Nonlinear model predictive control of a conductance-based neuron model via data-driven forecasting

Journal of Neural Engineering

PAPER

Nonlinear model predictive control of a conductance-based neuron model via data-driven forecasting

Christof Fehrman[1] and C Daniel Meliza[1,2,*]

[1] Psychology Department, University of Virginia, Charlottesville, VA, United States of America
[2] Neuroscience Graduate Program University of Virginia, Charlottesville, VA, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: cdm8j@virginia.edu

Keywords: model predictive control, data-driven forecasting, hodgkin-huxley, optimal control

OPEN ACCESS

RECEIVED
29 December 2023

REVISED
9 July 2024

ACCEPTED FOR PUBLICATION
23 August 2024

PUBLISHED
17 September 2024

Abstract

Objective. Precise control of neural systems is essential to experimental investigations of how the brain controls behavior and holds the potential for therapeutic manipulations to correct aberrant network states. Model predictive control, which employs a dynamical model of the system to find optimal control inputs, has promise for dealing with the nonlinear dynamics, high levels of exogenous noise, and limited information about unmeasured states and parameters that are common in a wide range of neural systems. However, the challenge still remains of selecting the right model, constraining its parameters, and synchronizing to the neural system. *Approach*. As a proof of principle, we used recent advances in data-driven forecasting to construct a nonlinear machine-learning model of a Hodgkin–Huxley type neuron when only the membrane voltage is observable and there are an unknown number of intrinsic currents. *Main Results*. We show that this approach is able to learn the dynamics of different neuron types and can be used with model predictive control (MPC) to force the neuron to engage in arbitrary, researcher-defined spiking behaviors. *Significance*. To the best of our knowledge, this is the first application of nonlinear MPC of a conductance-based model where there is only realistically limited information about unobservable states and parameters.

1. Introduction

1.1. Control of neural systems

Precise control of neural systems is a major goal of modern neuroscience, both as a means for experimental investigation of the brain and as a clinical method for treating neurological disorders [1]. In the most general sense, we seek to find a command signal that will force a specific neuron or network of neurons to follow a specified trajectory through state space [2]. More informally stated, how can we make the system do what we want it to do? 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 [3].

Achieving control presupposes the ability to predictably manipulate the system. In open-loop control, a command signal is chosen ahead of time based on a general model of the system and then applied to an individual instance (figure 1(a)). The outcome may inform 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 [4]. What makes these open-loop is that the intensity and duration of the pulse is not automatically adjusted if the stimulus fails to achieve the desired effect [5]. 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 [6]. 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.

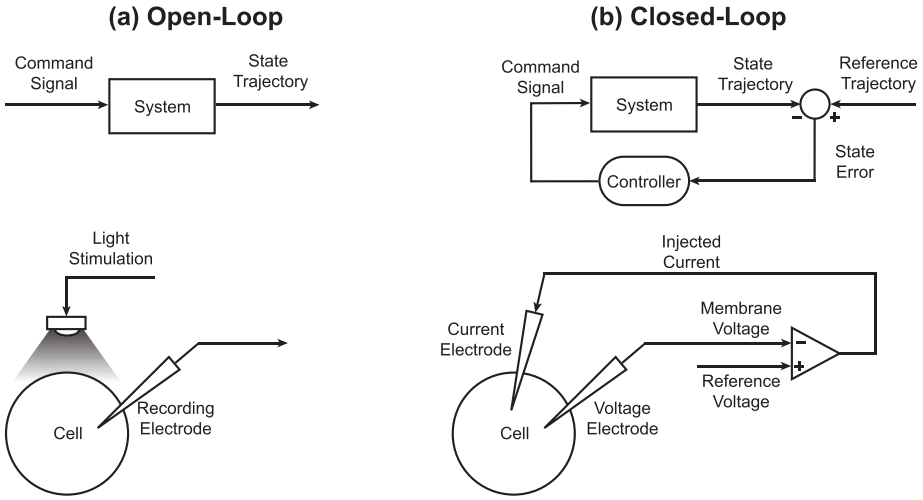

Figure 1. Open- vs Closed-Loop Control. (a) (Above) Block diagram of open-loop control. A command signal is applied to the system irrespective of the system output. (Below) Diagram of typical optogenetic stimulation experiment. A light source is applied to a cell expressing the appropriate light-sensitive 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.

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(b)). 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 [7]. Closed-loop controllers have the ability to adapt to unknown system disturbances and changes in system dynamics even when tracking complicated reference trajectories [8]. Because of this, considerable work has gone into incorporating feedback controllers in other areas of neuroscience such as brain-machine interfaces [9–12] and neuro-prosthetics [9, 13–17]. 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 [18] and population levels [19, 20].

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 control works exceptionally well with intracellular preparations that allow low-noise measurements of voltage and direct injection of current [7]; indeed, voltage-clamp experiments are foundational to almost all of what we know about the physiology of individual neurons. However, there are significant obstacles to using reactive control in networks of neurons, which can have highly nonlinear dynamics in much larger state spaces, more unobservable states and parameters, and proportionally fewer variables that can be experimentally controlled. Our goal in this study is to explore the application of *anticipatory* control to neural systems, using a single neuron with Hodgkin-Huxley dynamics as a proof of principle. Although this is a simple system that does not need sophisticated methods to control it, we are able to use it to address one of the major problems likely to arise when applying more sophisticated methods to complex circuits, namely the lack of ground-truth knowledge about the dynamics and the unmeasureable states of the system.

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

$$J(x_0) = \sum_{i=0}^{T} \ell(x_i, u_i), \tag{1}$$

with constraints

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

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,

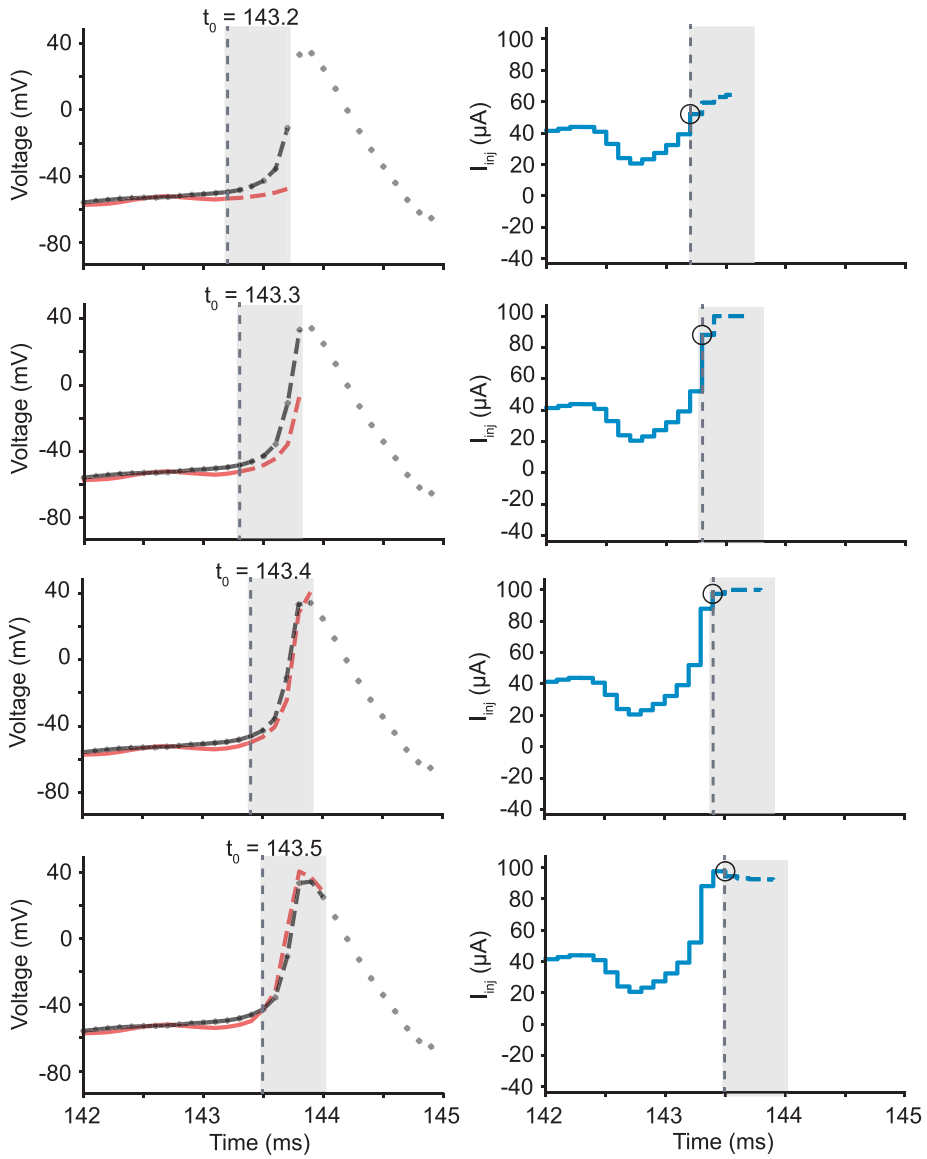

Figure 2. Receding Horizon of MPC. Starting at the top row, (left) the system state trajectory (red) is being controlled to follow the reference trajectory (black). At the current time step (vertical dotted line) the controller finds the optimal set of inputs that minimize the loss function for a specified time horizon (shown with a gray box). In this case, the controller looks ahead 5 time steps (black dashed curve). Given the state at the current time (t_0), 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).

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 [21], but this will not be explored here.

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 2). At each time step, the controller finds an optimal command signal 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 control signal 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 [22]. By finding an optimal input based on predictions of how the system will behave in the future, MPC is an *anticipatory* controller [23].

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 [24, 25].

A commonly used analogy to describe MPC is the game of chess [24], 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 [26]. 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 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 [27]. 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 [28]. MPC has been successfully applied to conductance models in previous work [29–31], 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 [28, 32, 33]. 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 [34] that would not be feasible if the goal is to control a specific neuron or network in a live experiment.

1.3. 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 to various inputs. These models are often referred to as forecasting models since the model predicts how the system will change across time. Data-driven approaches have been successfully applied to MPC problems in diverse fields [21, 35–38]. While there has been previous work using these approaches to model HH-type neuron models, the models were either used solely for prediction (instead of control) [39] or made unrealistic assumptions about which state variables were available in the training data or the extent to which the complement and functional forms of the intrinsic currents could be known [30, 31].

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 intrinsic currents, and generalize to a control scheme where command signals may be outside the training set. As a proof of principle, we 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. We used these observations to create a nonlinear data-driven model that accurately predicted the response of the system to command signal inputs and used this model for MPC. 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* [40], our goal was to demonstrate how data-driven modeling can be applied to nonlinear MPC of neural systems. To our knowledge, this is the first application of nonlinear MPC to a spiking neuron model where realistically limited knowledge of the system is known.

2. Methods

2.1. Connor-stevens model

As a model of single-unit responses to an injected current, we used the HH-type Connor-Stevens (CS) model for all simulations [41]. The CS model includes four intrinsic currents and an extrinsic injected current. We also included a noise current 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

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = I_{\mathrm{Na}} + I_{\mathrm{K}} + I_{\mathrm{A}} + I_{\mathrm{l}} + I_{\mathrm{noise}} + I_{\mathrm{inj}}, \qquad (2)$$

where

$$\begin{aligned} I_{\mathrm{Na}} &= g_{\mathrm{Na}} m^3 h\,(E_{\mathrm{Na}} - V) \\ I_{\mathrm{K}} &= g_{\mathrm{K}} n^4\,(E_{\mathrm{K}} - V) \\ I_{\mathrm{A}} &= g_{\mathrm{A}} a^3 b\,(E_{\mathrm{A}} - V) \\ I_{\mathrm{l}} &= g_{\mathrm{l}}\,(E_{\mathrm{l}} - V)\,, \end{aligned}$$

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. The activity of the neuron can be externally modulated by varying the injected current I_{inj}. For the noise current I_{noise}, random Poisson spike trains were convolved with an alpha function with a decay rate of 10 ms [41] to model the resulting post-synaptic potentials. Each CS model received inputs from both an excitatory and inhibitory Poisson neuron with a firing rate of 20 Hz. The amplitude of $I_{\mathrm{noise}}(\mathrm{t})$ was scaled in reference to the training and validation injected currents to have a constant signal-to-noise ratio (SNR) of 5. Note that all currents are functions of time but we 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 [28, 32, 42], 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 (figure 3). With $g_{\mathrm{A}} = 47.7\mathrm{mS}$ and $E_{\mathrm{l}} = -22$ 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 $g_{\mathrm{A}} = 0\mathrm{mS}$ (eliminating the A-type current) and $E_{\mathrm{l}} = -72.8$ 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, we used both the Type-I and Type-II CS models 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 [43]. Thus, the relationship between the membrane voltage and cellular currents can be expressed using the current conservation equation

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = \sum_i I_i(t)\,,$$

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

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = F(V, X, \Theta, t) + I_{\mathrm{inj}}, \qquad (3)$$

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. We stress that 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 (3).

Let $\mathbf{V} = [V_0, V_1, \ldots, V_T]$ denote a set of discretely sampled membrane voltages where $V_k = V(k\Delta t)$ and Δt is the sampling period. Similarly, let $\mathbf{I} = [I_0, I_1, \ldots, I_T]$ 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 $V_{n+1} = F_{\mathrm{DDF}}(\mathbf{V}, \mathbf{I})$ that can accurately map V_n to V_{n+1}. There are many possible models $F_{\mathrm{DDF}}(.)$ to choose from and as a general rule demand larger amounts of training data as the DDF model gets more complex [44]. Additionally, if one used both the membrane voltage and injected

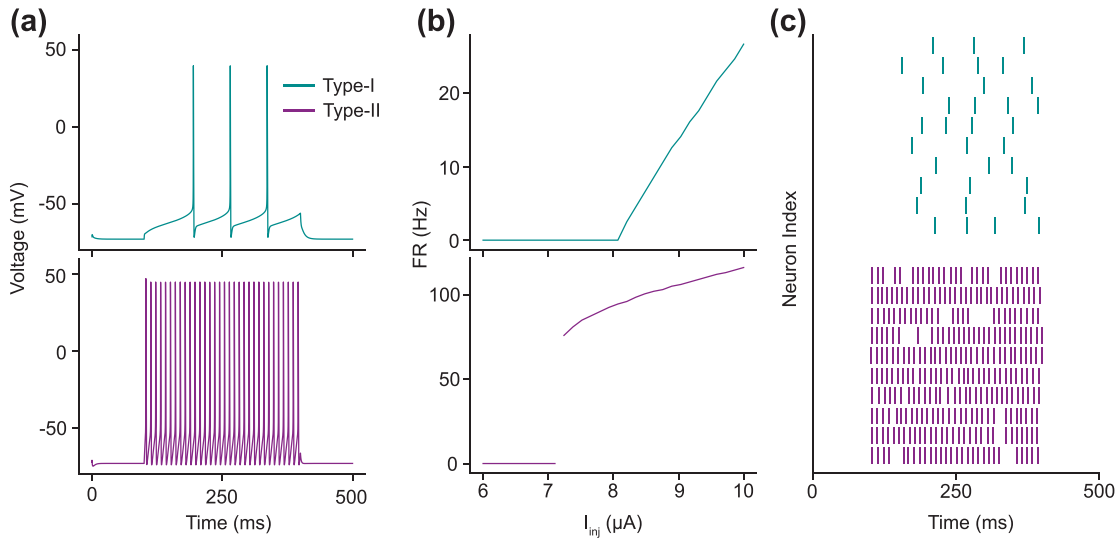

Figure 3. CS Model Behaviors. (a) The spiking pattern of a Type-I (above) and Type-II (below) CS model in response to a 300 ms 9 μA step current. (b) The firing rate of the CS model as a function of step current amplitude. Notice that the Type-I model's firing rate increases approximately linearly after 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.

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 [45] 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 [45], 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) [46]. This is a single hidden-layer artificial neural network (ANN) that typically uses a Gaussian as the nonlinear activation function. Although this is a classical approach largely superseded by more modern forecasting models such as LSTMs [47] and Transformers [48], 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 [49]. The general form of this DDF model is given by the equation

$$V_{n+1} = V_n + F_{\text{RBF}}(S_n) + \alpha(I_{n+1} + I_n), \qquad (4)$$

where V_n is the membrane voltage at the nth time sample, S_n is a time-embedding of V_n, $F_{\text{RBF}}(.)$ 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 appendix for a brief derivation of the DDF model and [45, 50] 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 $x(t)$ state of the Lorenz 63 system [51]. 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 [28]. The differential equations governing the system were scaled in time by $\boldsymbol{\tau}$ and the resulting $x(t)$ trajectory was scaled by amplitude $\mathbf{A}$:

$$\boldsymbol{\tau}\dot{x} = \sigma(y - x) \qquad (5)$$

$$\boldsymbol{\tau}\dot{y} = x(\rho - z) - y \qquad (6)$$

$$\boldsymbol{\tau}\dot{z} = xy - \beta z \qquad (7)$$

$$I_{\text{inj}}(t) = \mathbf{A}x(t) \qquad (8)$$

where $\boldsymbol{\tau} = 20$ and the amplitudes for the CS model Type-I and Type-II models were 1.8 and 0.5 respectively.

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 5, 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. RBFN hyperparameters

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

$$\psi_c(S_n) = \exp\left\{-R||S_n - \mu_c||^2\right\}. \tag{9}$$

The dimension of the time-embedding was chosen using the Simplex method as described in [52] with the `pyEDM` package. Using a time delay τ^* of 1, the optimal predictive embedding dimension D_e was found to be 2 for both DDF models (i.e. $S_n = [V_n, V_{n-1}]$). The center vectors μ_c (N = 50) were obtained by performing k-means clustering in the time-embedded space of the training data. For all RBFs, a length scaling parameter of $R = 0.01$ was used.

2.2.3. RBFN training

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

$$W = \left(X^T X + \lambda I\right)^{-1} X^T Y, \tag{10}$$

where

$$Y = \begin{bmatrix} V_1 - V_0 \\ V_2 - V_1 \\ \vdots \\ V_T - V_{T-1} \end{bmatrix},$$

$$X = \begin{bmatrix} \psi_1(S_0) & \psi_2(S_0) & \dots & \psi_N(S_0) & I_1 + I_0 \\ \psi_1(S_1) & \psi_2(S_1) & \dots & \psi_N(S_1) & I_2 + I_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \psi_1(S_{T-1}) & \psi_2(S_{T-1}) & \dots & \psi_N(S_{T-1}) & I_T + I_{T-1} \end{bmatrix},$$

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ \alpha \end{bmatrix}. \tag{11}$$

Model training was performed using the `sklearn` python package [53] with 10-fold cross-validation to obtain an optimal λ regularization parameter.

2.2.4. DDF model evaluation

Although previous work has shown that the DDF model has high accuracy for *in silico* and *in vitro* neurons [45], the sampling rate was much higher than our data ($\geqslant$ 50 kHz). To assess whether DDF would work on 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, we performed open-loop forecasting on novel injected currents. Because the DDF models had a time-embedding parameters $D_e = 2$ and $\tau^* = 1$, the first two Δt time samples were used to seed the model. As seen in figure 4, the DDF model was able to accurately predict the response of the CS neuron to a novel current injection. Although the DDF model did not accurately forecast every spike, it is important to note that it was only able to use the injected current to make predictions of the time-evolution of the system. We believed this was largely due to the unknown synaptic noise current rather than significant errors in the model. To test this, we compared the DDF predictions to the responses of the original CS neuron to noisy currents (figure 4, right). By running the CS model with different instantiations of the noise current, we can see that the prediction error of the DDF model is comparable to the variability due to the noise current.

2.3. Model predictive control using a DDF model

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

$$\underset{I_1, I_2, \dots, I_T}{\operatorname{argmin}} \mathbf{s} e_T^2 + \sum_{n=0}^{T-1} \mathbf{q} e_{n+1}^2 + \mathbf{r} \Delta I_{n+1}^2, \tag{12}$$

subject to the constraints

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

$$|I_n| \leqslant 100\,\mu A, \tag{13}$$

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 the arg min corresponds to controlling the I_{n+1} term in the DDF model. For the very first optimization loop, we set I_0 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 [54]. One could additionally add another term to the cost function that penalizes the squared magnitude of the input current.

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.

While in principle, one could measure the controlled system once to get the initial values and use the data-driven model forecasts as an estimate of the actual state for the entirety of the control loop, we update the DDF model at every sampling period with the corresponding membrane voltage values of the CS neuron. This was done due to the high amount of

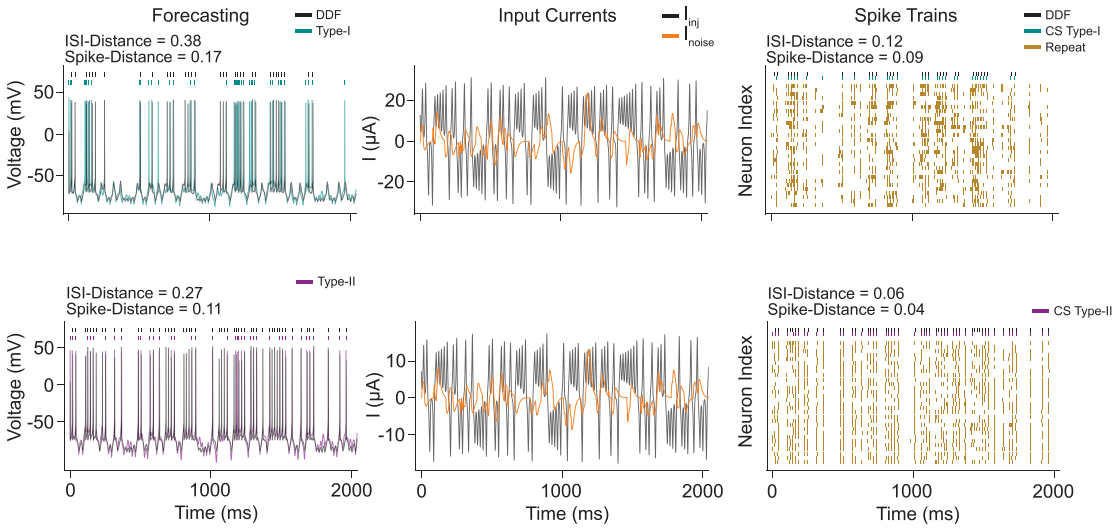

Figure 4. DDF Model Forecasts. Each CS model was stimulated with 5 seconds of a known $I_{\text{inj}}(t)$ and unknown $I_{\text{noise}}(t)$. Each of the DDF models were fit using only the observations of the CS model voltages and the $I_{\text{inj}}(t)$. To evaluate the fit of the DDF models, their forecasted membrane voltages and spike trains were compared to CS models stimulated by a validation set of injected and noise currents. The forecasts were completely open-loop, where the DDF model was not corrected based on errors in predictions. (Left) CS model state trajectories (colored) and DDF model forecasts (black) on 2000 ms of validation data. (Middle) The $I_{\text{inj}}(t)$ (black) and $I_{\text{noise}}(t)$ (orange) currents for the section of validation data. The injected currents used to train and validate the model were obtained using the chaotic Lorenz 63 system. Poisson neurons were used to produce the noise current and had balanced excitation and inhibition. The amplitudes of the noise currents were chosen to result in an SNR of 5 compared to the injected current. This can be seen in the differing scales of the injected and noise currents between the two models. (Right) In black are the DDF forecasted rasters and directly below are the CS model spikes when only stimulated by $I_{\text{inj}}(t)$. We see a strong similarity between the two spike trains indicating that the DDF model learned much of the CS model dynamics. Although the DDF model did not accurately forecast every spike in the validation data (a), this was largely due to the unknown sources of noise. Repeated simulations of the CS model with the same injected current but different noise currents (yellow) show that the CS model without $I_{\text{noise}}(t)$ is deficient in predicting the noisy spike trains.

noise present in the system. In systems were there is a low amount of noise, data-driven models that accurately forecast the dynamics could be used without the need of the constant monitoring of the system states.

All MPC optimizations and implementations were performed using the `do-mpc` python package [55]. This package utilizes `CasADi` [56] and `IPOPT` [57] for optimization and automatic differentiation methods. See table 1 in appendix for controller hyperpameters.

Note that the range in which the control input operates is higher than is typical of patch-clamp experiments [40]. 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 1 cm^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 [41] 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.

3. Results

3.1. Homogeneous system control

Our 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}). We refer to this as homogeneous system control because each CS model was forced to track a reference trajectory it previously produced. For each CS model, we simulated 50 trials of 1-second responses to a chaotic current similar to the one used in the training data. We then used MPC with the corresponding DDF model to find I_{inj} such that the errors in state tracking were minimized, thereby forcing the CS model to reproduce each of these 50 trials.

We repeated each trial 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 us a reference for the

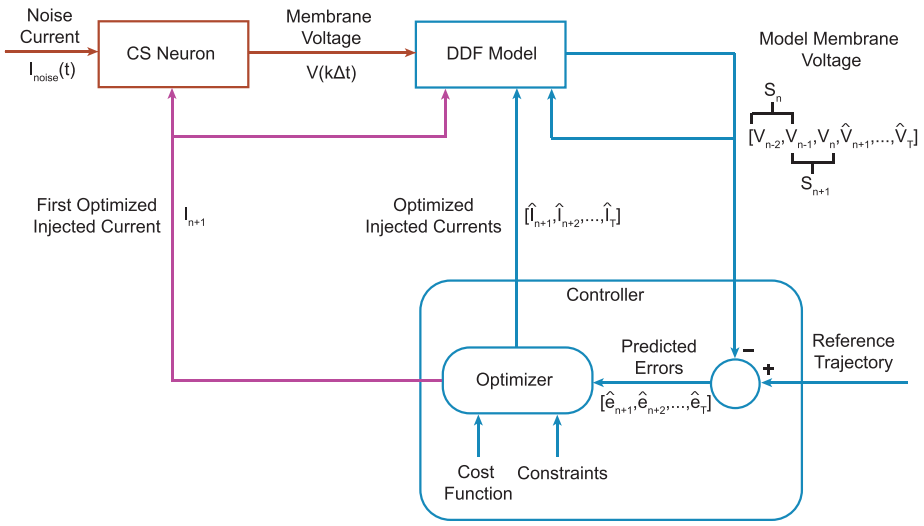

Figure 5. MPC via DDF Diagram. (Red) The CS neuron receives an input $I_{\text{inj}}(t)$ from the controller at time step n which is held constant across a time interval of Δt seconds. (Blue) The DDF model gets an update of the CS model membrane voltage every Δt seconds. Given the membrane voltage V_n, 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.

amount of variability we would expect to see without feedback control.

In order to quantify how well the MPC and open-loop control performed, we used three measures of fit: 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 [58]. 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 6, 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.

3.2. Heterogeneous system control

As a more difficult test of the MPC controller, we investigated whether it could force a CS neuron of one type to follow a reference trajectory generated by the other type. In other words, could a Type-I neuron be made to spike like a Type-II neuron? We refer to this as heterogeneous system control because one dynamical system is being forced to behave as a different dynamical system. We used the same MSE and spike train similarity metrics 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 7.

3.3. Spike train control and comparison to other control methods

In many neuroscience studies, the experimenter wants to make a neuron spike at specific times without caring too much about the subthreshold activity. Spike trains are a point process, whereas the DDF models forecast a continuous variable. To overcome this mismatch in data structure, we extracted the mean spike waveform from the training data of

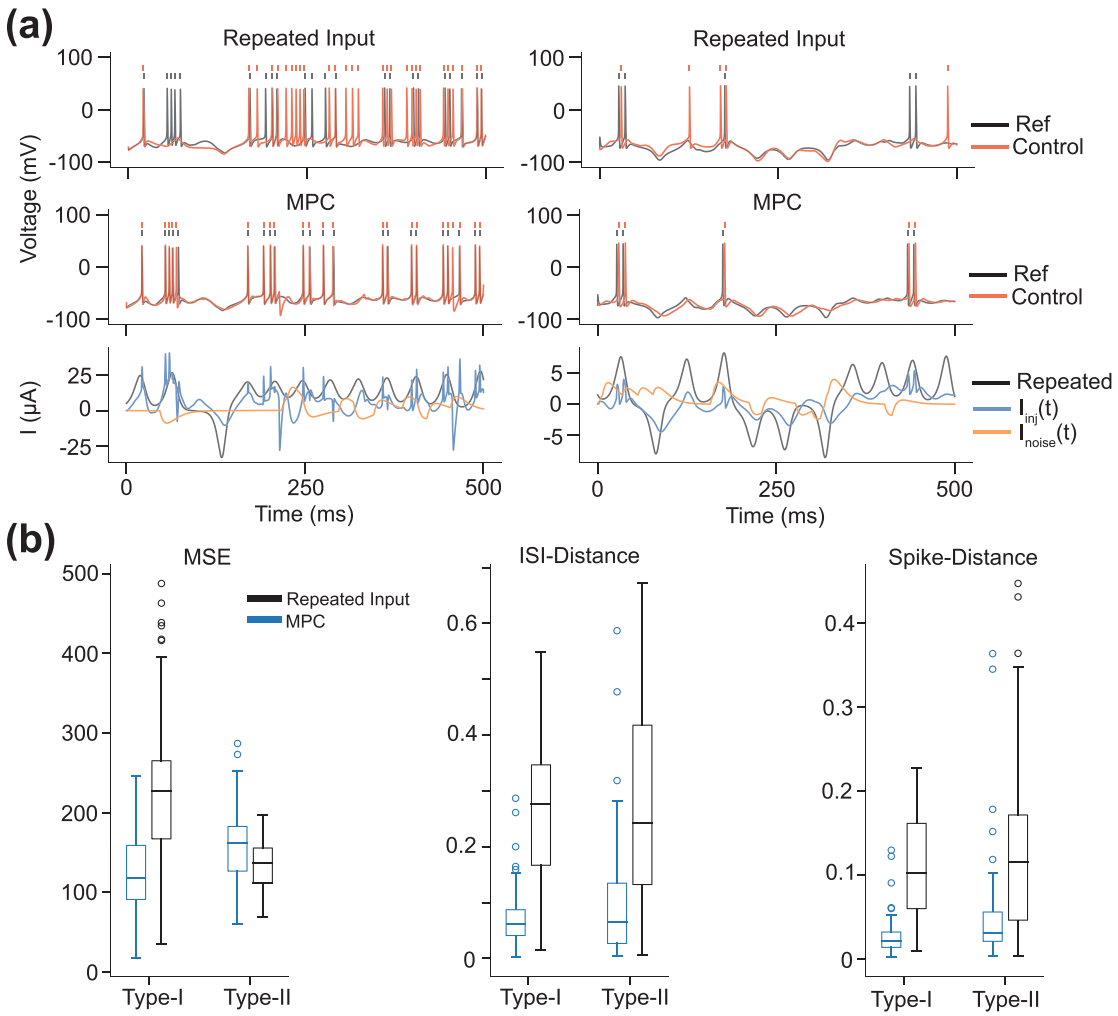

Figure 6. Results of Homogeneous System Control. (a) (Left) Example of a membrane voltage trajectory for a Type-I CS model with open-loop control. In black is the reference trajectory and in red is the controlled trajectory. Rasters above indicate spike times. The same $I_{\text{inj}}(t)$ used to generate the reference trajectory was used as the input for the open-loop control. However, the unknown noise current $I_{\text{noise}}(t)$ into the CS model resulted in the controlled membrane voltage deviating from the reference trajectory. (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 (orange) into the CS model, the $I_{\text{inj}}(t)$ used to generate the reference trajectory and as the open-loop input (black), and MPC optimized input used to control the CS model (blue). (Right) The same diagrams as on the left, but for a Type-II CS model. (b) Performance metrics comparing the open-loop and MPC methods of control. The MSE was calculated for each reference/control trajectory pair. ISI-distance and spike-distance are both on the interval [0,1] where 0 indicates identical spike trains. Although the MSE for the Type-II CS model is slightly lower compared to the open-loop method, the spike train similarity metrics are much better for MPC.

each CS model 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, we were able to convert any spike train into a reference trajectory with units of mV.

As shown in figure 8(a), MPC achieved good control of spike timing for both CS models. However, as previously stated, there are decades of work showing that controlling the firing of individual neurons is readily achievable with simple methods. With the increased use of data-driven methods for complex control schemes, it is important to consider when methods like MPC are more beneficial compared to traditional methods of control. To illustrate this, we also controlled the CS models with a proportional controller and open-loop pulse control. The proportional controller scaled the state error by a gain parameter of $K_p = -2$ for both CS models. While this controller does not explicitly use constraints to find the control signal, the maximum and minimum values were clamped to the same values as the MPC constraints. This simple closed-loop controller demonstrated exceptional performance (figure 8(b)) which is unsurprising given that this is essentially the methodology used in voltage-clamp [40]. Similarly, an entirely open-loop 2 ms pulse is able to produce consistent firing to the desired spike train of both CS

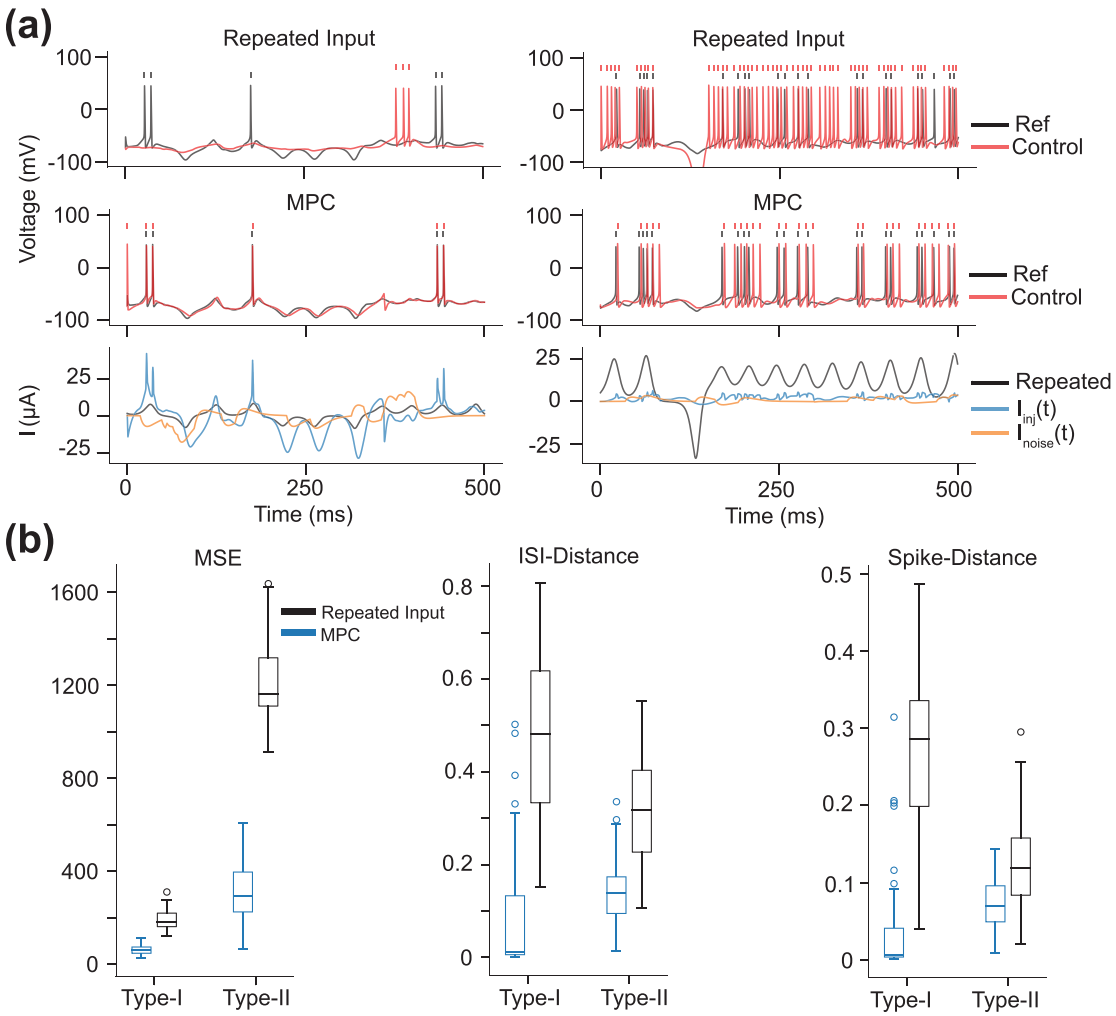

Figure 7. Results of Heterogeneous System Control. (a) (Left) Example of a membrane voltage trajectory for a Type-I CS model with open-loop control. The reference trajectory (black) was obtained from a Type-II CS model. In this example, the controlled trajectory (red) exhibited very little spiking and did not closely follow the reference trajectory. This was expected since the control input for the open-loop controller was the $I_{\text{inj}}(t)$ used to generate the reference trajectory. The Type-II model fires at a lower amplitude input than the Type-I model, and therefore the open-loop input would be weakly stimulating to the Type-I model. (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 (orange) into the CS model, the $I_{\text{inj}}(t)$ 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. (Right) The same diagrams as on the left, 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 since the control input is the $I_{\text{inj}}(t)$ used to generate the Type-I reference trajectory (which fires at higher input amplitudes compared to the Type-II model). However, MPC is able to more accurately control the Type-II neuron into following the Type-I reference trajectory. (b) Performance metrics comparing the open-loop and MPC methods of control. In all cases, MPC achieved much better control than the open-loop controller.

models (figure 8(c)). While this would not prevent spikes produced by the unknown noise current, it is completely adequate for low noise systems.

Although models of individual neurons are a good test bed for building data-driven models for use in MPC, we again stress that this methodology is not universally superior to simpler alternative methods. In practice, MPC is most useful in high dimensional systems and when there are many constraints [24]. Given the ever increasing amount of data that can be recorded from neural activity (e.g. [59]), we have no doubt that data-driven MPC will become an important experimental tool in neuroscience.

4. Discussion

Neural systems can be difficult to control because of their nonlinear dynamics and many hidden states [27]. Here we 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 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, we are

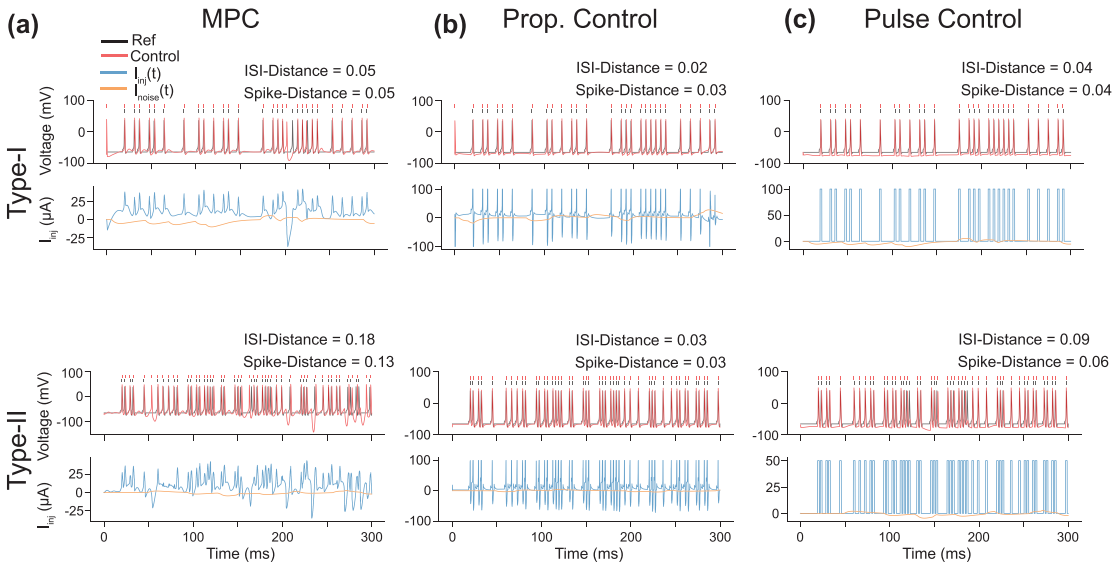

Figure 8. Results of Spike Train Control. (a) Membrane voltage trajectory controlled via MPC for a Type-I (top) and Type-II model (bottom) to follow a specific spike train. The reference trajectory (black) was obtained by taking the average spiking waveform for the corresponding CS model type and embedding it into a constant valued time series (−65 mV) at time points where a spike is desired. MPC was able to control CS models with their resulting membrane voltages (red) following the reference trajectories. (b) A proportional feedback controller is able to achieve much better performance compared to MPC in this example. The values of the control signal were clamped between −100 and 100 μA in accordance with the constraints but on the MPC controller. For both the Type-I and Type-II CS models, the proportional controller produces a sharp excitatory input at the desired spike time follwed by an strong inhibitory input. This inhibitory input is largely unneeed, but since the controller has no knowledge of the system dynamics (unlike MPC) it attempts to minimize the state error when the controlled spikes occur at a slightly delayed time compared to the reference spikes. (b) Open-loop control with 2 ms wide pulses of current also produce spike trains exteremly similar to the desired set. The amplitudes of the two pulses required empirical scaling to ensure that the neurons would both fire when desired and not fire more than once to a single pulse.

able to do this 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. 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 [40]. 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. To our knowledge, 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 [29–31, 33], 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 [60]. 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

$$J(x_0) = e_T^\intercal \mathbf{S} e_T + \sum_{n=0}^{T-1} e_{n+1}^\intercal \mathbf{Q} e_{n+1} + \Delta I_{n+1}^\intercal \mathbf{R} \Delta I_{n+1}, \tag{14}$$

where e_i and ΔI_i have the same meaning as in equation (12) but are now vector-valued with each element corresponding to an individual state and external input source respectively. The matrices **S**, **Q**, and **R** function largely the same as the scaling factors **s**, **q**, and **r** in equation (12), but now allow one to differentially weight the cost of each of the elements of vectors e and 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 **Q** and **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* [18] and simulation models [61, 62]. 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. We were able to construct DDF models 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.

Controlling networks may also require the inclusion of a state estimator in the control loop. In a whole-cell preparation, measurement noise is very low [40] 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 than 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 [63]. 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 [21]. State estimators can also transform the spike times arising from extracellular recording into estimates of a continuous latent state [64]. Latent factor models could also be used as a latent state estimator which would 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 factor model could express the coordinates of the network state and reference trajectory in the lower dimensional space [65] (often referred to as the neural manifold [66]).

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 nonlinear DDF models to approximate system dynamics, MPC has the potential to advance 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 given to 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 [67–70]. Other therapeutic

applications that would require network-level control include driving neural activity away from potentially pathological areas in state space (e.g. epilepsy [71, 72]). Because 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.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/melizalab/mpc-hh.

Appendix

A.1. DDF model derivation

We assume that the continuous time dynamics of the membrane voltage are given by

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = F(V,X,\Theta,t) + I_{\mathrm{inj}}. \tag{15}$$

Let t_n denote the nth time sample of measurement. Following [45], we can construct a DDF model by integrating over some time interval $\Delta t = t_{n+1} - t_n$,

$$V_{n+1} - V_n = \frac{1}{C}\int_{t_n}^{t_{n+1}} F(V,X,\Theta,t')\,\mathrm{d}t' + \frac{1}{C}\int_{t_n}^{t_{n+1}} I_{\mathrm{inj}}(t')\,\mathrm{d}t' \tag{16}$$

where we denote $V(t_n)$ as V_n for notational convenience.

Since the injected current I_{inj} is set by the researcher, it can be approximated by the trapezoidal rule,

$$V_{n+1} - V_n = \frac{1}{C}\int_{t_n}^{t_{n+1}} F(V,X,\Theta,t')\,\mathrm{d}t' + \alpha(I_{n+1} + I_n) \tag{17}$$

where $\alpha = \frac{\Delta t}{2C}$. We can now approximate the unknown integral with radial basis function network (RBFN) denoted by F_{RBF},

$$V_{n+1} = V_n + F_{\mathrm{RBF}}(S_n) + \alpha(I_{n+1} + I_n) \tag{18}$$

where S_n is a time-delay embedding of the membrane voltage. The time delay of each embedding τ and the number of delays D_e are hyperparameters that must be tuned. For all simulations, we set $\tau^* = 1$ and $D_e = 2$, i.e. $S_n = (V_n, V_{n-1})$. The weights of the RBFN can be obtained by minimizing the cost function

$$\sum_{i=0}^{N}\left[V_{i+1} - V_i - F_{\mathrm{RBF}}(S_i) - \alpha I_{i+1} - \alpha I_i\right]^2 \tag{19}$$

which can be obtained using standard least squares methods.

A.2. Connor-stevens neuron

CS model equations

$$\begin{aligned} C\frac{\mathrm{d}V}{\mathrm{d}t} &= I_{\mathrm{Na}} + I_{\mathrm{K}} + I_{\mathrm{A}} + I_{\mathrm{l}} + I_{\mathrm{noise}} + I_{\mathrm{inj}} \\ I_{\mathrm{Na}} &= g_{\mathrm{Na}} m^3 h\,(E_{\mathrm{Na}} - V) \\ I_{\mathrm{K}} &= g_{\mathrm{K}} n^4\,(E_{\mathrm{K}} - V) \\ I_{\mathrm{A}} &= g_{\mathrm{A}} a^3 b\,(E_{\mathrm{A}} - V) \\ I_{\mathrm{l}} &= g_{\mathrm{l}}\,(E_{\mathrm{l}} - V) \end{aligned}$$

CS model parameters

Parameter	Type-I (Type-II)
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}

The full list of parameter values and first-order kinetic equations can be found in [41].

A.3. MPC Hyperparameters

Table 1. MPC controller hyperparameters used for each experiment.

	q	s	r	T
Experiment I				
Type-I	5	1	7	5
Type-II	1	0	100	5
Experiment II				
Type-I	5	1	7	5
Type-II	1	0	100	5
Experiment III				
Type-I	5	1	7	5
Type-II	5	1	50	5

ORCID iDs

Christof Fehrman https://orcid.org/0000-0002-2743-404X
C Daniel Meliza https://orcid.org/0000-0002-9395-4369

References

[1] Schiff S J 2011 *Neural Control Engineering: the Emerging Intersection Between Control Theory and Neuroscience* (MIT Press)

[2] Kao T-C and Hennequin G 2019 Neuroscience out of control: control-theoretic perspectives on neural circuit dynamics *Curr. Opin. Neurobiol.* **58** 122–9
[3] Parkes L *et al* 2024 A network control theory pipeline for studying the dynamics of the structural connectome *Nat. Protoc.* (https://doi.org/10.1038/s41596-024-01023-w)
[4] Emiliani V *et al* 2022 Optogenetics for light control of biological systems *Nat. Rev. Methods Primers* **2** 55
[5] Grosenick L, Marshel J H and Deisseroth K 2015 Closed-loop and activity-guided optogenetic control *Neuron* **86** 106–39
[6] Zaaimi B *et al* 2022 Closed-loop optogenetic control of the dynamics of neural activity in non-human primates *Nat. Biomed. Eng.* **7** 559–75
[7] Nowotny T and Levi R 2014 Voltage-Clamp Technique *Encyclopedia of Computational Neuroscience* ed D Jaeger and R Jung (Springer) pp 1–5
[8] R T Stefani 2002 Design of feedback control systems *The Oxford Series in Electrical and Computer Engineering* 4th edn (Oxford University Press)
[9] Shanechi M M, Orsborn A L, Moorman H G, Gowda S, Dangi S and Carmena J M 2017 Rapid control and feedback rates enhance neuroprosthetic control *Nat. Commun.* **8** 13825
[10] Gilja V, Nuyujukian P, Chestek C A, Cunningham J P, Yu B M, Fan J M, Ryu S I and Shenoy K V 2012 A brain machine interface control algorithm designed from a feedback control perspective *2012 Annual Int. Conf. IEEE Engineering in Medicine and Biology Society* (IEEE) pp 1318–22
[11] Willett F R *et al* 2017 Feedback control policies employed by people using intracortical brain–computer interfaces *J. Neural Eng.* **14** 016001
[12] Zhang Q *et al* 2021 A prototype closed-loop brain–machine interface for the study and treatment of pain *Nat. Biomed. Eng.* **7** 533–45
[13] Shanechi M M, Orsborn A L and Carmena J M 2016 Robust brain-machine interface design using optimal feedback control modeling and adaptive point process filtering *PLOS Comput. Biol.* **12** 1004730
[14] Cunningham J P, Nuyujukian P, Gilja V, Chestek C A, Ryu S I and Shenoy K V 2011 A closed-loop human simulator for investigating the role of feedback control in brain-machine interfaces *J. Neurophys.* **105** 1932–49
[15] Wright J, Macefield V G, Van Schaik A and Tapson J C 2016 A review of control strategies in closed-loop neuroprosthetic systems *Front. Neurosci.* **10** 312
[16] Pandarinath C and Bensmaia S J 2022 The science and engineering behind sensitized brain-controlled bionic hands *Physiol. Rev.* **102** 551–604
[17] Pedrocchi A, Ferrante S, De Momi E and Ferrigno G 2006 Error mapping controller: a closed loop neuroprosthesis controlled by artificial neural networks *J. NeuroEng. Rehabil.* **3** 25
[18] Bolus M F, Willats A A, Rozell C J and Stanley G B 2021 State-space optimal feedback control of optogenetically driven neural activity *J. Neural Eng.* **18** 036006
[19] Bergs A C F *et al* 2023 All-optical closed-loop voltage clamp for precise control of muscles and neurons in live animals *Nat. Commun.* **14** 1939
[20] Newman J P, Fong M-f, Millard D C, Whitmire C J, Stanley G B and Potter S M 2015 Optogenetic feedback control of neural activity *eLife* **4** 07192
[21] Hewing L, Wabersich K P, Menner M and Zeilinger M N 2020 Learning-based model predictive control: toward safe learning in control *Annu. Rev. Control Robot. Auton. Syst.* **3** 269–96
[22] Holkar K and Waghmare L M 2010 An overview of model predictive control *Int. J. Control Autom.* **3** 47–63
[23] Lin L, Oncken J, Agarwal V, Permann C, Gribok A, McJunkin T, Eggers S and Boring R 2023 Development and assessment of a model predictive controller enabling anticipatory control strategies for a heat-pipe system *Prog. Nucl. Energy* **156** 104527
[24] S V Raković and W S Levine 2019 Handbook of model predictive control *Control Engineering* (Springer)
[25] Brunton S L and Kutz J N 2019 *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems and Control* (Cambridge University Press)
[26] Schwenzer M, Ay M, Bergs T and Abel D 2021 Review on model predictive control: an engineering perspective *Int. J. Adv. Manuf. Technol.* **117** 1327–49
[27] Rabinovich M I, Varona P, Selverston A I and Abarbanel H D I 2006 Dynamical principles in neuroscience *Rev. Mod. Phys.* **78** 1213–65
[28] Toth B A, Kostuk M, Meliza C D, Margoliash D and Abarbanel H D I 2011 Dynamical estimation of neuron and network properties I: variational methods *Biol. Cybern.* **105** 217–37
[29] Fröhlich F and Jezernik S 2005 Feedback control of Hodgkin-Huxley nerve cell dynamics *Control Eng. Pract.* **13** 1195–206
[30] Yue R, Tomastik R and Dutta A 2022 Non-linear model-based control of neural cell dynamics Research Square (https://doi.org/10.21203/rs.3.rs-580874/v2)
[31] Senthilvelmurugan N N and Subbian S 2023 Active fault tolerant deep brain stimulator for epilepsy using deep neural network *Biomed. Eng./Biomed. Tech.* **68** 373–92
[32] Kostuk M, Toth B A, Meliza C D, Margoliash D and Abarbanel H D I 2012 Dynamical estimation of neuron and network properties II: path integral Monte Carlo methods *Biol. Cybern.* **106** 155–67
[33] Ullah G and Schiff S J 2009 Tracking and control of neuronal Hodgkin-Huxley dynamics *Phys. Rev.* E **79** 040901
[34] Meliza C D, Kostuk M, Huang H, Nogaret A, Margoliash D and Abarbanel H D I 2014 Estimating parameters and predicting membrane voltages with conductance-based neuron models *Biol. Cybern.* **108** 495–516
[35] Bieker K, Peitz S, Brunton S L, Kutz J N, Dellnitz M 2019 Deep model predictive control with online learning for complex physical systems (arXiv:1905.10094)
[36] Kaiser E, Kutz J N and Brunton S L 2018 Sparse identification of nonlinear dynamics for model predictive control in the low-data limit *Proc. R. Soc.* A **474** 20180335
[37] Salzmann T, Kaufmann E, Arrizabalaga J, Pavone M, Scaramuzza D and Ryll M 2023 Real-time neural MPC: deep learning model predictive control for quadrotors and agile robotic platforms *IEEE Robot. Autom. Lett.* **8** 2397–404
[38] Zheng Y and Wu Z 2023 Physics-informed online machine learning and predictive control of nonlinear processes with parameter uncertainty *Indust. Eng. Chem. Res.* **62** 2804–18
[39] Plaster B and Kumar G 2019 Data-driven predictive modeling of neuronal dynamics using long short-term memory *Algorithms* **12** 203
[40] Sherman-Gold R 2012 *The Axon Guide, a Guide to Electrophysiology and Biophysics Laboratory Techniques* (San Jose: Molecular Devices, LLC)
[41] D Sterratt 2011 *Principles of Computational Modelling in Neuroscience* (Cambridge University Press)
[42] Knowlton C, Meliza C D, Margoliash D and Abarbanel H D I 2014 Dynamical estimation of neuron and network properties III: network analysis using neuron spike times *Biol. Cybern.* **108** 261–73
[43] Skinner F K 2006 Conductance-based models *Scholarpedia* **1** 1408
[44] Bourdeau M, Zhai X Q, Nefzaoui E, Guo X and Chatellier P 2019 Modeling and forecasting building energy consumption: a review of data-driven techniques *Sustain. Cities Soc.* **48** 101533
[45] Clark R, Fuller L, Platt J A and Abarbanel H D I 2022 Reduced-dimension, biophysical neuron models constructed from observed data *Neural Comput.* **34** 1545–87
[46] Lowe D and Broomhead D 1988 Multivariable functional interpolation and adaptive networks *Complex Syst.* **2** 321–55
[47] Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput.* **9** 1735–80

[48] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser Ł and Polosukhin I 2017 Attention is all you need *Advances in Neural Information Processing Systems* vol 30
[49] Park J and Sandberg I W 1991 Universal approximation using radial-basis-function networks *Neural Comput.* **3** 246–57
[50] Clark R, Abarbanel H D I, Fairbanks L, Sanchez R and Wacharanan P 2023 Data driven regional weather forecasting: Example using the shallow water equations (arXiv:2303.16363)
[51] Lorenz E N 1963 Deterministic nonperiodic flow *J. Atmos. sci.* **20** 130–41
[52] Sugihara G and May R M 1990 Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series *Nature* **344** 734–41
[53] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R and Dubourg V 2011 Scikit-learn: machine learning in python *J. Mach. Learn. Res.* **12** 2825–30
[54] Qin S J and Badgwell T A 2003 A survey of industrial model predictive control technology *Control Eng. Pract.* **11** 733–64
[55] Fiedler F, Karg B, Lüken L, Brandner D, Heinlein M, Brabender F and Lucia S 2023 do-mpc: towards FAIR nonlinear and robust model predictive control *Control Eng. Pract.* **140** 105676
[56] Andersson J A E, Gillis J, Horn G, Rawlings J B and Diehl M 2019 CasADi: a software framework for nonlinear optimization and optimal control *Math. Progr. Comput.* **11** 1–36
[57] Wächter A and Biegler L T 2006 On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming *Math. Program.* **106** 25–57
[58] Mulansky M and Kreuz T 2016 PySpike—a python library for analyzing spike train synchrony *SoftwareX* **5** 183–9
[59] Steinmetz N A, Aydin C, Lebedeva A, Okun M, Pachitariu M, Bauza M, Beau M, Bhagat J, Böhm C and Broux M 2021 Neuropixels 2.0: a miniaturized high-density probe for stable, long-term brain recordings *Science* **372** 4588
[60] Bottjer S W, Miesner E A and Arnold A P 1986 Changes in neuronal number, density and size account for increases in volume of song-control nuclei during song development in zebra finches *Neurosci. Lett.* **67** 263–8
[61] Milias-Argeitis A and Khammash M 2015 Adaptive model predictive control of an optogenetic system *2015 54th IEEE Conf. on Decision and Control (CDC)* (IEEE) pp 1265–70
[62] Fox Z R, Batt G and Ruess J 2023 Bayesian filtering for model predictive control of stochastic gene expression in single cells *Phys. Biol.* **20** 055003
[63] Bemporad A and Morari M 1999 Robust model predictive control: a survey *Robustness in Identification and Control* ed A Garulli and A Tesi (Springer) pp 207–26
[64] Smith A C, Scalon J D, Wirth S, Yanike M, Suzuki W A and Brown E N 2010 State-space algorithms for estimating spike rate functions *Comput. Intell. Neurosci.* **2010** 1–14
[65] Fehrman C and Meliza C D 2024 Model predictive control of the neural manifold (arXiv:2406.14801)
[66] Langdon C, Genkin M and Engel T A 2023 A unifying perspective on neural manifolds and circuits for cognition *Nat. Rev. Neurosci.* **24** 1–15
[67] Lambeth K, Singh M and Sharma N 2023 Robust control barrier functions for safety using a hybrid neuroprosthesis *2023 American Control Conf. (ACC)* (IEEE) pp 54–59
[68] Wolf D N and Schearer E M 2022 Trajectory optimization and model predictive control for functional electrical stimulation-controlled reaching *IEEE Robot. Autom. Lett.* 3093–8
[69] Singh M and Sharma N 2023 Data-driven model predictive control for drop foot correction *2023 American Control Conference (ACC)* (IEEE) pp 2615–20
[70] Bao X, Kirsch N, Dodson A and Sharma N 2019 Model predictive control of a feedback-linearized hybrid neuroprosthetic system with a barrier penalty *J. Comput. Nonlinear Dyn.* **14** 101009
[71] Chatterjee S, Romero O, Ashourvan A and Pequito S 2020 Fractional-order model predictive control as a framework for electrical neurostimulation in epilepsy *J. Neural Eng.* **17** 066017
[72] Brar H K, Exarchos I, Pan Y, Theodorou E and Mahmoudi B 2018 Seizure reduction using model predictive control *2018 40th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC)* (IEEE) pp 3152–5