Biological Cybernetics (2021) 115:219–235
https://doi.org/10.1007/s00422-021-00874-w

Biological Cybernetics

ORIGINAL ARTICLE

Leveraging deep learning to control neural oscillators

Timothy D. Matchen[1] · Jeff Moehlis[2]

Received: 17 December 2020 / Accepted: 10 April 2021 / Published online: 28 April 2021

Abstract
Modulation of the firing times of neural oscillators has long been an important control objective, with applications including Parkinson's disease, Tourette's syndrome, epilepsy, and learning. One common goal for such modulation is desynchronization, wherein two or more oscillators are stimulated to transition from firing in phase with each other to firing out of phase. The optimization of such stimuli has been well studied, but this typically relies on either a reduction of the dimensionality of the system or complete knowledge of the parameters and state of the system. This limits the applicability of results to real problems in neural control. Here, we present a trained artificial neural network capable of accurately estimating the effects of square-wave stimuli on neurons using minimal output information from the neuron. We then apply the results of this network to solve several related control problems in desynchronization, including desynchronizing pairs of neurons and achieving clustered subpopulations of neurons in the presence of coupling and noise.

Keywords Oscillators · Machine learning · Neurons · Clustering · Control · Dynamic programming

1 Introduction

Oscillators are ubiquitous in biological systems, from macroscopic processes like circadian rhythms (Saini et al. 2019) to specific internal processes such as cardiac pacemaker cells and motor control (Schnitzler and Gross 2005). Developing effective strategies to control these processes has emerged as an important objective. This is particularly relevant in neuroscience, where improper behavior of these oscillators may lead to pathological conditions. Existing strategies for mitigating this pathological activity, such as deep brain stimulation (DBS) and transcranial magnetic stimulation (TMS), provide a compelling motivation to further our understanding of the control of neural oscillators.

DBS in particular has seen numerous applications, from reducing some symptoms in Parkinson's disease (PD) (The Deep-Brain Stimulation for Parkinson's Disease Study Group 2001; Adamchic et al. 2014; Lysyansky et al. 2011, 2013) to treatment for Tourette Syndrome (Savica et al. 2012), essential tremor, and various other disorders. In DBS, neural dynamics are modulated by an electrode implanted in the brain tissue; pulsatile stimuli are sent via the electrode into the target brain region. Despite its clinical effectiveness, the specific mechanisms by which DBS influences brain dynamics are still an open question. Because of this, DBS calibration typically requires manual tuning of stimulus amplitude and frequency by a technician after electrode implantation, and this tuning may need to be occasionally adjusted as the patient's neurophysiology evolves over time.

Two of the primary challenges involved in identifying the precise therapeutic pathways involved in DBS are the high degree of complexity in neural systems and the relative lack of in vivo tracking data for these systems. Even relatively simple models of neurons are typically highly nonlinear and high-dimensional, making prediction of the effects of a stimulus difficult, thereby hampering the effectiveness of closed-loop approaches to control. Similarly, most measurements relating to diseases such as PD come from local field potential (LFP) readings taken between the first and second of the two successive surgeries required to implant the DBS

Communicated by Jonathan Rubin.

✉ Timothy D. Matchen
tmatchen@ucsb.edu

Jeff Moehlis
moehlis@ucsb.edu

[1] Department of Mechanical Engineering, University of California, Santa Barbara, CA 93106, USA

[2] Department of Mechanical Engineering, Program in Dynamical Neuroscience, University of California, Santa Barbara, CA 93106, USA

electrode, and we cannot in practice know the full state of the neuron, even if the underlying model is sound (Kühn et al. 2005; Holt and Netoff 2014).

References Uhlhaas and Singer (2006), Hua et al. (1998), Chen et al. (2007), Hammond et al. (2007), Levy et al. (2000) describe examples of pathological synchrony related to a number of different disorders, while references Tass (2003), Wingeier et al. (2006), Kühn et al. (2008), Bronte-Stewart et al. (2009), Wilson et al. (2011), Wilson and Moehlis (2015) further suggest both experimentally and in simulation that the primary mechanism involved in DBS is the desynchronization of populations of neurons, specifically in the subthalamic nucleus (STN). Because of this and other applications in neuroscience of both synchronization and desynchronization (Zhao et al. 2011; Titiz et al. 2017), the objective of influencing the configuration of neurons relative to each other—either toward an in-phase or anti-phase firing pattern—is specifically an important goal within the general field of neural control. Controlling populations or ensembles of oscillators is a topic area with an extensive literature, both in the context of biology and elsewhere (Roenneberg et al. 2005; Zalalutdinov et al. 2003; Diekman and Bose 2016). Clustering is additionally a problem of particular interest because of its widespread occurrence and applications. As mentioned previously, clustering configurations appear relevant to the study of pathological conditions, but it is also present in desirable brain activity, such as language development (Power et al. 2012) and visual identification (Liao et al. 2013). Spike timing within neuron ensembles is also fundamental, of course, to spiking-timing-dependent plasticity (Rodríguez-Pineda 2000), and modulation of this timing via clustering can encourage either potentiation or depression. This naturally extends into applications of Hebbian learning as well, where we may similarly influence outcomes by influencing entrainment of oscillators. Clustering possesses other biological applications as well; entrainment can have applications to photochemical oscillators (Taylor et al. 2008), and in Juul et al. (2018) the authors suggest entrainment may be used to influence embryonic patterning in mice. The applications of clustering control extend beyond the sphere of biology as well; the authors of Skardal and Arenas (2015) suggest applications to both macroscopic processes and emergent microgrid technologies.

While the presence and applications of oscillators and clustering are numerous, we choose to focus here on neural models in particular because they possess two important qualities that make them especially interesting in the study of oscillators: neural ensembles are typically not fully observable and are severely underactuated. Continuing with the example of Parkinson's disease, there are numerous models that attempt to characterize the dynamics of the basal ganglia, such as Holgado et al. (2010), Rubin and Terman (2004a), Hahn and McIntyre (2010), Otsuka et al. (2004). These models are developed over the course of numerous studies using both in vitro and in vivo measurements of relevant neural regions. However, when it comes to actually observing these systems, we may possess only a single biomarker, such as the local field potential. For this reason, system identification is a problem of much interest in biology and neuroscience, cf. Villaverde et al. (2019). In addition to this minimal observability, precise control of neural oscillators is made more challenging by the fact that a single stimulator may be influencing the dynamics of hundreds, if not thousands, of neurons, and it is simply impossible to control such a heavily underactuated system to an arbitrary state.

Even setting aside the challenges of low observability and underactuation, because of the high dimensionality of neural models, finding control strategies using traditional methods can prove difficult. One powerful technique for simplifying this process is phase model reduction, wherein the dynamics of an oscillator with a stable limit cycle are reduced to a one-dimensional system represented by a variable, θ, and parameterized by a natural frequency ω and a function specifying the responsiveness to external inputs known as the phase response curve (PRC) (Kuramoto 1984; Brown et al. 2004; Ermentrout 1996; Hansel et al. 1995; Winfree 2001). With this reduced system, it is straightforward to apply traditional optimization techniques, such as utilizing variational calculus to control an oscillator's period (Li et al. 2013; Moehlis et al. 2006) or applying control to manipulate a neural population's distribution (Monga and Moehlis 2020; Zlotnik and Li 2014; Zlotnik et al. 2016; Matchen and Moehlis 2018). In Wilson and Moehlis (2014a, b), the authors address the issue of desynchronization directly, optimizing the response of a population of neurons to inputs and determining that the best point in an oscillator's cycle to stimulate in order to achieve desynchronization is when the derivative $Z'(\theta)$ is maximized, cf. Holt et al. (2016).

Although phase model reduction can be a powerful technique, it is not without drawbacks. First and foremost, phase model reduction typically can only be considered valid close to the periodic orbit, especially in highly nonlinear systems (Kurebayashi et al. 2013). This can be mitigated by concurrently considering the system's isostables, an extension of the model intended to account in part for these nonlinearities (Wilson and Moehlis 2016; Monga and Moehlis 2019; Monga et al. 2019). As more dimensions are added to the system, however, so too does the need for accurate accounting of the state of the system if closed-loop control is desired; as previously noted, this is not necessarily feasible in the context of neural control.

We believe that machine learning, and specifically deep artificial neural networks (or ANNs), can be particularly effective in addressing the challenges present in the existing literature. Despite the moniker, neural networks are not yet widely used by the neural control community, though

interest is growing; recent results (Mitchell and Petzold 2018; Nagaraj et al. 2017; Narayanan et al. 2019; Yu et al. 2020) have begun leveraging reinforcement learning and Q-learning to generate control strategies. Because neuron response dynamics are highly nonlinear, we believe that deep neural networks may be able to better accommodate the range of possible responses than other regression methods. Additionally, the short timescales involved in neuron firing (in the case of the STN model presented here, about 30 ms) can allow a neural network training on neural data to quickly gather a suitably large training set for accurate prediction and cross-validation onto a developmental set. Lastly, the "feed-forward" nature of ANNs makes it ideal for the short-timescale implementation necessary for eventual in vivo application as it can be computed explicitly from a given input, unlike other nonlinear regression strategies.

An ANN consists of a system that maps an input vector to an output vector via one or more layers; the system is a "deep" network if at least one of these layers is "hidden," meaning it does not map directly to the output. Each layer is comprised of a number of neurons, each consisting of a matrix that linearly transforms the output from the previous layer followed by a nonlinear activation function. Training a neural network consists of running repeated iterations of two steps: a forward propagation step followed by a backward propagation step. In the forward propagation step, the current parameters of the neural network are used to make a prediction of the output based on the input, which is then compared to the actual output. This provides a loss function which is then fed back into the neural network to compute appropriate derivatives for updating the weights on the matrices in each layer of the neural network. A schematic of this process is provided in Fig. 1. An appealing aspect of neural networks is that it is well established that, for some finite number of artificial neurons, any smooth function can be accurately approximated by a neural network consisting of only a single hidden layer (Cybenko 1989). In practice, however, it is typically more efficient to sacrifice breadth—a large number of neurons in a layer—for depth, a larger number of hidden layers. Effectively, the deep neural network architecture allows nonlinearities to compound at each layer, providing for the approximation of highly nonlinear functions with less computational cost than a simpler, single-hidden layer network.

Challenges arise in the training of such networks, ranging from computational issues such as failure of convergence to the global minimum and vanishing or exploding gradients to broader model issues, such as over or underfitting. In general, local minima are not observed in deep networks because of the high dimensionality of the system; it has been shown that local minima are almost certainly global minima and instead the most problematic points are saddle points (Kawaguchi 2016). There is no universal strategy for eliminating the other issues that may arise, so neural network design is an

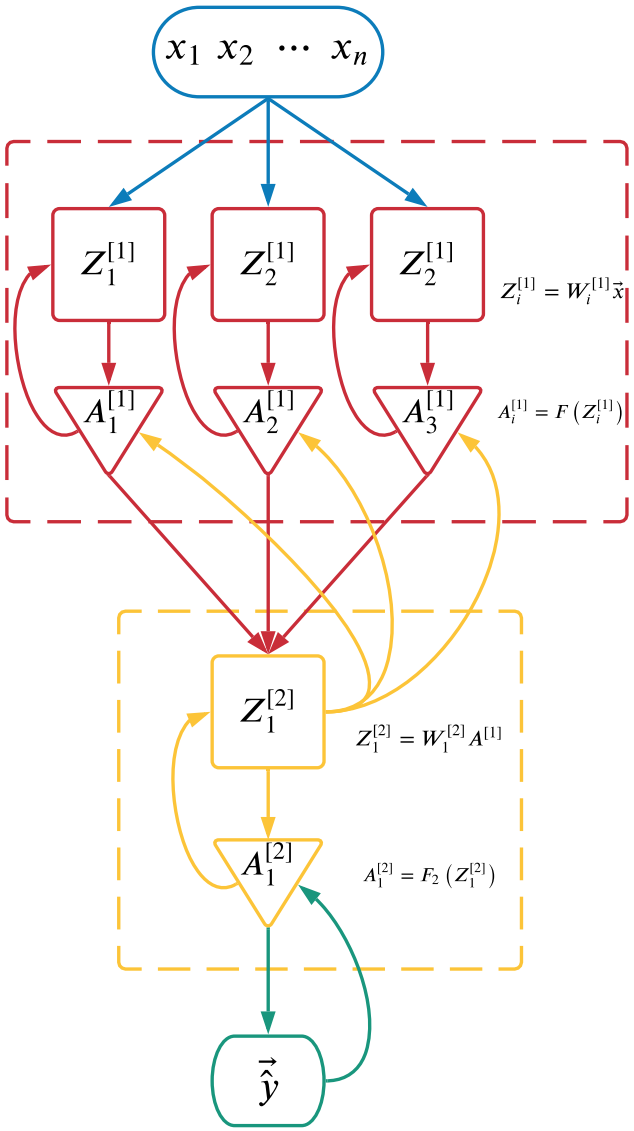

Fig. 1 Schematic of a neural network with one hidden layer. This neural network contains one hidden layer (in red) with three neurons and an output layer (in yellow) with one neuron. The input **x** (blue) is applied to each neuron in the hidden layer. The outputs of the hidden layer's activation function are passed to the output layer, which yields the system's estimate $\hat{y}$ (green). The result is then fed backward for back propagation (color figure online)

inherently iterative process, where various hyperparameters (number of layers, number of neurons in each layer, etc.) are tuned across trials to optimize results; once these hyperparameters are found, however, pre-existing architectures may be leveraged to solve similar problems (known as transfer learning), a common technique in other areas of machine learning such as image classification (Zhu et al. 2011; Quattoni et al. 2008).

In this paper, we seek to develop a marriage of traditional control techniques and approaches to problem solving with the functionality that machine learning offers. We specif-

ically look to develop predictions and control based on a scarcity of observable data (in this paper, we consider only firing times) and use that as the basis for control in both fully actuated, single-neuron control applications and underactuated, population-level control schemes. We propose a deep neural network for prediction of the effects of input stimuli on a neural oscillator. This network is relatively simple compared with the architectures commonly employed for problems such as natural language processing and image processing. While our network was trained on a model for STN neurons, similar or identical architectures (trained on appropriate data sets) should be able to obtain similar regressions for other models. We have intentionally separated the regression problem from the control applications to demonstrate how a single network can be applied to solve not just singular problems in engineering, but instead whole suites of interrelated problems, allowing users to adapt a pre-existing network to address novel challenges and questions.

Our paper will proceed as follows. We first describe the design and training of our neural network, with a brief analysis of the associated error and comparison to alternative regression strategies. Then, we describe how we applied the results of our regression model framework to three problems of interest: an optimal efficiency control wherein two neurons are desynchronized as efficiently as possible with a single square-wave input; a dynamic programming strategy wherein multiple square waves are used each period to maximally desynchronize a pair of neurons; and a larger-population clustering application where square-wave inputs are used to drive a group of neurons into a k-cluster state. We will show that the network was capable of not only successful applications to these various types of control applications but also that it is robust in the presence of effects not accounted for in the initial training protocol, specifically system noise and electrotonic coupling.

2 Artificial neural network design and training

A primary goal of this research was to utilize as little information as possible when training the neural network and designing the control. To this end, we assumed that the only measurement we could recover from the neuron was the timing of its spikes. The control signal used, however, was assumed to be known, and is characterized by three variables: signal amplitude I_0, signal width t_{width}, and signal delay time t_{delay}. The delay time is the length of time after the previous spike that the input stimulus is applied. The system's output was the expected value of the neuron's next firing time, t_{final}. The input $\mathbf{x}$ and output $\hat{y}$ of our neural network can therefore be written as:

$$\mathbf{x} = \left[I_0,\ t_{\text{width}},\ t_{\text{delay}} \right]^T, \tag{1}$$

$$\hat{y} = E\left[t_{\text{final}} | \mathbf{x} \right]. \tag{2}$$

The neural network was trained via random trials of $\mathbf{x}$ selected uniformly from the ranges $0 \leq I_0 \leq 25$ mA, $1 \leq t_{\text{width}} \leq 9$ ms, and $0 \leq t_{\text{delay}} \leq 33$ ms. For each $\mathbf{x}^i$, the neural model was simulated (here, a conductance-based model of the STN adapted from (Rubin and Terman 2004a); the STN model used is provided in Appendix 1) and the next spike time $y^i = t_{\text{final}}$ was recorded. Here, the superscript notation denotes the ith training example from the data.

Prior to being used to train the neural network, $\mathbf{x}$ was normalized to account for differences in the ranges of I_0, t_{width}, and t_{delay}. The transformed $\mathbf{x}$ was computed as:

$$\hat{x}_j = \frac{x_j - \mu_{x_j}}{\sigma_{x_j}}, \tag{3}$$

where μ_{x_j} and σ_{x_j} are the arithmetic mean and standard deviation, respectively, of the jth element of $\mathbf{x}$.

The model was trained using 10,000 random trials, with 70% of the trials used as the training set and the remaining 30% held out for cross-validation. Parameters for the neural network were initialized using the Xavier initialization protocol (Glorot and Bengio 2010) and updated via batch gradient descent. Hyperparameters for the number of hidden layers, the number of neurons per layer, and the learning rate were tuned using a Bayesian search optimization and optimizing the harmonic mean HM of the quadratic loss function $\mathcal{L}$ (commonly referred to as the mean-squared error, or MSE) applied to both the training set and the cross-validation set. More concretely:

$$\mathcal{L}\left(\hat{y}, y\right) = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^i - y^i \right)^2, \tag{4}$$

$$HM = \frac{2}{\frac{1}{\mathcal{L}_{train}} + \frac{1}{\mathcal{L}_{C-V}}}. \tag{5}$$

Here, $m = 7000$ for the training set and $m = 3000$ for the cross-validation set. This process yielded a neural network with 5 hidden layers and neuron numbers whose values are presented in Table 1. The hidden layers utilized $tanh$ activation functions, while the output layer contained a linear activation function to account for the complete range of possible period measurements. A learning rate of $\eta = 0.0015$ was used to achieve convergence, and the model was trained for 50,000 epochs.

After confirming the model's accuracy on the training set, we cross-validated against the held out data to check for overfitting. As can be seen in Fig. 2, the model maintained a high degree of accuracy in this cross-validation.

Table 1 Neural network hyperparameters

Layer #	1	2	3	4	5
# of neurons	80	100	60	30	80

We note that the use of a neural network to perform this regression has two primary advantages over other common regression strategies we considered. First and foremost, the neural network allows us to achieve a far greater degree of complexity in the nonlinearities than other general regression methods, such as polynomial regression or Fourier coefficient regression. Because neuron dynamics exist in a high-dimensional, highly complex space, this generality that the deep neural network provides is especially valuable in the context of modeling stimulus responses in neurons. A comparison of the accuracy of our neural network to two other regression strategies can be found in Table 2. Second, because future estimations are provided by an explicit, feed-forward model based on provided input values, the neural network produces an estimate that is efficient to compute once the initial training is complete. Other strategies, such as interpolating from the scattered data, are necessarily slower; this difference in processing speed can be significant in the context of on-line neural control, where computational efficiency is of particular importance.

2.1 Evaluation of multiple neurons simultaneously

To extend the ANN to consider a larger population of neurons, the firing times of all neurons in the population were defined relative to a reference time, selected as the firing time of the latest-firing neuron in the population. As such, the first firing time for the last-firing neuron was considered $t = 0$, and all other neurons were considered to have first fired at some offset negative ($t < 0$) time. The appropriate result estimation, then, was evaluated as t_{delay} with this offset subtracted. For example, if the reference neuron fired at 0 ms and a second neuron fired at −4 ms, then an input that occurs at $t_{\text{delay}} = 5$ for the reference neuron occurs at $t_{\text{delay}} = 9$ for the second neuron. The ANN therefore can be efficiently used for larger neural populations.

As an extension of this process, an "event horizon" mapping was also generated from the ANN's regression analysis. To extend the neural network to the case of multiple inputs within a single cycle, it was necessary to estimate not just when the neuron was stimulated but also when the stimulus ends relative to the limit cycle. This was accomplished by making the simplifying assumption that the neuron rapidly returns to the limit cycle following the end of a given stimulus. An event horizon time t_{hor} disallows subsequent input stimuli with start times $t_{\text{delay}} < t_{\text{hor}}$. The event horizon time

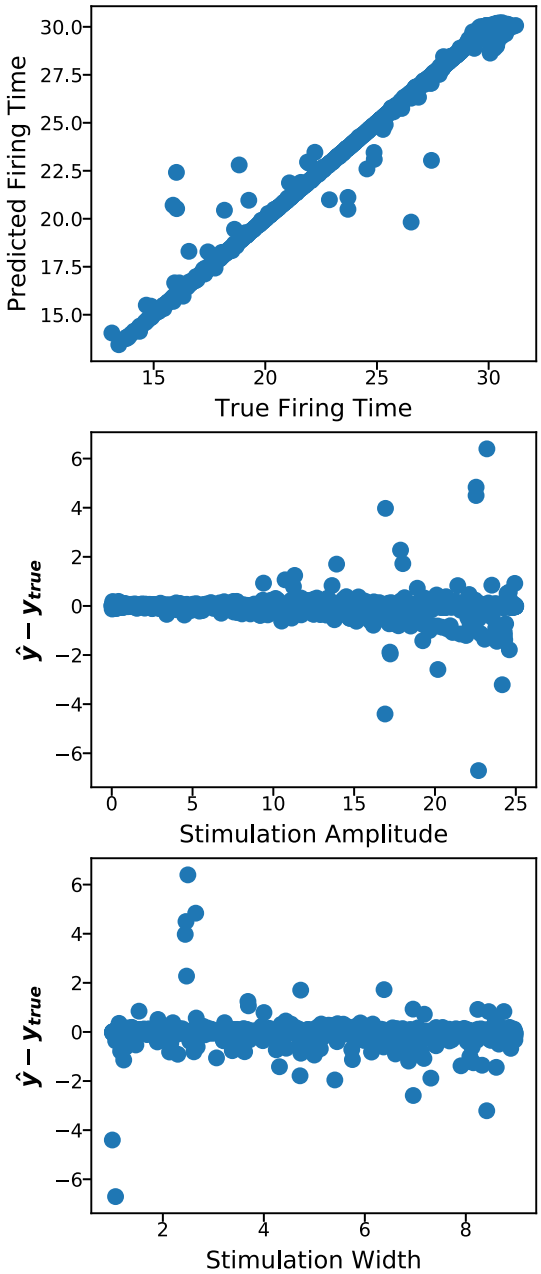

Fig. 2 Analysis of cross-validation data prediction. In the *top* panel, predictions are plotted against true values; perfect accuracy would return a straight line along $y = x$. As can be seen, almost all predictions have low error, with some outliers. The *center* and *bottom* panels plot the errors against amplitude and stimulation width, respectively, revealing that the most challenging stimuli to model tend to be those with high amplitude and short duration. These represent stimulations near the firing threshold for the neuron—sufficient stimulation will generate an all-or-nothing action potential, while insufficient stimulation will simply return to the limit cycle, meaning a small change in stimulation parameters can have a strong, divergent effect on the resulting firing time (color figure online)

was computed as:

$$t_{\text{hor}} = T_0 - \left(\hat{y} - t_{\text{delay}} - t_{\text{width}}\right), \tag{6}$$

Table 2 Comparison of performance of neural network to other regression methods

	Training error	C–V error
Polynomial Fit	0.7787	0.9297
MATLAB scattered interpolant	N/A	0.4674
Neural network	0.02688	0.0490

The error was calculated according to the loss function (4). The polynomial interpolation was estimated using a least-squares analysis of polynomial terms of the type $a_{j,k,l} I_0^j t_{\text{width}}^k t_{\text{delay}}^l$

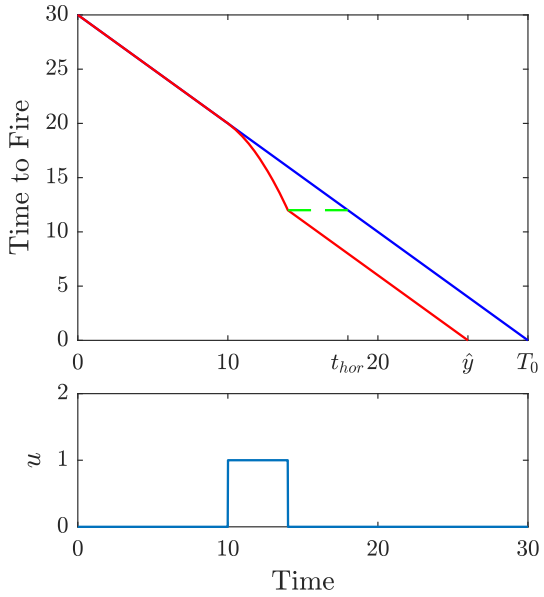

Fig. 3 Illustration of event horizon mapping. Time increases from 0 going left to right, with the remaining time until a firing event is recorded indicated by the y-axis. In the *top* panel, the unperturbed system is shown by the *blue* line, while the system subject to the input stimulus u (shown in the *bottom* panel) is represented by the *red* curve. The event horizon time is computed by translating the remaining time until the neuron fires after the completion of the input stimulus to the corresponding remaining time relative to the original firing time T_0, as indicated by the *green dashed* line. The predicted firing time $\hat{y}$, meanwhile, is the time at which the time to fire reaches 0 (color figure online)

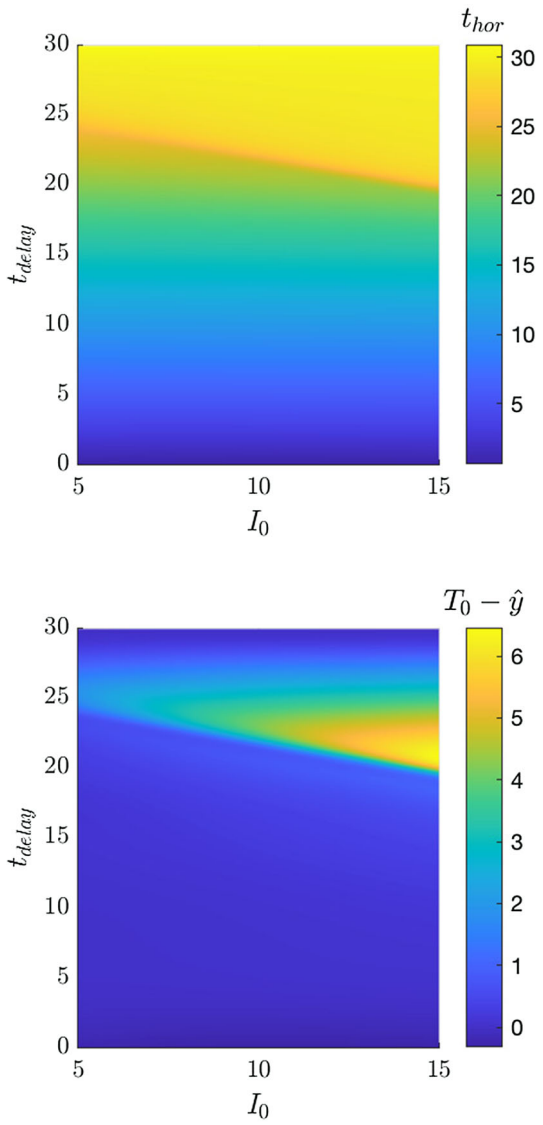

Fig. 4 Event horizon mapping (*top*) and period gain mapping (*bottom*) for $t_{\text{width}} = 1$. The event horizon mapping is calculated via (6); the period gain mapping shows the amplitude of the reduction in period generated by stimulating at the given t_{delay} and I_0. The natural period of the oscillator T_0 is roughly 30 ms (color figure online)

where $T_0 \approx 30$ ms is the natural period of the neuron. For example, if a stimulus at $t_{\text{delay}} = 15$ with $t_{\text{width}} = 3$ causes the period to change from 24 to 21, although the stimulus ends at $t = 18$, its event horizon time would be 21 because it ends 4 ms prior to the neuron's firing time. Any subsequent stimuli that may be applied would need to have a value of $t_{\text{delay}} > 21$ since all delay values were considered relative to the original firing time. This is illustrated in Fig. 3. Additionally, we note T_0 can itself be derived from the neural network by setting $t_{\text{delay}} \gg T_0$ for a candidate stimulus (if $t_{\text{delay}} > T_0$, the stimulus occurs after the next spike, and the network should predict a firing time that precedes the value of t_{delay}). A realization of this event horizon mapping for $t_{\text{width}} = 1$ is shown in Fig. 4.

As can be seen in Fig. 4, the relationship between stimulation parameters and firing time is nonlinear and timing-dependent. As such, the specific effect of an applied current varies depending on when and how strongly it is applied, but the underlying dynamics of the STN are not overridden by the applied stimulus in general. Rather, advancing the firing time corresponds to an earlier rise in I_{Na} that in turn causes the neuron to reach the threshold for an action potential firing, while stimuli that do not successfully advance the firing time show either an increase in the amplitude of I_K or negligible increase in I_{Na}. This can be seen for three different input stimuli in Fig. 5.

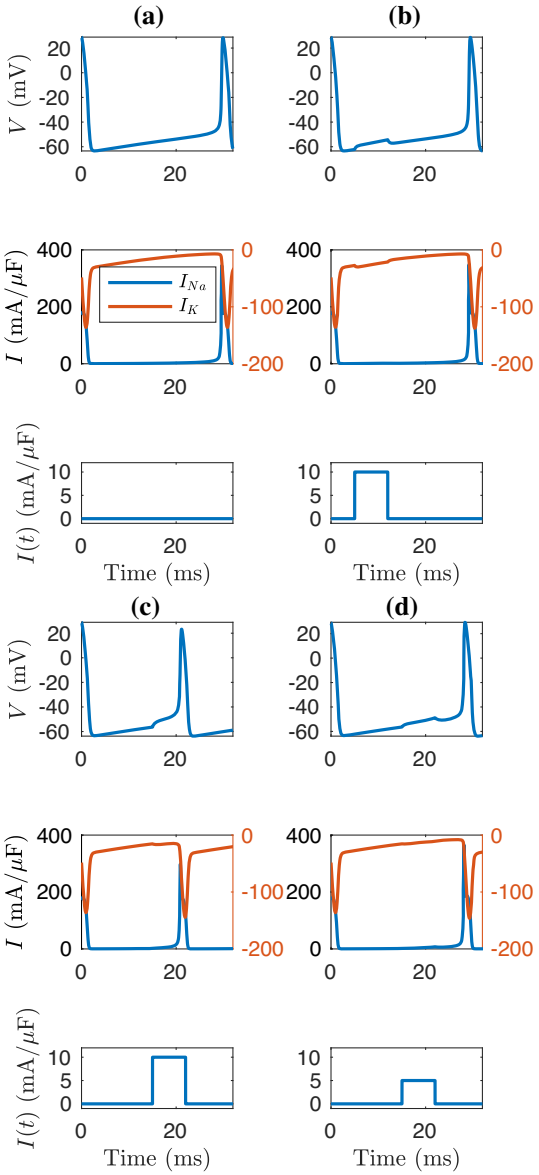

Fig. 5 Response of I_{Na}, I_K, and V to three different input stimuli. Four different examples are shown. The *top* panel of each triplet shows the voltage trace, while the *middle* shows the ion currents I_{Na} and I_K. The *bottom* panel shows the applied stimulus. As can be seen, the relationship between input and firing time is nonlinear and works by increasing the value and growth of I_{Na} (color figure online)

3 Maximally efficient desynchronization

We now turn our attention to application of our regression model to the desynchronization of two neural oscillators. We consider the voltage traces of two identical uncoupled STN neurons, V_1 and V_2. We define the spike time as the time at which V_i is locally maximal and above a threshold voltage (here, 0). The initial spike times for the neurons are separated by some small increment Δt; because the neurons are identical, uncoupled, and at steady state, this spike time difference will persist from cycle to cycle unless the neurons receive an input. We define, without loss of generality, V_2 to be the "leading" neuron, meaning V_2 fires before V_1 and the firing times can be related by:

$$t_{\text{spike}}^{V_1} = t_{\text{spike}}^{V_2} + \Delta t, \ \Delta t > 0. \tag{7}$$

If during the subsequent cycle the inter-firing times of V_1 and V_2 differ by Δt_{final}, the total separation between the two neurons is given by:

$$\Delta t_{\text{total}} = \Delta t + \Delta t_{\text{final}}. \tag{8}$$

Our goal is to increase the value of Δt_{final} as efficiently as possible when the two neurons are subject to a common stimulus. Following our previous work in Matchen and Moehlis (2018), we therefore design our control to minimize our value function $Q(x, y)$:

$$Q = \frac{-\left(t_{\text{final}}^{V_1} - t_{\text{final}}^{V_2}\right)}{\alpha \int_0^t I(t)^2 \, dt + \epsilon}, \tag{9}$$

where α is a weighting term, $\epsilon > 0$ is an offset to prevent division by zero if $I = 0$, and $t_{\text{final}}^{V_1} - t_{\text{final}}^{V_2} = -\Delta t_{\text{final}}$. The values of $t_{\text{final}}^{V_1}$ and $t_{\text{final}}^{V_2}$ are computed assuming the neuron previously fired at $t = 0$. We additionally note that, since we are only considering square waves, the integral can be replaced:

$$\int_0^t I(t)^2 \, dt = I_0^2 t_{\text{width}}. \tag{10}$$

Because the only difference between the two neurons is the initial offset Δt, the final time estimation is just the difference in the expected values:

$$t_{\text{final}}^{V_1} - t_{\text{final}}^{V_2} = E[y|\mathbf{x_1}] - E\left[y|\mathbf{x_2} = \mathbf{x_1} + [0, \ 0, \ \Delta t]^T\right]. \tag{11}$$

We minimized Q via gradient descent for multiple initial separations Δt subject to two restrictions:

1. Q was minimized while holding amplitude constant, then amplitude was varied; and
2. t_{width} was not allowed to decrease below $t_{\text{width}} = 0.5$.

The first condition was applied to recognize that control may be subject to specific constraints, such as a particular amplitude or performance characteristics (desynchronization time or total energy). The second condition was to ensure the model did not minimize the cost by reducing t_{width} to a physically unrealizable negative number.

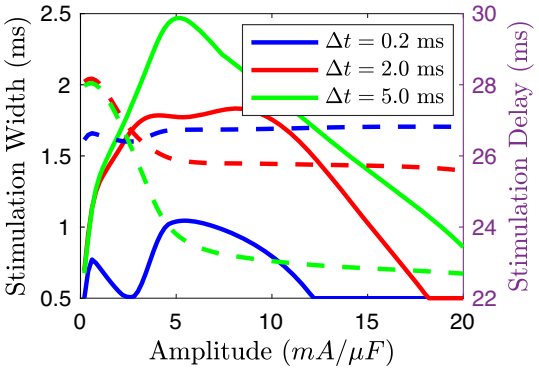

Fig. 6 Optimal t_{width} (*solid line*, scaling on left y-axis) and t_{delay} (*dashed line*, scaling on right y-axis) for desynchronizing a pair of neural oscillators as a function of stimulus amplitude I_0 for three different initial separations. As can be seen, all three separations show the same general trend in terms of the optimal signal, with shorter signals typically preferred at higher amplitude (color figure online)

The most energy-efficient signal for desynchronizing a pair of neural oscillators as a function of amplitude I_0 is shown in Fig. 6. To validate the results of the control, the error in the output when applied to the original ODE was calculated as:

$$\text{err}_{\text{rel}} = \frac{\sqrt{\left(\Delta t_{\text{final,est}} - \Delta t_{\text{final,sim}}\right)^2}}{\left|\Delta t_{\text{final,sim}}\right|}, \tag{12}$$

where $\Delta t_{\text{final,sim}}$ and $t_{\text{final,est}}$ are the simulation and machine learning-derived values of $t_{\text{final}}^{V_1} - t_{\text{final}}^{V_2}$, respectively. Both this error and the absolute error ($\Delta t_{final,est} - \Delta t_{final,sim}$) are shown in Fig. 7. We see that the model performs well outside of small-amplitude signals regardless of the separation: because the response to small-amplitude signals is relatively small, the relative effectiveness is limited by the underlying accuracy of the model. We note, however, that the model performs exceedingly well outside of these extreme cases, with relative errors under 10% and absolute errors on the order of less than 1% of the STN neuron's period.

4 Dynamic programming desynchronization

The use of multiple inputs to optimally desynchronize a pair of neurons over a single cycle was carried out by leveraging dynamic programming. In the interest of computational efficiency, we opted for a two-level approach to dynamic programming: on a cycle-to-cycle level, the control was greedy, attempting to maximize the desynchronization, while individual inputs within the cycle were found recursively by working backward from the final state, here $\Delta t_{\text{total}} = \frac{T_0}{2}$.

The process within a given cycle proceeded as follows: potential choices of I_0 and t_{delay} were gridded while t_{width} was

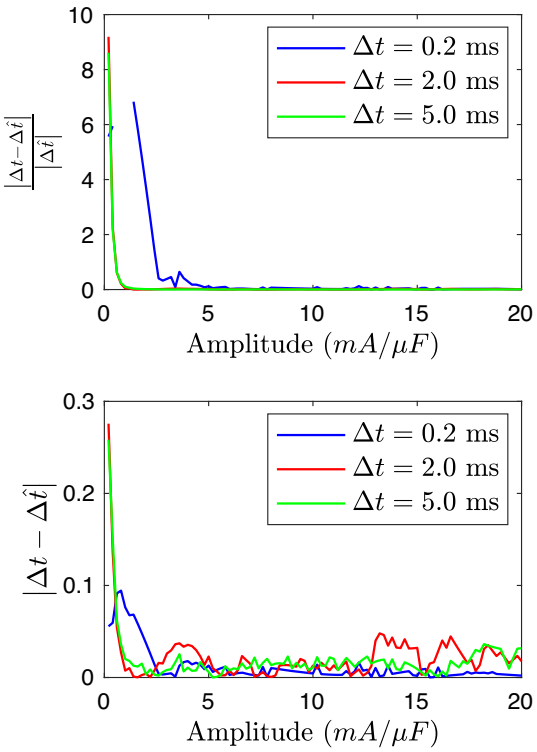

Fig. 7 Relative (*top*) and absolute (*bottom*) errors by amplitude for three different initial separations. Although absolute error is low across all samples and lowest for the smallest separation ($\Delta t = 0.2$ ms), the relative error for the smallest separation is highest because the true measured improvement is near 0; for a portion of the signal range, in fact, the change is sufficiently small that there is no measured change, causing the denominator of the relative error to be identically 0 (this can be seen in the plot up to roughly 2 mA). However, for most amplitudes and stimulation lengths, both the relative and absolute errors are insignificant and near 0 as stimulation amplitude increases (color figure online)

held constant. Additionally, a minimum time of 3 ms between input stimuli was imposed. For a selected (I_0, t_{delay}) pair, a binary search function was used to find the spike time difference that, when the pair of neurons is subjected to the given stimulus, would result in the desired final spike time difference. This process was then iteratively repeated for every input stimulus whose event horizon was less than $t_{\text{delay}} - 3$ (with the difference accounting for the enforced minimum time between firing).

The goodness of a given sequence of control stimuli was determined via a value function:

$$P = \Delta t_{\text{final}} - \Delta t_{\text{initial}} - \beta I_0^2, \tag{13}$$

where β represents a weighting of the input stimulus power relative to the time improvement and $\Delta t_{initial}$ represents the separation at the start of the given cycle; in general, a higher value of β should produce lower-energy desynchronizations requiring more cycles to complete, while a lower value of β

should produce higher-energy desynchronizations requiring fewer cycles to complete.

The input stimulus sequence that maximized P determined the endpoint for the previous cycle's optimization problem. For each cycle, the same process was applied until the start point for a given cycle terminated at a spike time difference of less than 1 ms. The process for a given cycle is presented in Fig. 8.

The success of our dynamic programming approach shows that the perturbations to the limit cycle introduced by the applied control were sufficiently weak to allow the neuron to return to its limit cycle in a short timeframe, thereby extending the functionality of the regression model to input sequences. The value of an input sequence was calculated using the neural network according to (13). The end separation Δt_{final} was taken to be half the neuron's natural period, or roughly 15 ms. Two different values of β were used to verify the existence of distinct control schemes depending on the relative weighting of energy cost. The faster-responding system used a value of $\beta = 0.01$, while the slower-responding system used a value of $\beta = 0.1$.

After the input sequence was calculated from dynamic programming applied to the ANN, it was applied to the full ODE to verify the result. Rather than using the input sequence as an open-loop stimulus, the firing time of the reference neuron was used as the basis for setting the next cycle's input sequence (that is, t_{delay} was computed relative to the recorded spike time, not the neural network's estimate of the spike time. Because of the observed accuracy limitations at small input magnitudes and small separations seen in Fig. 7, inputs for the dynamic programming analysis were constrained to the range of $5 \leq I_0 \leq 10$. Results can be seen in Fig. 9. These results are consistent with expectations: as in the case of the efficient stimulus calculation, the dynamic programming yields a highly accurate final result, and the smaller β value returns a more-responsive input sequence offset by a higher energy cost. We note that, again consistent with prior results, performance does degrade if I_0 is allowed to take on smaller values or if the starting condition requires the neurons to be significantly closer (on the order of $\Delta t = 0.1$ instead of $\Delta t \approx 1$).

Larger inputs can also present challenges for multiple-input control applications. This is a byproduct of the underlying neural network, which only predicts the *next* firing time; larger stimuli will yield a correct prediction of the firing time, but the firing of the *subsequent* spike will not occur exactly a full period later even if no additional stimulus is applied. Although beyond the scope of this paper, future work may be warranted on predicting these aftereffects of stimulation in addition to the immediate effect.

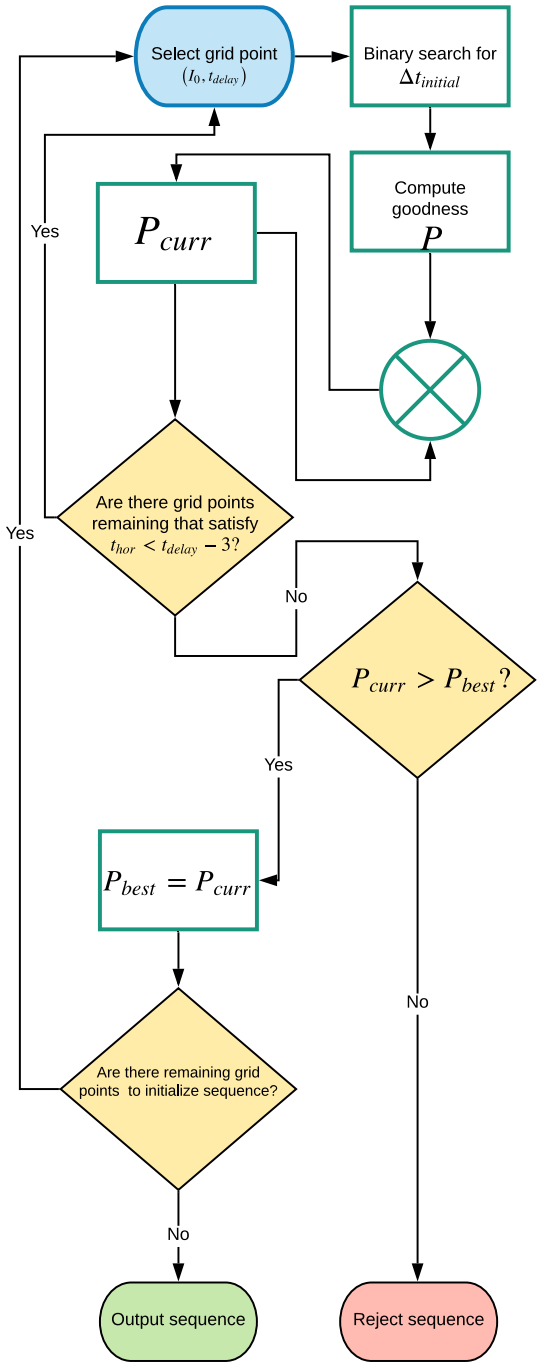

Fig. 8 Process flowchart for dynamic programming application. This process was carried out independently from cycle to cycle until the terminating condition was reached (color figure online)

5 Clustering of neurons

5.1 Control policy

Clustering, in which oscillators form discrete, finite groups with similar characteristic behaviors, was accomplished by defining a cost function related to the input stimulus and the

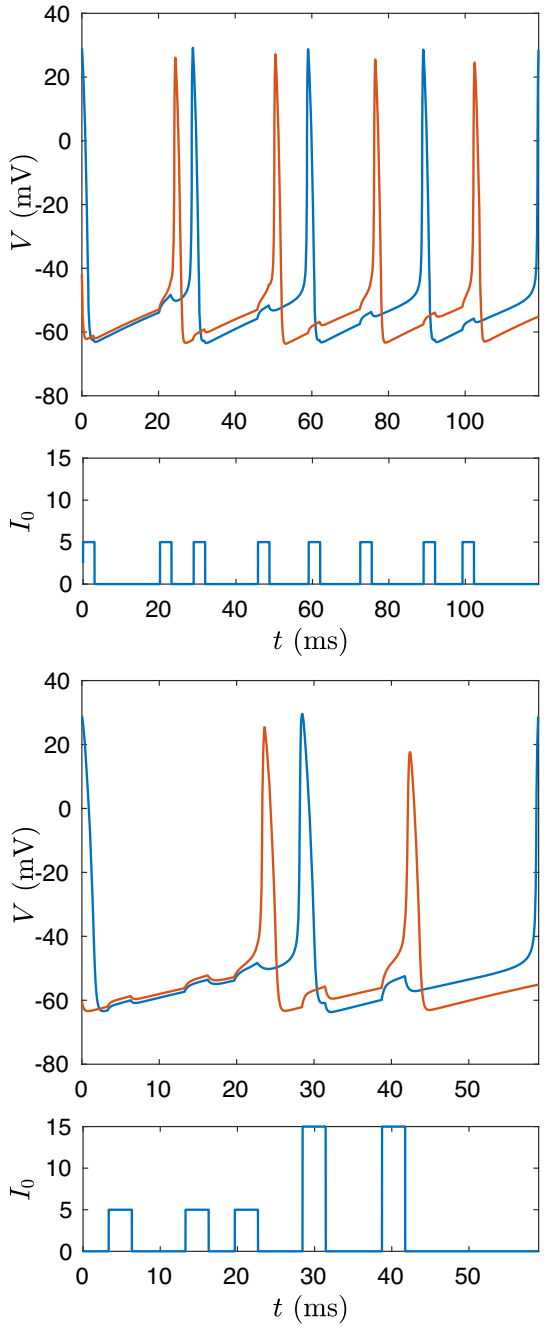

Fig. 9 Computed stimuli and output for dynamic programming approach to desynchronization, $\beta = 0.1$ (*top*) and $\beta = 0.0001$ (*bottom*). For both trials, $t_{\text{width}} = 3$. Note that both accurately yield a final target separation of ≈ 15 ms, but $\beta = 0.0001$ results in an input sequence that is higher in energy and faster in response (color figure online)

change in a state function. This state function was adapted from the standard order parameter $R_k\,(\grave{})$ where R_k represents the k^{th}-order parameter and is defined as Daido (1996):

$$|R_k\,(\grave{})| = \frac{1}{N}\left|\sum_{l=1}^{N} e^{ik\theta_l}\right|. \tag{14}$$

Clustering has emerged as a potential solution to problems such as Parkinson's disease. In Adamchic et al. (2014), a quadripolar stimulator providing identical but phase-separated stimuli at each of its four poles showed clinical improvement in patients, and Tass (2003) suggests clustering as the mechanism for coordinated reset. Here, we attempt to recreate the effects of coordinated reset—grouping of subpopulations of neurons—using an identical common input to all neurons instead of four distinct signals.

Because we are solely interested in the magnitude of R_k, for simplicity of notation in this chapter we will henceforth understand R_k to be the real-valued magnitude $|R_k\,(\grave{})|$ as defined in (14). Here, θ is interpolated from when the neuron previously fired relative to the natural period of the neuron. Assuming the l^{th} neuron fired at $t = 0$, then, θ_l is represented as a linear transformation:

$$\theta_l\,(t) = \frac{2\pi t}{T}. \tag{15}$$

Order parameters are frequently used for analysis in clustering problems, cf. (Tass 2003). In the order parameter context, $R_1\,(\grave{}) = 1$ implies the oscillators are perfectly clustered in a single-cluster configuration, while $R_1\,(\grave{}) = 0$ suggests desynchrony or a larger number of clusters instead. It should be noted, however, that one drawback of the order parameter is that, for $k \geq 2$, the distributions that generate $R_k = 1$ are non-unique. For example, the one-cluster case of $R_1 = 1$ also yields $R_2 = 1$, $R_3 = 1$, etc. In contrast, an optimally distributed two-cluster system, with an equal number of neurons in each cluster and the clusters π radians out of phase with each other, would still have $R_2 = 1$, but R_1 would instead equal 0. R_4, however, would also equal 1.

This ambiguity provides a challenge for utilizing order parameters as a basis for control rather than as simply a diagnostic tool for the effectiveness of a control. Controlling to a two-cluster state, for example, requires not only mandating $R_2 = 1$ but also that $R_1 = 0$; more generally, a k-cluster state is better defined according to:

$$R_l = \begin{cases} 0, & l < k \\ 1, & l = k \end{cases}. \tag{16}$$

In general, we seek a cost function $C_k\,(\grave{})$ that satisfies:

$$C_k\,(\grave{}) = \sum_{l=1}^{k} \alpha_l R_l\,(\grave{}), \tag{17}$$

such that that C_k is minimized when (16) is satisfied. A second constraint results from the existence of undesirable local minima resulting from the definition of the order parameters. Suppose, for example, $k = 2$ and $R_1 \approx 1$. In this case, $R_2 \approx 1$ as well. If $\alpha_1 = -\alpha_2 = 1$, then $C_k \approx 0$ and the

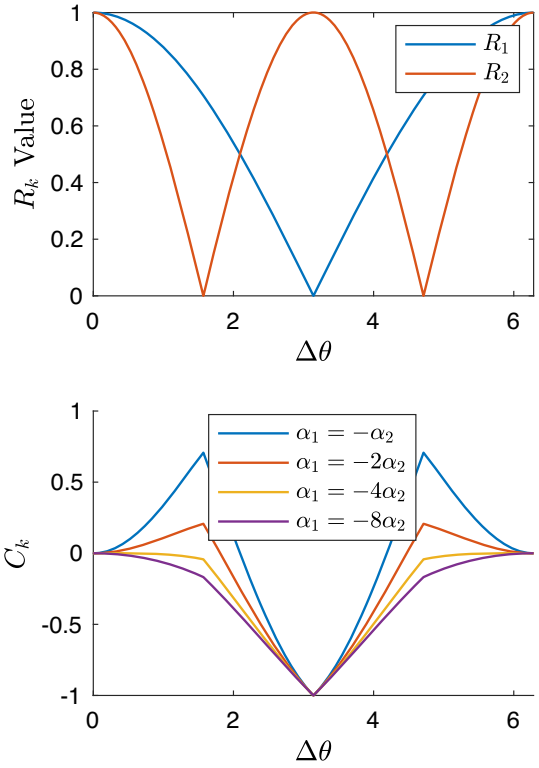

Fig. 10 Effect of α_l selection on existence of local minima in C_k. To illustrate the potential existence of local minima related to coefficient selection, here we examine the $k = 2$, $N = 2$ case. As can be seen in the *top* panel, for $\Delta\theta < \frac{\pi}{2}$, R_2 decreases more rapidly than R_1, creating a local minimum at $\Delta\theta = 0$ when $\alpha_1 = -\alpha_2$. The *bottom* panel shows C_k (normalized such that $C_k\,(\Delta\theta = 0) = 0$ and $\min C_k = -1$) for different choices of α_1 relative to α_2. At $\alpha_1 \approx -4\alpha_2$, the fixed point at $\Delta\theta = 0$ loses its stability (color figure online)

global minimum is $C_k = -1$. However, $R_1 = R_2 = 1$ is a local minimum of the cost function, as desynchronizing from the single-cluster state causes the value of R_2 to change more rapidly than that as R_1 (this can be shown by differentiation; for an example, see Fig. 10). Therefore, some care must be used in selecting α_l to derive a valid cost function.

We derive here an approximate sufficient condition for the $k = 2$ case, noting that a similar analysis can be iteratively carried out for $k > 2$ as well. At the undesirable, one-cluster fixed point, the derivative with respect to the jth neuron's phase θ_j may be written as:

$$\frac{\partial C_k}{\partial \theta_j} = \sum_{l=1}^{N} \left[2\frac{\alpha_1}{R_1} \left(\sin\theta_l \cos\theta_j - \cos\theta_l \sin\theta_j \right) + 4\frac{\alpha_2}{R_2} \left(\sin 2\theta_l \cos 2\theta_j - \cos 2\theta_l \sin 2\theta_j \right) \right]. \tag{18}$$

Additionally, at this undesired fixed point, $\frac{\partial C_k}{\partial \theta_j} = 0$. A necessary and sufficient condition for this point to be unstable is that, for $|\epsilon| \ll 1$, $\left.\frac{\partial C_k}{\partial \theta_j}\right|_{\theta_j+\epsilon} < 0$. We make use of the Taylor approximations:

$$\begin{aligned} \sin(x+\epsilon) &= \sin x + \epsilon\cos x + \mathcal{O}\left(\epsilon^2\right); \\ \sin(2x+2\epsilon) &= \sin 2x + 2\epsilon\cos 2x + \mathcal{O}\left(\epsilon^2\right); \\ \cos(x+\epsilon) &= \cos x - \epsilon\sin x + \mathcal{O}\left(\epsilon^2\right); \\ \cos(2x+2\epsilon) &= \cos 2x - 2\epsilon\sin 2x + \mathcal{O}\left(\epsilon^2\right). \end{aligned} \tag{19}$$

Additionally, we note that at the undesirable, one-cluster fixed point:

$$\begin{aligned} \sin\theta_j &\approx \sin\theta_l \quad \forall l \in N \\ &\text{and} \\ \cos\theta_j &\approx \cos\theta_l \quad \forall l \in N. \end{aligned} \tag{20}$$

These approximations allow us to rewrite the derivative as:

$$\left.\frac{\partial C_k}{\partial \theta_j}\right|_{\theta_j+\epsilon} = -2\epsilon\frac{\alpha_1}{R_1}\sum_{l\neq j}\left(\cos^2\theta_l + \sin^2\theta_l\right) - 8\epsilon\frac{\alpha_2}{R_2}\sum_{l\neq j}\left(\cos^2 2\theta_l + \sin^2 2\theta_l\right), \tag{21}$$

where we have additionally used the fact that (18) is equal to 0 at the fixed point. Further simplifying, we find:

$$-2\epsilon\,(N-1)\,\frac{\alpha_1}{R_1} - 8\epsilon\,(N-1)\,\frac{\alpha_2}{R_2} < 0; \tag{22}$$

rearranging and solving the inequality with $R_1 = R_2 = 1$ yields the condition:

$$\alpha_1 > -4\alpha_2. \tag{23}$$

To aid performance, for our analysis we empirically found a simpler and more conservative bound was useful; for the $N = 50$ system considered below, the coefficients α_l for the k-cluster problem were written as:

$$\alpha_l = \begin{cases} N^{k-l+1}, & l < k \\ -N, & l = k \end{cases}. \tag{24}$$

We note that we have made no claims or statements about reachability or controllability with this analysis, only that the condition provided is sufficient to overcome an empirically observed pitfall of order parameter-based control.

5.2 Population clustering simulation results

A population of 50 neurons was simulated using the neural network. For each cycle, an input was selected by finding the minimal value on a three-dimensional grid of values for

$(I_0, t_{\text{width}}, t_{\text{delay}})$ of the cost of the input according to the overall cost:

$$Q = C_k + \alpha I_0^2 t_{\text{width}}, \tag{25}$$

where α was set to 0.01 and C_k was of the form defined in (17).

Fifty different trials were run with initial firing times for each neuron randomly selected from a normal distribution ($\mu = 0, \sigma = \frac{T}{8}$). Once optimal input stimuli were generated via the neural network, the inputs were applied to the full ODE model. Phase values at the end of the simulation were calculated as the time that had elapsed since the neuron had last fired divided by the natural period T_0 and multiplied by 2π. Boxplots of these simulation results for two-cluster and three-cluster control objectives are shown in Fig. 11. For both control objectives, the median values for the desired order parameter (R_2 for two clusters and R_3 for three clusters) were statistically significantly higher than the other two-order parameters. The largest difference between the neural network's predicted results and the full simulation's output is that lower-order parameters, notably R_1, are not as effectively eliminated in the simulation when compared to the neural network's expectation. This is again consistent with the previous analysis of the regression; these errors are largely due to the inability to accurately distinguish the effects of input stimuli when two neurons are very close (< 1 ms). As a result, while in the neural network the correct number of neurons may end up in each cluster (ideally, balanced equally) which in turn leads the lower-order parameters to approach zero, the network simply does not have the precision in practice to correctly cleave the initial distribution when the stimulus is applied to the simulation, resulting in slightly imbalanced (but still strong) clusters. A characteristic resulting output for the three-cluster case (with values of R_1, R_2, and R_3 close to the median values for the ODE output) is additionally shown in Fig. 12. We note that despite the values of R_1 and R_2 both hovering around 0.2 for this realization, a distinct three-cluster state does emerge by the end of the simulation time.

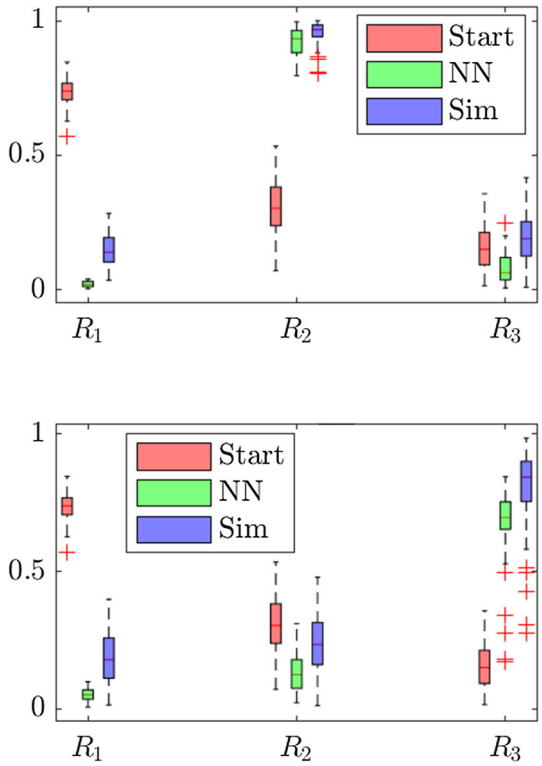

Fig. 11 Boxplots of order parameters for 50 trials of two-cluster (*top*) and three-cluster (*bottom*) control objectives. Although there are some errors when comparing the predictions from the neural network (in green) to the actual output from the ODE system (in red), we can see that the neural network performs well and successfully maximizes the correct order parameter in both control objectives (color figure online)

5.3 Extension to noisy oscillators

An important measure of robustness when designing control algorithms is sensitivity to noise, so we applied the neural network's generated stimulation patterns to a noisy version of the ODE to evaluate its response. The same initial conditions and generated stimulation patterns from the previous section were again used, but an additive Gaussian white noise term was included in the equation for $\dot{V}$:

$$\dot{V} = \cdots + \sigma \eta(t), \tag{26}$$

where σ is the standard deviation of the noise and $\eta(t)$ is the Gaussian distribution with mean 0 and standard deviation 1, and the sampling of the distribution is independent and identically distributed such that $\langle \eta(t)\, \eta(s) \rangle = \delta(t - s)$. Numerical simulation for the noisy oscillators was carried out using a 4th-order Runge–Kutta solver adapted for noise. The 2nd-order solver was explicitly derived in Honeycutt (1992), and the derivation there can, as noted in the original paper, be extended to higher-order Runge–Kutta solvers; a 4th-order adaptation is presented in Durham (2007). Differing magnitudes of noise were added to the simulation; as can be seen in Fig. 13, even for highly noisy systems, the control sequence developed under the no-noise condition retained success in achieving the appropriate clustering.

5.4 Control in the presence of coupling

The other important measure of robustness is how well the control maintains validity when we consider interactions between the neurons. To this end, we carried out the same protocol as previously but included all-to-all electrotonic coupling in addition to noise. For all trials, the standard deviation of the voltage noise was set to $\sigma = 5$. For the electrotonic

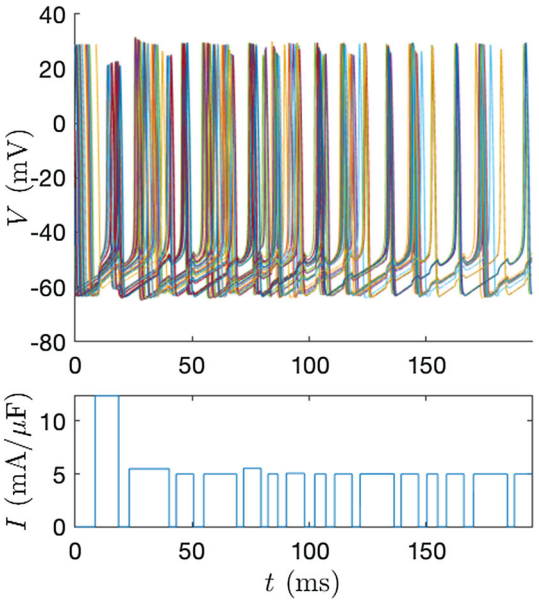

Fig. 12 ODE simulation output with final order parameter values $R_1 = 0.1747$, $R_2 = 0.2686$, and $R_3 = 0.8392$. This particular realization was selected for its closeness to the median results, suggesting it is representative of an average clustering sequence. The *top* panel shows voltage traces for the 50 simulated neurons; we note that though they start normally distributed, they end in a clear three-cluster configuration. The *bottom* panel shows the computed applied control input which was identically and simultaneously applied to all neurons to generate the three-cluster configuration (color figure online)

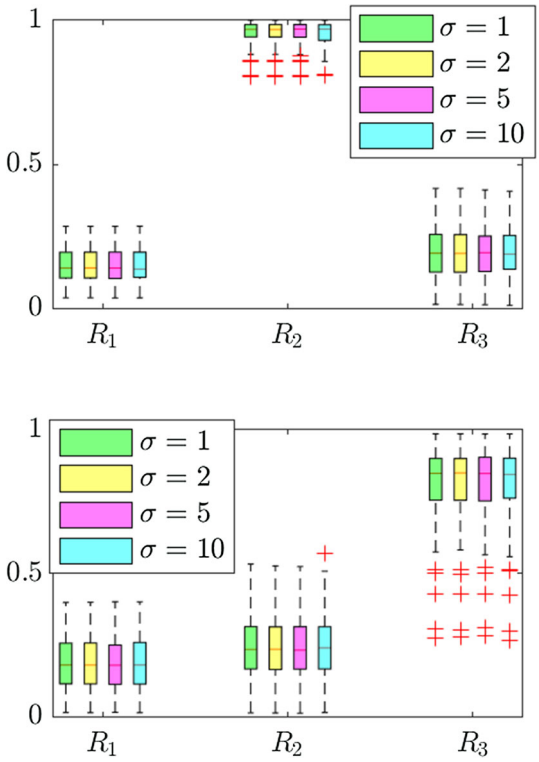

Fig. 13 Boxplots for 50 trials with additive Gaussian noise of differing magnitudes for two clusters (*top*) and three clusters (*bottom*). Even for the high-variance case $\sigma = 10$, the clustering control sequence is still successful in generating clusters, indicating the robustness of the control algorithm (color figure online)

coupling, we implement the interaction between neurons as in Johnston and Wu (1995) and considered different values of the coupling strength a_e:

$$\Delta \dot{V}_i = \frac{a_e}{N} \sum_{j=1}^{N} \left(V_j - V_i \right). \tag{27}$$

Otherwise, as in the case of considering noisy oscillators, the neurons were stimulated using the same stimulation parameters developed for the noise-free, uncoupled scenario. Three levels of coupling were considered: $a_e = 0.01$, $a_e = 0.1$, and $a_e = 1.0$, which correspond to weak, moderate, or strong coupling, respectively. As can be seen in Fig. 14, weak coupling has little impact on the results, while the control sequence still generally performs well with moderate coupling (though performance is degraded when compared to the uncoupled baseline). We additionally note that the two-cluster control objective outperforms the three-cluster objective, demonstrating it is easier to achieve fewer clusters. This is particularly pronounced in the case of moderate coupling, where there is far less of a dropoff in performance in the case of two clusters than for three, while strong coupling is impossible for the control sequences for either objective to overcome without modification. This indicates that when only moderate or weak coupling is expected, we need not modify our approach, but in applications where the coupling strength is expected to be much larger, it may be necessary to supplement the base model with some additional estimation of the effect of coupling. This is in contrast to the influence of noise, which showed far less effect on the outcome even for large values of the noise. Most likely, this is because noise does not introduce new stable configurations in the same way coupling does and its average effect is 0 in the case of white noise, independent of the magnitude of the standard deviation. Therefore, noise is roughly as likely to cause a neuron to fire earlier than anticipated as it is to cause the neuron to fire later than anticipated, so in the aggregate, its effect on the accuracy of the control is less pronounced. We would likely see larger effects if the neural network used the voltage traces of the neurons directly in making predictions, but since only the firing times are recorded, individual fluctuations have little effect on the overall system.

6 Conclusion

We developed a fully connected deep neural network capable of making accurate predictions of the firing time of a neu-

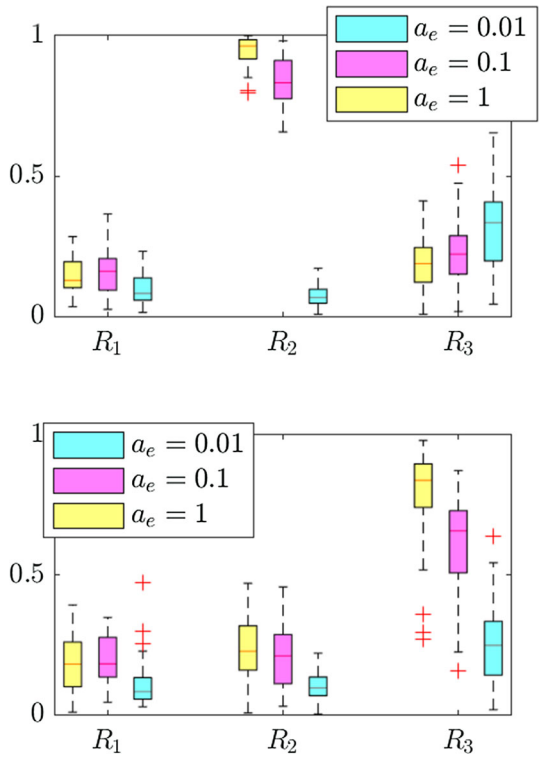

Fig. 14 Boxplots for weak, moderate, and strong coupling for two clusters (*top*) and three clusters (*bottom*). The results behave as expected, with the predictions of the neural network performing worse as coupling strength increases. Despite this, the control strategy shows significant resilience in the presence of weak to moderate coupling, with clear elevations in the appropriate order parameter compared to the other order parameters (color figure online)

ron model when trained on a corpus of data. This network significantly outperformed other feed-forward approaches to regression provided equivalent data, such as interpolation and polynomial regression. We found the network to be nearly uniformly accurate, struggling only in cases of short, high-amplitude signals on the edges of our training dataset.

Having verified the validity of the model, we considered three different strategies and objectives for control. In our first two examples, we sought to desynchronize a pair of neurons, first as efficiently as possible with a single input, then by implementing dynamic programming to develop an optimal sequence of stimulations. We showed the model was able to accurately desynchronize the neurons in both of these trials. Lastly, we shifted our focus to an underactuated ensemble of neural oscillators. After developing a control protocol based on the order parameters of the ensemble, we showed effective control could be generated to achieve either two- or three-cluster configurations, even in the presence of Gaussian white noise or weak to moderate electrotonic coupling.

We believe an interesting area for further development is generalizing the control input signal and finding ways to adaptively learn optimal signals. We intend to extend this work in future research to an orthogonal basis of input signal parameters, such as Legendre polynomials, that includes the square waves considered here but additionally allows for more varied control input schemas.

We additionally note that there are some considerations that have been excluded from this analysis but would be of interest in the future. First, although the control examples presented in this paper have centered around clustering neural populations, in applications such as DBS for Parkinson's disease, the specific mechanism by which stimulation alleviates symptoms are at best unclear, and clustering and/or desynchronization may be either unnecessary or undesirable. One advantage of the formulation presented here is that it is largely agnostic to the specific control objective: application of the strategies outlined here relies only on developing a function that evaluates the relative goodness of a particular configuration of oscillators and that can be measured via or associated with the firing times of the neurons.

Second, we note that the control strategies presented here should ultimately be verified experimentally, though this is beyond the scope of this paper. The soundness of maximally efficient desynchronization has previously been explored by Faramarzi and Netoff (2021), but its integration with machine learning and use in larger populations remains to be experimentally verified, as do the dynamic programming-based desynchronization algorithms presented here.

Acknowledgements This work was supported by National Science Foundation Grant No. NSF-1264535/1631170.

STN neuron model

The STN neuron model is adapted from the interconnected model of the basal ganglia presented in Rubin and Terman (2004a, b), in turn adapted from Terman et al. (2002). The governing ODEs are:

$$\dot{V} = -\left(I_\ell + I_{\mathrm{Na}} + I_{\mathrm{K}} + I_{\mathrm{AHP}} + I_{\mathrm{Ca}} + I_t\right) + I_0 + I\,(t) \tag{28}$$

$$\dot{h} = \phi \frac{h_\infty - h}{\tau_h} \tag{29}$$

$$\dot{n} = \phi \frac{n_\infty - n}{\tau_n} \tag{30}$$

$$\dot{r} = \phi_r \frac{r_\infty - r}{\tau_r} \tag{31}$$

$$[\dot{\mathrm{Ca}}] = \epsilon\phi\left(-I_{\mathrm{Ca}} - I_t - k_{\mathrm{Ca}}\,[\mathrm{Ca}]\right), \tag{32}$$

with $\phi = 0.75$, $\phi_r = 0.5$, and $\epsilon = 5E-5$. We note here that whereas in the prior two conductance-based models, the variables consisted of the voltage V and then a set of gating

Table 3 Constants for STN neuron model from Terman et al. (2002); Rubin and Terman (2004a, b)

Variable	Value	Description
V_{Na}	55 mV	Sodium current reversal potential
V_{K}	−80 mV	Potassium current reversal potential
V_{ℓ}	−60 mV	Leak current reversal potential
V_{Ca}	140 mV	Calcium current reversal potential
$\bar{g}_{\mathrm{Na}}$	37.5 nS/μm^2	Maximum sodium current conductance
$\bar{g}_{\mathrm{K}}$	45 nS/μm^2	Maximum potassium current conductance
$\bar{g}_{\ell}$	2.25 nS/μm^2	Maximum leak current conductance
$\bar{g}_{\mathrm{T}}$	0.5 nS/μm^2	Maximum T-type calcium current conductance
$\bar{g}_{\mathrm{AHP}}$	9 nS/μm^2	Maximum afterhyperpolarization potassium current conductance
$\bar{g}_{\mathrm{Ca}}$	0.5 nS/μm^2	Maximum high-threshold calcium current conductance
I_0	25 mA	Base current

variables, here [Ca] refers to the concentration of calcium ions within the neuron.

The auxiliary equations are:

$$s_\infty = \frac{1}{1+e^{-\frac{V+39}{8}}} \tag{33}$$

$$m_\infty = \frac{1}{1+e^{-\frac{V-30}{15}}} \tag{34}$$

$$h_\infty = \frac{1}{1+e^{\frac{v+39}{3.1}}} \tag{35}$$

$$n_\infty = \frac{1}{1+e^{\frac{v+32}{-8}}} \tag{36}$$

$$\tau_h = 1 + \frac{500}{1+e^{\frac{V+57}{3}}} \tag{37}$$

$$\tau_n = 1 + \frac{100}{1+e^{\frac{v+80}{26}}} \tag{38}$$

$$\tau_r = 7.1 + \frac{17.5}{1+e^{\frac{V-68}{2.2}}} \tag{39}$$

$$T_\infty = \frac{1}{1+e^{\frac{V+63}{-7.8}}} \tag{40}$$

$$r_{\mathrm{new}} = \frac{1}{1+e^{\frac{r-0.25}{-0.07}}} - \frac{1}{1+e^{-\frac{0.25}{-0.07}}}. \tag{41}$$

Using these, we can calculate the various currents as:

$$I_\ell = \bar{g}_\ell\,(V - V_\ell) \tag{42}$$

$$I_{\mathrm{Na}} = \bar{g}_{\mathrm{Na}} {m_\infty}^3 h\,(V - V_{\mathrm{Na}}) \tag{43}$$

$$I_K = \bar{g}_K n^4\,(V - V_K) \tag{44}$$

$$I_{\mathrm{AHP}} = \bar{g}_{\mathrm{AHP}}\,(V - V_K)\,\frac{[\mathrm{Ca}]}{[Ca] + k_1} \tag{45}$$

$$I_{\mathrm{Ca}} = \bar{g}_{\mathrm{Ca}} {s_\infty}^2\,(V - V_{\mathrm{Ca}}) \tag{46}$$

$$I_T = \bar{g}_T {T_\infty}^3 {r_{\mathrm{new}}}^2\,(V - V_{\mathrm{Ca}})\,. \tag{47}$$

The associated constants are listed in Table 3.

References

Adamchic I, Hauptmann C, Barnikol UB, Pawelczyk N, Popovych O, Barnikol TT, Silchenko A, Volkmann J, Deuschl G, Meissner WG, Maarouf M, Sturm V, Freund HJ, Tass PA (2014) Coordinated reset neuromodulation for Parkinson's disease: proof-of-concept study. Mov Disord 29(13):1679–1684

Bronte-Stewart H, Barberini C, Koop MM, Hill BC, Henderson JM, Wingeier B (2009) The STN beta-band profile in Parkinson's disease is stationary and shows prolonged attenuation after deep brain stimulation. Exp Neurol 215(1):20–28

Brown E, Moehlis J, Holmes P (2004) On the phase reduction and response dynamics of neural oscillator populations. Neural Comput 16(4):673–715

Chen CC, Litvak V, Gilbertson T, Kühn A, Lu CS, Lee ST, Tsai CH, Tisch S, Limousin P, Hariz M, Brown P (2007) Excessive synchronization of basal ganglia neurons at 20 Hz slows movement in Parkinson's disease. Exp Neurol 205(1):214–221

Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Math Control Signals Syst 2(4):303–314

Daido H (1996) Onset of cooperative entrainment in limit-cycle oscillators with uniform all-to-all interactions: bifurcation of the order function. Phys D Nonlinear Phenom 91(1–2):24–66

Diekman CO, Bose A (2016) Entrainment maps: a new tool for understanding properties of circadian oscillator models. J Biol Rythm 31(6):598–616

Durham JW (2007) Controlling canards using ideas from the theory of mixed-mode oscillations. PhD thesis, University of California, Santa Barbara

Ermentrout B (1996) Type I membranes, phase resetting curves, and synchrony. Neural Comput 8(5):979–1001

Faramarzi S, Netoff TI (2021) Closed-loop neuromodulation for clustering neuronal populations. J Neurophysiol 125(1):248–255

Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. Int Conf Artif Intell Stat 9:249–256

Hahn PJ, McIntyre CC (2010) Modeling shifts in the rate and pattern of subthalamopallidal network activity during deep brain stimulation. J Comput Neurosci 28(3):425–441

Hammond C, Bergman H, Brown P (2007) Pathological synchronization in Parkinson's disease: networks, models and treatments. Trends Neurosci 30(7):357–364

Hansel D, Mato G, Meunier C (1995) Synchrony in excitatory neural networks. Neural Comput 7(2):307–337

Holgado AJN, Terry JR, Bogacz R (2010) Conditions for the generation of beta oscillations in the subthalamic nucleus–globus pallidus network. J Neurosci 30(37):12340–12352

Holt AB, Netoff TI (2014) Origins and suppression of oscillations in a computational model of Parkinson's disease. J Comput Neurosci 37(3):505–521

Holt AB, Wilson D, Shinn M, Moehlis J, Netoff TI (2016) Phasic burst stimulation: a closed-loop approach to tuning deep brain stimulation parameters for Parkinson's disease. PLoS Comput Biol 12(7):1–14

Honeycutt RL (1992) Stochastic Runge-Kutta algorithms. I. White noise. Phys Rev A 45(2):600–603

Hua SE, Lenz Fa, Zirh Ta, Reich SG, Dougherty PM (1998) Thalamic neuronal activity correlated with essential tremor. J Neurol Neurosurg Psychiatry 64(2):273–276

Johnston D, Wu SMS (1995) Foundations of cellular neurophysiology, 1st edn. MIT Press, Cambridge, MA

Juul JS, Krishna S, Jensen MH (2018) Entrainment as a means of controlling phase waves in populations of coupled oscillators. Phys Rev E 98(6):1–9

Kawaguchi K (2016) Deep learning without poor local minima. In: Conference on neural information processing systems (NIPS), pp 586–594

Kühn A, Trottenberg T, Kivi A, Kupsch A, Schneider Gh, Brown P (2005) The relationship between local field potential and neuronal discharge in the subthalamic nucleus of patients with Parkinson's disease. Exp Neurol 194:212–220

Kühn AA, Kempf F, Brücke C, Doyle LG, Martinez-Torres I, Pogosyan A, Trottenberg T, Kupsch A, Schneider GH, Hariz MI, Vandenberghe W, Nuttin B, Brown P (2008) High-frequency stimulation of the subthalamic nucleus suppresses oscillatory β activity in patients with Parkinson's disease in parallel with improvement in motor performance. J Neurosci 28(24):6165–6173

Kuramoto Y (1984) Chemical oscillations, waves, and turbulence, springer series in synergetics, vol 19. Springer, Berlin Heidelberg, Berlin, Heidelberg

Kurebayashi W, Shirasaka S, Nakao H (2013) Phase reduction method for strongly perturbed limit cycle oscillators. Phys Rev Lett 111(21)

Levy R, Hutchison W, Lozano A, Dostrovsky J (2000) High-frequency synchronization of neuronal activity in the subthalamic nucleus of Parkinsonian patients with limb tremor. J Neurosci 20(20):7766–7775

Li JS, Dasanayake I, Ruths J (2013) Control and synchronization of neuron ensembles. IEEE Trans Autom Control 58(8):1919–1930

Liao HI, Wu DA, Halelamien N, Shimojo S (2013) Cortical stimulation consolidates and reactivates visual experience: neural plasticity from magnetic entrainment of visual activity. Sci Rep 3(1):2228

Lysyansky B, Popovych OV, Tass PA (2011) Desynchronizing anti-resonance effect of m:n ON-OFF coordinated reset stimulation. J Neural Eng 8(3)

Lysyansky B, Popovych OV, Tass PA (2013) Optimal number of stimulation contacts for coordinated reset neuromodulation. Front Neuroeng 6:5

Matchen TD, Moehlis J (2018) Phase model-based neuron stabilization into arbitrary clusters. J Comput Neurosci 44(3):363–378

Mitchell BA, Petzold LR (2018) Control of neural systems at multiple scales using model-free, deep reinforcement learning. Sci Rep 8(1):1–12

Moehlis J, Shea-Brown E, Rabitz H (2006) Optimal inputs for phase models of spiking neurons. J Comput Nonlinear Dyn 1(4):358–367

Monga B, Moehlis J (2019) Optimal phase control of biological oscillators using augmented phase reduction. Biol Cybern pp 161–178

Monga B, Moehlis J (2020) Supervised learning algorithms for controlling underactuated dynamical systems. Phys D Nonlinear Phenom 412

Monga B, Wilson D, Matchen T, Moehlis J (2019) Phase reduction and phase-based optimal control for biological systems: a tutorial. Biol Cybern 113(1–2):11–46

Nagaraj V, Lamperski A, Netoff TI (2017) Seizure control in a computational model using a reinforcement learning stimulation paradigm. I J Neural Syst 27(07):1750012

Narayanan V, Ritt JT, Li JS, Ching S (2019) A learning framework for controlling spiking neural networks. In: 2019 American control conference (ACC), IEEE, pp 211–216

Otsuka T, Abe T, Tsukagawa T, Song WJ (2004) Conductance-based model of the voltage-dependent generation of a plateau potential in subthalamic neurons. J Neurophysiol 92(1):255–264

Power AJ, Mead N, Barnes L, Goswami U (2012) Neural entrainment to rhythmically presented auditory, visual, and audio-visual speech in children. Front Psychol 3(July):1–13

Quattoni A, Collins M, Darrell T (2008) Transfer learning for image classification with sparse prototype representations. In: 26th IEEE conference on computer vision and pattern recognition, CVPR (March 2008)

Rodríguez-Pineda JA (2000) Competitive Hebbian learning through spiking-timing dependent plasticity (STDP). Thesis and Dissertation 3:919–926

Roenneberg T, Dragovic Z, Merrow M (2005) Demasking biological oscillators: properties and principles of entrainment exemplified by the Neurospora circadian clock. In: Proceedings of the national academy of sciences of the United States of America 102(21):7742–7747

Rubin JE, Terman D (2004a) High frequency stimulation of the subthalamic nucleus eliminates pathological thalamic rhythmicity in a computational model. J Comput Neurosci 16(3):211–235

Rubin JE, Terman D (2004b) High frequency stimulation of the subthalamic nucleus eliminates pathological thalamic rhythmicity in a computational model. J Comput Neurosci 16(3):211–235

Saini R, Jaskolski M, Davis SJ (2019) Circadian oscillator proteins across the kingdoms of life: structural aspects. BMC Biol 17(1):1–39

Savica R, Stead M, Mack KJ, Lee KH, Klassen BT (2012) Deep brain stimulation in Tourette syndrome: a description of 3 patients with excellent outcome. Mayo Clin Proc 87(1):59–62

Schnitzler A, Gross J (2005) Normal and pathological oscillatory communication in the brain. Nat Rev Neurosci 6(4):285–296

Skardal PS, Arenas A (2015) Control of coupled oscillator networks with application to microgrid technologies. Sci Adv 1(7):1–7

Tass PA (2003) Desynchronization by means of a coordinated reset of neural sub-populations—a novel technique for demand-controlled deep brain stimulation. Progress Theor Phys Suppl 150(150):281–296

Taylor AF, Kapetanopoulos P, Whitaker BJ, Toth R, Bull L, Tinsley MR (2008) Phase clustering in globally coupled photochemical oscillators. Eur Phys J Spec Top 165(1):137–149

Terman D, Rubin JE, Yew AC, Wilson CJ (2002) Activity patterns in a model for the subthalamopallidal network of the basal ganglia. J Neurosci 22(7):2963–2976

The Deep-Brain Stimulation for Parkinson's Disease Study Group (2001) Deep-brain stimulation of the subthalamic nucleus or the pars interna of the globus pallidus in Parkinson's disease. N Engl J Med 345(13):956–963

Titiz AS, Hill MR, Mankin EA, Aghajan ZM, Eliashiv D, Tchemodanov N, Maoz U, Stern J, Tran ME, Schuette P, Behnke E, Suthana NA, Fried I (2017) Theta-burst microstimulation in the human entorhinal area improves memory specificity. eLife 6:1–18

Uhlhaas PJ, Singer W (2006) Neural synchrony in brain disorders: relevance for cognitive dysfunctions and pathophysiology. Neuron 52(1):155–168

Villaverde AF, Tsiantis N, Banga JR (2019) Full observability and estimation of unknown inputs, states and parameters of nonlinear biological models. J Royal Soc Interface 16(156)

Wilson CJ, Beverlin B, Netoff T (2011) Chaotic desynchronization as the therapeutic mechanism of deep brain stimulation. Front Syst Neurosci 5:50

Wilson D, Moehlis J (2014a) Locally optimal extracellular stimulation for chaotic desynchronization of neural populations. J Comput Neurosci 37(2):243–257

Wilson D, Moehlis J (2014b) Optimal chaotic desynchronization for neural populations. SIAM J Appl Dynam Syst 13(1):276–305

Wilson D, Moehlis J (2015) Clustered desynchronization from high-frequency deep brain stimulation. PLoS Comput Biol 11(12):1–26

Wilson D, Moehlis J (2016) Isostable reduction of periodic orbits. Phys Rev E 94(5):1–7

Winfree AT (2001) The geometry of biological time, interdisciplinary applied mathematics, vol 12. Springer, New York, NY

Wingeier B, Tcheng T, Koop MM, Hill BC, Heit G, Bronte-Stewart HM (2006) Intra-operative STN DBS attenuates the prominent beta rhythm in the STN in Parkinson's disease. Exp Neurol 197(1):244–251

Yu YC, Narayanan V, Ching S, Li JS (2020) Learning to control neurons using aggregated measurements. In: Proceedings of the American control conference, pp 4028–4033

Zalalutdinov M, Aubin KL, Pandey M, Zehnder AT, Rand RH, Craighead HG, Parpia JM, Houston BH (2003) Frequency entrainment for micromechanical oscillator. Appl Phys Lett 83(16):3281–3283

Zhao C, Wang L, Netoff T, Yuan LL (2011) Dendritic mechanisms controlling the threshold and timing requirement of synaptic plasticity. Hippocampus 21(3):288–297

Zhu Y, Chen Y, Lu Z (2011) Heterogeneous transfer learning for image classification. In: AAAI conference on artificial intelligence, pp 1304–1309

Zlotnik A, Li JS (2014) Optimal subharmonic entrainment of weakly forced nonlinear oscillators. SIAM J Appl Dynam Syst 13(4):1654–1693

Zlotnik A, Nagao R, Kiss IZ, Li JS (2016) Phase-selective entrainment of nonlinear oscillator ensembles. Nat Commun 7:1–7