DATA-DRIVEN APPROACHES TO MODEL PREDICTIVE CONTROL OF NEURAL SYSTEMS

Christof Kane Fehrman

Charlottesville, Virginia

Bachelor of Science, Middle Tennessee State University, 2015

Master of Arts, Middle Tennessee State University, 2017

A Dissertation submitted to the Graduate Faculty
of the University of Virginia in Candidacy for the Degree of
Doctor of Philosophy

Department of Psychology

University of Virginia

May 2024

Data-Driven Approaches to Model Predictive Control of Neural Systems

Christof Kane Fehrman

(ABSTRACT)

Achieving precise control of neural activity is a major focus of modern neuroscience. This challenge persists due to the diverse nonlinearities in neuronal dynamics and the largely unknown biophysical mechanisms governing large populations of neurons. Increasing our ability to control these systems would be of enormous benefit, both in terms of basic theory and clinical applications. Model Predictive Control (MPC) is a powerful control technique that uses mathematical models of a system to find optimal inputs to get a desired output. Using data-driven methods, models can be empirically fit to neural activity for use in MPC. This allows for control of a neural system even when there is very limited knowledge about the dynamics present. This dissertation provides a framework for how these data-driven models can be fit and utilized for MPC of neural systems without needing extensive *a priori* information.

Dedication

To my ladies, Melissa and Marbles.

Acknowledgments

I want to acknowledge first and foremost the unwavering love and support from my wife Melissa. Without her, I would be lost. I can't wait to see what the future holds for us (besides more cats). I would also like to thank my Mom and Dad, not only for being great parents but for always encouraging a curiosity of the world. Thank you both for the many wonderful days of finding snakes under rocks, and the nights spent naming the constellations. I of course need acknowledge my advisor Dan Meliza. It is an understatement to say that joining his lab was the most important and best decision I made in graduate school. I can't imagine being in any other lab. I will never be able to thank him enough for his mentorship, encouragement, and humor. Finally, I would like to thank Jacy and Milan for the countless days of 'band' practice that turned into calculus lessons on a dirty white board. Simply put, they're the best.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Control Theory in Neuroscience

Precise control of neural systems is a major focus of modern neuroscience, both as a means for experimental investigation of the brain and as a clinical method for treating neurological disorders. In the most general sense, how can we make the system do what we want it to do? The 'neural system' could be from a wide variety of structures in the nervous system. For example, we may want to control the membrane voltage of a single neuron so that it fires in accordance with a particular spike train. On a larger scale, we may want to force a population of neurons to produce activity that elicits a desired behavior from the organism as a whole. This level of control would allow us to experimentally test the predictions of hypotheses in neural systems and potentially restore normal function to a circuit that has gone into a pathological state.

If we take a dynamical systems approach to neuroscience, the field of control theory gives us the vocabulary and tools needed to deal with these problems. In this view, neural systems are modeled as a set of differential equations of the general form,

$$\frac{dx}{dt} = f(x(t), u(t)) \tag{1.1}$$

Many passages and figures in this dissertation are freely adapted from the preprint Fehrman and C Daniel Meliza 2023. In particular, Chapter 2 contains much of the content found in this article.

where $x(t)$ is the trajectory of system state and may be a scalar or vector-valued function. At any given point in time, the state error is given by

$$e(t) = x_{ref}(t) - x(t) \tag{1.2}$$

where $x_{ref}(t)$ is the reference trajectory and is the desired behavior of the system. The problem of control can now be formalized as finding some command signal $u(t)$ such that it will force the state of the system to follow the reference trajectory.

Broadly speaking, there are two strategies for finding the command input $u(t)$: open- and closed-loop control. In open-loop control, the command signal is chosen ahead of time based on a general model of the system and then applied to an individual instance (Figure 1.1 **A**). The outcome may provide additional information on how to update the general model, but this occurs offline. In neurophysiology, examples of open-loop control include current-clamp intracellular recording as well as most optogenetics experiments, where a pulse of current or light is used as the command signal to force a neuron to spike or prevent it from spiking (Emiliani et al. 2022). What makes these open-loop is that the intensity and duration of the command signal is not automatically adjusted if the stimulus fails to achieve the desired effect (Grosenick, Marshel, and Deisseroth 2015). Although open-loop control has the advantages of being fast and simple to implement, it is not robust to unknown disturbances or errors in command signal calculation (Zaaimi et al. 2022). Because neurons *in vivo* receive many spontaneously active excitatory and inhibitory inputs, there may be significant trial-to-trial variability in the number of spikes evoked during application of the command signal. More broadly, variability in the actual effects of a manipulation reduces the power to make causal inferences in experimental settings.

In contrast, for closed-loop (or feedback) control, the command signal is dynamically adjusted as a function of the difference between the actual (or estimated) state of a specific system and the desired reference trajectory (Figure 1.1 **B**). This can be expressed as

$$u(t) = g(e(t)) \tag{1.3}$$

where the function $g(.)$ acts as the controller for the system, turning state errors into control inputs. A common (but limited) implementation of feedback control is with a proportional controller,

$$u(t) = Ke(t) \tag{1.4}$$

where K is a gain hyperparameter. This is the approach employed in voltage-clamp experiments, where the difference between the actual and the desired membrane voltage (the state error) is scaled by a gain factor and used as the command signal of electrical current injected into the neuron (Nowotny and Levi 2014). Closed-loop controllers have the ability to adapt to unknown system disturbances and changes in system dynamics even when tracking complicated reference trajectories (Stefani 2002). Because of this, considerable work has gone into incorporating feedback controllers in other areas of neuroscience such as brain-machine interfaces (Shanechi, Orsborn, Moorman, et al. 2017; Gilja et al. 2012; Willett et al. 2017; Zhang et al. 2021) and neuro-prosthetics (Shanechi, Orsborn, and Carmena 2016; Shanechi, Orsborn, Moorman, et al. 2017; Cunningham et al. 2011; Wright et al. 2016; Pandarinath and Bensmaia 2022; Pedrocchi et al. 2006). In particular, there have been recent advancements in using feedback controllers with optogenetic stimulation to give more fine-tuned and reliable control of neural spiking at both the individual neuron (Bolus et al. 2021) and population levels (Bergs et al. 2023; Newman et al. 2015).

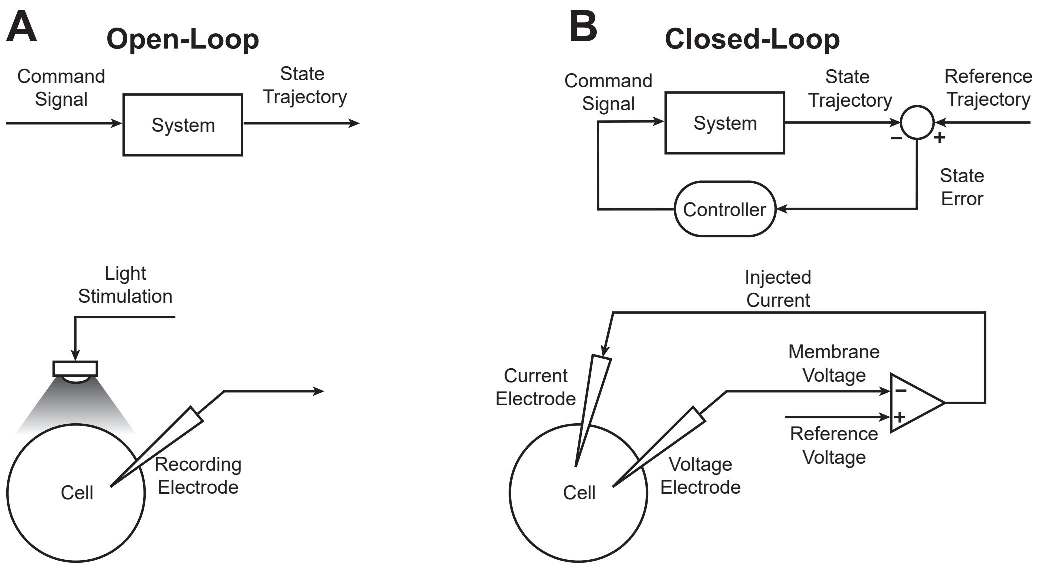


Figure 1.1: **Open- vs Closed-Loop Control**. **A)** (Above) Block diagram of open-loop control. A command signal is applied to the system irrespective of the system output. (Below) Diagram of typical optogenetic stimulation experiment. A light source is applied to a cell expressing the appropriate light-sensitive opsin. For open-loop stimulation, the intensity of the light is determined before recording and may or may not cause the cell to fire. **B)** (Above) Block diagram of closed-loop control. A command signal is calculated by the controller based on the state error - the difference between the system state trajectory and reference trajectory. (Below) Diagram of a voltage-clamp experiment. The cell is held at a specified voltage by injecting current determined by an online comparison of the membrane voltage with the desired reference voltage.

Although a promising avenue of research, there are still many issues when using feedback controllers with complicated systems. Most implementations of feedback control are purely *reactive*, where the command signal is a function of the present and/or past state error terms. Reactive controllers are often sensitive to abrupt changes in the reference trajectory and time delays in the system (Afram and Janabi-Sharifi 2014). Another issue with classical feedback control schemes is that the technique is either rigorously justified only for linear systems or requires strong assumptions about the structure of the nonlinearities of the system (Zhao and Guo 2022). These challenges are particularly prominent in the control of neural systems, because the desired behavior usually has abrupt changes in state (i.e., spikes) and the systems exhibit strong and diverse nonlinear dynamics (Johnston and S. M.-s. Wu 1995; Bjoring and C. Daniel Meliza 2019; Chen and C. Daniel Meliza 2018; Martinez et al. 2023). Additionally, having only reactive controllers would be detrimental to problems of preventing pathological neural activity. As a concrete example, this would mean we could only try and stop an epileptic seizure after it begins instead of preventing it from even occurring. In order to have greater control over complex neural systems, more sophisticated feedback control schemes are needed.

1.2 Model Predictive Control

One promising method of feedback control to deal with these problems is model predictive control (MPC), which is a type of optimal controller. It is optimal in the sense that the control input u minimizes an objective function of the form

$$J(x_0) = \sum_{i=0}^{T} \ell(x_i, u_i), \tag{1.5}$$

with constraints

$$\begin{aligned}
x_{n+1} &= f(x_n, u_n) \\
x_{LB} &\leq x \leq x_{UB} \\
u_{LB} &\leq u \leq u_{UB},
\end{aligned}$$

where $\ell(x_i, u_i)$ is the loss associated with ith time step, which is a function of the state variable(s) x and input(s) u. Many types of loss functions are possible, but typically involve the state error and energy cost of the command signal. The constraints allow one to specify the dynamics of the system and to give lower and upper bounds for the state variables and inputs. More sophisticated versions of MPC allow for additional constraints where knowledge of any measurement or process noise can be incorporated (Hewing et al. 2020).

The controller uses a discrete-time model of the system $f(x_n, u_n)$ to predict what command inputs would best force the system to follow the reference trajectory over some time horizon T (Figure 1.2). At each time step, the controller finds an optimal sequence of command signals by minimizing the total loss given the constraints. The total loss is calculated by summing the actual and predicted losses across the time horizon. However, only the first time step in the optimized sequence is applied to the system, and the optimization is performed again in the next time step. This process repeats at each discrete time step, which leads some to refer to MPC as receding horizon control (Holkar and Waghmare 2010). By finding an optimal input based on predictions of how the system will behave in the future, MPC is an *anticipatory* controller (Lin et al. 2023). Although the command signal is only guaranteed to be globally optimal for linear systems with convex loss functions, MPC has been widely used in nonlinear system control (Raković and Levine 2019; Steven L. Brunton and Jose Nathan Kutz 2019).

A commonly used analogy to describe MPC is the game of chess (Raković and Levine 2019), where the player (the controller) wants to find a set of moves (the command signal) to win the game (the objective function). When selecting a move, the player must use a model of their opponent to anticipate how that opponent will respond to their moves. Although the player may have mapped out their moves for the next T turns (the time horizon), the player can only implement the first of these moves during their turn. The player may update their planned moves based on a variety of factors. Their opponent may have selected a different move than predicted, or after completing their turn the player is able to think one more move ahead (the receding horizon) and finds a new optimal set of moves. Intuitively, being able to think more moves ahead (extending the length of the time horizon) should produce a more optimal set of moves to win the game. However, this comes at the cost of increased computational complexity for the player, and errors in modeling how the opponent will respond can accumulate when incorporating these errors across the extended time horizon. This leads to a balancing act in MPC where not having a large enough time horizon may result in suboptimal moves in the long run, whereas too long of a time horizon is expensive and sensitive to modeling errors.

One of the primary considerations in implementing MPC is choosing a good model of the system one wants to control (Schwenzer et al. 2021). At first glance, this might not seem to be a problem for applications in neuroscience since constructing mathematical models of neural systems is one of the main research areas. For example, models based on voltage-dependent ionic conductances using the Hodgkin-Huxley

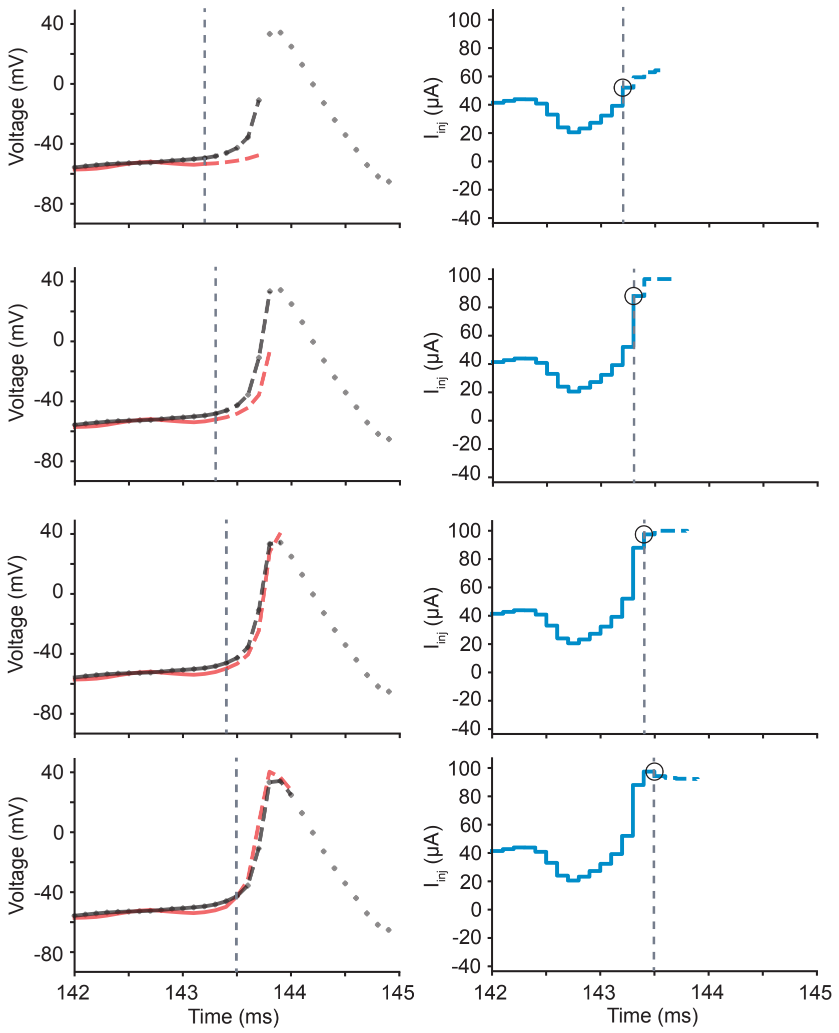


Figure 1.2: **Receding Horizon of MPC.** Starting at the top row, (left) the system state trajectory (red) is being controlled to follow the reference trajectory (black). At the current time step (vertical dotted line) the controller finds the optimal set of inputs that minimize the loss function for a specified time horizon. In this case, the controller looks ahead 5 time steps (black dashed curve). Given the state at the current time, the controller uses a model to predict where the system will be across the future time horizon (red dashed curve). The inputs into the system (right) are optimized in discrete-time, and the input into the system is held constant between model time steps (solid blue curve). The predicted optimal future inputs (dashed blue curve) are calculated across the future time horizon. However, only the first of these values (circled in black) is used as input in the next time step before the optimization procedure begins again. From the top to bottom rows, we see how the controller may pick new optimal inputs given updates in the model predictions and by having access to new reference trajectory values (black dotted curve).

(HH) framework can accurately predict how the membrane voltage of a neuron with a given morphology and complement of currents will respond to an arbitrary input. However, building a conductance model of a specific neuron is far from trivial (Rabinovich et al. 2006). The types of currents must be chosen along with dozens to hundreds of free parameters that govern the maximal conductances of the intrinsic currents and their voltage-dependent kinetics, and there are many state variables of which only the membrane voltage is typically observable (Toth et al. 2011). MPC has been successfully applied to conductance models in previous work (Fröhlich and Jezernik 2005; Yue, Tomastik, and Dutta 2022; Senthilvelmurugan and Subbian 2023), but always with the assumption that the number, type, and/or functional forms of all the intrinsic currents are known *a priori*. Under this assumption, there are many data assimilation methods that can be used to estimate the unknown parameters and hidden states of the model (Toth et al. 2011; Kostuk et al. 2012; Ullah and Schiff 2009). However, in most biological preparations, this is an unrealistic assumption, because neurons express a large, diverse complement of voltage-gated channels whose physiological properties can depend on variations in isoform composition, modulatory subunits, phosphorylation state, and subcellular localization. Choosing even a general form of a model to be used with a specific type of cell requires significant hand-tuning (C Daniel Meliza et al. 2014) that would not be feasible if the goal is to control a specific neuron or network in a live experiment. This issue is only compounded when trying to model larger neural systems where precise control over population dynamics is desired. Relatively fewer biophysically detailed models exist for neural circuits compared to individual neurons and those that do would have even more parameters to estimate than their HH counterparts.

1.2.1 Data-Driven Approaches to Modeling

An alternative to constructing a detailed biophysical model is to use a data-driven approach where the dynamics of the system are modeled based on empirical data with minimal reference to the underlying biology of the neuron. Using standard machine-learning approaches, unknown parts of the system can be modeled via function approximation and used to predict the time-evolution of the system in response to various inputs.

Fitting these models is achieved by observing a temporal sequence of the state and input variables with some sampling period Δt,

$$\mathbf{X} = [x_0, x_1, ..., x_N], \mathbf{U} = [u_0, u_1, ..., u_N] \tag{1.6}$$

where $x_n = x(n\Delta t)$. A discrete-time model can be parameterized such that,

$$\hat{x}_{n+1} = f_\theta(x_n, u_n) \tag{1.7}$$

These types models are often referred to as forecasting models since the model predicts how the system will change across time. The goal is to find a set of parameters θ such that given some initial state value x_0,

$$[x_0, \hat{x}_1, ..., \hat{x}_N] \approx [x_0, x_1, ..., x_N] \tag{1.8}$$

for any general temporal data sequence produced from the true dynamical system.

Data-driven approaches have been successfully applied to MPC problems in diverse fields (Bieker et al. 2019; Kaiser, J. N. Kutz, and S. L. Brunton 2018; Hewing et al.

2020; Salzmann et al. 2023; Zheng and Z. Wu 2023) and the field is rapidly growing. In order for data-driven models to be useful for MPC applications in neuroscience, these models must be able to accurately predict the states to be controlled based only on observable state measurements, be agnostic to the number of hidden states, and generalize to a control scheme where command signals may be outside the training set. These challenges are not unique to neuroscience, but are still important to consider when selecting a data-driven approach to model the system dynamics. However, there are several unresolved issues that have prevented the widespread adoption of MPC for neural systems control. Neural activity often occurs at very fast time scales ($\sim$ milliseconds) which creates a computational issues since the optimization step of MPC must be completed at every discrete time point. While a rigorous treatment of hardware and software improvements for increased controller speed is beyond the scope of this dissertation, there will be an emphasis on data-driven model simplicity to reduce the number of calculations needed at every time step of the optimizer. Another challenge is the high dimensionality of states in neural populations and of naturalistic stimuli which may be used as the source of a command signal. This is not only an issue in terms of computational complexity for optimization problem but also increases the difficultly of preventing overfitting in the data-driven dynamics model.

1.3 Aim of Dissertation

In this dissertation, I demonstrate that MPC of complex neural systems is possible using data-driven dynamics models derived from observationally limited sources of neuronal measurement. I first show that MPC can be applied to control the voltage of a biophysical conductance-based neuron model. This will provide the fundamental framework for how data-driven dynamics models can be constructed using only observations that would be realistically obtainable in an *in vivo* or *in vitro* experiment. I then extend these results to control of a spiking neural network which simulates the activity of a large circuit of neurons that are driven by exogenous naturalistic images. Population level control of the network is a achieved by controlling the latent dynamics of the network rather than the activity in the original measurement space. I show that this not only has beneficial practical results (i.e. reduced model complexity) but also allows for future experiments interrogating the nature and function of neural manifolds. In all simulations, there is an emphasis on how MPC can be achieved using experimental designs that are currently feasible in terms of both measurement and system perturbation.

Chapter 2

MPC of a Single Conductance-Based Neuron Model

As a proof of principle, I conducted a simulation study to control the membrane voltage of an HH-type neuron through current injection when the parameters of the model were unknown, and only the membrane voltage was observable. These observations were used to construct a nonlinear data-driven dynamics model for MPC that accurately predicted the response of the system to command signal inputs. While there has been previous work using these approaches to model HH-type neurons, the models were either used solely for prediction instead of control (Plaster and Kumar 2019), or made unrealistic assumptions about which state variables were available in the training data and the extent to which the complement and functional forms of the intrinsic currents could be known (Yue, Tomastik, and Dutta 2022; Senthilvelmurugan and Subbian 2023). The model made no assumptions about the nature of the intrinsic currents and still allowed the controller to force the membrane voltage to follow a reference trajectory. Although control of single unit voltage activity is achievable with proportional feedback control both *in vivo* and *in vitro* (Sherman-Gold 2012), the goal was to demonstrate how data-driven modeling can be applied to nonlinear MPC of neural systems.

2.1 Connor-Stevens Model

As a model of single-unit responses to an injected current, the HH-type Connor-Stevens (CS) model was used for all simulations (Sterratt 2011). The CS model includes four intrinsic currents and an extrinsic injected current. A noise current was also included that modeled the unknown, variable synaptic inputs that contribute to the trial-to-trial variability neurons tend to exhibit *in vivo*. The model is given by the equations

$$C\frac{dV}{dt} = I_{Na} + I_K + I_A + I_l + I_{noise} + I_{inj}, \tag{2.1}$$

where

$$\begin{aligned} I_{Na} &= g_{Na} m^3 h(E_{Na} - V) \\ I_K &= g_K n^4 (E_K - V) \\ I_A &= g_A a^3 b(E_A - V) \\ I_l &= g_l(E_l - V), \end{aligned}$$

where I_{Na}, I_K, and I_A are the voltage-dependent sodium, potassium, and A-type intrinsic ionic currents, and I_l is the intrinsic leak current. For the noise current I_{noise}, pink noise was used since it has been shown to model neuron response stochasticity (Destexhe et al. 2001). The activity of the neuron can be externally modulated by varying the injected current I_{inj}. Note that all currents are functions of time but I omit making this explicit in the equations for simplicity. Each of the three voltage-gated currents depend on one or more unobservable state variables that model the activation state (m, n, a) and inactivation state (h, b) of the channels. Each of these state variables is governed by a first-order differential equation with unique parameters that determine its kinetics. While in principle one could estimate the values of the state variables and model parameters using data assimilation techniques (Toth et al. 2011; Kostuk et al. 2012; Knowlton et al. 2014), in an actual biological preparation one would be unlikely to know all of the channels a specific neuron expresses.

The CS model is able to produce distinct firing dynamics by changing the parameter E_l and g_A parameter values. With $g_A = 47.7$ mS and $E_l = -22$ mV, the model exhibits Type-I excitability, which is characterized by a smooth increase in firing rate when the input currents exceed the firing threshold. When $g_A = 0$ mS (eliminating the A-type current) and $E_l = -72.8$ mV, the model instead exhibits Type-II excitability, which is characterized by a discontinuous jump in firing rate for input currents that exceed the firing threshold. This difference in spiking behavior reflects two distinct dynamical topologies that undergo qualitatively different kinds of bifurcations: Type-I spiking is indicative of a saddle-node bifurcation, whereas Type-II spiking is caused by an Andronov-Hopf bifurcation. To show that data-driven approaches to MPC can extend to various types of neural dynamics and number of intrinsic currents, both the Type-I and Type-II CS models were used in all simulated experiments.

2.2 Data-Driven Forecasting of CS Model

The HH model and its variants are conductance-based models where the cell membrane is modeled as a capacitor (Skinner 2006). Thus, the relationship between the membrane voltage and cellular currents can be expressed using the current conservation equation

$$C\frac{dV}{dt} = \sum_i I_i(t),$$

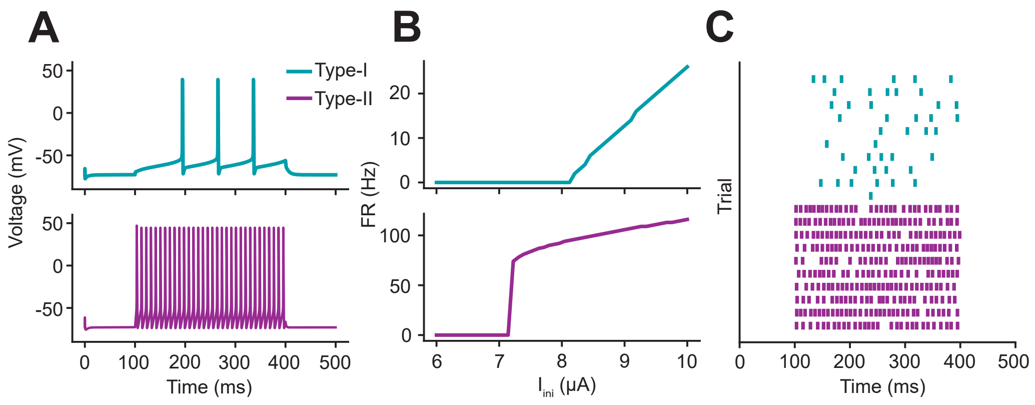


Figure 2.1: **CS Model Behavior. A)** The spiking pattern of a Type-I (above) and Type-II (below) CS model in response to a 300 ms 8.5 μA step current. **B)** The firing rate of the CS model as a function of step current amplitude. Notice that the Type-I model's firing rate increases approximately linearly after the input passes the firing threshold while the Type-II model abruptly jumps in firing rate. **C)** The effect of the noise current on the CS models when stimulated with the same step current from panel A.

where the time derivative of the membrane voltage V is proportional to the sum of all currents through the membrane. These currents may be externally applied (e.g. electrode injected currents) or intrinsic to the neural dynamics themselves (e.g. arising from voltage- and ligand-gated ion channels). To construct a forecasting model, we only assumed the dynamics of the membrane voltage were given by

$$C\frac{dV}{dt} = F(V, X, \Theta, t) + I_{inj}, \tag{2.2}$$

where C is the membrane capacitance and $F(.)$ is an unknown time-varying function of membrane voltage, with unknown states X and unknown parameters Θ. This assumption would hold not only for the CS model, but any conductance-based model because of the additivity of the currents. For the CS model, this $F(.)$ would be the intrinsic currents, X would be the intrinsic state variables, and Θ would be the model parameters. The goal in data-driven forecasting (DDF) is not to estimate these unknowns but to approximate F such that one can accurately predict how the neuron will respond to an arbitrary input current I_{inj} by integrating Equation (2.2).

Let $\mathbf{V} = [V_0, V_1, ..., V_N]$ denote a set of discretely sampled membrane voltages where $V_n = V(n\Delta t)$ and Δt is the sampling period. Similarly, let $\mathbf{I} = [I_0, I_1, ..., I_N]$ be the set of discretely sampled injected currents. Given only $\mathbf{V}$ and $\mathbf{I}$, the goal is to find a DDF model of the form $V_{n+1} = F_{DDF}(\mathbf{V}, \mathbf{I})$ that can accurately map V_n to V_{n+1}. There are many possible models $F_{DDF}(.)$ to choose from and as a general rule demand larger amounts of training data as the DDF model gets more complex (Bourdeau et al. 2019). Additionally, if one used both the membrane voltage and injected currents as input into black-box function approximator, it would be difficult to separate the effects of the intrinsic dynamics of the system (membrane voltage) from the effects of the external force (injected current). An approach taken by (Clark, Fuller, et al. 2022) when modeling the dynamics of HH-type neurons was to exploit the fact that the I_{inj} term is additive and remove that from the function approximation step. In (Clark, Fuller, et al. 2022), they were able to achieve a good forecasting model by using time-embeddings of $\mathbf{V}$ in conjunction with a radial basis function network (RBFN). Although this is a classical approach largely superseded by more modern forecasting models such as LSTMs and Transformers, their DDF model generalized to *in vitro* recordings across many different neuron types. In contrast to more complex models, RBFNs are easier to train while still being universal function approximators (Park and Sandberg 1991). The general form of this DDF model is given by the equation

$$V_{n+1} = V_n + F_{RBF}(S_n) + \alpha(I_{n+1} + I_n), \tag{2.3}$$

where V_n is the membrane voltage at the nth time sample, S_n is a time-embedding of V_n, $F_{RBF}(.)$ is a RBFN with learned parameters, I_n is the injected current at the nth time sample, and α is a learned scaling parameter. See (Clark, Fuller, et al. 2022; Clark, Fairbanks, et al. 2022) for a more detailed treatment.

2.2.1 Simulating Training Data

Separate DDF models were trained on data from the Type-I and Type-II CS neuron models. The injected currents used to stimulate the CS neurons were obtained from the Lorenz63 system (Lorenz 1963). This chaotic current has been shown to cover a large frequency spectrum and has been used to drive *in vitro* neurons across a sufficient extent of their state space to support accurate data assimilation (Toth et al. 2011). All simulations were performed using `scipy.integrate.odeint` with a time window $h = 0.02$ ms. Five seconds of simulated data were used as training data for each of the DDF models. Because MPC is computationally expensive, we would not expect to be able to run the optimization process (Figure 2.3, blue loop) at the sampling rate of the recording, and so we down-sampled the membrane voltage and injected currents to 10 kHz, which corresponds to a sampling period of $\Delta t = 0.1$ ms. This is a relatively low sampling rate for voltage-clamp experiments and demonstrates we are still able to control these systems with less data than typically used to build biophysical conductance-based models.

2.2.2 Choosing RBFN Hyperparameters

A Gaussian was used as the radial basis function in the RBFNs, which is of the form

$$\psi_c(S_n) = exp\{-R||S_n - \mu_c||^2\}. \tag{2.4}$$

The time-embedding was chosen to have an embedding dimension D_e of 3 and time delay τ of 1 for both DDF models (i.e., $S_n = [V_n, V_{n-1}, V_{n-2}]$). The center vectors μ_c (N = 500) were obtained by performing k-means clustering in the time-embedded space of the training data. For all RBFs, a scaling parameter $R = 0.1$ was used.

2.2.3 Training RBFN

The RBFNs were trained via Ridge regression (also known as Tikhonov regularization) with the solution given by

$$W = (X^T X + \lambda I)^{-1} X^T Y, \tag{2.5}$$

where

$$Y = \begin{bmatrix} V_1 - V_0 \\ V_2 - V_1 \\ \vdots \\ V_N - V_{N-1} \end{bmatrix}, X = \begin{bmatrix} \psi_1(S_0) & \psi_2(S_0) & \dots & \psi_N(S_0) & I_1 + I_0 \\ \psi_1(S_1) & \psi_2(S_1) & \dots & \psi_N(S_1) & I_2 + I_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \psi_1(S_{N-1}) & \psi_2(S_{N-1}) & \dots & \psi_N(S_{N-1}) & I_N + I_{N-1} \end{bmatrix}, W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ \alpha \end{bmatrix}. \tag{2.6}$$

Model training was performed using the `sklearn` python package (Pedregosa et al. 2011) with 10-fold cross-validation to obtain an optimal λ regularization parameter.

2.2.4 Validating Forecasting Model

Although previous work has shown that the DDF model has high accuracy for *in silico* and *in vitro* neurons (Clark, Fuller, et al. 2022), the sampling rate was much higher than data used here ($\geq$ 50 kHz). To assess whether DDF would work on

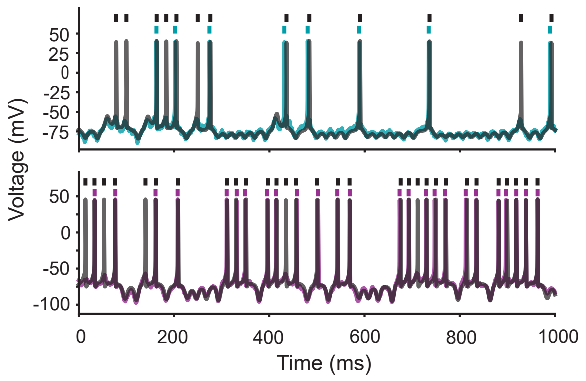


Figure 2.2: **DDF Model Forecasts.** Each of the DDF models were fit with five seconds of training data and evaluated based on their ability to accurately predict the voltage and spiking behavior of the corresponding CS model to testing input current. The forecasts are completely open-loop, where the DDF model does not get corrected based on errors in predictions. The spike trains of the CS model (colored rasters) and DDF model (black rasters) are shown above the voltage predictions. (Above) Predicted membrane voltage and rasters of the Type-I DDF model in response to one second of testing data. (Below) Predicted membrane voltage and rasters of the Type-II DDF model in response to one second of testing data.

the CS model using data with a lower sampling rate more in line with the control loop speed we might expect to achieve in a live, biological preparation, open-loop forecasting was performed on novel injected currents. Because the DDF models had a time-embedding parameters $D_e = 3$ and $\tau = 1$, the first three Δt time samples were used to seed the model.

2.3 Model Predictive Control of CS Neuron

Controlling the membrane voltage of a CS neuron via MPC was performed by finding an optimal set of injected current inputs that minimized the cost function

$$J(V_0) = \mathbf{s}e_T^2 + \sum_{n=0}^{T-1} \mathbf{q}e_n^2 + \mathbf{r}\Delta I_{n+1}^2, \tag{2.7}$$

subject to the constraints

$$V_{n+1} = V_n + F_{RBF}(S_n) + \alpha(I_{n+1} + I_n)$$

$$|I_n| \leq 100\,\mu A, \tag{2.8}$$

where V_0 is the membrane voltage at the current time step, e_n is the error between the membrane voltage V_n and the reference trajectory V_n^{ref} at the nth time step relative to V_0, and $\Delta I_{n+1} = I_{n+1} - I_n$ (also relative to V_0). Note that solving this optimization problem corresponds to controlling the I_{n+1} term in the DDF model. For the very first optimization loop, I_0 was set to 0 μA. The controller hyperparameters $\mathbf{s}, \mathbf{q}$ and $\mathbf{r}$ allow one to differentially weight errors in control and errors in input fluctuations. Setting $\mathbf{r} = 0$ can result in rapid input fluctuations which may make the controller perform poorly (Qin and Badgwell 2003).

At the beginning of the control loop, the controller uses a model of the system to simulate T time steps into the future in order to find the optimal set of inputs to minimize the cost function. Recall that the DDF model is working in 0.1 ms time steps resulting in the control input I_{inj} applied to the CS neurons being kept constant for that time window. Reducing the width of this window would enable one to control systems at faster time scales but at the cost of increased computational load.

All MPC optimizations and implementations were performed using the `do-mpc` python package (Fiedler et al. 2023). This package utilizes CasADi (Andersson et al. 2019) and IPOPT (Wächter and Biegler 2006) for interior-point optimization and automatic differentiation methods. All controllers used the following controller hyperparameters: $T = 5, \mathbf{s} = 5, \mathbf{q} = 1, \mathbf{r} = 0.5$.

Note that the range in which the control input operates is higher than is typical of patch-clamp experiments (Sherman-Gold 2012). The CS model has parameters that are normalized by soma surface area which result in different conductance and capacitance values when modeling cells of different sizes. For simplicity all CS models corresponded to a neuron with a soma surface area of 1cm^2. In practice, the size of the neuron would have a significant impact on the magnitudes of the input currents used. Larger cells will require larger values of input current to produce meaningful changes in the membrane voltage compared to smaller cells (Sterratt 2011) and smaller cells would not survive larger injected currents. However, in a real patch clamp scenario the researcher would know a reasonable range of voltages that would produce neural firing. Given enough data, the DDF model would be able to learn the relationship between the magnitude of input needed to drive the neuron without needing an estimate of the soma size.

2.4 Results

2.4.1 Experiment I: Homogeneous System Control

The first test of MPC for neural control was to force a CS neuron to reproduce a previously recorded voltage trajectory (the reference trajectory V^{ref}). This is referred to as homogeneous system control because each CS model was forced to track a reference trajectory it previously produced. For each CS model, 50 trials were simulated of 1-second responses to a chaotic current similar to the one used in the training data. MPC was then performed with the corresponding DDF model to find I_{inj} such that the errors in state tracking were minimized, thereby forcing the CS model to

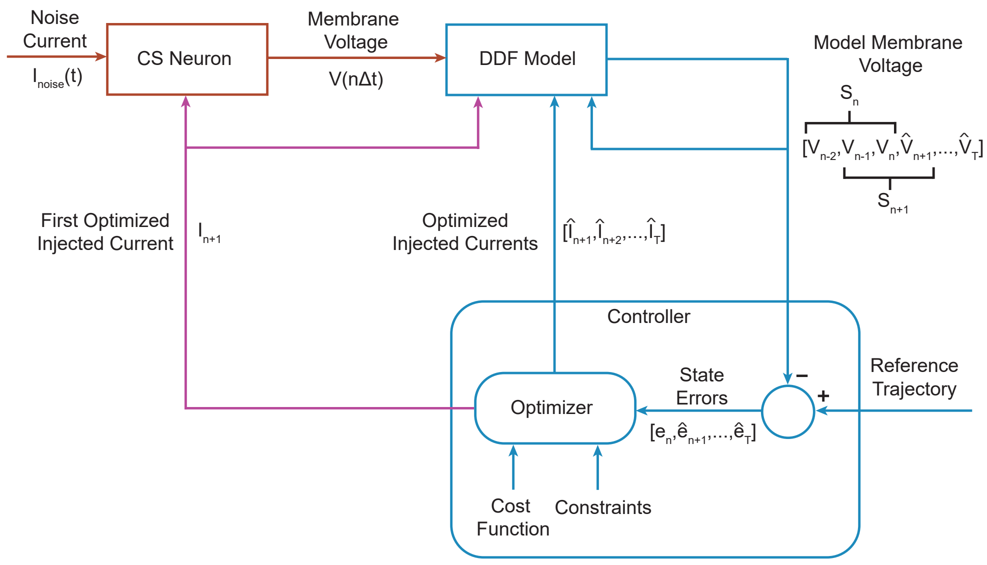


Figure 2.3: **MPC via DDF Diagram.** (Red) The CS neuron receives an input $I_{inj}(t)$ from the controller at time step n which is held constant across a time interval of Δt seconds. (Blue) The DDF model gets an update of the CS model membrane voltage every Δt seconds. Given the membrane voltage V_n, time-embedded state history S_n, and discrete-time input I_n, the controller finds an optimal input for the next time step I_{n+1}. Given this optimized input, the DDF model makes a prediction of the CS model membrane voltage at the next Δt time step, V_{n+1}. The controller uses the DDF model to simulate 5 Δt time steps into the future (the time horizon) to find a sequence of optimized inputs by minimizing the loss function. (Purple) The first of these optimized inputs I_{n+1} is used as the next injected current into the CS neuron.

reproduce each of these 50 trials.

As a control, each trial was performed in an open-loop condition; that is, by injecting the same input current that produced the reference trajectory. If the neuron were deterministic, then the voltage trace would perfectly match the reference. However, because I_{noise} varies in each trial, the same injected current will not produce the same voltage trace or pattern of spiking. Thus, the open-loop condition gives a reference for the amount of variability we would expect to see without feedback control.

In order to quantify how well the MPC and open-loop control performed, three measures of fit were used: MSE, ISI-distance, and spike-distance. The MSE was calculated by comparing the reference trajectory V^{ref} with the voltage trajectory V the controlled CS model produced. The ISI- and spike-distances are measures of spike train similarity (Mulansky and Kreuz 2016). Rather than directly comparing the trajectories, these measures use the times that the CS model spiked and compare them to corresponding spikes in V^{ref}. Spike times were obtained by recording when the voltage exceeded a threshold (30 mV). The ISI-distance measures the similarity between the inter-spike intervals (ISI) of two spike trains, and the spike-distance measures the similarity of the spike timing between the two spike trains.

As seen in Figure 2.4, MPC performed much better than open-loop control for both CS model types. This is clear from visual inspection of the membrane voltages and from the quantitative measures of performance. This result is remarkable because the underlying DDF model in the controller does not have any knowledge of the biophysical details of the system it is controlling.

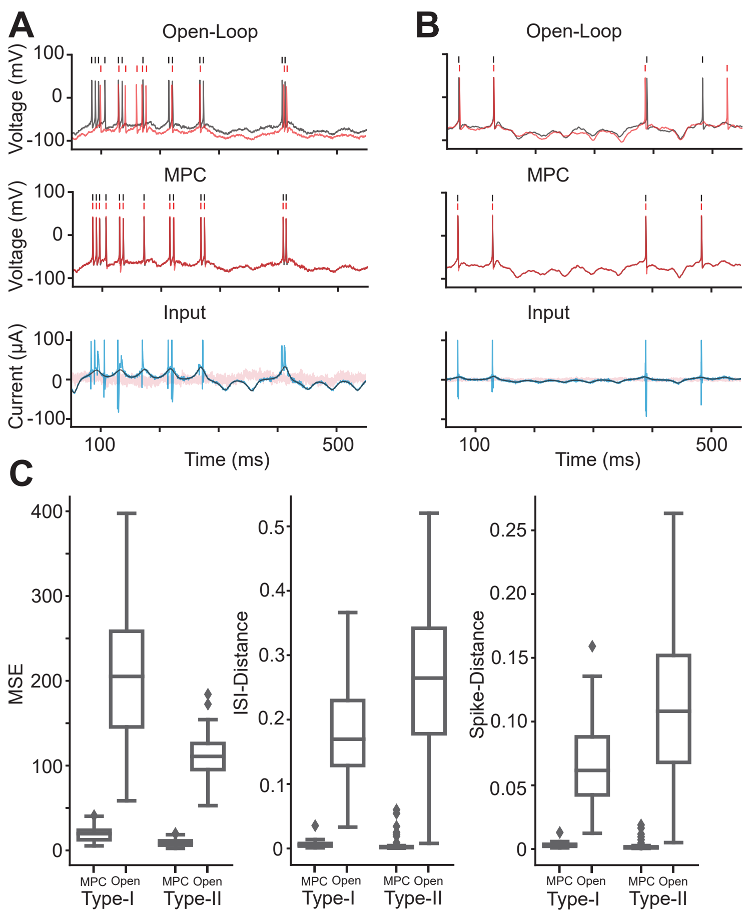


Figure 2.4: **Results of Homogeneous System Control. A)** (Top) Example of a membrane voltage trajectory (red) for a Type-I CS model with open-loop control. Rasters above indicate spike times. The same $I_{inj}(t)$ used to generate the reference trajectory was used as the input for the open-loop control. However, the unknown noise current $I_{noise}(t)$ into the CS model resulted in the controlled membrane voltage deviating from the reference trajectory (black). (Middle) Example of a membrane voltage trajectory for a Type-I CS model controlled via MPC. The controlled membrane voltage tracks the reference trajectory extremely well compared to the open-loop controller. (Bottom) The unknown noise current (pink) into the CS model, the $I_{inj}(t)$ used to generate the reference trajectory and as the open-loop input (black), and MPC optimized input used to control the CS model (blue). **B)** The same diagrams as in **A**, but for a Type-II CS model. **C)** Performance metrics comparing the open-loop and MPC methods of control. The MSE was calculated for each reference/control trajectory pair. For both types of CS models, MPC achieved much better performance than the open-loop controller. ISI-distance and spike-distance are both on the interval [0,1] where 0 indicates identical spike trains. Once again, MPC outperformed the open-loop controller. Notice that for all three metrics, the variances are also much lower for MPC compared to open-loop control, showing that MPC is not only better on average but is also a consistent controller.

2.4.2 Experiment II: Heterogeneous System Control

As a more difficult test of the MPC controller, the CS neuron of one type was forced to follow the activity of the other type. In other words, could a Type-I neuron be made to spike like a Type-II neuron? I refer to this as heterogeneous system control because one dynamical system is being forced to behave as a different dynamical system. The same MSE and spike train similarity metrics from Experiment I were used to compare MPC performance with open-loop control. In this case, open-loop control was performed by taking the I_{inj} that produced the V^{ref} in a particular CS model type and using that as the command signal into the other CS model.

Unsurprisingly, open-loop control performed poorly for this task, because each CS model was a distinct dynamical system and thus responded differently to the same command signal. Type-I neurons often failed to fire at all when driven by injected currents used to stimulate Type-II neurons, presumably because the A-type current in the Type-I model counteracted the depolarizing injected current. Conversely, the Type-II neurons had a tendency to produce too many spikes when injected with currents used to stimulate Type-I neurons. Despite the dissimilar intrinsic currents and dynamical topologies of the two CS models, MPC was able to force each CS model to follow the trajectory of the other model as seen in Figure 2.5, with almost the same level of performance seen in the homogeneous system control task.

2.4.3 Experiment III: Spike-Train Control

In many neuroscience studies, the experimenter wants to make a neuron spike at specific times without caring too much about the subthreshold activity. To see if MPC could be used in such experiments, I tested whether MPC could force the CS

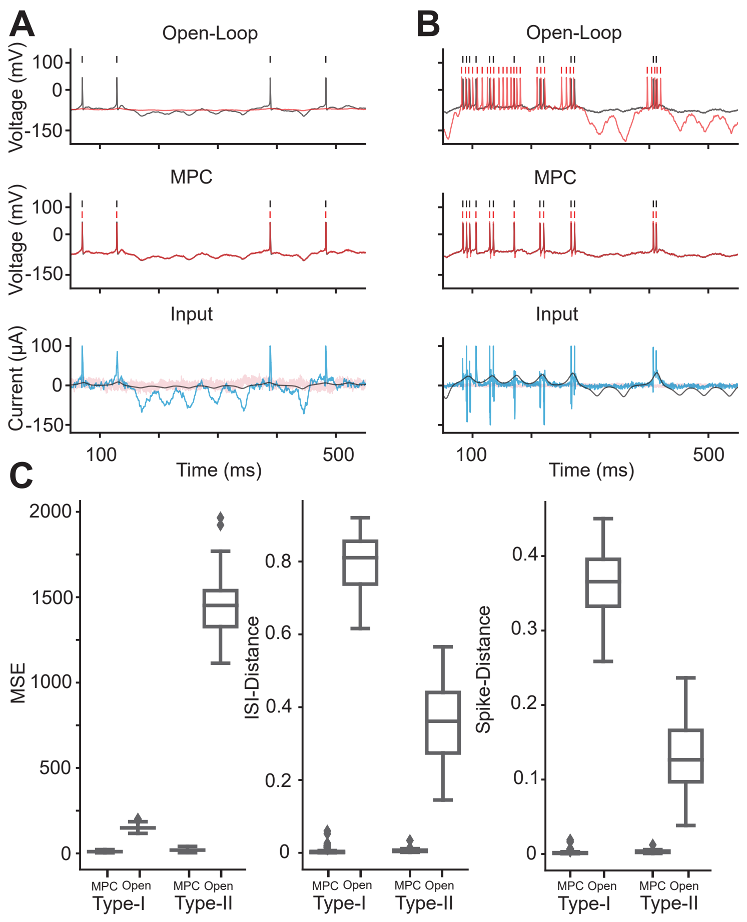


Figure 2.5: **Results of Heterogeneous System Control. A)** (Top) Example of a membrane voltage trajectory for a Type-I CS model with open-loop control. The reference trajectory (black) was obtained from a Type-II CS model. In this example, the controlled trajectory (red) exhibited no spiking and did not closely follow the reference trajectory. (Middle) Example of a membrane voltage trajectory for a Type-I CS model controlled via MPC. Notice that MPC is able to force the Type-I model to follow a Type-II model reference trajectory. (Bottom) The unknown noise current (pink) into the CS model, the $I_{inj}(t)$ used to generate the reference trajectory and as the open-loop input (black), and MPC optimized input used to control the CS model (blue). The MPC input drastically deviates from the open-loop control input in order to make the Type-I model follow the Type-II reference trajectory. **B)** The same diagrams as in **A**, but now a Type-II CS model is controlled to follow a reference trajectory taken from a Type-I model. In open-loop control (Top), the Type-II model fires noticeably more than the reference trajectory. However, MPC is able to control the Type-II neuron into following the Type-I reference trajectory. **C)** Performance metrics comparing the open-loop and MPC methods of control. In all cases, MPC achieved much better control than the open-loop controller. Notice the change in metric ranges compared to the homogeneous system control condition. Heterogeneous open-loop control did much worse than the homogeneous condition for all three metrics, while MPC remained at similar levels and had extremely small variances across trials compared to open-loop control.

models to produce arbitrary spike trains. Spike trains are a point process, whereas the DDF models forecast a continuous variable. To overcome this mismatch in data structure, the mean spike waveform from the training data of each CS model was extracted and embedded it into a time series of constant value chosen to be below the threshold. The peaks of these embedded waveforms matched the spike times of the reference spike train. By doing so, any spike train could be converted into a reference trajectory with units of mV.

Because precise control of the subthreshold activity was not the objective, the performance in this task was quantified using spike jitter instead of MSE. Jitter was calculated by subtracting the reference trajectory spike times from the controlled spike times; thus positive values indicate the controlled spikes fired late, and negative values indicate the spiking was too early. As shown in Figure 2.6, MPC achieved precise control of spike timing for both CS models. Interestingly, the Type-I model never fired early while the Type-II model fired both early and late. However, both models had similar values of jitter, and the absolute maximum value was around 0.2 ms. These results demonstrate how MPC can be used to control a neuron to fire in accordance with an arbitrary spike train.

2.5 Conclusions

Neural systems can be difficult to control because of their nonlinear dynamics and many hidden states (Rabinovich et al. 2006). Here I demonstrated that nonlinear MPC can control the well-characterized Connor-Stevens neuron model via an injected current using only measurements of the membrane voltage. The controller was able to force the model to reproduce a previously observed voltage trajectory in the presence

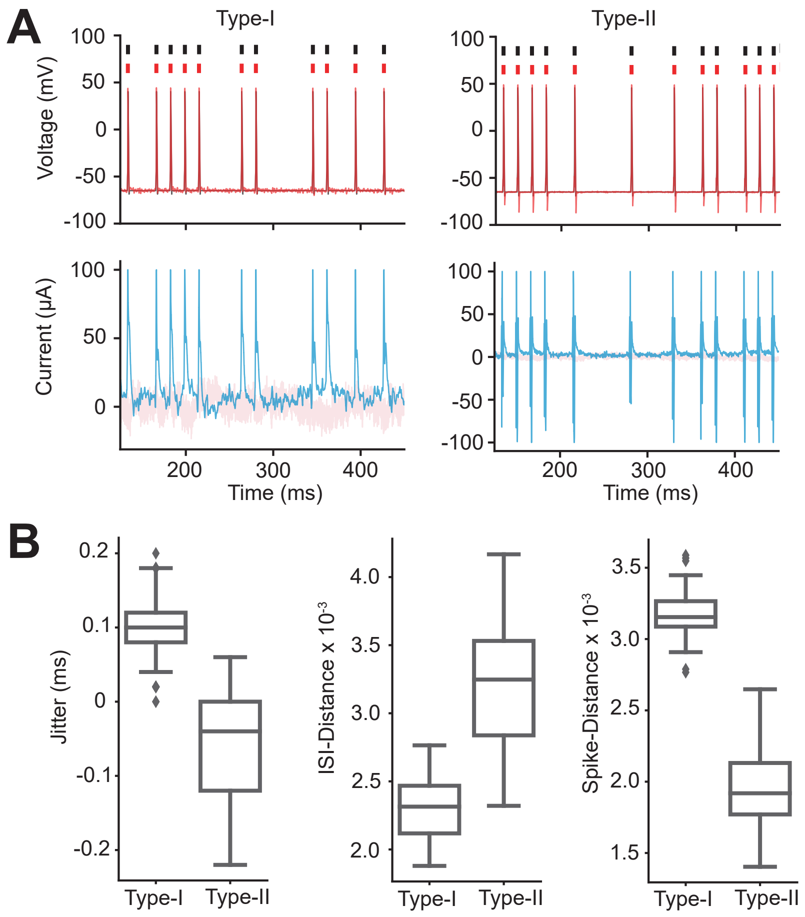


Figure 2.6: **Results of Spike-Train Control. A)** Example of membrane voltage trajectory controlled via MPC for a Type-I (top left) and Type-II model (top right). The reference trajectory (black) was obtained by taking the average spiking waveform for the corresponding CS model type and embedding it into a constant valued time series (-72 mV) at time points where a spike is desired. MPC was able to control CS models with their resulting membrane voltages (red) closely following the reference trajectories. **B)** MPC performance metrics. Since capturing the spike timing was the goal rather than a strict following of the reference trajectory and subthreshold voltages, we used spike jitter instead of MSE. Jitter was calculated by subtracting the spike times of reference trajectory from the controlled trajectory where positive values indicate the model fired late and negative values indicate the model fired early. The Type-I model had strictly positive values across all trials, however the maximum value of jitter was approximately 0.2 ms which is an extremely small time difference in terms of spike train dissimilarity. The Type-II model had both positive and negative values of jitter, but with a negative average indicating that MPC produced spikes earlier on average than desired. The ISI- and spike-distances were similar to the values of the homogeneous and heterogeneous control conditions and had extremely small variances. Notice the difference in scale compared to the previous conditions.

of unknown intrinsic noise, follow a reference trajectory produced by a model with a different dynamical topology, and produce an arbitrary spike train with high temporal precision. Importantly, this was achieved without any knowledge of the biophysical details of the Connor-Stevens model by using a data-driven forecasting model fit to a few seconds of noisy current-clamp data.

This study represents one step toward the ultimate goal of controlling biological networks of neurons *in vivo* to experimentally probe the mechanisms of neural computations and ameliorate pathological circuit states like epilepsy. The system tested here involves only a single neuron in the equivalent of a whole-cell patch recording, which enables an experimenter to make low-noise, high-bandwidth measurements of membrane potential while injecting current through an access resistance much smaller than the resting (input) resistance of the cell membrane. This preparation is easily controllable in practice however, and modern intracellular amplifiers are able to clamp cell voltage using relatively simple proportional feedback controllers implemented in analog circuitry (Sherman-Gold 2012). The purpose of this study was not to improve on amplifier design but rather as a proof of principle for how data-driven nonlinear MPC can achieve control of a neuron's membrane voltage without any prior knowledge of the intrinsic ionic currents expressed by a specific neuron. All of the prior studies applying MPC to conductance-based neuron models have assumed knowledge about the neuron such as the states of current gating variables or the parameters and functional forms of their kinetics (Fröhlich and Jezernik 2005; Yue, Tomastik, and Dutta 2022; Ullah and Schiff 2009; Senthilvelmurugan and Subbian 2023), which would not be known in a real experiment. The results in this study show that this information is not necessary, bolstering confidence that data-driven MPC can be extended to neural networks in which there is likely to be even less knowledge about the full state and parameters of the system.

To illustrate some of the ways in which the principles in this study could be extended to networks, consider as an example the zebra finch's HVC, a bilateral premotor nucleus which contains around 36,000 highly interconnected neurons in each hemisphere (Bottjer, Miesner, and Arnold 1986). Precise patterns of neural activity in HVC collectively result in the bird singing, and the ability to experimentally control HVC to produce arbitrary trajectories in its state space would produce major insights into how this system orchestrates vocal communication. Present technology allows simultaneous measurement of activity in a few hundred of these cells using calcium imaging or high-density extracellular electrophysiology, and activity could be manipulated in a (potentially different) subset of neurons using optogenetic stimulation. The state of the system would now be a vector, naively with one component for each of the neurons the experimenter was monitoring. The input would also be a vector corresponding to the neurons the experimenter was manipulating, and the loss function would generalize to a form along the lines of

$$J(x_0) = \mathbf{e_T^\intercal S e_T} + \sum_{n=0}^{T-1} \mathbf{e_n^\intercal Q e_n} + \mathbf{\Delta I_{n+1}^\intercal R \Delta I_{n+1}}, \tag{2.9}$$

where $\mathbf{e_i}$ and $\mathbf{\Delta I_i}$ have the same meaning as in equation (8) but are now vector-valued with each element corresponding to a individual state and external input source respectively. The matrices $\mathbf{S}$, $\mathbf{Q}$, and $\mathbf{R}$ function largely the same as the scaling factors $\mathbf{s}$,$\mathbf{q}$, and $\mathbf{r}$ in (8), but now allow one to differentially weight the cost of each of the elements of vectors $\mathbf{e}$ and $\mathbf{I}$. For example, there may be a subset of neurons in a network that have an outsized impact on the population activity as a whole. By using larger values in the $\mathbf{Q}$ and $\mathbf{S}$ matrices that correspond to this subset of neurons in error vector **e**, the controller will view state errors in these neurons as more costly than the other units in the network. It should be noted that this kind of loss function could be applied across many different modalities of neural activity. Similar loss functions for linear MPC have been applied to optogenetics both *in vivo* (Bolus et al. 2021) and simulation models (Milias-Argeitis and Khammash 2015; Fox, Batt, and Ruess 2023). It would be straightforward (at least mathematically) to use behaviorally derived states in vector **e** while maintaining cellular inputs in vector **I**. This would allow researchers to explore how specific patterns of neural activity control organism behavior.

Extending control to neural circuits may necessitate the use of more complicated function approximators to obtain a good forecasting model. However, there are several advantages of simpler models like the RBFN compared to more complex models. Time is often a constraint in neuroscience experiments and the ability to estimate and use a model in a short time frame is a necessity. When controlling a neuron (or neural circuit) with MPC, the forecasting model would need to be estimated quickly and in a data-efficient manner. Because neurons exhibit a large diversity of dynamics, a forecasting model trained on one neuron is unlikely to generalize to a different neuron. The more complex and time-consuming the forecasting model is to train, the less time there would be to control the neuron. In this study, DDF models were estimated using only 5 seconds of data and the training time was negligible compared to the amount of time for a typical patch-clamp experiment. Modern data-driven models often have many more parameters and more computationally expensive training algorithms which limit their ability to be practical in an experimental setting. Additionally, RBFNs can be estimated in an online setting (e.g., recursive least squares) which allow the DDF models to adapt to changes in the neural dynamics which can come from many sources such as electrode drift, tissue damage, and intrinsic plasticity. However, architectures such as RNNs and Transformers routinely achieve state of the art performance on time-series forecasting benchmarks and may be necessary when using MPC on large coupled networks. Latent factor models could also be used to reduce the dimensionality of the cost function thereby reducing the computational complexity of the optimization. Instead of using MPC to control the activity of every unit in the network, the latent model could express the coordinates of the network state and reference trajectory in the lower dimensional space.

Chapter 3

Controlling a Spiking Neural Network

In the previous chapter, I demonstrated how a single neuron could be controlled with MPC using a data-driven dynamics model. However, an organism's behaviors often do not arise from the activity of a single neuron but rather through complex interactions of large networks of neurons. Understanding the causal pathways between behavior and neural activity will not be understood by controlling one neuron at a time; instead simultaneous control of many neurons will be required. Although there exist many techniques to stimulate and silence populations of neurons (e.g. optogenetics, transcranial magnetic stimulation, drug interventions), achieving precise control over neural activity is still in relative infancy. One of the main reasons for this is the diversity of technological constraints imposed by each method of neural control. For example, controlling the membrane voltage of a single neuron through patch clamp is an extremely well established closed-loop control technique. It is tempting to assume that controlling the voltage of multiple neurons could be obtained by patching onto all the neurons concurrently. With few notable exceptions (Jäckel et al. 2017), this is currently technologically infeasible. While other methods allow for the simultaneous measurement and perturbation of networks of neurons, each come with their own limitations and technological considerations.

Notwithstanding these issues, there are also specific challenges with applying the MPC framework to network-level control. Recall that MPC requires a dynamics model that can predict the state evolution of the system for a given initial condition and control input. Unlike individual neurons, few experimentally validated biophysical models of networks exist. Fortunately, there is a wealth of work showing that the complex dynamics of populations of neurons can be approximated using data-driven approaches (Beniaguev, Segev, and London 2021; Sun et al. 2023; Taylor et al. 2024). In principle, this should allow one to produce a dynamics model for MPC of a network of neurons. However, neural systems exist in a high dimensional state space both theoretically and experimentally with extracellular probes able to simultaneously record the activity of hundreds to thousands of neurons. This presents a problem for practical implementation of MPC, because increasing the dimensionality of the state space can lead to increased computational cost of the optimizer. For example, given the quadratic loss function

$$\ell(\mathbf{e_i}, \mathbf{u_i}) = \mathbf{e_i^\intercal Q e_i} + \mathbf{u_i^\intercal R u_i} \tag{3.1}$$

a large state space would lead to many hyperparameters in $\mathbf{Q}$. In order to overcome this issue, I propose leveraging recent work in neural manfiolds which can both reduce the dimensionality of the problem and reframe the control problem in general.

3.1 Neural Manifolds

It has been well documented that activity from neural populations can be embedded into lower dimensional subspaces that preserve much of the information contained in the original full observation dimension (Pang, Lansdell, and Fairhall 2016). These subspaces (termed ‘neural manifolds’) correlate with cognitive, behavioral, and stimulus activity across many experimental settings (Chung and Abbott 2021). While the entirety of the network can be viewed as a high dimensional dynamical system, activity on the manifold can be expressed as a latent dynamics model. Broadly speaking, there are two main perspectives on how to model the time evolution of neural manifolds: descriptive or generative modeling. Using linear subspaces with autonomous dynamics as an illustrative example, the descriptive perspective can be seen as classic dimensionality reduction with

$$\mathbf{z_t} = \mathbf{G}\mathbf{x_t} \tag{3.2}$$

where $\mathbf{x_t}$ is a column vector of the activity of n neurons at time t and $\mathbf{z_t}$ is a reduced dimension representation of $\mathbf{x_t}$ given by the linear transformation $\mathbf{G}$. This model is descriptive because the latent trajectories $\mathbf{z_t}$ are largely a recapitulation of the covariances that exist in $\mathbf{x_t}$ and provide limited information for inference or mechanistic understanding (Langdon, Genkin, and Engel 2023).

In contrast, the generative perspective can be modeled as a latent factor model of the form

$$\mathbf{x_t} = \mathbf{F}\mathbf{z_t} + \epsilon_\mathbf{t} \tag{3.3}$$

where $\mathbf{x_t}$ is a column vector of the activity of n neurons at time t, $\mathbf{z_t}$ are the latent factors that span the neural manifold with some smaller dimension k, $\mathbf{F}$ are the factor loadings, and $\epsilon_\mathbf{t}$ is a sample from some distribution (often Gaussian). This perspective views the measured neural activity as a function of the latent dynamics of $\mathbf{z_t}$. Early work on neural manifolds used linear methods such as principal components analysis (as in equation 3.2) and factor analysis (as in equation 3.3), but there is now a broader consensus that neural manifolds are nonlinearly embedded in the full state space and require more sophisticated methods (Fortunato et al. 2023).

The nature of these subspaces in relation to neural computation is still debated, and neural manifolds are viewed either as fundamental units of computation or merely a convenient representation of high dimensional activity. Much of this debate is due to the purely correlational nature of most neural manifold studies (Langdon, Genkin, and Engel 2023). Regardless of the true nature of neural manifolds, we can exploit a property present in both interpretations: much of the recorded activity is redundant in describing the computations of the network. We can now reframe our general problem of controlling a network of neurons to be more specific. Instead of trying to control the state of each neuron in the network, I will control the latent dynamics obtained from the activity on the neural manifold. In other words, the desired state trajectory for the network will be expressed using coordinates in reference to the neural manifold. This only partially solves the optimization dimensionality problem, however. The control inputs $\mathbf{u}$ in equation 3.1 could be very high dimensional, leading to increased computational cost and matrix $\mathbf{R}$ to require many hyperparameters. Using similar dimensionality reduction techniques, we could find an optimal set of MPC inputs in a lower dimensional space and project back into the original dimension of the stimulus input to drive the network.

3.2 Simulating an Extracellular Recording Experiment

A simulation of an extracellular recording experiment was chosen to demonstrate how control over latent neural dynamics may be achieved. This modality of experimentation was chosen due to the fact that extracellular probes are able to record from dozens to hundreds of neurons simultaneously in both anesthetized and awake ani-

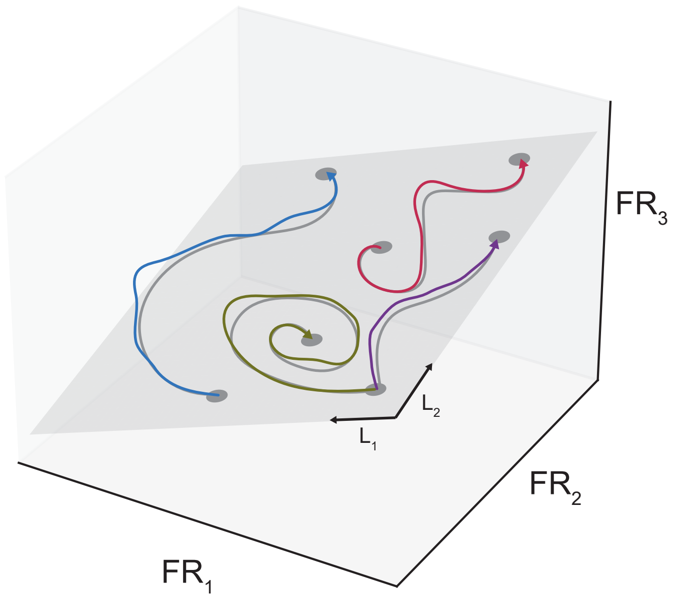


Figure 3.1: **Neural Manifolds.** An example of a linear neural manifold. The simultaneous firing rates (FR) of three neurons can be represented in 3D state-space. Different trajectories in this state space (colored lines) could arise from differing initial conditions and/or different external perturbations such as changes in stimuli. However, this activity often exhibits lower dimensional structure than the original measurement space. The trajectories can be projected onto a 2D embedding given by the basis vectors L_1 and L_2. This linear subspace is often referred to as a neural manifold and the activity of the neurons is largely preserved in this lower dimensional space (gray lines).

mals. In a typical extracellular experiment, a probe is inserted into neural tissue to record electrical activity. The recorded signals are complex and noisy mixtures of the activity in the tissue and must be separated into unique components (Rey, Pedreira, and Quiroga 2015). This separation is referred to as spike sorting and results in the continuous electrical signals being reduced into spike trains (time sequences where each element is a spike time). Each spike train is purportedly from a unique neuron and both offline and online methods of spike sorting exist (Wouters, Kloosterman, and Bertrand 2018), with online methods being necessary for real-time monitoring and control of the neural activity. In this simulation, only a subset of the neurons simulated were used to fit a latent dynamics model of the whole network since extracellular probes can only measure a fraction of the neurons in the surrounding tissue.

3.2.1 Artificial Circuit

Architecture

The activity of an artificial circuit (AC) evoked by an external visual stimulus was simulated with a spiking neural network (SNN) composed of three layers: sensory, reservoir, and output. Each neuron in the AC was modeled with a recurrent leaky integrate-and-fire (rLIF) model, with the discrete time approximation

$$V_{n+1} = \begin{cases} \beta V_n + w^T X_{n+1} + r^T S_n, & \text{if } V_n < \Theta \\ 0, & \text{if } V_n \geq \Theta \end{cases} \tag{3.4}$$

where

V_n : membrane voltage at the nth time step

Θ : spiking threshold

β : decay parameter

X_n : feedforward input vector at the nth time step

w : feedforward weights

S_n : layer spiking vector at the nth time step

r : recurrent weights

Whenever the V variable was reset to 0, a spike was recorded at that time step. For a given layer with N neurons, this produced a binary vector $S_n = [s_1, s_2, ..., s_N]^T$ where the value of each element was either a 0 or 1 indicating if the corresponding neuron had fired at that time step. This allowed the activity of each neuron in a layer to be affected not only by its own firing (i.e., spiking inhibition/facilitation), but also to receive inputs from the other neurons in that layer.

Each neuron in the sensory layer received feedforward input in the form of a grayscale image reshaped into a 784 dimensional vector. This input served as the external stimulus that was the primary driver of AC activity. Gaussian noise was also added to the feedforward inputs of each neuron in every layer to produce stochasticity in activity. This noise modeled the effects of natural variability in neural firing and the effects of unknown exogenous inputs. Thus the subthreshold activity of the three layers was given by the equations

$$\textbf{Sensory} : V_{n+1}^{sen} = \beta V_n^{sen} + w_{sen}^T (I_n + \epsilon) + r_{sen}^T S_n^{sen} \tag{3.5}$$

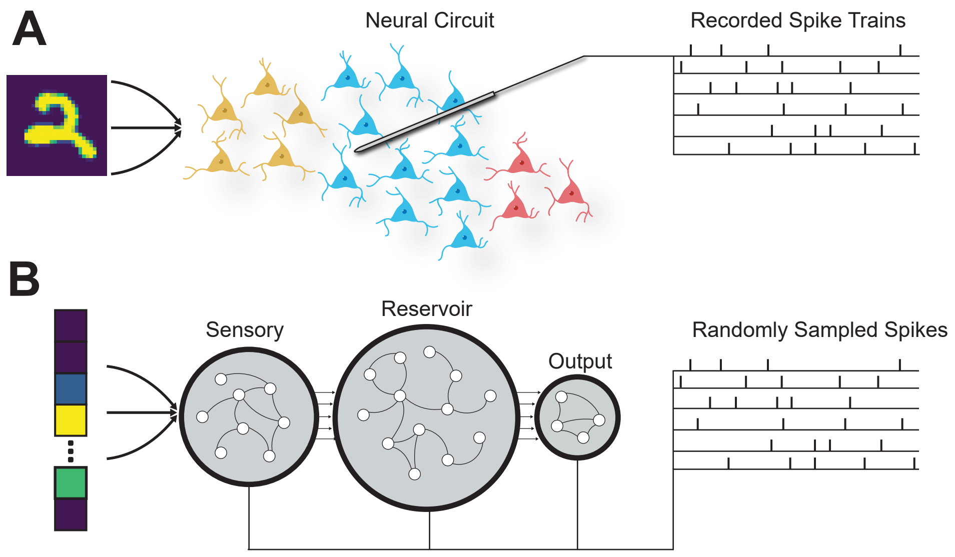


Figure 3.2: **Simulation of Extracellular Experiment.** **A)** Simplified model of an extracellular recording. A visual stimulus (MNIST digit) evokes firing from a neural circuit of unknown composition. In this example, the circuit contains three distinct neural subpopulations (colored neurons) whose electrical activity is recorded with an inserted electrode. Only a random subset of the neural activity is sorted into distinct spike trains. **B)** A simulation of the circuit above using an SNN. The artificial circuit is composed of three compartments: sensory (n = 100), reservoir (n = 600), output (n = 10). The sensory layer is driven by MNIST digit stimuli in vector form. Each layer is composed of rLIF neurons and receives all-to-all connections from the previous layer. Gaussian noise is added to the input of each neuron to produce stochasiticity in the activity and model the effects of unknown extrinsic and intrinsic inputs. Only a random subset of the spiking activity of the artificial circuit is recorded to mimic the results of spike sorting.

$$\textbf{Reservoir}: V_{n+1}^{res} = \beta V_n^{res} + w_{res}^T(S_n^{sen} + \epsilon) + r_{res}^T S_n^{res} \tag{3.6}$$

$$\textbf{Output}: V_{n+1}^{out} = \beta V_n^{out} + w_{out}^T(S_n^{res} + \epsilon) + r_{out}^T S_n^{out} \tag{3.7}$$

where $\epsilon \sim N(0, \eta^2)$ for each element in the feedforward input and I_n is the stimulus image presented at the nth time step.

Training the Network

While in principle the feedforward and recurrent weights for each neuron could be randomly distributed, these weights were trained to perform a classification task. This was done to ensure that the activity of the AC was strongly correlated on similar types of images. Using the Python package `snntorch` (Eshraghian et al. 2023), the AC was trained to accurately classify digits from the MNIST data set (LeCun et al. 1998). Training SNNs requires additional considerations compared to traditional artificial neural networks. The resetting of a neuron's membrane voltage when it reaches the threshold Θ produces a non-differentiable function (Eshraghian et al. 2023) making training with gradient descent impossible. One solution to this problem is to use surrogate gradient descent (Neftci, Mostafa, and Zenke 2019), where the non-differentiable function is preserved in the forward pass of the network but is replaced with a sigmoid function during the backward pass. This results in a function differentiable everywhere and allows the network to be trained with valid gradients.

The MNIST data set is composed of handwritten images of the digits 0 through 9. Classification of the images was performed by associating the label of the image with a corresponding neuron in the output layer of the AC (e.g., label 4 with neuron 4). For time step n, the cross-entropy ℓ_n was given by

$$p_n^i = \frac{\exp(V_n^i)}{\sum_{k=0}^{9} \exp(V_n^k)} \tag{3.8}$$

$$\ell_n = -\sum_{i=0}^{9} y_i log(p_n^i), \tag{3.9}$$

where V_n^i is the membrane voltage of the ith neuron which corresponds to the prediction of label i, and y is a one-hot encoded vector of the true label. Due to the inherently temporal nature of SNNs, one must specify how many time steps a stimulus is be presented before class prediction takes place. This can be interpreted as a combination of reaction time and evidence accumulation. Thus the loss function to be minimized is given by

$$\mathcal{L}_{CE} = \sum_t \ell_t, \tag{3.10}$$

where the cross-entropy loss at each time step is summed for some trial time length t. This forces the neuron of the associated predicted class to have the highest firing rate compared to the other neurons in the output layer. For the given time window that the image is presented, the feedforward and recurrent weights in the AC can be updated using backpropagation through time (BPTT).

The first half of the MNIST data set (n = 35,000) was used for the training and validation of the AC with an 80/20 split. Due to the stochastic nature of the noise added to every neuron in the AC, performance after training was assessed by presenting the stimuli 30 times to obtain an accuracy distribution. Accuracy on the training (n=28,000, M = 43.7%, s = 8.4%) and validation (n = 7,000, M = 43.9%, s = 8.4%) set were largely similar, indicating there was no overfitting to the training data. Although the accuracies were far below what would be considered competitive performance on a classification task, the purpose of training the AC was to ensure the connections between the neurons were not random. See Appendix.2 for rLIF and AC training hyperparameters.

3.3 Dimensionality Reduction of Stimuli and Neural States

The dimensionalities of the MNIST digit stimuli and AC neural activity were reduced using variational autoencoders (VAEs). The fundamental idea behind VAEs is that an observed data vector $\mathbf{x_i} \in \mathbb{R}^n$ is some nonlinear transformation of a latent random variable $\mathbf{z_i} \in \mathbb{R}^k$ with $k << n$. This latent variable is typically parameterized as a Gaussian of the form

$$\mathbf{z_i} = \mu_\mathbf{i} + \sigma_\mathbf{i} \odot \epsilon, \epsilon \sim N(0, I), \tag{3.11}$$

where $\odot$ denotes element-wise multiplication. The original data vector can now be written as

$$\mathbf{x_i} = h_\phi(\mathbf{z_i}), \tag{3.12}$$

where $h_\phi(.)$ has parameters that must be learned from the data. Since there are an infinite set of solutions to this parameterization if $\mu_\mathbf{i}$ and $\sigma_\mathbf{i}{}^2$ were any arbitrary value, an additional nonlinear transformation is learned such that,

$$\mu_\mathbf{i}, \sigma_\mathbf{i}^\mathbf{2} = f_\theta(\mathbf{x_i}). \tag{3.13}$$

The functions $f_\theta(.)$ and $h_\phi(.)$ are referred to as the encoder and decoder respectively. The encoder takes a data vector $\mathbf{x_i}$ and embeds it into a latent representation $\mathbf{z_i}$. This latent variable can then be projected back into the original dimension by applying the decoder transformation. The parameters of the encoder and decoder can be found by minimizing the loss function

$$\ell_{VAE} = \ell_{recon} + \alpha \ell_{KL} \tag{3.14}$$

$$\ell_{recon} = \sum_{i=1}^{B} ||\mathbf{x_i} - h_\phi(\mathbf{z_i})||_2^2 \tag{3.15}$$

$$\ell_{KL} = \sum_{i=1}^{B} \sum_{j=1}^{k} (1 + log\, \sigma_{ij}^2 - \mu_{ij}^2 - \sigma_{ij}^2) \tag{3.16}$$

where α is a hyperparameter that scales the importance the of KL-divergence term in the loss function and B is the batch size used for gradient descent. Due to the latent variable $\mathbf{z}$ being parameterized by a Gaussian, minimizing this loss function is equivalent to maximizing the evidence lower bound (ELBO) (Odaibo 2019).

The VAE framework was chosen for two primary reasons. First, VAEs embed high-dimensional data into nonlinear low-dimensional subspaces. This allows for a flexible approach for finding the latent dynamics of the AC and obtaining a more generalizable compression of the data compared to linear transformations (Gomari et al. 2022). Second, the use of KL-divergence promotes nearby values of $\mathbf{z}$ in the latent space to be decoded into similar values in the original space of $\mathbf{x}$ (Kingma and Welling 2013). By doing so, this allows for easier interpolation between a set of training data points in latent space when constructing the dynamics model and finding the optimal latent inputs with MPC. If a basic autoencoder (i.e., $\alpha = 0$) was used instead of a VAE, nearby points in latent space are not guaranteed to be similar in the original measurement space.

The control problem can now be expressed as given a latent embedding of the neural states $\mathbf{x}$ and stimulus states $\mathbf{u}$,

$$\mathbf{z_n} = \mathbb{E}(f_\theta^{\text{neural}}(\mathbf{x_n})), \mathbf{v_n} = \mathbb{E}(f_\theta^{\text{stimulus}}(\mathbf{u_n})) \tag{3.17}$$

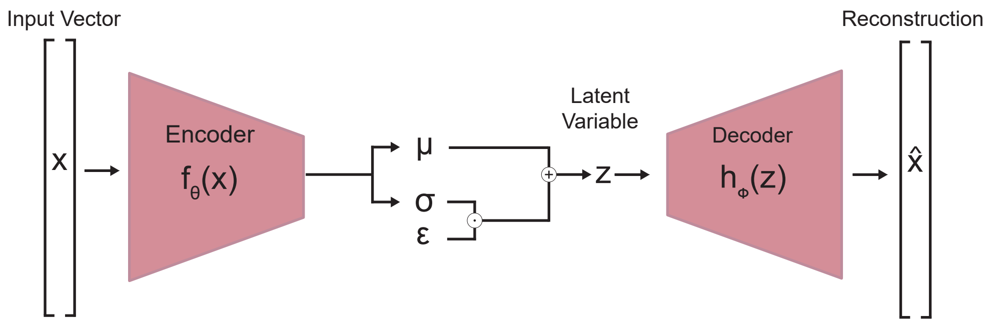


Figure 3.3: **Variational Autoencoder Architecture**. The VAE encoder produces a latent representation of input vector $\mathbf{x}$ by projecting it in a nonlinear subspace. This embedding is achieved by learning a parameterization of a normal distribution where the mean and variance vectors associated with the input $\mathbf{x}$ are the outputs of a multilayer perceptron. The latent representation $\mathbf{z}$ is sampled from this distribution where $\epsilon \sim N(0, I)$. The decoder takes this latent representation and projects it back into the original dimension of $\mathbf{x}$. Sampling directly from the latent space can produce novel decoded vectors that will share many of the statistical properties of the $\mathbf{x}$ vectors used to train the network.

we seek to find an optimal set of latent inputs

$$\mathbf{v}^*_{1:T} = \arg\min_{\mathbf{v}_{1:T}} \sum_{n=0}^{T} \ell(\mathbf{z_n}, \mathbf{v_n}) \tag{3.18}$$

with the dynamics model

$$\mathbf{z_{n+1}} = g(\mathbf{z_n}, \mathbf{v_n}). \tag{3.19}$$

The AC can then be stimulated with the decoded latent inputs

$$\mathbf{u_n} = h_\phi^{\text{stimulus}}(\mathbf{v^*_n}) \tag{3.20}$$

to produce the neural states $\mathbf{x}$ at the next time step.

3.3.1 Stimulus VAE

A VAE was trained to find a low dimensional representation of MNIST digit stimuli (sVAE). The sVAE encoder was composed of two single-channel convolutional layers and a single feedforward layer. Each of the three layers used a $ReLU$ activation function. The size of the latent space $\mathbf{v}$ was chosen to be 2. The sVAE decoder was symmetric to the architecture of the encoder with one key difference; a sigmoid activation function was used on the final output layer to ensure that the values of the reconstructed stimuli were between 0 and 1 (the bounds of all pixel values in the training images). See Appendix.3.1 for layer hyperparameters.

The second half of the MNIST data set (n = 35,000) was used for training and validation of the sVAE. Eighty percent of this data (n = 28,000) was used for training and the remaining 20% for validation (n = 7,000). The sVAE was trained via gradient descent in `Pytorch` using the Adam optimizer. See Appendix.3.2 for training hyperparameters.

A sequence of latent inputs for latent dynamics model identification was constructed using the validation set of images. Discrete points in the latent sVAE space were obtained by running k-means clustering (n=100) on the low dimensional embedding of the validation images. Half of the centers were used for the dynamics model training (V_{train}) and half for testing (V_{test}). These discrete points were converted to a continuous time series by one of three methods: step function, fast interpolation, slow interpolation. The step function method took a sequence of centers and held each center constant for 500 time steps (ms). The fast and slow methods took the sequence and linearly interpolated values between each element in the sequence but at different time scales (200 ms for the fast and 1000 ms for the slow). Both the V_{train}

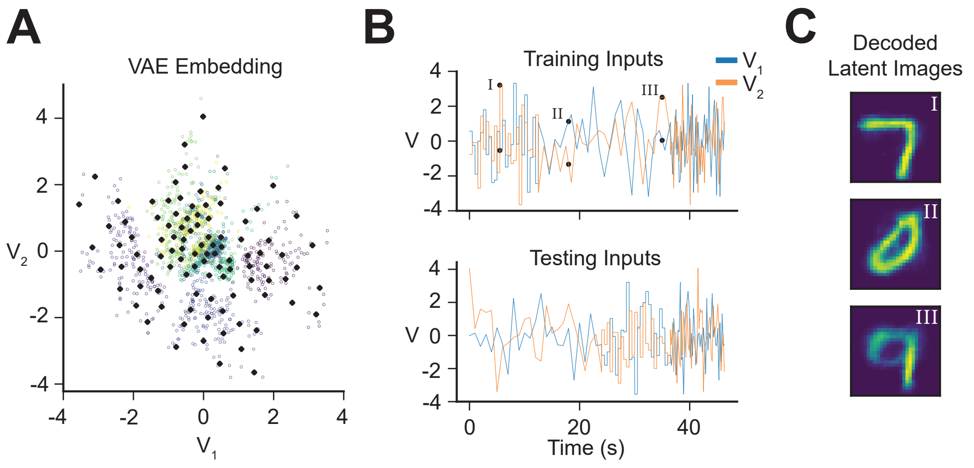


Figure 3.4: **Latent Embedding of MNIST Digits. A)** Each colored dot is one of the MNIST digits embedded in the 2-dimensional nonlinear subspace of the sVAE encoder. Notice the clustering of the digits by label (color), indicating that digits with identical labels were often embedded in nearby latent space. In order to generate a latent sequence of inputs used to stimulate the artificial circuit, points from this latent space where sampled using k-means clustering (100 centers). **B)** Training and testing latent sequences where each generated using half of the centers from **A**. **C)** Three points from the training inputs decoded into the original stimulus dimension.

and V_{test} sequences were 46.3 seconds long (46,300 time steps). The use of these three methods was to have a input sequence that could show how the latent states of the AC responded to inputs changing at different time scales and frequencies. See Figure 3.4 for visualization of the latent input sequences and their sVAE decoded values.

3.3.2 Neural VAE

The AC was stimulated using the sVAE-decoded V_{train} and V_{test} input sequences with the resulting spiking activity used to fit a VAE for the neural states (nVAE). A random sample of 20% of the neurons (n=122) in the AC were used to build the model. These neurons were the only units in the AC that were measured for the entire experiment. Recall that in a typical extracellular recording, only a subset of

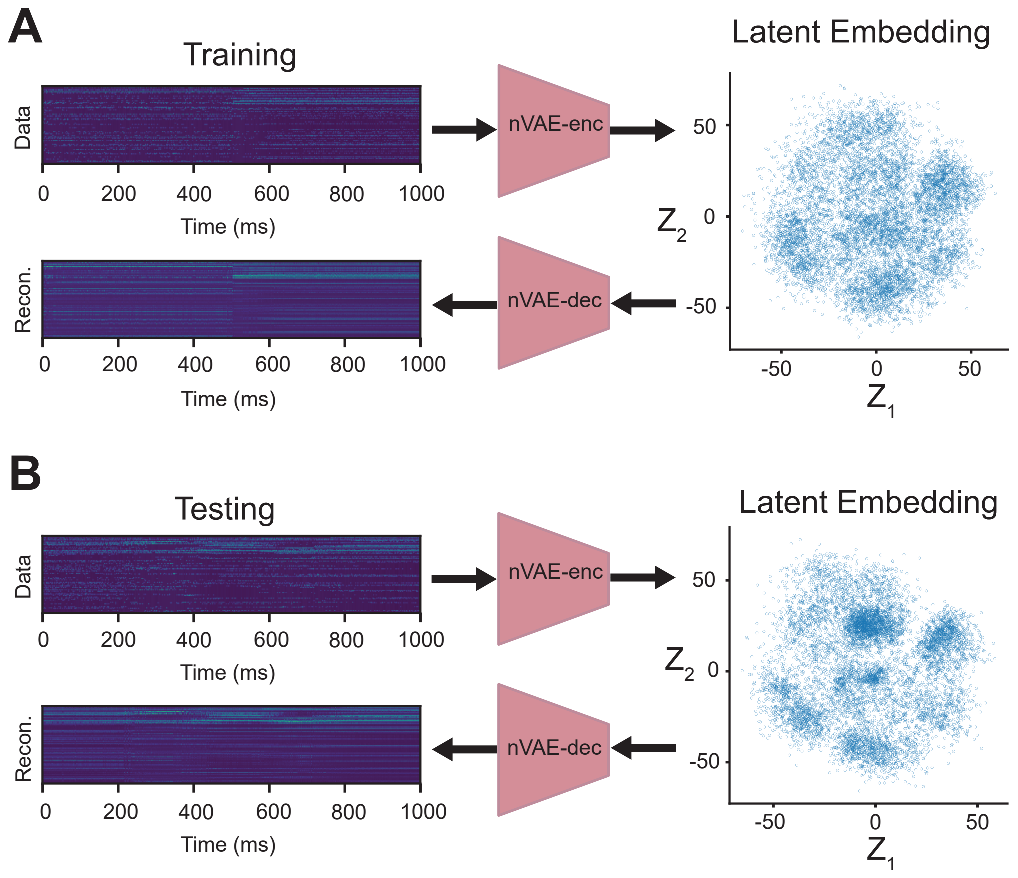


Figure 3.5: **Latent Embedding of Neural Activity. A)** Results of the VAE on training SNN neural activity (nVAE). The exponentially filtered spikes are embedded in a latent 2D space through the nVAE encoder. Activity in this latent space can be projected back into the original dimension with the nVAE decoder. **B)** Results of the nVAE on the testing SNN neural activity.

the neurons are observable. The purpose of this random sampling was to mimic the incomplete information that would be obtained in a real recording.

The measured binary spiking states were converted to continuous states with an exponential filter, where the smoothed state $\mathbf{x_n}$ of spiking state $\mathbf{y_n}$ is given by

$$\mathbf{x_{n+1}} = \omega \mathbf{y_n} + (1 - \omega)\mathbf{x_n} \tag{3.21}$$

where ω was chosen to be 0.5 and $\mathbf{x_0} = \mathbf{y_0}$.

The nVAE encoder and decoder were symmetric, with each having five feedforward layers. The smoothed state $\mathbf{x_n}$ was z-scored when entering the first layer of the encoder. A small value $\epsilon = 1 \times 10^{-5}$ was added to the denominator of the standardization since some time steps had near identical values. Each hidden layer used a *ReLU* activation function followed by a batch normalization layer. The size of the latent dimension $\mathbf{z}$ was chosen to be 2. Since the addition of noise to the rLIF neurons resulted in temporal spiking jitter to the same input stimulus, the variance of the latent representation $\mathbf{z}$ was constrained to be above 1. This was achieved by having the encoder learn the log of the variance instead of the variance directly and then applying the *softplus* function to the estimate. By using this minimum variance, it acted as a regularizing parameter to the noise present in the training data. See Appendix.4 for layer and training hyperparameters.

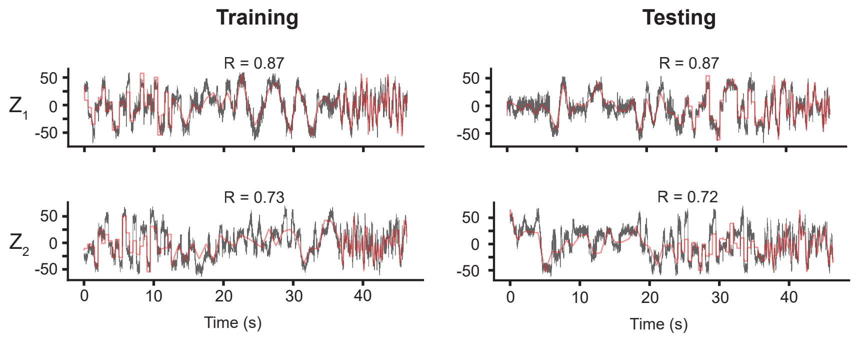


Figure 3.6: **Forecasting Performance of Latent Dynaimcs Model**. (Left) Latent state $\mathbf{z}$ (red) forecasting on the training data. On top is the forecast for $\mathbf{z_1}$ and on bottom $\mathbf{z_2}$. In black is the actual latent trajectory of the training data. The product-moment correlation between the actual and predicted latent states is shown above each figure. (Right) Same as on the left, but with the testing data. The fits between the training and testing forecast are largely identical.

3.4 Latent Dynamics Model

A linear latent dynamics model of the form

$$\mathbf{z_{n+1}} = A\mathbf{z_n} + B\mathbf{v_n} \tag{3.22}$$

was estimated using ridge regression with the L2-hyperparameter chosen via leave-one-out cross-validation. All model fitting was performed using the `sklearn` Python package. The dynamics model was fit with the data produced by the V_{train} input sequence. Model performance was assessed by using an initial value of $\mathbf{z}$ and forecasting for the entire training sequence length. At every time step, the known value of V_{train} and the model's previous prediction of $\mathbf{z}$ was used to forecast the next value. An inadequate model would produce a time-series that was a poor approximation of the actual latent state trajectory produced by V_{train} and could possibly even diverge due to compounding errors.

The forecasted values of the training data were a close fit to the actual $\mathbf{z}$ training

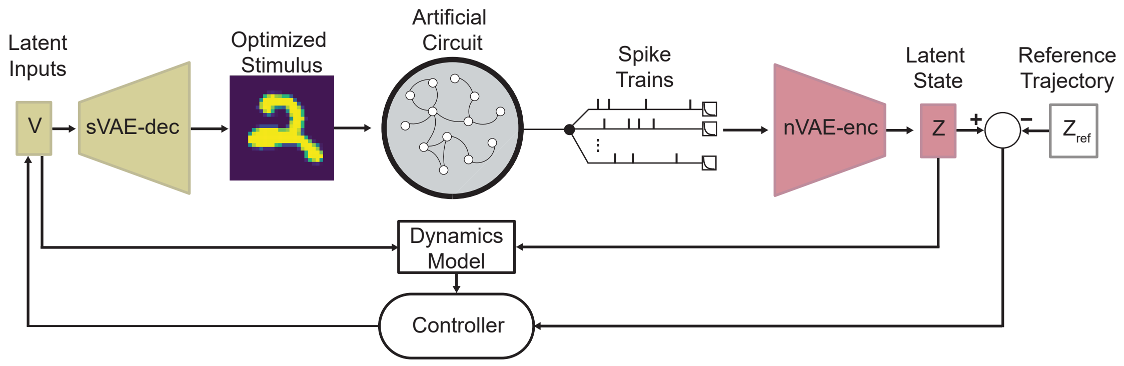


Figure 3.7: **MPC Control Loop of Artificial Circuit.** Exponentially filtered spike trains are encoded into the latent state $\mathbf{z}$ and is compared to a reference trajectory $\mathbf{z_{ref}}$ to produce an error signal. The controller uses a model of the latent dynamics to find an optimal input that minimizes a loss function of the state error. This input is then projected back into the original stimulus dimension using the sVAE decoder which stimulates the artificial circuit.

trajectories as measured with the product-moment correlation ($R_{z_1} = 0.87$, $R_{z_2} = 0.73$). The possibility of overfitting was assessed by forecasting with the latent states elicited from V_{test}. The resulting predicted trajectory was also similar to the actual latent trajectory ($R_{z_1} = 0.87$, $R_{z_2} = 0.72$) indicating that the dynamics model would be useful for control with MPC. See Figure 3.6 for the forecasted latent trajectories of the training and testing data.

3.5 Control Loop

After training the VAEs and obtaining the latent dynamics model, MPC could now be performed. Given some latent input $\mathbf{v}$, the sVAE decoder projects the vector into the original dimension of the MNIST digits. This image serves as a visual stimulus that elicits spikes from the AC. These spikes are exponentially filtered to produce a continuous state $\mathbf{x}$. This state is then projected into a latent representation $\mathbf{z}$ with the nVAE encoder and compared to some reference value $\mathbf{z_{ref}}$. The controller uses the latent dynamics model and latent state errors to find an optimal sequence of latent inputs $\mathbf{v}_{1:T}$ for some time horizon T. The process can repeat for as long as control is desired. See Figure 3.7 for a graphical representation of the control loop.

3.6 Experiment I: Latent Set-Point Control

As a simple first experiment, the latent dynamics were controlled to follow a step function. This reference trajectory was composed of two set-points in latent space, each held constant for 500 ms. The values were chosen by running k-means clustering on the latent state training data and using the resulting centroids as the set-points. The controller had a predictive time horizon T of 30 time steps and optimized the loss function

$$J(\mathbf{z_0}) = \mathbf{z_T^\intercal S z_T} + \sum_{i=0}^{T-1} \mathbf{z_i^\intercal Q z_i} + \Delta\mathbf{v_i^\intercal R} \Delta\mathbf{v_i} \tag{3.23}$$

where

$$\mathbf{Q}, \mathbf{S} = \begin{bmatrix} 150 & 0 \\ 0 & 150 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 10,000 & 0 \\ 0 & 10,000 \end{bmatrix}. \tag{3.24}$$

Unlike the optimization problem in Chapter 2, no other constraints were included. Due to the stochastic nature of the AC responses, 50 independent trials of MPC were performed. The normalized mean square error (nMSE) was used to quantify the controller performance of the latent state, which normalizes mean square error by the difference between the maximum and minimum $\mathbf{z_{ref}}$ values. This was done for ease of comparing controller performance across dimensions with different scales. See Table 3.1 for the results of control across the 50 trials and the root mean square (RMS) of the latent inputs used.

	$\mathbf{nMSE_{z_1}}$	$\mathbf{nMSE_{z_2}}$	$\mathbf{RMS_{v_1}}$	$\mathbf{RMS_{v_2}}$
Mean (SD)	1.58 (0.32)	0.96 (0.19)	0.56 (0.02)	1.56 (0.05)
Min/Max	1.03/2.79	0.58/1.39	0.52/0.61	1.43/1.69

Table 3.1: **Results of Set-Point Control.**

The controller was able to achieve good performance especially when considering that only 20% of the neurons of the AC were observable and the noise present in the system. Even though the noise that was added to all neuron inputs was Gaussian, the nonlinearities in the rLIF models propagate highly complex noise structures throughout the network. See Figure 3.8 for the controller performance in the latent space.

Although the control problem was formulated in the latent space, it produced informative activity in the measurement space. As seen in Figure 3.9 (**A**), the spike count of the observable neurons abruptly changed when the reference trajectory switched from set-point 1 to 2. However, the population spike count then returned to approximately the same levels as the first set-point. An inspection of the spike trains from one of the trials shows that the spiking patterns are qualitatively different between the two set-points. The set-points also appeared to correspond to two distinct types of visual inputs with the decoded latent inputs for set-point 1 producing an image similar to a morph between digits 4 and 9 and the digit 2 for set-point 2 (Figure 3.9 (**B**).

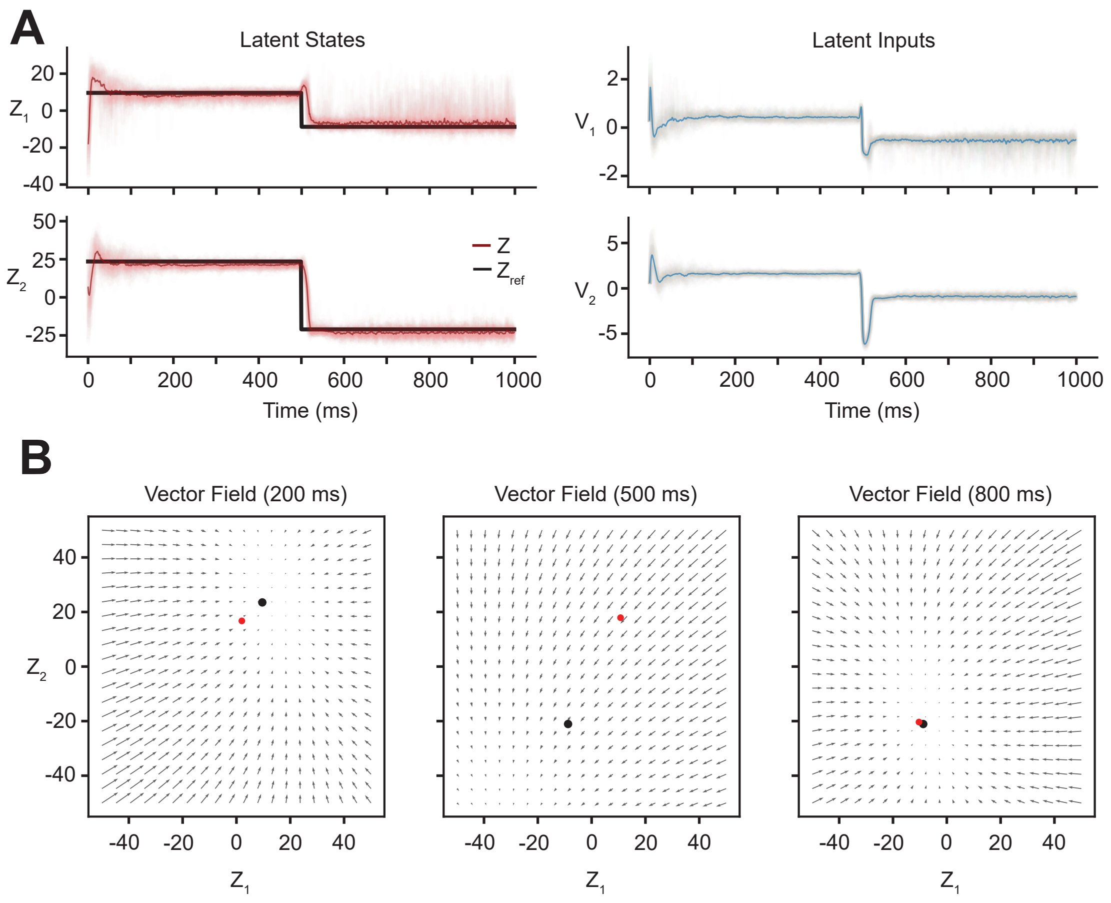
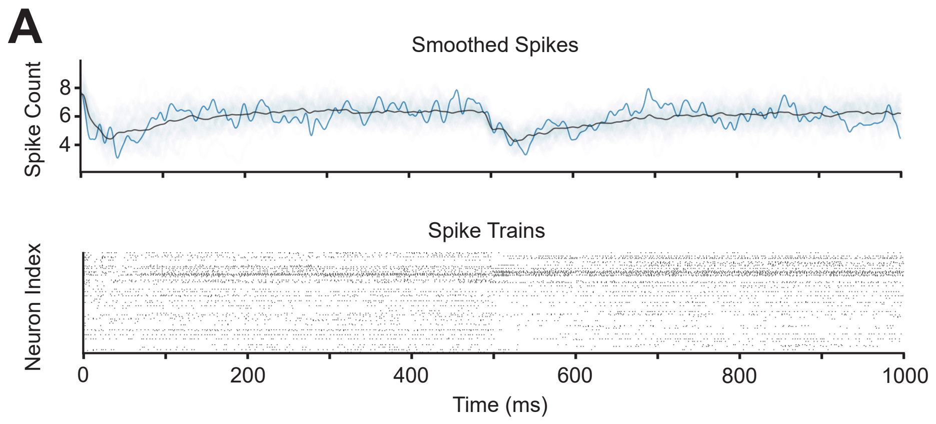


Figure 3.8: **Control of Artificial Circuit in Latent Space**. **A)** (Left) The results of MPC control of the AC across 50 trials. In black are the reference trajectories the latent dynamics were forced to follow. The reference trajectories were composed of two set points that changed at 500 ms. In light red are the controlled latent trajectories across the 50 trials and the average of these trajectories are shown in dark red. (Right) The latent inputs produced by the optimizer. In light blue are the inputs used across the 50 trials and the average input shown in dark blue. **B)** At each time step, the latent dynamics model predicts a vector field on the latent states. Three snapshots of this vector field are shown at 200, 500, and 800 ms. The desired reference points are shown by black dots and the controlled latent states are indicated by red dots. Notice the magnitude of the vectors when the reference point changes at 500 ms.



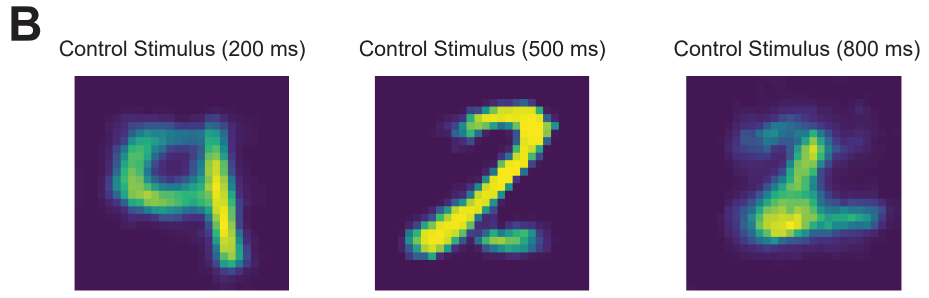

B
Control Stimulus (200 ms)
Control Stimulus (500 ms)
Control Stimulus (800 ms)

Figure 3.9: **Control of Artificial Circuit in Measurement Space**. **A)** (Above) The smoothed spike trains of the measured AC neurons across the 50 trials are shown in light blue with the average shown in black. The smoothed spikes of a sample trial is shown in dark blue. (Below) The measured spike trains of the sampled trial corresponding to the dark blue curve above. Notice the qualitative changes in the spike trains when the reference point changes at 500 ms. Interestingly, while there is an initial drop in the smoothed spike trains at reference point change, it increases back to a similar value to the previous reference point. This indicates that the latent states are not just functions of the population firing rate, but are functions of particular firing patterns and activity from specific neurons. **B)** Decoded latent inputs from the optimizer across three time points. These are the visual stimuli that drove activity in the network. Notice the images corresponding to the second reference point are similar to the digit '2', but are at different magnitudes and sizes.

3.7 Experiment II: Effects of Partial Observation on Control

To investigate how robust the control strategy was to partial observation of neural states, the procedure from Experiment I was performed on different proportions of measurable neurons in the AC. In order to ensure a fair comparison, the architectures of the nVAEs were kept constant across observation percentages (except for the size of the input layer). For the case where only 1% of the neurons of the AC were observed (n=6), the nVAE achieved almost perfect reconstruction for both training and testing data. The ability of the neural data to be accurately reconstructed is important to establish that the latent space produced by the nVAE encoder preserves much of the neural activity. A poor reconstruction could indicate that the latent space is stripping away important information on the dynamics of the network. As expected, keeping the number of latent dimensions to 2 for the nVAE reduced the reconstruction performance for models with high levels of neural observability (i.e., higher dimensionality). However, there was a floor to this degradation of performance and the reconstruction errors plateaued around 40% observability. Interestingly, having a good nVAE reconstruction performance did not guarantee a good dynamics model could be found. In fact, the models between 1% and 10% had the best nVAE reconstructions but the worst forecasts with their dynamics models. This is likely due to the observed neurons not capturing enough of the network dynamics, which include multiple unmeasured inputs and complex recurrent connections. See Figure 3.10 for nVAE and dynamics model performance across neuron observability percentages.

As in Experiment I, the latent dynamics of each model was controlled to follow a step function of two set-points obtained from *k*-means clustering. Fifty trials of

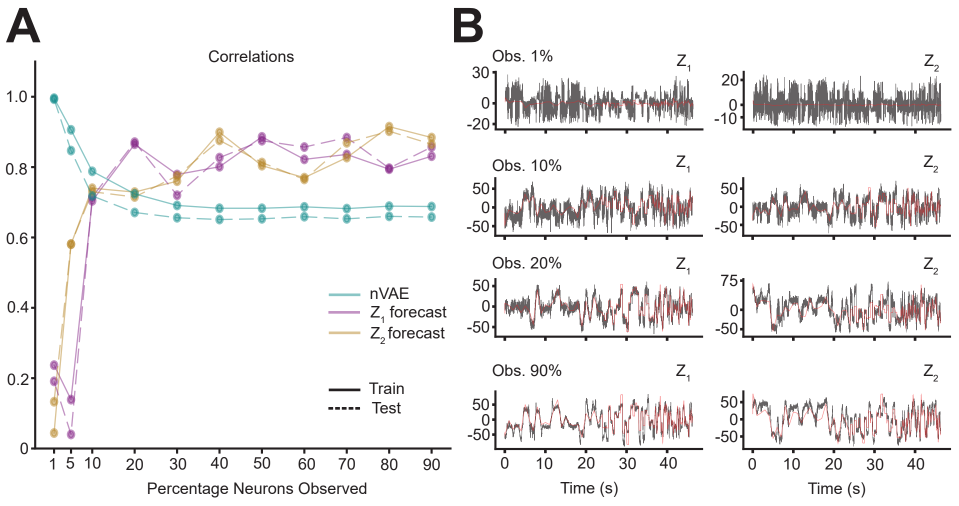


Figure 3.10: **Effects of Partial Observation in Measurement Space. A)** The performance of the nVAE reconstructions as a function of percentage of neurons measured is shown in cyan. For small numbers of measurable neurons, the nVAE finds transformations that preserve much of the information in the original dimension. Since the latent dimension for all nVAEs was constrained to two, as the number of neurons increases more compression is applied, resulting in decreased nVAE reconstruction performance. The latent dynamics models forecasts are shown for both states $\mathbf{z_1}$ (magenta) and $\mathbf{z_2}$ (yellow). **B)** Forecasting models on testing data for four of the levels of observation percentage. In black is the true latent trajectory with the forecasted in red.

MPC were run for each of the models using the same previous loss function and hyperparameters. While in practice, the hyperparameters of the controller would be tuned to the specific dynamics model used, using the same values allowed for easy comparisons in performance.

Controller performance suffered in the models with small observation percentages with 2 of the trials for the 1% observation model diverging. However, the relationship between controller performance and observation percentage was complex. On average, the controllers were able to force the latent dynamics to follow the corresponding reference trajectories, but with differing levels of variability between trials. This can be seen in Figure 3.11 where the controlled latent states are shown for each observation level and across all 50 trials. See Tables 3.2 for quantification of the controlled trajectories and 3.3 of the latent control inputs. Only models up to 60% observability are shown since even this level is extremely unlikely to occur in an actual extracellular experiment.

Model	Z_1 **M(SD)**	Z_1 **Min/Max**	Z_2 **M(SD)**	Z_2 **Min/Max**
Obs 1%	13.81 (0.62)	12.63/15.44	5.91 (0.64)	4.22/7.63
Obs 5%	7.03 (0.75)	5.58/8.67	6.47 (0.67)	5.25/7.99
Obs 10%	86.88 (12.06)	63.82/109.77	5.28 (1.19)	3.71/7.71
Obs 20%	1.58 (0.32)	1.03/2.79	0.96 (0.19)	0.58/1.39
Obs 30%	2.27 (0.12)	2.03/2.55	3.36 (0.22)	2.91/3.77
Obs 40%	0.39 (0.09)	0.24/0.65	0.34 (0.08)	0.21/0.49
Obs 50%	2.17 (0.33)	1.67/2.89	2.45 (0.32)	1.87/3.42
Obs 60%	0.99 (0.13)	0.77/1.38	0.98 (0.08)	0.80/1.88

Table 3.2: **Normalized Mean Squared Error of Latent State Across Models.** *Note*: Two divergent control paths in Obs. 1% model are omitted.

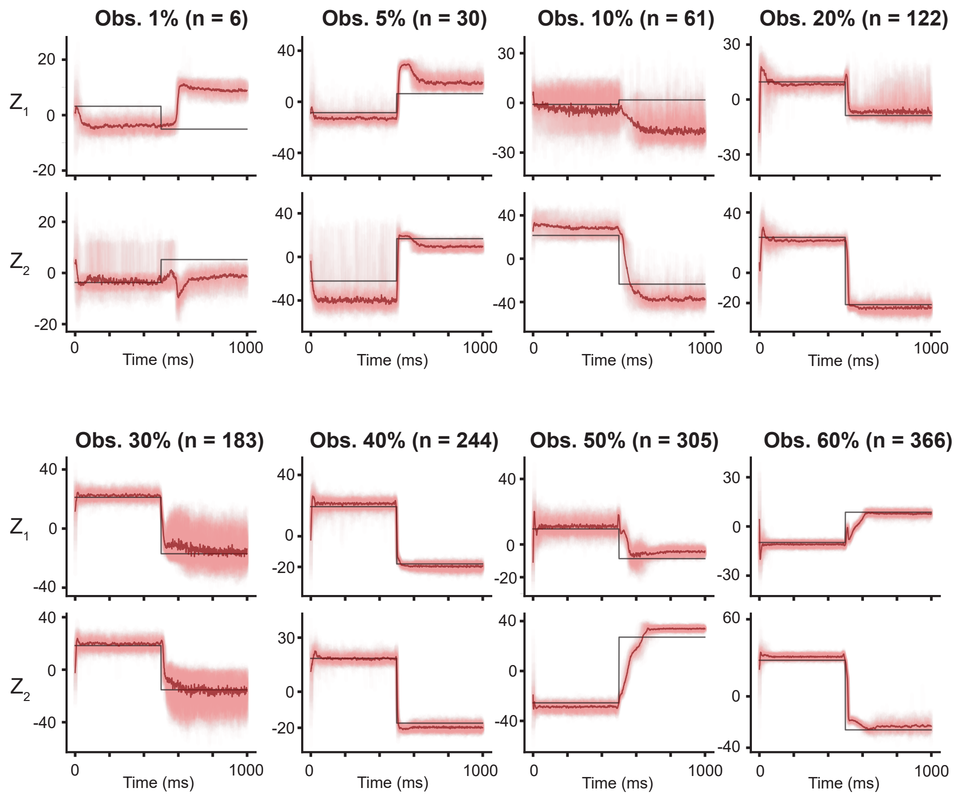


Figure 3.11: **MPC Performance Across Observation Percentage**. Reference (black) and controlled (red) trajectories for models with differing proportions of neurons observed. Notice the poor performance of the controllers using dynamics models estimated with under 20% of neurons observed.

Model	V_1 **M(SD)**	V_1 **Min/Max**	V_2 **M(SD)**	V_2 **Min/Max**
Obs 1%	28.62 (0.14)	28.34/28.93	6.45 (0.04)	6.37/6.55
Obs 5%	3.49 (0.03)	3.40/3.56	0.85 (0.03)	0.76/0.92
Obs 10%	1.23 (0.05)	1.13/1.33	0.96 (0.11)	0.69/1.18
Obs 20%	0.56 (0.02)	0.52/0.61	1.56 (0.05)	1.43/1.69
Obs 30%	1.30 (0.03)	1.23/1.41	1.60 (0.07)	1.45/1.78
Obs 40%	0.48 (0.03)	0.43/0.55	1.11 (0.07)	0.97/1.26
Obs 50%	1.83 (0.09)	1.60/2.03	1.90 (0.16)	1.58/2.28
Obs 60%	1.31 (0.03)	1.25/1.39	1.77 (0.06)	1.65/1.92

Table 3.3: **Root Mean Square of Latent Inputs Across Models.** *Note*: Two divergent control paths in Obs. 1% model are omitted.

3.8 Experiment III: Comparing Latent Reference Trajectories

Although the previous experiments demonstrated that the latent states of the network could be controlled to specific set-points, this offers limited ability to investigate the structure of the latent dynamics. For a true understanding of the dynamics, it is not sufficient to characterize the start and end points of a trajectory, but instead what specific path was taken. As a final experiment, the reference trajectory was replaced with two time-varying functions (reference trajectory 1 and 2). Each of these reference trajectories had the same initial (s_0) and final values (s_f), but took different paths through state space. Using a parameterized function of a circle passing through points s_0 and s_f, trajectories 1 and 2 were the opposite arcs of the resulting circle. This

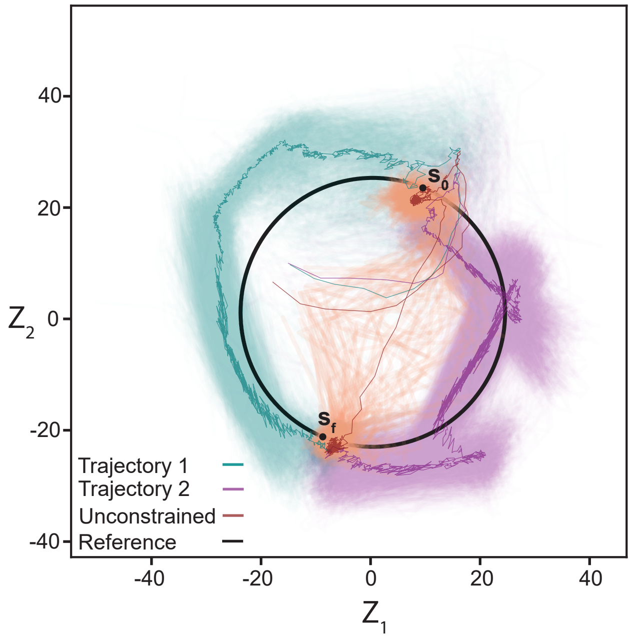


Figure 3.12: **Comparing the Control Performance Between Trajectories**. The controlled paths for the two reference trajectories. Each of the trajectories had the same initial and final values (s_0 and s_f respectively). Using these values, the two trajectories were obtained by fitting a circle between them. The paths for 50 trials with trajectory 1 as the reference are shown in cyan and in purple for trajectory 2. For comparison, the paths from Experiment I 200 ms before and after the change in set-point are shown in red. The set-points were identical to the s_0 and s_f used here, but the path between them was not constrained to follow certain values. The dark colors show the average of the respective path types.

ensured that both trajectories were of equal length through the latent state space. The observation model from Experiment I was used for this task (20% observability) and the values of s_0 and s_f were the set-points from the same experiment. Fifty control trials were performed for each of the reference trajectories with each trial having the same MPC hyperparameters from Experiment I and II. See Figure 3.12 for control performance in latent space. As seen in Table 3.4, while there were slight differences in the average errors in control between the two reference trajectories, there were large differences in the RMS of the latent inputs used to control the system. In particular, for both latent inputs $\mathbf{v_1}$ and $\mathbf{v_2}$ the minimum RMS for trajectory 2 was larger than the maximum RMS for trajectory 1. This indicated that more energy in the latent space was needed to control the system to follow trajectory 2, even though the lengths of the two trajectories were the same. Interestingly, both sets of trajectories had regions of state space that produced large oscillations in the latent states which can be easily seen in Figure 3.13. The errors in the latent states were also qualitatively different between the two reference trajectories, producing strikingly different average controlled paths. Examining the activity in the measurement space, we see obvious differences in the spiking behavior of the AC (Figure 3.14 **A**). The visual stimuli produced from decoding the latent inputs also show distinct differences between the two reference trajectories (Figure 3.14 **B**). This demonstrates that not only do the two trajectories require different spiking patterns to enter different regions of state space, but qualitatively different stimuli are needed. One implication of this is that MPC of the latent dynamics could reveal if specific kinds of stimuli correspond to particular latent trajectories or if the activity in the latent space is driven by the differences in the stimulus at every time step (e.g. prediction error, Egner, Monti, and Summerfield 2010).

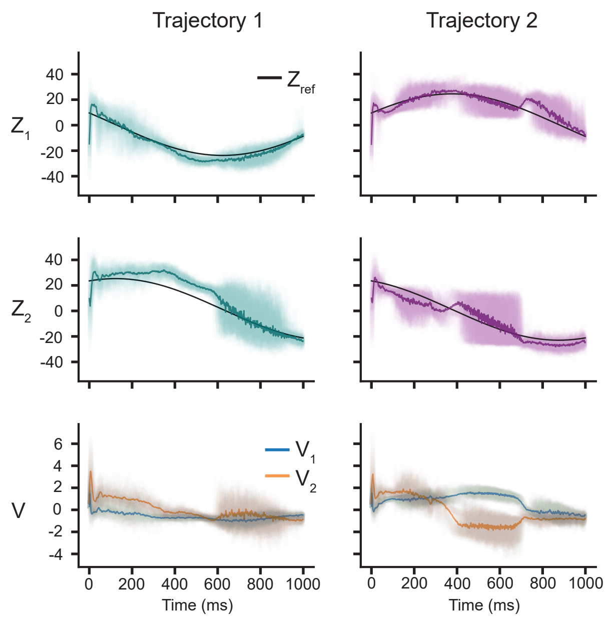


Figure 3.13: **Latent States and Inputs Between Trajectories**. (Left) The controlled latent states are shown in cyan with reference trajectory 1 (black) across the 50 trials. The corresponding latent inputs obtained from the optimizer are shown in blue and orange. (Right) The controlled latent states (purple) and inputs for reference trajectory 2. The dark colors correspond to the average of the respective path types.

	Trajectory 1	**Trajectory 2**
$\mathbf{nMSE_{Z_1}}$		
M(SD)	0.64 (0.18)	1.05 (0.11)
Min/Max	0.36/1.05	0.84/1.22
$\mathbf{nMSE_{Z_2}}$		
M(SD)	1.42 (0.11)	1.34 (0.18)
Min/Max	1.13/1.70	0.97/1.80
$\mathbf{RMS_{V_1}}$		
M(SD)	0.76 (0.04)	1.07 (0.05)
Min/Max	0.69/0.86	0.96/1.15
$\mathbf{RMS_{V_2}}$		
M(SD)	1.04 (0.05)	1.42 (0.05)
Min/Max	0.93/1.14	1.31/1.53

Table 3.4: **Performance of Control Between Trajectories.**

3.9 Conclusions

In this chapter, I have demonstrated that MPC can be used to control the latent states of an SNN using data-driven models of the dynamics. Control is possible even when there is only a limited number of neurons that are observable and there are unknown sources of noise in the network. By reducing the dimensionality of the neural activity and the visual stimuli used to elicit neuron spiking, MPC becomes more computationally tractable for high-dimensional neural systems. These latent states can be fixed to certain set-points, across many levels of neuron observability,

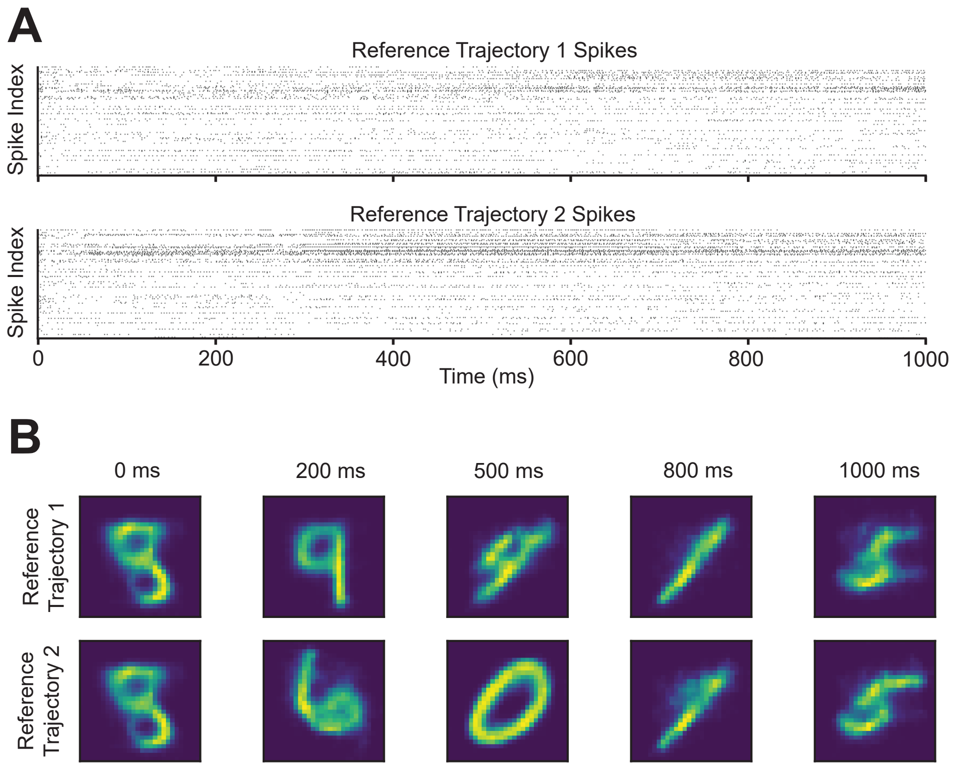


Figure 3.14: **Comparison of Control Inputs and AC Spiking Across Reference Trajectories. A)** Spike trains of a single trial for the reference trajectory 1 and 2 conditions. Note that the high densities of spiking in the Trajectory 2 condition correspond to the same time window as the high oscillations in the latent space. **B)** The decoded latent inputs that are used to stimulate the neurons in **A** during specific time points. The first and last images are largely similar due to s_0 and s_f being identical in both reference trajectories. However, in the intermediate times there are obvious differences in what kinds of images are needed to control the network along the desired trajectory.

and forced to follow time-varying reference trajectories. The results of this simulation provide a framework for the use of MPC for real-time control of neural activity in an experimental setting using extracellular recordings.

In a realistic experiment using extracellular recordings, the proportion of neurons in the circuit that are directly measured is extremely small. The results of Experiment II show that there is a deleterious effect of limited neuron observation in constructing a dynamics model and the subsequent application of MPC. However, it should be noted that only a single sample was used to construct the latent dynamics models for each level of observation percentage. It is possible that the particular samples from the lower observation percentage models were not representative of the population activity. In the case of the 1% observation model this would result in nearly 7×10^{13} possible combinations of neurons that could be used to learn the latent subspace and dynamics models. Future work should examine if this decrease in performance is mainly due to having too few neurons or if small representative samples are adequate for controlling the latent activity. It is also well-known that when observing a subset of states of a dynamical system, time-delay embedding the measurements gives information on the full dynamics (Clark, Fuller, et al. 2022). In each of the experiments here, time-delay embedding was not used to fit the VAEs or latent dynamics models for simplicity. Although this did not appear to impact the controller performance for the higher observation percentage models, it may have resulted in poorer performance as the percentage decreased in value. It would be interesting to examine if using time delays would result in increased performance even when the number of observed neurons is very low. Other work has shown the success of using time-lagged autoencoders to find latent dynamics models (Wehmeyer and Noé 2018) and methods exist to find optimal time delays and the number of embedding dimensions (Sugihara and May 1990). This would introduce additional complexity in practice however, because both the dimensionality reduction and dynamics model need to be fit quickly when recording from living neurons, otherwise the experiment may not be feasible in a laboratory setting due to cell death or electrode drift.

The results of Experiment III showed that different trajectories in latent space required distinct sequences of latent inputs and produced varying levels of control. There are several reasons that this could be the case and will require extensive work to investigate. One possibility is the latent dynamics model made better predictions for certain regions of latent state space. The use of a nonlinear dynamics model may result in a better approximation of the latent vector field, but at the cost of complexity in both model fitting and MPC optimization. Other work has shown reduced dimension transformations that preserve much of the information in the full dimensional space may not be optimal for controlling these latent dynamics (Lusch, J Nathan Kutz, and Steven L Brunton 2018). By fitting the VAEs and latent dynamics models in two distinct phases, this may have produced dynamics models that were not optimal for the subspaces found by the VAEs. An alternative approach would be to simultaneously learn the dimensionality reduction and latent dynamics model in a single step. While this would likely improve the performance of the controller it would also introduce greater complexity when fitting the model in this manner. Additional hyperparameters would be needed to scale the influences of the VAE reconstruction, latent forecasting, and measurement space forecasting. When recording from actual biological neurons, this may result in a hyperparameter space complexity that is too prohibitive in practice.

As previously mentioned, the latent subspace of the SNN dynamics can be seen as a model of a neural manifold. In biological neural networks, neural manifold activity has been found to strongly correlate with organism behaviors across many different contexts. It is still unknown however if the activity in these latent spaces are causally connected to these behaviors. The MPC framework in developed in this chapter provides a method for experimentally probing the activity on these manifolds to better understand their structure and function. If manifold activity is in fact casual, it may be possible to produce specific organism behaviors by controlling the latent neural dynamics. This would be an enormously beneficial tool in furthering the understanding of how complex large-scale behaviors arise from neural activity.

Chapter 4

Conclusions and Future Directions

As our ability to collect vast quantities of neural data has surpassed our theoretical knowledge of the system dynamics, data-driven methods will be essential in increasing our ability to control the activity of the nervous system. By using data-driven methods for approximating system dynamics, MPC has the potential to revolutionize the field of neural control without needing a deep *a priori* knowledge of the biophysics. This would allow for new experimental designs and reduce the need to have hand-engineered patterns of neural stimulation. Instead of the usual methods where an input is injected into the system and the resulting behavior recorded, a complex and precise behavior could be defined in advance with the necessary stimulation found via MPC. There would be numerous therapeutic uses as well, with applications of MPC in neuroprosthetics being an especially promising area of research (Lambeth, Singh, and Sharma 2023; Wolf and Schearer 2022; Singh and Sharma 2023; Bao et al. 2019). Other therapeutic uses include driving neural activity away from a potentially pathological area in state space (e.g. epilepsy (Chatterjee et al. 2020; Brar et al. 2018)). Since MPC is an anticipatory controller, the dynamics model could forecast this activity before it happens and preemptively send a control signal to prevent the pathological state from occurring. The ability to have this level of control over therapeutic and experimental interventions will allow researchers to explore and validate new theories of neural dynamics as well as the relationship between extrinsic and intrinsic modulation of network activity.

One notable area where closed-loop control has been successfully deployed in optogenetic stimulation (Grosenick, Marshel, and Deisseroth 2015; Bergs et al. 2023). In (Newman et al. 2015), the collective firing rate of a population of neurons was controlled using a proportional-integral controller. This work not only demonstrated that real-time closed-loop control of multiple neurons was possible but necessary to overcome the unknown exogenous inputs into the network. Further work by (Bolus et al. 2021) showed through *in vivo* and *in silico* experiments that MPC was possible with optogenetics, however only single neuron control was considered. The results in Chapter 3 could be extended to this experimental modality; instead of finding latent inputs that correspond to images, the decoded input would now be a grid of light intensities over a population of neurons.

Controlling these complex neural systems may require the inclusion of a state estimator in the control loop. In a whole-cell preparation, measurement noise is very low (Sherman-Gold 2012) and one can assume that the measured values of the membrane voltage and injected current are the true values. In a neural circuit, it is not possible to achieve whole-cell access to more that a couple neurons at best, so it is necessary to use extracellular electrophysiology, which only reveals the timing of action potentials, or optical signals of calcium concentration, which tend to be slow and much noisier. Similarly, optogenetic stimulation of neural activity is much less precise than direct current injection. These sources of variability can corrupt the measurements of the system's state, leading to incorrect calculations of the optimal control inputs. If the structure of the noise can be assumed, techniques from robust MPC may lead to improved performance (Bemporad and Morari 1999). Robust MPC can provide a safer control scheme since the effect of disturbances is explicitly modeled and the controller tries to ensure the system does not enter a region of state space that is infeasible or potentially dangerous (Hewing et al. 2020). State estimators can also transform the spike times arising from extracellular recording into estimates of a continuous latent state (Smith et al. 2010).

The work presented here is an incremental step toward the larger goal of precise control of the nervous system. The interdisciplinary collaboration between neuroscientists, engineers, and control theorists will be essential for translating future advancements into practical applications that can benefit both basic research and clinical interventions. Achieving a comprehensive understanding of neural control mechanisms holds immense potential for advancing our ability to treat neurological disorders, enhance brain-computer interfaces, and unlock new frontiers in neuroscience and neurotechnology.

Bibliography

Lorenz, Edward N (1963). "Deterministic nonperiodic flow". In: *Journal of atmospheric sciences* 20.2, pp. 130–141.

Bottjer, Sarah W., Elizabeth A. Miesner, and Arthur P. Arnold (1986). "Changes in neuronal number, density and size account for increases in volume of song-control nuclei during song development in zebra finches". In: *Neuroscience Letters* 67.3, pp. 263–268. ISSN: 0304-3940. DOI: https://doi.org/10.1016/0304-3940(86)90319-8. URL: https://www.sciencedirect.com/science/article/pii/0304394086903198.

Sugihara, George and Robert M May (1990). "Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series". In: *Nature* 344.6268, pp. 734–741.

Park, J. and I. W. Sandberg (June 1991). "Universal Approximation Using Radial-Basis-Function Networks". en. In: *Neural Computation* 3.2, pp. 246–257. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1991.3.2.246. URL: https://direct.mit.edu/neco/article/3/2/246-257/5580 (visited on 10/29/2023).

Johnston, Daniel and Samuel Miao-sin Wu (1995). *Foundations of cellular neurophysiology*. Cambridge, Mass: MIT Press. ISBN: 978-0-262-10053-3.

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

Bemporad, Alberto and Manfred Morari (1999). "Robust model predictive control: A survey". In: *Robustness in identification and control*. Ed. by A. Garulli and A. Tesi. London: Springer London, pp. 207–226. ISBN: 978-1-84628-538-7.

Destexhe, A et al. (Nov. 2001). “Fluctuating synaptic conductances recreate in vivo-like activity in neocortical neurons”. en. In: *Neuroscience* 107.1, pp. 13–24. ISSN: 03064522. DOI: 10.1016/S0306-4522(01)00344-X. URL: https://linkinghub.elsevier.com/retrieve/pii/S030645220100344X (visited on 10/29/2023).

Stefani, Raymond T., ed. (2002). *Design of feedback control systems*. 4th ed. The Oxford series in electrical and computer engineering. New York: Oxford University Press. ISBN: 978-0-19-514249-5.

Qin, S.Joe and Thomas A. Badgwell (July 2003). “A survey of industrial model predictive control technology”. en. In: *Control Engineering Practice* 11.7, pp. 733–764. ISSN: 09670661. DOI: 10.1016/S0967-0661(02)00186-7. URL: https://linkinghub.elsevier.com/retrieve/pii/S0967066102001867 (visited on 10/29/2023).

Fröhlich, Flavio and Sašo Jezernik (Sept. 2005). “Feedback control of Hodgkin–Huxley nerve cell dynamics”. en. In: *Control Engineering Practice* 13.9, pp. 1195–1206. ISSN: 09670661. DOI: 10.1016/j.conengprac.2004.10.008. URL: https://linkinghub.elsevier.com/retrieve/pii/S096706610400214X (visited on 10/29/2023).

Pedrocchi, Alessandra et al. (Dec. 2006). “Error mapping controller: a closed loop neuroprosthesis controlled by artificial neural networks”. en. In: *Journal of NeuroEngineering and Rehabilitation* 3.1, p. 25. ISSN: 1743-0003. DOI: 10.1186/1743-0003-3-25. URL: https://jneuroengrehab.biomedcentral.com/articles/10.1186/1743-0003-3-25 (visited on 10/29/2023).

Rabinovich, Mikhail I. et al. (Nov. 2006). “Dynamical principles in neuroscience”. en. In: *Reviews of Modern Physics* 78.4, pp. 1213–1265. ISSN: 0034-6861, 1539-0756. DOI: 10.1103/RevModPhys.78.1213. URL: https://link.aps.org/doi/10.1103/RevModPhys.78.1213 (visited on 10/29/2023).

Skinner, Frances K (2006). “Conductance-based models”. In: *Scholarpedia* 1.11, p. 1408.

Wächter, Andreas and Lorenz T. Biegler (Mar. 2006). “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. en. In: *Mathematical Programming* 106.1, pp. 25–57. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/s10107-004-0559-y. URL: http://link.springer.com/10.1007/s10107-004-0559-y (visited on 10/29/2023).

Ullah, Ghanim and Steven J. Schiff (Apr. 2009). “Tracking and control of neuronal Hodgkin-Huxley dynamics”. en. In: *Physical Review E* 79.4, p. 040901. ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.79.040901. URL: https://link.aps.org/doi/10.1103/PhysRevE.79.040901 (visited on 10/29/2023).

Egner, Tobias, Jim M Monti, and Christopher Summerfield (2010). “Expectation and surprise determine neural population responses in the ventral visual stream”. In: *Journal of Neuroscience* 30.49, pp. 16601–16608.

Holkar, KS and Laxman M Waghmare (2010). “An overview of model predictive control”. In: *International Journal of Control and Automation* 3.4, pp. 47–63.

Smith, Anne C et al. (2010). “State-space algorithms for estimating spike rate functions”. In: *Computational Intelligence and Neuroscience* 2010, pp. 1–14.

Cunningham, John P. et al. (Apr. 2011). “A closed-loop human simulator for investigating the role of feedback control in brain-machine interfaces”. en. In: *Journal of Neurophysiology* 105.4, pp. 1932–1949. ISSN: 0022-3077, 1522-1598. DOI: 10.1152/jn.00503.2010. URL: https://www.physiology.org/doi/10.1152/jn.00503.2010 (visited on 10/28/2023).

Pedregosa, Fabian et al. (2011). “Scikit-learn: Machine learning in Python”. In: *The Journal of Machine Learning Research* 12, pp. 2825–2830.

Sterratt, David, ed. (2011). *Principles of computational modelling in neuroscience*. OCLC: ocn690090171. Cambridge ; New York: Cambridge University Press. ISBN: 978-0-521-87795-4.

Toth, Bryan A. et al. (Oct. 2011). “Dynamical estimation of neuron and network properties I: variational methods”. en. In: *Biological Cybernetics* 105.3-4, pp. 217–237. ISSN: 0340-1200, 1432-0770. DOI: 10.1007/s00422-011-0459-1. URL: http://link.springer.com/10.1007/s00422-011-0459-1 (visited on 10/29/2023).

Gilja, V. et al. (Aug. 2012). “A brain machine interface control algorithm designed from a feedback control perspective”. In: *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. San Diego, CA: IEEE, pp. 1318–1322. ISBN: 978-1-4577-1787-1 978-1-4244-4119-8. DOI: 10.1109/EMBC.2012.6346180. URL: http://ieeexplore.ieee.org/document/6346180/ (visited on 10/28/2023).

Kostuk, Mark et al. (Mar. 2012). “Dynamical estimation of neuron and network properties II: path integral Monte Carlo methods”. en. In: *Biological Cybernetics* 106.3, pp. 155–167. ISSN: 0340-1200, 1432-0770. DOI: 10.1007/s00422-012-0487-5. URL: http://link.springer.com/10.1007/s00422-012-0487-5 (visited on 10/29/2023).

Sherman-Gold, R (2012). “The axon guide, a guide to electrophysiology and biophysics laboratory techniques”. In: *San Jose: Molecular Devices, LLC*.

Kingma, Diederik P and Max Welling (2013). “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114*.

Afram, Abdul and Farrokh Janabi-Sharifi (Feb. 2014). “Theory and applications of HVAC control systems – A review of model predictive control (MPC)”. en. In: *Building and Environment* 72, pp. 343–355. ISSN: 03601323. DOI: 10.1016/j.

buildenv.2013.11.016. URL: https://linkinghub.elsevier.com/retrieve/pii/S0360132313003363 (visited on 10/28/2023).

Knowlton, Chris et al. (June 2014). “Dynamical estimation of neuron and network properties III: network analysis using neuron spike times”. en. In: *Biological Cybernetics* 108.3, pp. 261–273. ISSN: 0340-1200, 1432-0770. DOI: 10.1007/s00422-014-0601-y. URL: http://link.springer.com/10.1007/s00422-014-0601-y (visited on 10/29/2023).

Meliza, C Daniel et al. (Aug. 2014). “Estimating parameters and predicting membrane voltages with conductance-based neuron models.” English. In: *Biol Cybern* 108.4, pp. 495–516. DOI: 10.1007/s00422-014-0615-5. URL: http://link.springer.com/10.1007/s00422-014-0615-5.

Nowotny, Thomas and Rafael Levi (2014). “Voltage-Clamp Technique”. en. In: *Encyclopedia of Computational Neuroscience*. Ed. by Dieter Jaeger and Ranu Jung. New York, NY: Springer New York, pp. 1–5. ISBN: 978-1-4614-7320-6. DOI: 10.1007/978-1-4614-7320-6_137-2. URL: http://link.springer.com/10.1007/978-1-4614-7320-6_137-2 (visited on 10/27/2023).

Grosenick, Logan, James H. Marshel, and Karl Deisseroth (Apr. 2015). “Closed-Loop and activity-guided optogenetic control”. en. In: *Neuron* 86.1, pp. 106–139. ISSN: 08966273. DOI: 10.1016/j.neuron.2015.03.034. URL: https://linkinghub.elsevier.com/retrieve/pii/S0896627315002585 (visited on 10/27/2023).

Milias-Argeitis, Andreas and Mustafa Khammash (Dec. 2015). “Adaptive Model Predictive Control of an optogenetic system”. In: *2015 54th IEEE Conference on Decision and Control (CDC)*. Osaka: IEEE, pp. 1265–1270. ISBN: 978-1-4799-7886-1. DOI: 10.1109/CDC.2015.7402385. URL: http://ieeexplore.ieee.org/document/7402385/ (visited on 10/29/2023).

Newman, Jonathan P et al. (July 2015). “Optogenetic feedback control of neural activity”. en. In: *eLife* 4, e07192. ISSN: 2050-084X. DOI: 10.7554/eLife.07192. URL: https://elifesciences.org/articles/07192 (visited on 10/28/2023).

Rey, Hernan Gonzalo, Carlos Pedreira, and Rodrigo Quian Quiroga (2015). “Past, present and future of spike sorting techniques”. In: *Brain research bulletin* 119, pp. 106–117.

Mulansky, Mario and Thomas Kreuz (2016). “PySpike—A Python library for analyzing spike train synchrony”. en. In: *SoftwareX* 5, pp. 183–189. ISSN: 23527110. DOI: 10.1016/j.softx.2016.07.006. URL: https://linkinghub.elsevier.com/retrieve/pii/S2352711016300255 (visited on 10/29/2023).

Pang, Rich, Benjamin J Lansdell, and Adrienne L Fairhall (2016). “Dimensionality reduction in neuroscience”. In: *Current Biology* 26.14, R656–R660.

Shanechi, Maryam M., Amy L. Orsborn, and Jose M. Carmena (Apr. 2016). “Robust Brain-Machine Interface Design Using Optimal Feedback Control Modeling and Adaptive Point Process Filtering”. en. In: *PLOS Computational Biology* 12.4. Ed. by Olaf Sporns, e1004730. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1004730. URL: https://dx.plos.org/10.1371/journal.pcbi.1004730 (visited on 10/28/2023).

Wright, James et al. (July 2016). “A Review of Control Strategies in Closed-Loop Neuroprosthetic Systems”. In: *Frontiers in Neuroscience* 10. ISSN: 1662-453X. DOI: 10.3389/fnins.2016.00312. URL: http://journal.frontiersin.org/Article/10.3389/fnins.2016.00312/abstract (visited on 10/28/2023).

Jäckel, David et al. (2017). “Combination of high-density microelectrode array and patch clamp recordings to enable studies of multisynaptic integration”. In: *Scientific reports* 7.1, p. 978.

Shanechi, Maryam M., Amy L. Orsborn, Helene G. Moorman, et al. (Jan. 2017). “Rapid control and feedback rates enhance neuroprosthetic control”. en. In: *Nature Communications* 8.1, p. 13825. ISSN: 2041-1723. DOI: 10.1038/ncomms13825. URL: https://www.nature.com/articles/ncomms13825 (visited on 10/28/2023).

Willett, Francis R et al. (Feb. 2017). “Feedback control policies employed by people using intracortical brain–computer interfaces”. In: *Journal of Neural Engineering* 14.1, p. 016001. ISSN: 1741-2560, 1741-2552. DOI: 10.1088/1741-2560/14/1/016001. URL: https://iopscience.iop.org/article/10.1088/1741-2560/14/1/016001 (visited on 10/28/2023).

Brar, Harleen K et al. (2018). “Seizure reduction using model predictive control”. In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, pp. 3152–3155.

Chen, Andrew N. and C. Daniel Meliza (Mar. 2018). “Phasic and tonic cell types in the zebra finch auditory caudal mesopallium”. en. In: *Journal of Neurophysiology* 119.3, pp. 1127–1139. ISSN: 0022-3077, 1522-1598. DOI: 10.1152/jn.00694.2017. URL: https://www.physiology.org/doi/10.1152/jn.00694.2017 (visited on 10/29/2023).

Kaiser, E., J. N. Kutz, and S. L. Brunton (Nov. 2018). “Sparse identification of nonlinear dynamics for model predictive control in the low-data limit”. en. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2219, p. 20180335. ISSN: 1364-5021, 1471-2946. DOI: 10.1098/rspa.2018.0335. URL: https://royalsocietypublishing.org/doi/10.1098/rspa.2018.0335 (visited on 10/29/2023).

Lusch, Bethany, J Nathan Kutz, and Steven L Brunton (2018). “Deep learning for universal linear embeddings of nonlinear dynamics”. In: *Nature communications* 9.1, p. 4950.

Wehmeyer, Christoph and Frank Noé (2018). "Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics". In: *The Journal of chemical physics* 148.24.

Wouters, Jasper, Fabian Kloosterman, and Alexander Bertrand (2018). "Towards online spike sorting for high-density neural probes using discriminative template matching with suppression of interfering spikes". In: *Journal of neural engineering* 15.5, p. 056005.

Andersson, Joel A. E. et al. (Mar. 2019). "CasADi: a software framework for nonlinear optimization and optimal control". en. In: *Mathematical Programming Computation* 11.1, pp. 1–36. ISSN: 1867-2949, 1867-2957. DOI: 10.1007/s12532-018-0139-4. URL: http://link.springer.com/10.1007/s12532-018-0139-4 (visited on 10/29/2023).

Bao, Xuefeng et al. (Oct. 2019). "Model Predictive Control of a Feedback-Linearized Hybrid Neuroprosthetic System With a Barrier Penalty". en. In: *Journal of Computational and Nonlinear Dynamics* 14.10, p. 101009. ISSN: 1555-1415, 1555-1423. DOI: 10.1115/1.4042903. URL: https://asmedigitalcollection.asme.org/computationalnonlinear/article/doi/10.1115/1.4042903/632790/Model-Predictive-Control-of-a-FeedbackLinearized (visited on 10/29/2023).

Bieker, Katharina et al. (2019). "Deep Model Predictive Control with Online Learning for Complex Physical Systems". In: Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.1905.10094. URL: https://arxiv.org/abs/1905.10094 (visited on 10/29/2023).

Bjoring, Margot C. and C. Daniel Meliza (Jan. 2019). "A low-threshold potassium current enhances sparseness and reliability in a model of avian auditory cortex". en. In: *PLOS Computational Biology* 15.1. Ed. by Arthur Leblois, e1006723. ISSN:

1553-7358. DOI: 10.1371/journal.pcbi.1006723. URL: https://dx.plos.org/10.1371/journal.pcbi.1006723 (visited on 10/29/2023).

Bourdeau, Mathieu et al. (July 2019). "Modeling and forecasting building energy consumption: A review of data-driven techniques". en. In: *Sustainable Cities and Society* 48, p. 101533. ISSN: 22106707. DOI: 10.1016/j.scs.2019.101533. URL: https://linkinghub.elsevier.com/retrieve/pii/S2210670718323862 (visited on 10/29/2023).

Brunton, Steven L. and Jose Nathan Kutz (2019). *Data-driven science and engineering: machine learning, dynamical systems, and control*. Cambridge: Cambridge University Press. ISBN: 978-1-108-42209-3.

Neftci, Emre O., Hesham Mostafa, and Friedemann Zenke (2019). "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks". In: *IEEE Signal Processing Magazine* 36.6, pp. 51–63. DOI: 10.1109/MSP.2019.2931595.

Odaibo, Stephen (2019). "Tutorial: Deriving the standard variational autoencoder (vae) loss function". In: *arXiv preprint arXiv:1907.08956*.

Plaster, Benjamin and Gautam Kumar (Sept. 2019). "Data-Driven Predictive Modeling of Neuronal Dynamics Using Long Short-Term Memory". en. In: *Algorithms* 12.10, p. 203. ISSN: 1999-4893. DOI: 10.3390/a12100203. URL: https://www.mdpi.com/1999-4893/12/10/203 (visited on 10/29/2023).

Raković, Saša V. and William S. Levine, eds. (2019). *Handbook of Model Predictive Control*. en. Control Engineering. Cham: Springer International Publishing. ISBN: 978-3-319-77488-6 978-3-319-77489-3. DOI: 10.1007/978-3-319-77489-3. URL: http://link.springer.com/10.1007/978-3-319-77489-3 (visited on 10/29/2023).

Chatterjee, Sarthak et al. (2020). “Fractional-order model predictive control as a framework for electrical neurostimulation in epilepsy”. In: *Journal of Neural Engineering* 17.6, p. 066017.

Hewing, Lukas et al. (May 2020). “Learning-Based Model Predictive Control: Toward Safe Learning in Control”. en. In: *Annual Review of Control, Robotics, and Autonomous Systems* 3.1, pp. 269–296. ISSN: 2573-5144, 2573-5144. DOI: 10.1146/annurev-control-090419-075625. URL: https://www.annualreviews.org/doi/10.1146/annurev-control-090419-075625 (visited on 10/29/2023).

Beniaguev, David, Idan Segev, and Michael London (2021). “Single cortical neurons as deep artificial neural networks”. In: *Neuron* 109.17, pp. 2727–2739.

Bolus, M F et al. (June 2021). “State-space optimal feedback control of optogenetically driven neural activity”. In: *Journal of Neural Engineering* 18.3, p. 036006. ISSN: 1741-2560, 1741-2552. DOI: 10.1088/1741-2552/abb89c. URL: https://iopscience.iop.org/article/10.1088/1741-2552/abb89c (visited on 10/28/2023).

Chung, SueYeon and Larry F Abbott (2021). “Neural population geometry: An approach for understanding biological and artificial neural networks”. In: *Current opinion in neurobiology* 70, pp. 137–144.

Schwenzer, Max et al. (Nov. 2021). “Review on model predictive control: an engineering perspective”. en. In: *The International Journal of Advanced Manufacturing Technology* 117.5-6, pp. 1327–1349. ISSN: 0268-3768, 1433-3015. DOI: 10.1007/s00170-021-07682-3. URL: https://link.springer.com/10.1007/s00170-021-07682-3 (visited on 10/28/2023).

Zhang, Qiaosheng et al. (June 2021). “A prototype closed-loop brain–machine interface for the study and treatment of pain”. en. In: *Nature Biomedical Engineering* 7.4, pp. 533–545. ISSN: 2157-846X. DOI: 10.1038/s41551-021-00736-7.

URL: https://www.nature.com/articles/s41551-021-00736-7 (visited on 10/28/2023).

Clark, Randall, Luke Fairbanks, et al. (Nov. 2022). *Data Driven Regional Weather Forecasting*. preprint. Predictability, probabilistic forecasts, data assimilation, inverse problems/Climate, atmosphere, ocean, hydrology, cryosphere, biosphere/Big data and artificial intelligence. DOI: 10.5194/egusphere-2022-1222. URL: https://egusphere.copernicus.org/preprints/2022/egusphere-2022-1222/ (visited on 10/29/2023).

Clark, Randall, Lawson Fuller, et al. (June 2022). "Reduced-Dimension, Biophysical Neuron Models Constructed From Observed Data". en. In: *Neural Computation* 34.7, pp. 1545–1587. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco_a_01515. URL: https://direct.mit.edu/neco/article/34/7/1545/111332/Reduced-Dimension-Biophysical-Neuron-Models (visited on 10/29/2023).

Emiliani, Valentina et al. (July 2022). "Optogenetics for light control of biological systems". en. In: *Nature Reviews Methods Primers* 2.1, p. 55. ISSN: 2662-8449. DOI: 10.1038/s43586-022-00136-4. URL: https://www.nature.com/articles/s43586-022-00136-4 (visited on 10/27/2023).

Gomari, Daniel P et al. (2022). "Variational autoencoders learn transferrable representations of metabolomics data". In: *Communications Biology* 5.1, p. 645.

Pandarinath, Chethan and Sliman J. Bensmaia (Apr. 2022). "The science and engineering behind sensitized brain-controlled bionic hands". en. In: *Physiological Reviews* 102.2, pp. 551–604. ISSN: 0031-9333, 1522-1210. DOI: 10.1152/physrev.00034.2020. URL: https://journals.physiology.org/doi/10.1152/physrev.00034.2020 (visited on 10/28/2023).

Wolf, Derek N. and Eric M. Schearer (Apr. 2022). "Trajectory Optimization and Model Predictive Control for Functional Electrical Stimulation-Controlled Reaching". In: *IEEE Robotics and Automation Letters* 7.2, pp. 3093–3098. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2022.3145946. URL: https://ieeexplore.ieee.org/document/9691884/ (visited on 10/29/2023).

Yue, Rongting, Ryan Tomastik, and Abhishek Dutta (May 2022). *Non-linear Model-based Control of Neural Cell Dynamics*. preprint. In Review. DOI: 10.21203/rs.3.rs-580874/v2. URL: https://www.researchsquare.com/article/rs-580874/v2 (visited on 10/29/2023).

Zaaimi, B. et al. (Oct. 2022). "Closed-loop optogenetic control of the dynamics of neural activity in non-human primates". en. In: *Nature Biomedical Engineering* 7.4, pp. 559–575. ISSN: 2157-846X. DOI: 10.1038/s41551-022-00945-8. URL: https://www.nature.com/articles/s41551-022-00945-8 (visited on 10/27/2023).

Zhao, Cheng and Lei Guo (Aug. 2022). "Towards a theoretical foundation of PID control for uncertain nonlinear systems". en. In: *Automatica* 142, p. 110360. ISSN: 00051098. DOI: 10.1016/j.automatica.2022.110360. URL: https://linkinghub.elsevier.com/retrieve/pii/S0005109822002102 (visited on 10/29/2023).

Bergs, Amelie C. F. et al. (Apr. 2023). "All-optical closed-loop voltage clamp for precise control of muscles and neurons in live animals". en. In: *Nature Communications* 14.1, p. 1939. ISSN: 2041-1723. DOI: 10.1038/s41467-023-37622-6. URL: https://www.nature.com/articles/s41467-023-37622-6 (visited on 10/27/2023).

Eshraghian, Jason K. et al. (2023). "Training Spiking Neural Networks Using Lessons From Deep Learning". In: *Proceedings of the IEEE* 111.9, pp. 1016–1054. DOI: 10.1109/JPROC.2023.3308088.

Fehrman, Christof and C Daniel Meliza (2023). "Nonlinear Model Predictive Control of a Conductance-Based Neuron Model via Data-Driven Forecasting". In: *arXiv preprint arXiv:2312.14274*.

Fiedler, Felix et al. (Nov. 2023). “do-mpc: Towards FAIR nonlinear and robust model predictive control”. en. In: *Control Engineering Practice* 140, p. 105676. ISSN: 09670661. DOI: 10.1016/j.conengprac.2023.105676. URL: https://linkinghub.elsevier.com/retrieve/pii/S0967066123002459 (visited on 10/29/2023).

Fortunato, Cátia et al. (2023). “Nonlinear manifolds underlie neural population activity during behaviour”. In: *bioRxiv*.

Fox, Zachary R, Gregory Batt, and Jakob Ruess (Sept. 2023). “Bayesian filtering for model predictive control of stochastic gene expression in single cells”. In: *Physical Biology* 20.5, p. 055003. ISSN: 1478-3967, 1478-3975. DOI: 10.1088/1478-3975/ace094. URL: https://iopscience.iop.org/article/10.1088/1478-3975/ace094 (visited on 10/29/2023).

Lambeth, Krysten, Mayank Singh, and Nitin Sharma (May 2023). “Robust Control Barrier Functions for Safety Using a Hybrid Neuroprosthesis”. In: *2023 American Control Conference (ACC)*. San Diego, CA, USA: IEEE, pp. 54–59. ISBN: 9798350328066. DOI: 10.23919/ACC55779.2023.10155862. URL: https://ieeexplore.ieee.org/document/10155862/ (visited on 10/29/2023).

Langdon, Christopher, Mikhail Genkin, and Tatiana A Engel (2023). “A unifying perspective on neural manifolds and circuits for cognition”. In: *Nature Reviews Neuroscience*, pp. 1–15.

Lin, Linyu et al. (Feb. 2023). “Development and assessment of a model predictive controller enabling anticipatory control strategies for a heat-pipe system”. en. In: *Progress in Nuclear Energy* 156, p. 104527. ISSN: 01491970. DOI: 10.1016/j.pnucene.2022.104527. URL: https://linkinghub.elsevier.com/retrieve/pii/S0149197022004012 (visited on 10/28/2023).

Martinez, Sebastian et al. (2023). “Dynamical models in neuroscience from a closed-loop control perspective”. In: *IEEE Reviews in Biomedical Engineering* 16, pp. 706–

721. ISSN: 1937-3333, 1941-1189. DOI: 10.1109/RBME.2022.3180559. URL: https://ieeexplore.ieee.org/document/9791075/ (visited on 10/27/2023).

Salzmann, Tim et al. (Apr. 2023). “Real-Time Neural MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms”. In: *IEEE Robotics and Automation Letters* 8.4, pp. 2397–2404. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2023.3246839. URL: https://ieeexplore.ieee.org/document/10049101/ (visited on 10/29/2023).

Senthilvelmurugan, Nambi Narayanan and Sutha Subbian (Aug. 2023). “Active fault tolerant deep brain stimulator for epilepsy using deep neural network”. en. In: *Biomedical Engineering / Biomedizinische Technik* 68.4, pp. 373–392. ISSN: 0013-5585, 1862-278X. DOI: 10.1515/bmt-2021-0302. URL: https://www.degruyter.com/document/doi/10.1515/bmt-2021-0302/html (visited on 10/29/2023).

Singh, Mayank and Nitin Sharma (May 2023). “Data-driven Model Predictive Control for Drop Foot Correction”. In: *2023 American Control Conference (ACC)*. San Diego, CA, USA: IEEE, pp. 2615–2620. ISBN: 9798350328066. DOI: 10.23919/ACC55779.2023.10156600. URL: https://ieeexplore.ieee.org/document/10156600/ (visited on 10/29/2023).

Sun, Weinan et al. (2023). “Learning produces a hippocampal cognitive map in the form of an orthogonalized state machine”. In: *bioRxiv*, pp. 2023–08.

Zheng, Yingzhe and Zhe Wu (Feb. 2023). “Physics-Informed Online Machine Learning and Predictive Control of Nonlinear Processes with Parameter Uncertainty”. en. In: *Industrial & Engineering Chemistry Research* 62.6, pp. 2804–2818. ISSN: 0888-5885, 1520-5045. DOI: 10.1021/acs.iecr.2c03691. URL: https://pubs.acs.org/doi/10.1021/acs.iecr.2c03691 (visited on 10/29/2023).

Taylor, Luke et al. (2024). “Temporal prediction captures retinal spiking responses across animal species”. In: *bioRxiv*, pp. 2024–03.

Appendices

.1 Connor-Stevens Model Parameters

Parameter	**Type-I (Type-II)**	**Unit**
C	1	μFcm^{-2}
E_{Na}	50	mV
E_K	-77	mV
E_A	80	mV
E_l	-22 (-72.8)	mV
g_{Na}	120	mScm^{-2}
g_K	20	mScm^{-2}
g_A	47.7 (0)	mScm^{-2}
g_l	0.3	mScm^{-2}

.2 Artificial Circuit Hyperparameters

$\boldsymbol{\eta}$	$\boldsymbol{\beta}$	**Trial Length (ms)**	**Learning Rate**	**Number of Epochs**	**Batch Size**
0.5	0.99	50	5×10^{-4}	3	128

Table 1: **Training Hyperparameters**

.3 sVAE Hyperparameters

.3.1 sVAE Layers

Layer	**Filters**	**Kernel Size**	**Stride**	**Padding**	**Output Size**
Conv1	32	4	2	1	14x14
Conv2	64	4	2	1	7x7
Linear Layer	-	-	-	-	256

Table 2: **Convolutional VAE Layer Hyperparameters**

.3.2 sVAE Training

Learning Rate	**Batch Size**	**Number of Epochs**
1×10^{-3}	64	20

Table 3: **Training Hyperparameters**

.4 nVAE Hyperparameters

Hidden Layer Sizes	**Learning Rate**	**Batch Size**	**Number of Epochs**
$[128, 256, 1024, 256, 128]$	1×10^{-5}	256	2000

Table 4: **Training Hyperparameters**