative view of the filter's estimation accuracy against the actual dynamics of the Lorenz system over time. The red line shows the transition from observations being provided to pure prediction. Once the observation is removed, the system is qualitatively similar, but drifts from the GT. The states are normalized between [-1, 1]. A population of 1000 neurons was used for the estimation.

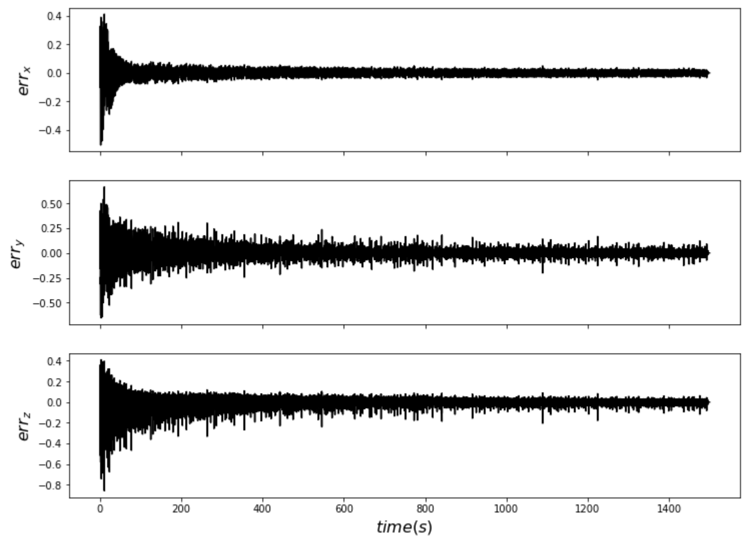


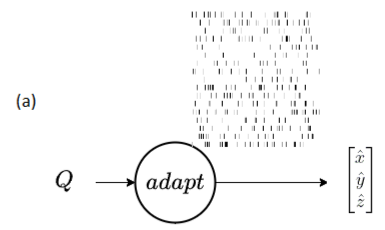
Figure 6.5: **Estimation error across time.** The x-axis is the simulation time and the y-axis shows the error. The three panels correspond to each of the three states of the Lorenz Attractor. The estimation error reduces with more learning.

the prediction and ground truth by the end of 1500s when the observation is present. Furthermore, after training, reintroducing observations realign divergent predictions with the filter's pre-test accuracy (see Figure 6.7).

The plots highlights the effect of the Lorenz system's inherent chaos. Minor prediction deviations amplify over time, causing increasing divergence from the true value. This dynamic showcases the filter's ability to model chaotic nonlinearities, rather than simply replicating an integrator or communication channel. Were it only relaying state information, the decoding during testing would stagnate at the last observed state. Similarly, if merely approximating an integrator, accumulating errors would quickly lead to system saturation. However, the exhibited oscillations, characteristic of a Lorenz system, highlight the filter's ability to capture the system dynamics.

It should also be noted that the gradual switch from training to testing phase is not critical to good performance. The effect of the gradual switch can be appreciated more in the forthcoming plants with control.

In sum, the results demonstrate the filter's ability to predict the Lorenz system dynamics, even with its chaotic nature. Given the manner we have set up this Lorenz system scenario, irrespective of poor predictions, the ground truth propagation remains uninfluenced by the estimate produced. This is useful as we were able to scrutinize the filter performance by turning off the observations. Now we shall increase the complexity to further test the filter performance by using it in two control systems: a damped pendulum, and a two-link arm.



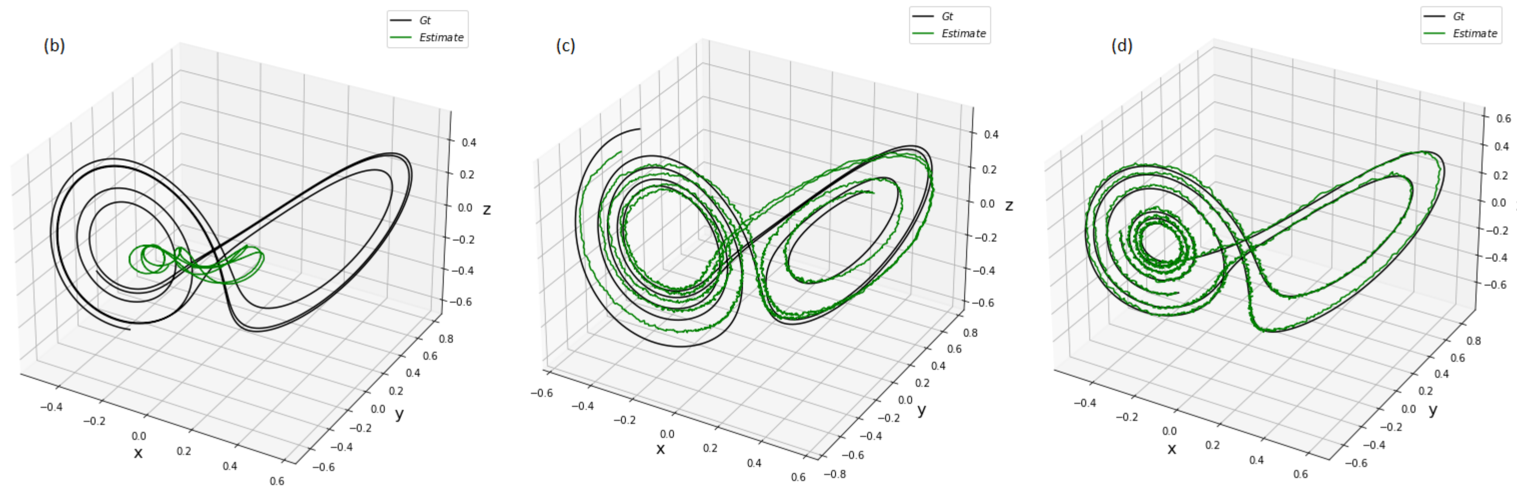


Figure 6.6: **Filter Estimation in 3D through time**: The top panel (a) shows that the filter is learning to predict the system states from a population of spiking neurons. The bottom panel shows the filter's estimation (in green) during (b) initial 10 s of training (b) after 150 s of training (d) after 1000s of training. During the first few seconds, the predictions are really poor and with more training data, the prediction gets better and by the end of training, the estimation is close to the ground truth (in black).

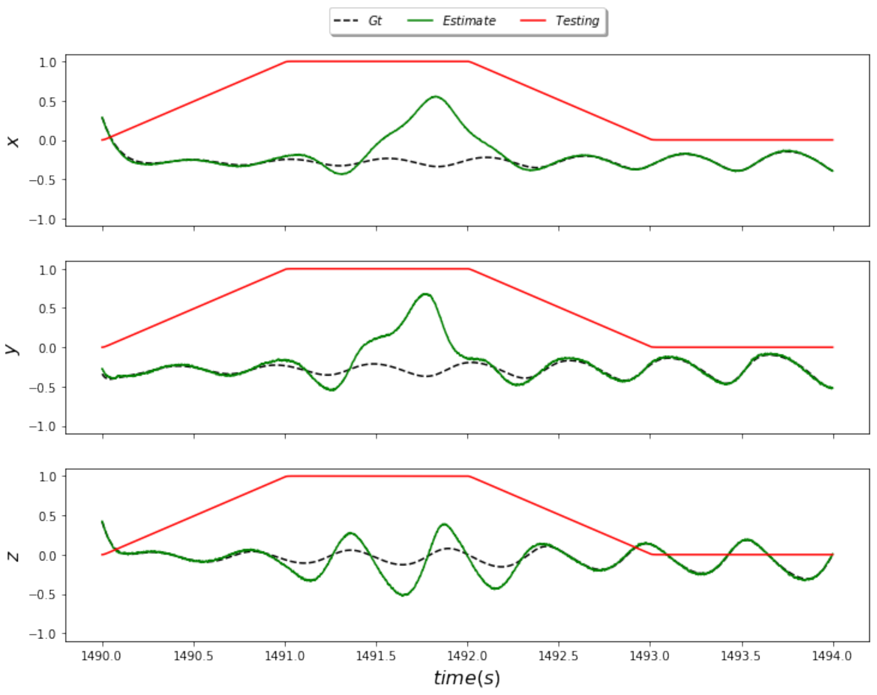


Figure 6.7: **Trajectory after observation reintroduction**: The estimations diverge from the ground truth (in dashed black) when testing (red) is turned on and no observations are given. As the observations are reintroduced by turning off the test signal, the estimations realign with the observations with the filter's pre-test accuracy.

The benchmark comparison for all the systems are performed collectively after specifying the dynamics.

6.3.2 Forced Damped Pendulum

Transitioning from the Lorenz system, an open-loop nonlinear model without control inputs, we now shift our focus to a more challenging system that incorporates external forces. We delve into the dynamics of a forced damped pendulum, commonly referred to as a *one link arm.* This progression allows us to evaluate the Neural Adaptive Filter's performance in scenarios where control inputs play a pivotal role, adding another layer of complexity to the estimation challenge.

The dynamics of the pendulum is given by:

$$J\ddot{q} - b|\dot{q}|\dot{q} - mgl\,sin(q) = u \tag{6.29}$$

where m, l and J are the mass, length and inertia of the pendulum, g is the gravitational term and b corresponds to the damping coefficient. u corresponds to the input torque. The nonlinearities here arise from the trigonometric and damping terms.

Methods

The general working of the pendulum is as follows. The pendulum is initialized at a random angle from [-$\pi/2$, $\pi/2$] from the vertical at zero velocity. A random target is chosen and a minimum jerk trajectory is generated to reach for the target with an average velocity of 1rad/sec. Upon reaching the target within the prescribed threshold, or after a wait time, a new target is generated. The observed states are the angle (q) and the angular velocity ($\dot{q}$) of the pendulum. The estimator now has to predict the location and the velocity of the pendulum and the resulting estimates $[q, \dot{q}]$, are sent to the controller to move the pendulum along the desired target trajectory. A unit mass, length, and damping coefficients are chosen, and the gravity is set to 10 m/s^2. For the sake of simplicity, the controller is designed to compensate for the gravitational term. It is important to note that, this simplification while reducing the complexity on the controller side, does not negate the impact of the gravity on the pendulum's state propagation. The controller still has to compensate for gravity, but it is baked in rather than learning to compensate.

Similar to the Lorenz system evaluation, the simulation for the damped pendulum system is executed in Python with a consistent time-step of dt = 1 ms. The states of the pendulum, namely the angle (q) and the angular velocity ($\dot{q}$), are normalized before feeding them into the filter. A 'seed' parameter, allows for variability in initial conditions of the dynamic system and the initialization of neural population weights. A typical proportional derivative controller is used ($K_p = 100, K_d = 10$) to follow a minimum-jerk trajectory.

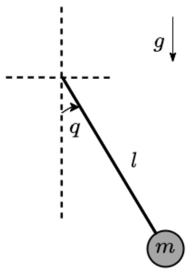


Figure 6.8: **Forced damped pendulum dynamics**: The above diagram illustrates the dynamics of a damped pendulum. The pendulum consists of a bob with mass m suspended from a pivot by a rod of length l. The angle q represents the pendulum's displacement from the vertical. Gravity acts downward, influencing the pendulum's motion.

The filter's implementation for the pendulum system closely resembles that for the Lorenz attractor. However, an essential modification involves incorporating a control input to influence the pendulum based on estimated states. Two Legendre Delay Networks (LDNs) represent the states, while a separate LDN handles the control torque. These combined Legendre representations are dispatched to the 'adapt' population, which, consisting of 1000 neurons, learns the pendulum's dynamics. A window size of $\theta = 0.01$ s and the number of polynomials $q = 4$ for both the states and control.

It is imperative to highlight that this closed-loop control of the pendulum is significantly more challenging than the open-loop Lorenz system. Initially, before the estimator has effectively learned state predictions, large discrepancies emerge between estimated and true states. Consequently, the controller receives flawed predictions, resulting in huge and high frequency control torques that jeopardize system stability. A conventional, open-world random sweep of the pendulum might prove counterproductive.

To avoid this problem of immediate instability at the initial period, a target scheduling is employed. Random targets are generated at the lower range initially. Once a target is produced, a minimum jerk trajectory is generated to move at a constant average velocity. This choice of constant average velocity is to limit the span of the velocity for easy scaling and representation in the neural population. When the pendulum lands on the given target within a position and velocity threshold, it is considered a 'hit' and only then the target range is increased by a 10% of the overall sweeping range.

Moreover, during dynamic estimation, the challenge of catastrophic forgetting arises. Traversing through the state space can mean that neural weights optimize for a localized region in the state space, inadvertently unlearning vital representations for other regions. An excessively high learning rate can exacerbate this, blending observation noise into the learning process and erasing learned features rapidly. Striking a balance is crucial: the learning rate should ensure quick convergence of estimates, while maintaining estimation capability across the entire state space. After evaluation, a learning rate of 1e-4 proved optimal.

Results

The pendulum is simulated in Python with a simulation time-step of dt = 1ms. The Figure 6.9 shows an example the state propagation of the pendulum and the estimator prediction. The pendulum is made to reach for the targets with the help of the estimation generated. In the first few trails, when the system estimates are poor, and the system interaction has not yet been captured, the control is quite unstable. Large control and velocity values are estimated even for a small range of targets. With more training data, the model learns to estimate the angular position and velocity. This can also be seen in the Figure 6.10, where the estimation error was high in the beginning, and by the end of the training the error has decreased significantly. The number of hits increases per unit time and the target range is updated faster (not shown). As evident in Figure 6.10 and 6.9, by the end of 1000s, the estimation errors have significantly decreased, with an accurate prediction of the estimate (see figure 6.9) and almost all targets being hit by the end (hit rates not shown). This demonstrates the estimator's ability to control the arm by precise estimate of the dynamics.

To evaluate the performance of the filter after training, similar to our previous testing, we turn the testing signal on, cutting off the observations to the filter. Now when the estimator recursively predicts the position of the pendulum, we control the pendulum by applying torques depending upon this recursive estimate. From figure 6.9, the pendulum is driven along the minimum jerk for most of the reaches. The threshold of the pendulum was chosen as 0.1, and IAE_{ttt} metric for the pendulum is on average 1.713 seconds for q and 1.972s for $\dot{q}$ [std dev σ_q= 0.471s, $\sigma_{\dot{q}}$ = 0.255s]. From the Figure 6.9, we can also see that the estimates lie within close range of the ground truth well into the testing phase. What is more interesting is that the estimate, even after diverging, has a similar structure and trend to the ground truth. This further bolsters the claim that the filter has indeed learnt to capture the system dynamics based on the control input and the state space. Upon reintroducing the observation, it only takes a few seconds to make the estimate follow the ground truth.

Now we have tested the performance of the filter using a nonlinear system that is a pendulum model, with torque control, we can increase the complexity of the system to further test the filter's performance.

6.3.3 Two-link Arm

As a way to increase the complexity of the system dynamics and a step towards implementation of the sensory motor control system, we now test the working of the NAF with a two link arm model. We test the control of the arm by observing the the end effector in the Cartesian coordinates and controlling the arm by applying torques.

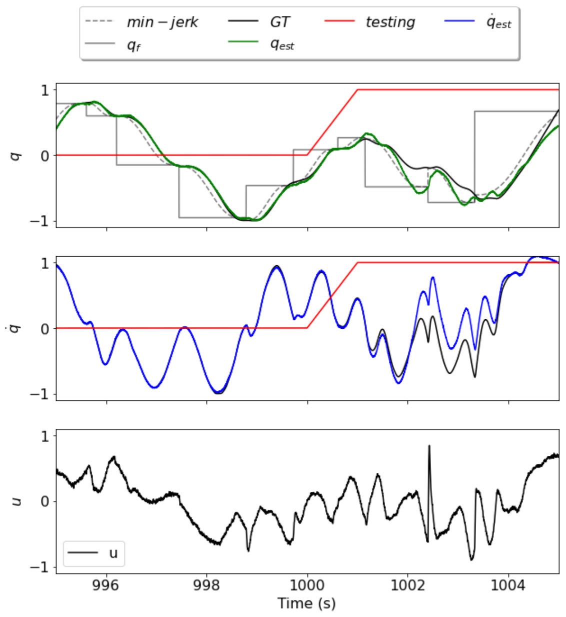


Figure 6.9: **Pendulum states across time**: The top panel shows the angular position q rad, the middle panel shows the angular velocity ($\dot{q}$) rad/s and the bottom panel shows the control torque u Nm. The solid grey line shows the final target and the dotted grey line shows the reference minimum-jerk trajectory. The solid black line represents the actual state of the pendulum and the green line represents position estimate and blue, the velocity estimate. The red line shows the 'testing' signal when the observation is gradually turned off and estimation occurs recursively.

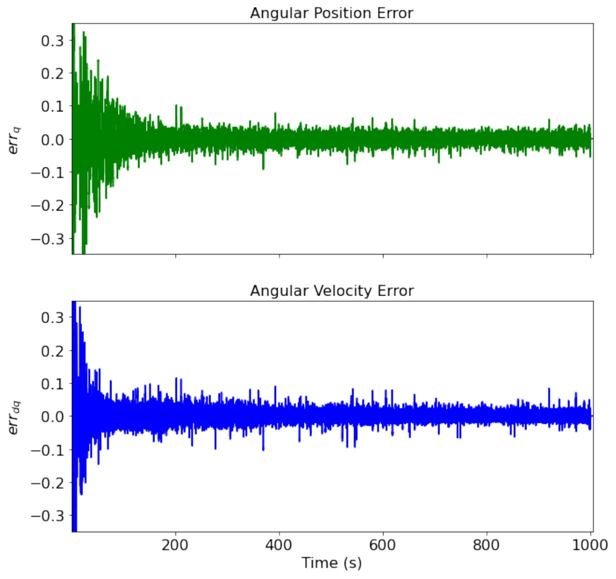


Figure 6.10: **Pendulum States Estimation Error**: The figure illustrates the propagation of error through time. The top panel (green) shows the error in angular position estimation and the bottom panel shows the angular velocity estimation error. The increased error in the beginning starts out with inaccurate estimation and control and gradually the filter learns to estimate the state accurately moving the pendulum along the desired trajectory.

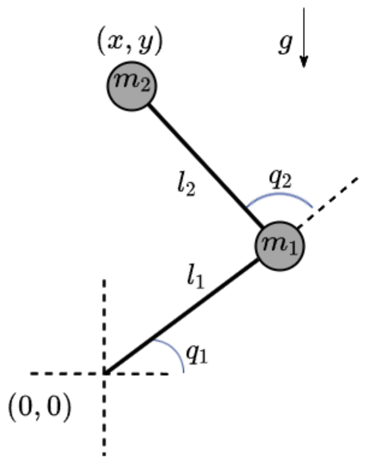


Figure 6.11: **Two-link arm dynamics**: The above diagram illustrates the dynamics of a two-link arm. The arm consists of masses m_1 and m_2 for upper arm and forearm of link lengths l_1 and l_2 from the shoulder at (0,0). The angle q_1 and q_2 represents the angle at shoulder and elbow respectively. The end-effector location is given by (x, y), measure with respect to the shoulder.

Methods

The dynamics of a two-link arm involves moving a double pendulum by applying joint torques to move on the horizontal plane by making reaching movements in the task space.

$$M\ddot{q} - C(\dot{q}, q) - G(q) = \tau \tag{6.30}$$

where M is the inertial matrix, C is the Coriolis and the centrifugal matrix and G is the gravitational term. The masses of both lower and upper arms are each 0.1 kg, the lengths l_1, and l_2 are 1 m. Since these are planar reaches on the horizontal plane, gravity is set to 0. The τ corresponds to the torque applied on the joints.

The position and the velocity of the end-effector i.e., the tip of the second link in the plane is given by (see figure 6.11):

$$x = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2)$$
$$y = l_1 \sin(q_1) + l_2 \sin(q_1 + q_2)$$

Considering the previous problem of pendulum, this system is more difficult to control since moving in task space needs both the Cartesian coordinates of the end-effector and the joint coordinates of the arm need to be taken into account. Hence, while testing of the Cartesian estimator, the ground truth joint angles are fed to the controller. Targets appear in pseudo random order to 8 different directions along a circle of radius 0.5m. A minimum jerk trajectory is generated to move for the target in 1s. A position and velocity threshold is set to 0.01 and 0.05 respectively. Similar to the previous pendulum control we also employ a progressive target range increase to avoid instability during the initial stages.

The controller is a simple PD controller with $K_p = 200$ and $K_d = 50$. Given the joint states and the end effector states, the controller calculates the Jacobian and moves the arm accordingly. In addition to the typical controller, an adaptive controller was used in parallel to increase stability during the initial control. This adaptive control learnt the control as a function of the estimated Cartesian coordinates, to follow the minimum jerk trajectory. The observations are fed to the estimator and a $\theta_s = 0.02$s and $q_s = 3$ is chosen. For the control, $\theta_u = 0.05$s and $q_u = 5$, the window size and the number of polynomials respectively. A population of 2000 neurons was used to learn the estimation.

Results

The simulation is run in Python with a simulation timestep of 1ms. Figure 6.12 shows the estimation control results of the two-link arm. When controlling the arm by estimating the end-effector Cartesian coordinates, the control is initially unstable, hitting high velocities to reach for the target. As more observations are obtained, the estimate gradually becomes less noisy catching up with the observations and providing a more stable control of the arm. As the control grows more stable, the targets are reached well within the position and velocity thresholds before the wait time and the target range was increased to hit the maximum of 0.8 units radius.

After 1000s, the filter is able to discern the interaction of the states and better control the two link arm with precision (see Figure 6.13. As soon as the testing signal is turned on, the estimation continued to stay close for a considerable amount of time well within the testing phase. The threshold for the metric was set at 0.1 and the IAE_{ttt} metric for the position was hit at 1.38 seconds and the velocity at 1.31 seconds.

6.4 Benchmark comparison

Having discussed the workings and performance of the Neural Adaptive Filter (NAF) across various dynamic systems, it is crucial to juxtapose its capabilities against other nonlinear estimation that incorporates system identification. This section aims to provide a comparative analysis of the NAF, highlighting its strengths, potential areas of improvement, and overall efficiency. Before delving into the comparative analysis, it is essential to understand the context and select an appropriate model for comparison.

Prominent models in nonlinear filtering, such as the Extended Kalman Filter (EKF) and Particle Filters, initially appear as suitable benchmarks. The EKF linearizes the model around its current estimate, offering a Gaussian approximation of the posterior distribution, whereas the Particle Filter uses stochastic

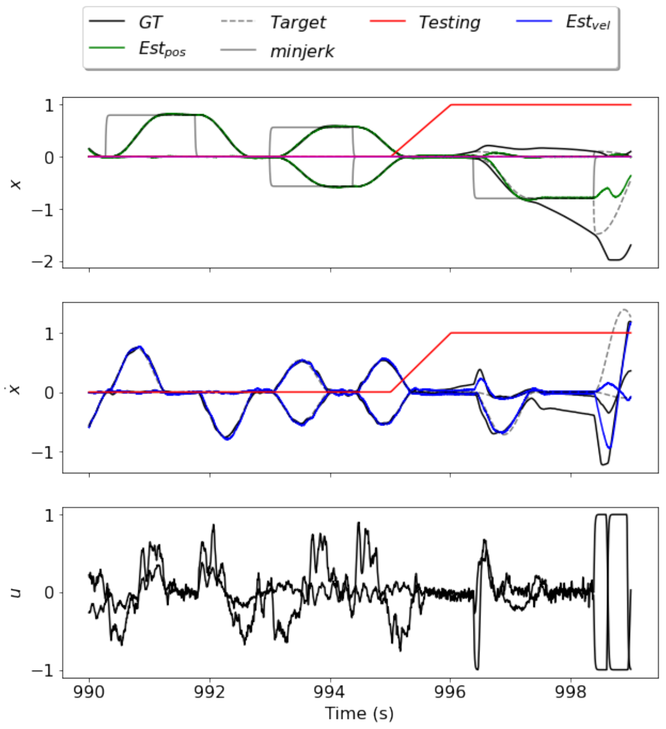


Figure 6.12: **Two-link arm states across time**: The figure illustrates the performance of the estimation driven control of the two-link arm. The top panel shows the Cartesian position x, y, the middle panel shows the Cartesian velocity $(\dot{x}, \dot{y})$ and the bottom panel shows the control torque u_1, u_2. The solid grey line shows the final target and the dotted grey line shows the reference minimum-jerk trajectory. The solid black line represents the actual state of the arm and the green line represents position estimate and blue, the velocity estimate. The red line shows the 'testing' signal when the observation is gradually turned off and estimation occurs recursively.

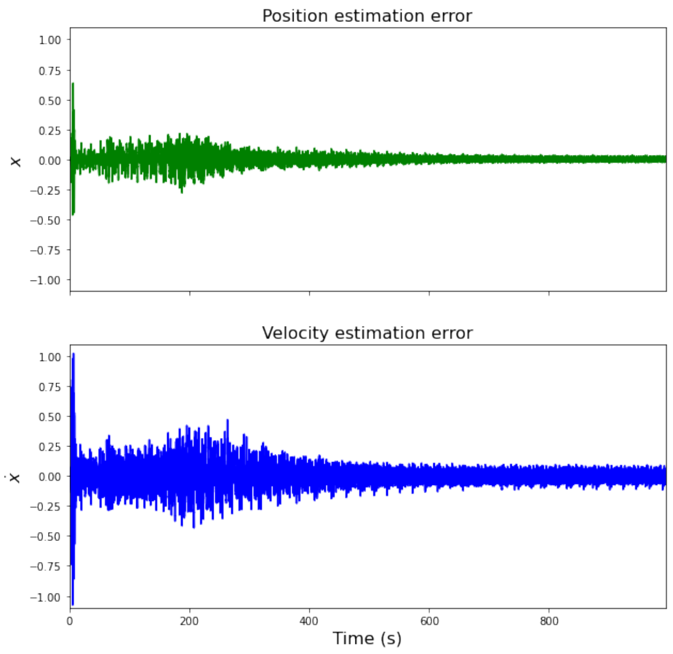


Figure 6.13: **Two-link arm Estimation Error**: The top panel (green) shows the error in position estimation and the bottom panel shows the velocity estimation error. The increased error in the beginning starts out with inaccurate estimation and control and gradually the filter learns to estimate the state accurately moving the pendulum along the desired trajectory.

weighted samples or ‘particles’ for this purpose. Aside from the limitations on the specifics of the working of each of these methods, both methods pose an inherent challenge. Both EKF and Particle Filter require intrinsic knowledge of the system they operate upon.

Unlike these traditional methods, NAF operates by learning and predicting the behaviour of any given dynamical system based solely on observations, devoid of any prior knowledge of the system’s underlying mechanics. The model-centric nature of these traditional methods, not only limits their applicability but also renders comparisons with a system like NAF irrelevant, where the dynamics of the system is learnt.

Dynamical systems identification methods offer an alternative avenue for comparison with the NAF. These methods analyze system interactions based on observation sequences, aiming to determine the underlying dynamics without a preset model. A recent and notable method in this category is SINDy-c. Using sparse regression techniques, the Sparse Identification of Nonlinear Dynamics with Control (SINDy-c) [Budišić et al., 2012; Kaiser et al., 2018] approximates the equations of the dynamic system. By regressing over a function library, which includes polynomials and trigonometric functions, it derives system dynamics from a minimal set of state observations. However, even SINDy-C, with its advanced approach, differs fundamentally from our filtering method. It conducts regression offline, separating estimation from control. The system’s deductions rely on observations made with ground truth control, without real-time feedback. Moreover, the current SINDy-c formulation does not support closed-loop control. When evaluated, SINDy-c typically uses ODE solvers, such as Runge Kutta, for recursive predictions. This approach contrasts with NAF’s simpler integration techniques.

This leaves us in a unique position. Traditional filters are challenging benchmarks due to their intrinsic differences from the NAF. Conversely, comparing NAF with dynamical discovery systems seems inequitable, given their focus on identification and absence of other constraints. The comparison has to be made to a unique operational mechanism. Without prior system knowledge, the expected system has to learn to approximate dynamics online, can adapt to changes, and can control the system in a closed-loop manner. A contemporary estimation technique we compare our model against is the Feedback-based Online Local Learning Of Weights (FOLLOW) mechanism we discussed in Sec. 2.5.2. This mechanism is an apt comparison for a filter that works online, controls the plant, and estimates the system propagation of the unknown dynamics. Furthermore, though not a primary comparison point, the NAF’s constraints stem from biological plausibility, which is also something that is discussed in FOLLOW, making it an ideal comparison.

To draw a more apt comparison between NAF and SINDy-c, we have modified the evaluation method for SINDy-c. Specifically we have adjusted it to rely purely on simple numerical integration, moving away from any sophisticated numerical interpolations. We test the performance within the scope of stationary dynamics at a given time. Distinctly, both the NAF and the modified SINDy-c are trained on ground truth control, differing from earlier sections where metrics were based on estimation control. It is important to note that the regression, and system identification from the original SINDy-c's working are still preserved but the numerical integration is simplified to match the working of NAF for better comparison. In addition to this, another aspect of SINDy-c is that the regression is also performed on a carefully selected choice of polynomials and with numerically obtained derivatives. In contrast, NAF uses neuronal bases to approximate system dynamics and computes its derivatives from observations. Hence we can expect the SINDy-c to exactly capture the system dynamics and predict the system better. However SINDy-c remains a better comparison to benchmark NAF against, than any other contemporary standards.

As far as the comparison to FOLLOW, the dynamics were seeded similarly and the training time, neuron type and number are kept identical. For all our tests, we ensured identical training settings. While SINDy-c is given the ground truth state propagation of the NAF, since closed loop control was not available, FOLLOW was made to control the plant from its estimate. In all the cases, we normalized the states and control. While not essential (more neurons could represent a broader value spectrum), this normalization simplifies the process and reduces computational demands. Such scaling should not significantly alter SINDy's predictions, as normalized states correspond to proportionally scaled dynamics, rather than presenting a completely different system. Following the supplementary code provided, the simulations are run and the results are reported below.

From Table 6.14, we have the IAE_{ttt} values for the Lorenz attractor, the pendulum and the two-link arm. It can be seen that the NAF's performance is comparable to a direct regression method such as SINDy-C and FOLLOW.

As expected, SINDy-c often better capture the system dynamics since it performs a regression over the ground truth states offline with numerical derivatives. Despite this, the NAF produces comparative predictions after training online. In both scenarios, the predictions diverged from the original ground truth within 2secs of unavailability of the observation. This indicates the difficult nature of predicting system states of a chaotic nonlinear system in the absence of observations.

For the comparison of FOLLOW with NAF, in the Lorentz and pendulum system the metrics are quiet comparable. The differences in IAE_{ttt} was mostly from the state of the system during the transition from training to testing and the amplitude of the control signal. Irrespective of the slight changes, both the systems while handling a chaotic system were only able to control the system effectively to only a few seconds in the absence of observations, being different techniques mechanistically. Interestingly, the difference in metrics from FOLLOW and NAF are also close compared to the differences between the offline and the online methods (seen both better and worse performances in Lorenz and Pendulum System).

The qualitative observations can further be reinforced from a pairwise t-test performed for the three systems and the three models. For the Lorenz attractor, the t-test results indicate significant differences in prediction errors between the SINDy and NAF models for x (t = 4.106, p = 0.003), y (t = 4.489,

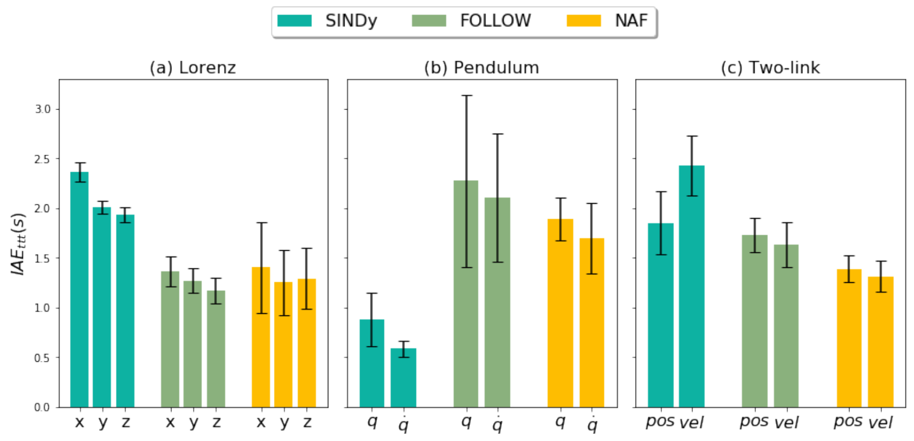


Figure 6.14: **Benchmark comparison for SINDy-c vs FOLLOW vs NAF**: The three panels show the performance of the three methods in capturing the dynamic propagation of (a) Lorenz attractor (b) Forced damped pendulum (c) Two-link arm model. The IAE_{ttt} metric is the time for the prediction error to hit a threshold, hence longer times implies better model at prediction.

p = 0.002), and z coordinates (t = 4.131, p = 0.003). Similarly, SINDy and FOLLOW models show significant differences in errors across all coordinates, with the most pronounced difference in z-coordinate (t = 4.695, p = 0.002). In contrast, there are no significant differences between the NAF and FOLLOW models for any of the coordinates, as evidenced by higher p-values (x: p = 0.671, y: p = 0.785, z: p = 0.127). As for the pendulum system, the IAE_{ttt} show significant differences between the SINDy and NAF models for both q (t = -5.822, p = 0.000) and $\dot{q}$ (t = -6.115, p = 0.000). Similarly, significant differences are observed between the SINDy and FOLLOW models for q (t = -3.059, p = 0.016) and $\dot{q}$ (t = -4.663, p = 0.002). However, no significant differences are found between the NAF and FOLLOW models for either q (t = -0.861, p = 0.414) or $\dot{q}$ (t = -1.116, p = 0.297), indicating similar performance on these metrics. A t-test on two-link arm model show significant differences between the SINDy and NAF models for both position (t = 2.705, p = 0.027) and velocity (t = 6.643, p = 0.000). The SINDy and FOLLOW models differ significantly in velocity (t = 4.257, p = 0.003), but not in position (t = 0.695, p = 0.507). While, there are also significant differences between the NAF and FOLLOW models in both position (t = -3.127, p = 0.014) and velocity (t = -2.301, p = 0.050), they are marginal compared to the difference between the combined online prediction to the SINdy offline model .The model performance tests reveal that SINDy while consistently differs in its predictions compared to both NAF and FOLLOW, both NAF and Follow exhibit similar performance characteristics across various scenarios.

The SINDy-c model works with the careful selection of polynomial basis. F provided the polynomials are chosen appropriately in the dynamic system can be approximated with better accuracy and this is demonstrated in the case of the Lorenz attractor and two-link arm. To demonstrate this, we choose the pendulum dynamics with a specific nonlinear function for the damping coefficient ($|x|$) which is not a part of the typical choice of polynomials. While capturing this dynamics using SINDy-c, the metrics shows inadequacy in the performance. On the other hand, the prediction accuracy remain consistent for NAF across different systems. This indicates robustness in the working of NAF.

Given that we have a predicting mechanism capable of estimating the system states, we shall implement this in our linear model to upgrade our model of the motor control system.

Chapter 7

Sensorimotor Control Model

In this chapter a biologically plausible sensorimotor control model is presented. Compared to the previous Kalman filter version, this model is built with a nonlinear prediction system driving the perceptual system.

7.1 Model description

In this section, we present a detailed description of the sensorimotor control model and how each of the components in our model maps to different anatomical organizations responsible for the overall adaptive behaviour. The model is described in Figure 7.2. The components and the organization of this nonlinear version are very similar to the previous linear sensorimotor control model (see Section 5.4.3 Figure 5.9). In contrast to the linear version, where the arm was modelled as a unit mass system, we here update the dynamics to a nonlinear two-link arm model (see Figure7.1) controlled through the application of joint torques.

The environment for our experiments consists of a two-link arm model that moves on a frictionless 2D surface, along the transverse plane and perpendicular to gravity. As this experiment is conducted on a tabletop, gravity is not made to affect the system. The arm is made to move along the surface by applying torques to each of the joints to reach for the provided targets (see Figure 7.1). The forearm and upper-arm are of equal unit length and of mass 0.1kg. Although gravity is neglected, Coriolis and centripetal components of the dynamics are kept intact.

The overall framework of the model can be seen in Figure 7.2. There are two modalities of observation, namely vision and proprioception. The visual observations (also referred to as the 'cursor'), z_{vision}, contain the end-effector Cartesian coordinate position and velocity $[x, y, \dot{x}, \dot{y}]$. On the hand, the proprioceptive observations, contain the joint angle coordinate position and velocity

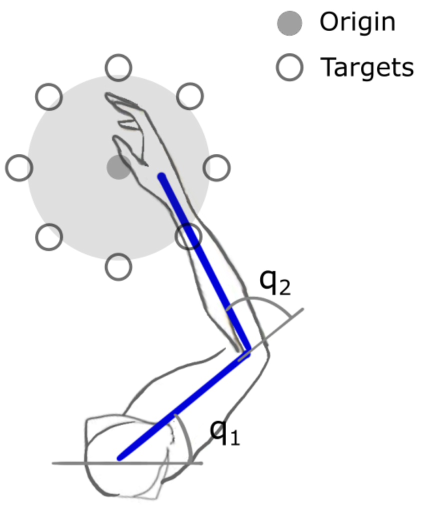


Figure 7.1: **Experimental and Model Setup**: The arm is modeled by a two-link arm centered at an origin aligned with the shoulder. There 8 targets distributed around the origin that appear pseudo randomly. The circular shaded region around the origin is the opaque zone where the end-effector feedback (cursor) is not provided during an outward reach.

$z_{prop} = [q_1, q_2, \dot{q}_1, \dot{q}_2]$ of the upper and forearm respectively (see Figure 7.1). From our literature (see Section 3.1), it is evident that we have individual perceptual system for each of our modalities that predict the arm states in its own coordinate frames of reference. Hence, for each perceptual system, we employ a Neural Adaptive Filter to estimate the state, given the previous state and a copy of our control inputs $[u_1, u_2]$. The 'Visual Estimator' predicts the cursor location in Cartesian coordinates x^+_{vision} in figure and the 'Proprioceptive Estimator' predicts the joint angle location of the arm (q^+_{prop}). Both of these components that are subjected to learning on-the-fly from the observations. These estimator components learn by comparing the predictions and the observations. Both of these estimators have a fast learning rate of 1×10^{-4}, owing to quick recovery and convergence of the estimates to the observations. If observations are either uncertain, or unavailable, the learning is inhibited and these system recursively predict from the previous prediction.

To obtain the end-effector estimate from joint angle location, we have the Angular End-effector Estimator. This component learns the forward kinematics as a function of the joint angles from the visual cursor estimate. Any changes to link lengths, or kinematic transformation is captured here and component has a learning of 5×10^{-6}. The errors here correspond to misalignment of proprioceptive and visual end-effector estimates. When the visual observations are uncertain, or unavailable, this learning is mitigated to avoid learning from the recurrent predictions. Now we have the Cartesian estimate of our hand from

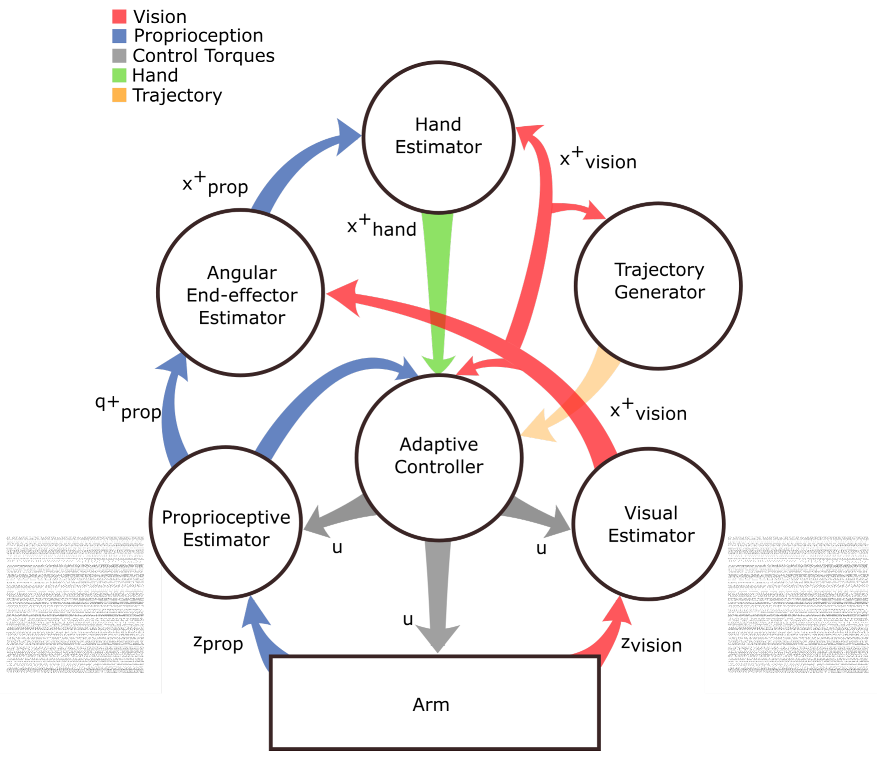


Figure 7.2: **Sensorimotor control model**: From the two noisy observation of the hand, vision and proprioception and the efference copy from the controller two estimates are obtained using the Kalman filters. These are further fused together to know the estimate of the hand position and velocity. A combined movement vector is learned from the hand heading direction and correction realized from the end point error.

proprioception, which is available even when visual observation is absent. The Hand Estimator component fuses these two estimates to provide a combined estimate. This uses a gain 'α' to manipulate reliance on one modality over the other.

The Trajectory Generator produces a minimum-jerk trajectory from the current location of the cursor and target location. The time required to reach for the target can be modified and there are no learning elements in this component. The Adaptive Controller uses the combined Hand estimate and joint angles to generate movement vectors to control the arm's end effector to move along the desired trajectory. It also compares the cursor's location and learns to map an added adaptive component that learns the new 'vector-to-aim-differently' if the current control torques are insufficient to follow the desired trajectory. The learning rate for this component is 1×10^{-5}. These movement vectors in Cartesian coordinate are transformed to control torques and are applied to the shoulder and elbow of the arm.

7.2 Anatomical parallels

When comparing our current architecture with the neuroscience literature previously discussed in Section 3.1, we can draw parallels between the model's components and specific brain regions.

1. **Visual Cortex and S1**: The Visual Cortex is vital for processing visual cues [Grill-Spector and Malach, 2004]. In contrast, S1 is pivotal for processing touch, spatial orientation, and proprioception[Hsiao et al., 2002]. While we do not directly replicate this in our model, we operate under the assumption that such processed information is accessible to it.

2. **Dorsal Premotor Cortex (PMd)**: PMd plays an instrumental role in preparing for movement, particularly those movements guided by external cues [Wise et al., 1997]. This aligns with the functionality of our Trajectory Generator.

3. **Parietal Cortex (PPC)**: Sensory consequence prediction, especially in the visual stream, is primarily associated with PPC [Colby et al., 1992; Haarmeier et al., 1997]. PPC estimates the arm's current state during movement, inclusive of its position, velocity, and orientation [Buneo and Andersen, 2006]. Although, the Middle Temporal Area is implicated in the brain's capacity to infer and project motion paths from prior observations [new, 1988], it is mostly the PPC that is involved in sensory consequence prediction. Especially the **Inferior Parietal Lobule (IPL)** and **Superior Parietal Lobule (SPL)**. Both are engaged in visual and proprioceptive estimates of limb positions (see Section 3.1). Specifically, IPL facilitates the transformation of visual data into diverse reference frames, allowing for multi-sensory information integration [Buneo and Andersen, 2006]. Consequently, our Angular End-effector Estimator and Hand Estimator mirror functionalities found within the PPC region.

4. **Cerebellum**: The cerebellum is responsible for generating error signals during movement control to regulate trajectories. It receives feedback from the sensory systems about the actual movement and compares it to the intended movement. Any discrepancies between the two generate an error signal that is used to adjust future movements. [Wolpert et al., 1995] [Miall and Wolpert, 1996]. Many of the error generation, forward models and efference copies replicates the Cerebellum's functionality.

5. **Primary Motor Cortex (M1)**: As highlighted in various studies, M1's primary role is akin to a controller [Penfield and Boldrey, 1937; Georgopoulos et al., 1989; Sanes and Donoghue, 2000]. Further, the cerebellum aids in rectifying intended movements. Our Adaptive Controller mirrors

the M1 and the cerebellar subsections responsible for error generation and corrections.

Collectively from the functionality and evidence of different error generation and the working of the individual components, we can draw parallels to the well defined and specific cortical structures. The entirety of the model is reproduced to avail biologically plausibility. For the sake of simulation, majority of the predictive, perceptual and the adaptive control are reproduced in spiking neurons, while the arm dynamics, trajectory and the conventional controller are simulated numerically.

7.3 Experimental Setup

7.3.1 Single trial

The simulation is run with a *dt* of 1ms. The targets appear in a pseudo random order in 8 directions distributed at 45°along a unit circle centered at a home location or 'origin' chosen at arbitrary distance in front of the shoulder (see Figure 7.1). The visual information provides the end-effector states with respect to the origin and the proprioceptive information relays the joint angle information. A single trial is defined as the movement of the cursor, that represents the visual feedback, from the origin to the target. Only a cursor, representing the hand, is provided. The arm itself is hidden from view in the experiments performed on humans and animals that is being reproduced. The next trial begins when the hand is moved back to the home location and the new target appears. A target is made to appear when the hand falls within a given velocity and position threshold or after 2s of wait period.

7.3.2 Rotation

The overall setup during rotation is preserved from before (see Section 5.4.2). As a recap, the proprioceptive observation is measured unperturbed while the visual observations are rotated during the rotation trials. The rotation is made at a chosen angle in a counter clockwise direction with respect to the origin. Once the rotation is introduced, the visual feedback is provided only at the beginning and the end of the reach and is made invisible during the majority duration of the reach.

To recall the experimental setup, there are three stages, namely: (a) Baseline Stage(16 trials) where all feedbacks remain undistorted; (b) Rotation Stage (80 trials) where the visual feedback is rotated with respect to the origin; (c) Washout Stage (80 trials) where the feedback is returned to its original unperturbed nature. The time course of the change of heading direction for each of the reaches is monitored across the stages to study the extent of adaptation.

7.4 Results

The system undergoes simulation in the specified environment, progressing through the baseline, exposure, and washout stages. This progression allows for the testing of adaptation when rotation is introduced.

7.4.1 Baseline Stage

For a single reach out trial to the target in the baseline condition, the arm's end effector follows the minimum jerk trajectory. The perception systems estimate the propagation of the visual cursor, the joint angles and the hand states with great accuracy (see Fig.7.3a). When the visual observation is unaltered, the ground truth, the visual cursor, and the hand location coincide with each other. Upon making consistent reaches, no learning occurs during this phase, since the predictions are accurate and no error is generated. The estimates are continuously compared with the observations and reaches are made to the targets.

7.4.2 Rotation Stage

Single trial

When the counterclockwise (CCW) rotation is introduced, adaptation kicks in, because of the generation of multiple error signals. The visual feedback representing the end effector is turned off during the beginning of the reach. It is is restored only at the end of the reach, but it is now rotated about the origin to be misaligned from the ground truth. Hence, this observation is different from the visual estimate that has been learned. The resulting trajectories can be seen in Figure 7.3b. During the reach where the visual feedback is unavailable, the estimation system completely relies on the recursive prediction until the cursor is observed again. Once the feedback is provided again, the visual perceptual system updates its estimate to coincide with the observation. The error referred to here as the 'visual estimation error' helps in learning to better predict the new dynamics of the visual feedback. It is important to note that this is not a linear transformation for the perceptual system, but rather a change in dynamics, that has to be captured to better predict the visual cursor's movement.

In addition, there is a hand estimation error. The predicted hand location is learned as a function of the proprioceptive estimate being mapped to the visual estimate. This perceived hand location is now offset from the visual cursor feedback, resulting in hand estimation learning. In other words, this is the error pointing towards the mismatch of visual and proprioceptive sensation of the hand. This adaptation process helps in updating the estimate of the hand position, given available information. Notably, the visual estimate has caught up with the new observation, and yet the combined hand location estimate remains closer to the previously learned and expected hand location. This is because the learning rates of the combined hand estimator and the visual estimator

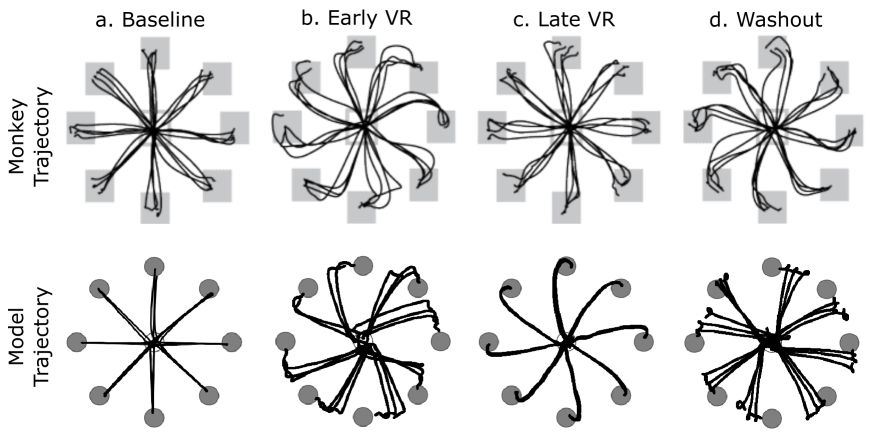


Figure 7.3: **Trajectory comparison:** Experimental data from [Perich et al., 2018] (top) and model (bottom) during visuomotor rotation: (a) the initial baseline trials without any rotation (b) early adaptation with onset of rotation at 30^o CCW direction (c) later stage of adaptation to rotation (d) washout trials when the rotation is returned to 0^o

are different. The objective of the prediction system is to estimate the states that are being observed, while the hand perceptual system is attempting to get an optimal inference from multiple modalities and to keep track of where the hand is. In this scenario, this can be understood as the estimator trusting the proprioceptive information more than the uncertain visual feedback.

Since the goal of the task is to land the cursor on the target location, a third error is generated that creates a need to change the overall movement vector to now compensate for the visual task error. While the primary controller has maintained the end-effector along the minimum-jerk trajectory, its control inputs are no longer adequate to complete the task goal. Thus, the visual adaptive control overrides the movement vector so as to bring the cursor to the target. This error correction can be seen in action from the curved trajectories made closer to the target (see Figure 7.3 b and c) as soon as the visual feedback is made available. During the reach back to the origin between trials, the feedback is made available throughout the reach to start the next reach from the origin.

Across trials

As the various errors drive adaptations, the reaching behaviour changes too. Initially, when the perceptual systems are introduced to the rotation, high directional errors are made by the subject. This is also true for the model, as indicated by the typical curved trajectories and the incorrect heading direction (see Figure 7.3b). With multiple trials of rotational exposure, the model compensates for directional error by changing the trajectory and learning the additional movement vector as a function of the hand location estimate(see Fig-

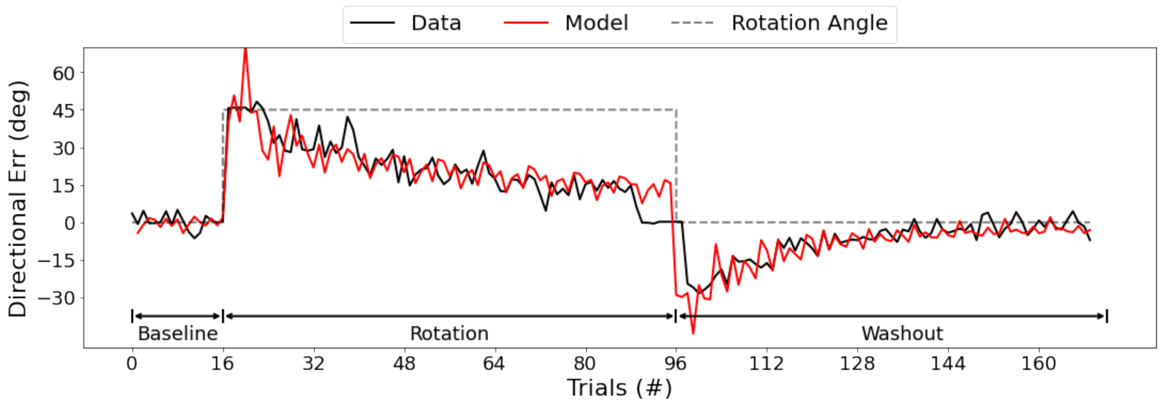


Figure 7.4: **Behaviour comparison**: Time course of directional error at threshold of the model's reaches (red) compared against experimental data (blue) (reproduced from the graphs from [Mazzoni and Krakauer, 2006]. The grey dashed line shows the angle of rotation of the cursor with respect to the hand location.

ure 7.3c). From Figure 7.4, we can see that the rate of adaptation is also similar to the behavioural data seen in the experiment. By the end of the rotation stage, the reaches have marginal directional error and the reaches become less curved compared to the earlier reaches.

7.4.3 Washout Stage

Following the rotational exposure phase, the washout stage is initiated. In this phase, the rotational perturbation is eliminated, and the visual feedback is realigned with the ground truth. Although the observations mirror the original feedback congruence observed in the baseline phase, subjects make directional errors opposite to the current perturbation (see Figure 7.3 d). Some trajectories along a few directions, have a loop towards the end of the reach in the model. This is a result of added noise from the combined hand location estimate and a harsh correction from the model. Although this makes the reaches slightly different to the experimental data in Figure 7.3d, the angular error remains consistent across trials.

Given that the experimental data is for one monkey, it is not uncommon to see such reaches in reaching studies. This modelled behaviour suggests that the sensorimotor system has adjusted to the previous transformation, and the controller has adapted to the perturbations. The model also exhibit the same behaviour and resembles experimental data across time. By the end of the washout phase, the model has fully adapted to the rotation, producing trajectories consistent with baseline trials.

7.5 Discussion

We have presented a biologically plausible model of the primate sensorimotor control system that begins to unravel the intricacies of sensorimotor mechanisms through the lens of adaptation to visuomotor rotation. The updated NAF sensorimotor control model successfully consolidated many features of the system that were previously unaccounted for by the linear model.

Firstly, the plant that approximates the limb uses two link arm dynamics. This introduces nonlinearity in the dynamics similar to what is observed in primate arm, and unlike a unit mass system. Owing to this change, the control and estimation systems are also moved to the nonlinear realm. The perceptual systems can now predict the propagation of the system states of the nonlinear system in multiple coordinates even in the absence of observations. Previously, the hand estimate was a linear combination of vision and proprioception. The current model learns this forward mapping on-the-fly. As supported by evidence, over time, subjects come to believe that the actual location of the end-effector closely aligns with the visual feedback, as reflected in the model. Furthermore, this shift into nonlinearity is justified by the task-specific goals that individuals encounter in everyday settings. The mapping from proprioceptive joint angles to the end-effector states are nonlinear. Overall, the nonlinear implementation has solely added details to many of the model components, underwriting appropriate behaviours where linear approximation would fail.

Another important aspect of the sensorimotor control model is plasticity. When the predictive systems observed the visual cursor to be present in a location farther from the estimated state, the previous Kalman filter though updated the state when observations were present, was not able to predict differently as trials progressed. However the NAF system is able to actively change the prediction of the visual estimate with every trial. This can be seen in Figure 7.5, where the visual estimates predict differently before and after adaptation. This is faithful to the experimental findings where the cortex adaptively updates its predictions for the visual estimates as trials progress.

In combining the various system state estimates, and the motor pathways responsible, we were also able to identify the different error signals that drive the adaptations. There are multiple mechanisms adaptively orchestrating stable control of the arm. Our model was able to reproduce the behaviour and the timeline of the overall adaptation, as well as propose the mechanisms involved in each pathway, various errors involved and capture the learning that dictates the behaviours consistent with the experimental observations. For example, when the contradiction between visual and proprioceptive observations is first seen, the adaptive controller learns to change the movement vector as a function of the hand location. This control adaptation kicks in at a faster rate than the correct prediction of hand location. If the hand location prediction updated faster than the control, then it would result in conflicting controls. Essentially, the adaptive controller would be providing obsolete additive torques that worked for the previous hand-location estimation when it was not aligned with the cursor. This can result in unstable control of the arm. In addition, comparing the

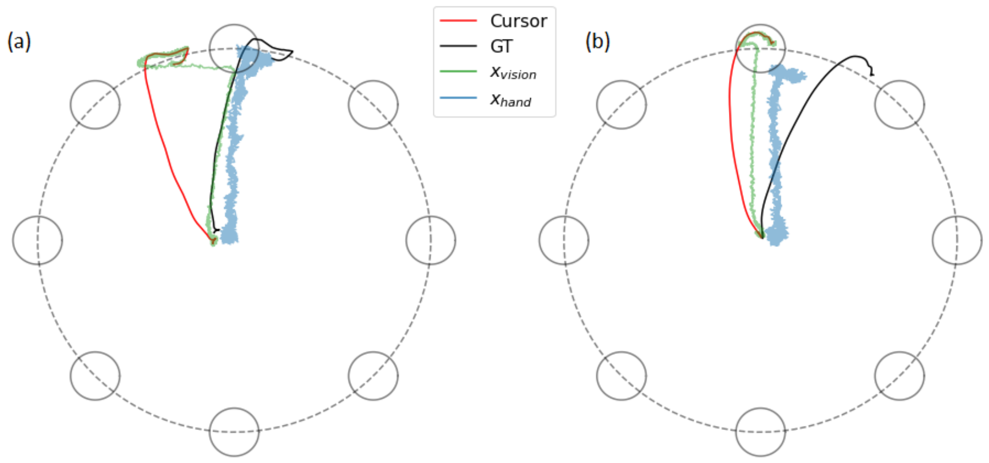


Figure 7.5: **Demonstration of online adaptation**: The figure shows the prediction of different systems (a) before and (b) after adaptation. With the introduction of rotation and the absence of visual feedback, the visual prediction, and hand estimate align with the ground truth location of the hand. After adaptation, the visual prediction captures the cursor location better. Also, the hand estimate is now in between the ground truth and the visual estimate

working of each mechanism and the biological evidence also helps us in drawing parallels to the cortical structures and how they are organized in producing the observed behaviour. This model successfully predicts and be tested for many of the behaviours ranging from neuronal representation of movement vectors and system states to different behavioural aspects that shed light into the working of the motor control system.

Many studies point towards the presence of internal models and that the brain updates them (see Section 3.1). In addition, many studies show that in the absence of sensory feedback, the prediction diverges quickly from the ground truth and controlling the arm becomes difficult. In [Cole and Sedgwick, 1992], the authors discuss a case study where the subject who lost almost all proprioceptive and tactile feedback below the neck due to a rare condition, learned to move with visual guidance. However, in the absence of visual feedback, his movement became very imprecise, highlighting the crucial role of sensory feedback in refining motor control and how the predictive system can quickly from the ground truth. A similar study is also found in [Taub et al., 1966], where de-afferented monkeys had significant difficulty in modulating grasp forces appropriately in the absence of sensory feedback. Observations like these are evident from the model. The reach performance and stable movement was only guaranteed when prediction were modulated by the sensory observations. Without these signals, the plant control becomes significantly

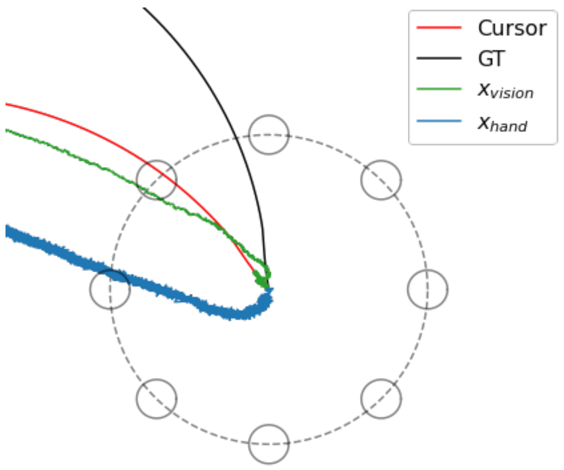


Figure 7.6: **Unstable control in the absence of observations**: The control starts with maintaining the arm at origin with minimal velocity. As soon as both the observations are cut-off, the estimations drift, driving the system unstable, out of the task-space.

more difficult and typically fail within a few seconds in the absence of any feedback (see Figure 7.6). The code for reproducing the work can be found at: *https://github.com/nvaidyan1/phd_thesis*

Chapter 8

Conclusions

8.1 Thesis Contributions

Unlike linear systems, which adhere to principles of superposition and homogeneity, nonlinear systems exhibit a plethora of behaviors that are complex and less predictable. With the study and control of nonlinear systems being highly ubiquitous, the estimation of such systems has become highly relevant. On top of this, evolving system dynamics or dealing with unknown systems only adds to the complexity. Currently there are several system identification techniques [Sussillo and Abbott, 2009; Brunton et al., 2016; Raissi et al., 2018; Lusch et al., 2018; Dong et al., 2023; Meidani and Farimani, 2023]. These methods are system identification methods, which collects state propagation data and employ statistical methods such a regression or use neural networks to capture the equations governing the dynamics. But, this involves separate offline training regimes to deduce the dynamics interactions. There are also episodic techniques [Fraccaro et al., 2017; Yu et al., 2017; Murray, 2019], which use reinforcement learning methods. These also do not fall under the online category since the training happens offline between episodes. Unlike these methods, the NAF performs system estimation online as the observations manifest and controlling the plant using the estimates. If there are additional disturbances, changes in the dynamics evolution, then these systems would fails and simply they are not designed for simultaneous learning and estimation.

There are very few online estimation techniques that handle both nonlinearities and unknown system dynamics, where [Alemi et al., 2018] and [Gilra and Gerstner, 2017] are two examples of these contemporary methods, which belong to a new bio-inspired approach for control and estimation engineering. Though these methods are online methods, these vary in the basic working operation compared to the working of NAF. Alemi et al. implement this online learning of the plant, using a teacher-student recurrent network. This network uses a feedforward connection for the input signal, and fast connections for efficient distribution of spikes and slow connections for learning the student network. While they implement this method to instill spike efficiency, they also use dendritic nonlinearities, and use multiple connections to realize this estimation. As for Gilra et al., although they were able to estimate the system propagation, they have learning connections both in the forward connection of the input signal and another recurrent learning in the state representation as well. In contrast to the two online methods, the NAF implements this prediction scheme by capturing the windowed time representation of the input and state variables. Thus the learning of system dynamics happen at a single layer where the system states are linearly decoded from the neuronal space, additionally with recurrence at the level of feedback and Legendre representation whose recurrence dynamics is fixed. By capturing the entire window of the state space this potentially also is helpful for learning systems that are more complex with high time dependencies or even when the system dynamics information is not readily aligned in time. For the case of Gilra et al. (FOLLOW) we have also seen that the benchmarks (see Sec. 6.4) shows comparable performance of the two filter under similar environments. Furthermore, to the best of our knowledge these other online methods have only been tested on toy problems.

With modern control challenges, more often than not we see solutions for this problem come from biological inspirations. Along similar lines, the NAF proposed here is a novel prediction technique, that adds to this particular niche. Unlike the past methods, the NAF has been employed to model a subtle learning paradigm in motor control (see Sec. 7.1). While this mechanism is setup to work in a biologically realistic spiking neuron implementation, the framework can be broadly applied numerically to estimate the given system as manifests in real-time. The NAF helps in tackling the estimation of unknown nonlinear systems while controlling the plant simultaneously. It does so by representing the signal through time, and learning the dynamics by continuously updating through a learning mechanism similar to gradient descent with every new observation. Hence, for control engineering tasks of complex changing dynamics, unknown kinematics or sensory uncertainties, NAF's performance has shown that it is an strong candidate in the application of close loop online estimation and control of an unknown plant.

When delving into the realm of sensorimotor control, various models are mostly fragmented in nature (see Sec. 4) in offering explanations for the underlying dynamics. While many cater to specific functionalities (see Sec. 4.1, a comprehensive, unified model remains elusive. The NAF-based sensorimotor control model suggests a new direction that may a change this scenario. It not only encapsulates a multitude of mechanisms but also does so in a manner that aligns with biological plausibility. It is one of the first large scale functional models that involves an adaptive forward estimation systems combined with sensory feedback, in contrast to many of the current models that solely work on feedback control. The model was able to explain how the prediction and adaptation happens at multiple levels (joint estimation, end-effector estimation and jacobian learning). It was able to explain prediction, perceptual and adaptations at visual and proprioceptive levels and also explored how these would interact to explain higher level behaviors such as visuomotor rotations. Furthermore, similar to how a visual decorrelation study was conducted, this model can be used to simulate other behaviors that work with visual and proprioceptive alterations, ablations or inhibitions even at the spiking level of the model. Such a comprehensive model encompassing different parts of the sensorimotor system on its own adds substantial ability to glean deeper insights into inherent biological mechanisms, paving the way for the further development and offers better explanation of motor control pathways involved in the primate movement execution.

8.2 Future work

In the prevailing landscape of filtering methods, we observe a plethora of specialized techniques, each optimized for a singular purpose. On one hand, we have nonlinear filters designed to estimate system states, leveraging detailed knowledge of system dynamics. On the other, we see system identification methods that focus primarily on extracting the underlying system interaction solely from observable data. The introduction of the Neural Adaptive Filter (NAF) has bridged this dichotomy. The NAF, when fed an online stream of observations, is adept at estimating the system's state and subsequently helping to control it in real-time. What sets the NAF apart is its resilience; it continues to guide control even in the absence of observations, and is agile enough to adapt online to shifts in dynamic interaction. This inherent flexibility propels its applicability beyond just biological realms. Systems characterized by frequent uncertainties or those prone to non-stationarity stand to benefit significantly from the NAF's capabilities.

However, it is important to note the limitations of the NAF. Its efficacy is determined by the spectrum of functions that the neuronal subspace can approximate. Furthermore, as the system's order elevates, there's an increased demand for polynomials, which can inadvertently compromise the filter's accuracy and convergence speed. In tandem, the efficacy of the NAF is sensitive to hyperparameter selection, necessitating tuning for optimal performance. Looking ahead, there is potential in leveraging the Legendre network. By implementing the Legendre network to compare trajectories, and aligning error signals in time, it might be possible to capture systems with inherent delays, thereby extending the reach and applicability of the NAF.

By modelling the sensorimotor control, we were able to explain many of the behaviors and also put together a comprehensive biologically plausible model of a good chunk of the primate sensory motor control systems. Yet, as with any early endeavor, this model is nowhere close to complete. For instance, the dynamics, in its current form, are approximated using a two-link arm system. To enrich the model's depth and realism, there is scope to integrate more layers of complexity both in terms of muscular interaction and synaptic interactions. Additionally, sensorimotor delays are deemed negligible in the current model. Integrating observational and control delays could significantly augment the realism and utility of the sensorimotor control framework and can potentially help in extrapolating it to engineering solutions.

Another important aspect that could be expanded is the error source attribution. Error source attribution refers to the process by which the brain continuously compares our intended movement to the actual movement and attributes the cause for the observed error. The anterior cingulate cortex (ACC) is largely responsible for this and plays a pivotal role in motor adaptation [Brockett et al., 2020]. For instance, if you are throwing a dart and you miss the target, knowing whether the error was due to your arm's motion or a gust of wind will determine how you adjust your next throw. The current model is primed for pursuing this line of work with multiple errors driving adaptation and gain terms that are currently tuned for availability of observation. The ability to perform error source attribution has important ramifications for unraveling how the motor system helps in skill acquisition and getting cognitive feedback about the working of the environment we live in to comprehend physics more generally.

In conclusion, the Neural Adaptive Filter and the NAF-based sensorimotor control model, with their unique attributes and capabilities, mark a significant step forward in our desire to understand the sensorimotor control systems. As we navigate the landscape of system dynamics and sensorimotor control, these tools not only offer a nuanced understanding but also highlight the how much remains for future exploration.

Bibliography

(1988). A selective impairment of motion perception following lesions of the middle temporal visual area (mt). *Journal of Neuroscience*, 8(6):2201–2211.

Akaike, H. (1969). Fitting autoregressive models for prediction. *Annals of the institute of Statistical Mathematics*, 21(1):243–247.

Alemi, A., Machens, C., Deneve, S., and Slotine, J.-J. (2018). Learning nonlinear dynamics in efficient, balanced spiking networks using local plasticity rules. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Alexander, R. M. (1997). A minimum energy cost hypothesis for human arm trajectories. *Biological cybernetics*, 76(2):97–105.

Bedford, F. L. (1993). Perceptual and cognitive spatial learning. *Journal of experimental psychology: Human perception and performance*, 19(3):517.

Bekolay, T., Kolbeck, C., and Eliasmith, C. (2013). Simultaneous unsupervised and supervised learning of cognitive functions in biologically plausible spiking neural networks. In *Proceedings of the annual meeting of the cognitive science society*, volume 35.

Berniker, M. and Kording, K. (2008). Estimating the sources of motor errors for adaptation and generalization. *Nature neuroscience*, 11(12):1454–1461.

Berniker, M. and Penny, S. (2019). A normative approach to neuromotor control. *Biological cybernetics*, 113:83–92.

Bhattacharyya, S. P., Datta, A., and Keel, L. H. (2018). *Linear control theory: structure, robustness, and optimization.* CRC press.

Bliss, T. V. and Collingridge, G. L. (1993). A synaptic model of memory: long-term potentiation in the hippocampus. *Nature*, 361(6407):31–39.

Brockett, A. T., Tennyson, S. S., deBettencourt, C. A., Gaye, F., and Roesch, M. R. (2020). Anterior cingulate cortex is necessary for adaptation of action plans. *Proceedings of the National Academy of Sciences*, 117(11):6196–6204.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937.

Budišić, M., Mohr, R., and Mezić, I. (2012). Applied koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):047510.

Buneo, C. A. and Andersen, R. A. (2006). The posterior parietal cortex: sensorimotor interface for the planning and online control of visually guided movements. *Neuropsychologia*, 44(13):2594–2606.

Churchland, M. M., Afshar, A., and Shenoy, K. V. (2006). A central source of movement variability. *Neuron*, 52(6):1085–1096.

Colby, C., Goldberg, M., et al. (1992). The updating of the representation of visual space in parietal cortex by intended eye movements. *Science*, 255(5040):90–92.

Cole, J. and Sedgwick, E. (1992). The perceptions of force and of movement in a man without large myelinated sensory afferents below the neck. *The Journal of physiology*, 449(1):503–515.

Conditt, M. A., Gandolfo, F., and Mussa-Ivaldi, F. A. (1997). The motor system does not learn the dynamics of the arm by rote memorization of past experience. *Journal of Neurophysiology*, 78(1):554–560.

Cueva, C. J. and Wei, X.-X. (2018). Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *arXiv preprint arXiv:1803.07770*.

Danziger, Z. and Mussa-Ivaldi, F. A. (2012). The influence of visual motion on motor learning. *Journal of Neuroscience*, 32(29):9859–9869.

Denève, S., Alemi, A., and Bourdoukan, R. (2017). The brain as an efficient and robust adaptive learner. *Neuron*, 94(5):969–977.

DeWolf, T., Jaworski, P., and Eliasmith, C. (2020). Nengo and low-power ai hardware for robust, embedded neurorobotics. *Frontiers in Neurorobotics*, 14:568359.

DeWolf, T., Patel, K., Jaworski, P., Leontie, R., Hays, J., and Eliasmith, C. (2023). Neuromorphic control of a simulated 7-dof arm using loihi. *Neuromorphic Computing and Engineering*, 3(1):014007.

DeWolf, T., Stewart, T. C., Slotine, J.-J., and Eliasmith, C. (2016). A spiking neural model of adaptive arm control. *Proceedings of the Royal Society B: Biological Sciences*, 283(1843):20162134.

Dong, A., Starr, A., and Zhao, Y. (2023). Neural network-based parametric system identification: a review. *International Journal of Systems Science*, 54(13):2676–2688.

Eliasmith, C. and Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems.* MIT press.

Fishbach, A., Roy, S. A., Bastianen, C., Miller, L. E., and Houk, J. C. (2007). Deciding when and how to correct a movement: discrete submovements as a decision making process. *Experimental brain research*, 177:45–63.

Flash, T. and Hogan, N. (1985). The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of neuroscience*, 5(7):1688–1703.

Flash, T. and Sejnowski, T. J. (2001). Computational approaches to motor control. *Current opinion in neurobiology*, 11(6):655–662.

Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. (2017). A disentangled recognition and nonlinear dynamics model for unsupervised learning. *Advances in neural information processing systems*, 30.

Gandolfo, F., Li, C.-S., Benda, B., Schioppa, C. P., and Bizzi, E. (2000). Cortical correlates of learning in monkeys adapting to a new dynamical environment. *Proceedings of the National Academy of Sciences*, 97(5):2259–2263.

Georgopoulos, A. P., Lurito, J. T., Petrides, M., Schwartz, A. B., and Massey, J. T. (1989). Mental rotation of the neuronal population vector. *Science*, 243(4888):234–236.

Gilra, A. and Gerstner, W. (2017). Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network. *Elife*, 6:e28295.

Goodbody, S. J. and Wolpert, D. M. (1998). Temporal and amplitude generalization in motor learning. *Journal of Neurophysiology*, 79(4):1825–1838.

Grill-Spector, K. and Malach, R. (2004). The human visual cortex. *Annu. Rev. Neurosci.*, 27:649–677.

Haarmeier, T., Thier, P., Repnow, M., and Petersen, D. (1997). False perception of motion in a patient who cannot compensate for eye movements. *Nature*, 389(6653):849–852.

Haith, A., Jackson, C., Miall, C., and Vijayakumar, S. (2008). Interactions between sensory and motor components of adaptation predicted by a bayesian model. *Workshop on Advances in Computational Motor Control (ACMC 2008).*

Heuer, H. and Hegele, M. (2008). Adaptation to a nonlinear visuomotor amplitude transformation with continuous and terminal visual feedback. *Journal of Motor Behavior*, 40(5):368–379.

Hsiao, S. S., Lane, J., and Fitzgerald, P. (2002). Representation of orientation in the somatosensory system. *Behavioural brain research*, 135(1-2):93–103.

Huang, V. S. and Shadmehr, R. (2007). Evolution of motor memory during the seconds after observation of motor error. *Journal of neurophysiology*, 97(6):3976–3985.

Imamizu, H. and Kawato, M. (2009). Brain mechanisms for predictive control by switching internal models: implications for higher-order cognitive functions. *Psychological Research PRPF*, 73:527–544.

Inoue, M. and Kitazawa, S. (2018). Motor error in parietal area 5 and target error in area 7 drive distinctive adaptation in reaching. *Current Biology*, 28(14):2250–2262.

Ivry, R. B. and Keele, S. W. (1989). Timing functions of the cerebellum. *Journal of cognitive neuroscience*, 1(2):136–152.

Izawa, J. and Shadmehr, R. (2008). On-line processing of uncertain information in visuomotor control. *Journal of Neuroscience*, 28(44):11360–11368.

Johnson, K. O. (2001). The roles and functions of cutaneous mechanoreceptors. *Current opinion in neurobiology*, 11(4):455–461.

Kaiser, E., Kutz, J. N., and Brunton, S. L. (2018). Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, 474(2219):20180335.

Kang, T., He, J., and Tillery, S. I. H. (2005). Determining natural arm configuration along a reaching trajectory. *Experimental brain research*, 167:352–361.

Kirk, D. E. (2004). *Optimal control theory: an introduction*. Courier Corporation.

Koch, C. and Segev, I. (1998). *Methods in neuronal modeling: from ions to networks*. MIT press.

Körding, K. P. and Wolpert, D. M. (2004). Bayesian integration in sensorimotor learning. *Nature*, 427(6971):244–247.

Krakauer, J. W. (2009). Motor learning and consolidation: the case of visuomotor rotation. In *Progress in motor control*, pages 405–421. Springer.

Krakauer, J. W., Pine, Z. M., Ghilardi, M.-F., and Ghez, C. (2000). Learning of visuomotor transformations for vectorial planning of reaching trajectories. *Journal of neuroscience*, 20(23):8916–8924.

Lackner, J. R. and Dizio, P. (1994). Rapid adaptation to coriolis force perturbations of arm trajectory. *Journal of neurophysiology*, 72(1):299–313.

Lee, C.-H. L., Liu, A., and Chen, W.-S. (2006). Pattern discovery of fuzzy time series for financial prediction. *IEEE Transactions on Knowledge and data Engineering*, 18(5):613–625.

Li, C.-S. R., Padoa-Schioppa, C., and Bizzi, E. (2001). Neuronal correlates of motor performance and motor learning in the primary motor cortex of monkeys adapting to an external force field. *Neuron*, 30(2):593–607.

Liepelt, R., Cramon, D., and Brass, M. (2008). What is matched in direct matching? intention attribution modulates motor priming. *Journal of Experimental Psychology: human perception and performance*, 34(3):578.

Lim, B. and Zohren, S. (2021). Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209.

Lu, D. and Weng, Q. (2007). A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28(5):823–870.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. (2021). Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228.

Lundquist, C., Sjanic, Z., and Gustafsson, F. (2015). *Statistical Sensor Fusion: Exercises*. Studentlitteratur AB.

Lusch, B., Kutz, J. N., and Brunton, S. L. (2018). Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):4950.

MacNeil, J. B., Kearney, R., and Hunter, I. (1992). Identification of time-varying biological systems from ensemble data (joint dynamics application). *IEEE Transactions on Biomedical Engineering*, 39(12):1213–1225.

Mante, V., Sussillo, D., Shenoy, K. V., and Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474):78–84.

Manto, M., Bower, J. M., Conforto, A. B., Delgado-García, J. M., Da Guarda, S. N. F., Gerwig, M., Habas, C., Hagura, N., Ivry, R. B., Mariën, P., et al. (2012). Consensus paper: roles of the cerebellum in motor control—the diversity of ideas on cerebellar involvement in movement. *The Cerebellum*, 11:457–487.

Masetty, B., Mirsky, R., Deshpande, A., Mauk, M., and Stone, P. (1989). Is the cerebellum a model-based reinforcement learning agent?

Mazzoni, P. and Krakauer, J. W. (2006). An implicit plan overrides an explicit strategy during visuomotor adaptation. *Journal of neuroscience*, 26(14):3642–3645.

Meidani, K. and Farimani, A. B. (2023). Identification of parametric dynamical systems using integer programming. *Expert Systems with Applications*, 219:119622.

Miall, R., Weir, D. J., Wolpert, D. M., and Stein, J. (1993). Is the cerebellum a smith predictor? *Journal of motor behavior*, 25(3):203–216.

Miall, R. C., Christensen, L. O., and Owen Cain, J. S. (2007). Disruption of state estimation in the human lateral cerebellum. *PLoS biology*, 5(11).

Miall, R. C. and Wolpert, D. M. (1996). Forward models for physiological motor control. *Neural networks*, 9(8):1265–1279.

Morgan, R. (2015). Linearization and stability analysis of nonlinear problems. *Rose-Hulman Undergraduate Mathematics Journal*, 16(2):5.

Murray, J. M. (2019). Local online learning in recurrent networks with random feedback. *Elife*, 8:e43299.

Nichols, T. and Houk, J. (1976). Improvement in linearity and regulation of stiffness that results from actions of stretch reflex. *journal of Neurophysiology*, 39(1):119–142.

Ohyama, T., Nores, W. L., Murphy, M., and Mauk, M. D. (2003). What the cerebellum computes. *Trends in neurosciences*, 26(4):222–227.

Olberg, R. M., Worthington, A. H., Fox, J. L., Bessette, C., and Loosemore, M. P. (2005). Prey size selection and distance estimation in foraging adult dragonflies. *Journal of comparative physiology A*, 191:791–797.

Padé, H. (1892). Sur la représentation approchée d'une fonction par des fractions rationnelles. In *Annales scientifiques de l'Ecole normale supérieure*, volume 9, pages 3–93.

Penfield, W. and Boldrey, E. (1937). Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation. *Brain*, 60(4):389–443.

Perich, M. G., Gallego, J. A., and Miller, L. E. (2018). A neural population mechanism for rapid learning. *Neuron*, 100(4):964–976.

Prilutsky, B. I. and Zatsiorsky, V. M. (2002). Optimization-based models of muscle coordination. *Exercise and sport sciences reviews*, 30(1):32.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2018). Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707.

Reuschel, J., Drewing, K., Henriques, D. Y., Rösler, F., and Fiehler, K. (2010). Optimal integration of visual and proprioceptive movement information for the perception of trajectory geometry. *Experimental brain research*, 201(4):853–862.

RobertC, M., Nicoll, and A, R. (1999). Long-term potentiation–a decade of progress? *Science*, 285(5435):1870–1874.

Rushworth, M., Nixon, P., and Passingham, R. (1997). Parietal cortex and movement i. movement selection and reaching. *Experimental brain research*, 117(2):292–310.

Sadtler, P. T., Quick, K. M., Golub, M. D., Chase, S. M., Ryu, S. I., Tyler-Kabara, E. C., Yu, B. M., and Batista, A. P. (2014). Neural constraints on learning. *Nature*, 512(7515):423–426.

Sanes, J. N. and Donoghue, J. P. (2000). Plasticity and primary motor cortex. *Annual review of neuroscience*, 23(1):393–415.

Saunders, J. A. and Knill, D. C. (2003). Humans use continuous visual feedback from the hand to control fast reaching movements. *Experimental brain research*, 152(3):341–352.

Scellier, B. and Bengio, Y. (2017). Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24.

Schaeffer, R., Khona, M., and Fiete, I. (2022). No free lunch from deep learning in neuroscience: A case study through models of the entorhinal-hippocampal circuit. *bioRxiv*, pages 2022–08.

Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28.

Scott, S. H. (2004). Optimal feedback control and the neural basis of volitional motor control. *Nature Reviews Neuroscience*, 5(7):532–545.

Sejnowski, T. J., Churchland, P. S., and Movshon, J. A. (2014). Putting big data to good use in neuroscience. *Nature neuroscience*, 17(11):1440–1441.

Shadmehr, R. and Mussa-Ivaldi, F. A. (1994). Adaptive representation of dynamics during learning of a motor task. *Journal of neuroscience*, 14(5):3208–3224.

Shadmehr, R. and Wise, S. P. (2004). *The computational neurobiology of reaching and pointing: a foundation for motor learning.* MIT press.

Slotine, J.-J. E. and Li, W. (1987). On the adaptive control of robot manipulators. *The international journal of robotics research*, 6(3):49–59.

Stengel, R. F. (1994). *Optimal control and estimation.* Courier Corporation.

Sussillo, D. and Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557.

Taub, E., Ellman, S. J., and Berman, A. (1966). Deafferentation in monkeys: effect on conditioned grasp response. *Science*, 151(3710):593–594.

Taylor, J. A. and Ivry, R. B. (2011). Flexible cognitive strategies during motor learning. *PLoS computational biology*, 7(3):e1001096.

Telgen, S., Parvin, D., and Diedrichsen, J. (2014). Mirror reversal and visual rotation are learned and consolidated via separate mechanisms: recalibrating or learning de novo? *Journal of Neuroscience*, 34(41):13768–13779.

Thrun, S. (2002). Probabilistic robotics. *Communications of the ACM*, 45(3):52–57.

Todorov, E. and Jordan, M. I. (2002). Optimal feedback control as a theory of motor coordination. *Nature neuroscience*, 5(11):1226–1235.

Uno, Y., Kawato, M., and Suzuki, R. (1989). Formation and control of optimal trajectory in human multijoint arm movement. *Biological cybernetics*, 61(2):89–101.

Vaidyanathan, N., Penny, S., and Berniker, M. (2020). Planned straight or biased to be so? the influence of visual feedback on reaching movements. *Journal of Motor Behavior*, 52(2):236–248.

van Beers, R. J., Wolpert, D. M., and Haggard, P. (2002). When feeling is more important than seeing in sensorimotor adaptation. *Current biology*, 12(10):834–837.

Vaziri, S., Diedrichsen, J., and Shadmehr, R. (2006). Why does the brain predict sensory consequences of oculomotor commands? optimal integration of the predicted and the actual sensory feedback. *Journal of Neuroscience*, 26(16):4188–4197.

Victor, J. D. and Purpura, K. P. (1996). Nature and precision of temporal coding in visual cortex: a metric-space analysis. *Journal of neurophysiology*, 76(2):1310–1326.

Voelker, A., Kajić, I., and Eliasmith, C. (2019). Legendre memory units: Continuous-time representation in recurrent neural networks. *Advances in neural information processing systems*, 32.

Von Helmholtz, H. (1867). *Handbuch der physiologischen Optik*, volume 9. Voss.

Wang, J. and Sainburg, R. L. (2004). Interlimb transfer of novel inertial dynamics is asymmetrical. *Journal of Neurophysiology*, 92(1):349–360.

Warrant, E. (2004). Vision in the dimmest habitats on earth. *Journal of Comparative Physiology A*, 190:765–789.

Wehmeyer, C. and Noé, F. (2018). Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *The Journal of chemical physics*, 148(24):241703.

Williams, M. O., Kevrekidis, I. G., and Rowley, C. W. (2015). A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25:1307–1346.

Wise, S., Moody, S., Blomstrom, K., and Mitz, A. (1998). Changes in motor cortical activity during visuomotor adaptation. *Experimental Brain Research*, 121(3):285–299.

Wise, S. P., Boussaoud, D., Johnson, P. B., and Caminiti, R. (1997). Premotor and parietal cortex: corticocortical connectivity and combinatorial computations. *Annual review of neuroscience*, 20(1):25–42.

Wolpert, D. M., Ghahramani, Z., and Jordan, M. I. (1995). An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882.

Wolpert, D. M., Goodbody, S. J., and Husain, M. (1998a). Maintaining internal representations: the role of the human superior parietal lobe. *Nature neuroscience*, 1(6):529–533.

Wolpert, D. M., Miall, R. C., and Kawato, M. (1998b). Internal models in the cerebellum. *Trends in cognitive sciences*, 2(9):338–347.

Yu, W., Tan, J., Liu, C. K., and Turk, G. (2017). Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*.