Physica D 412 (2020) 132621

Contents lists available at ScienceDirect

Physica D

journal homepage: www.elsevier.com/locate/physd

Supervised learning algorithms for controlling underactuated dynamical systems☆

Bharat Monga [a,*], Jeff Moehlis [a,b]

[a] Department of Mechanical Engineering, Engineering II Building, University of California Santa Barbara, Santa Barbara, CA 93106, United States
[b] Program in Dynamical Neuroscience, University of California Santa Barbara, Santa Barbara, CA 93106, United States

ARTICLE INFO

Article history:
Received 29 August 2019
Received in revised form 26 May 2020
Accepted 13 June 2020
Available online 18 June 2020

Keywords:
Supervised learning
Underactuated dynamical systems
Bang bang control
Coupled oscillators
Machine learning
Binary classification

ABSTRACT

Control of underactuated dynamical systems has been studied for decades in robotics, and is now emerging in other fields such as neuroscience. Most of the advances have been in model based control theory, which has limitations when the system under study is very complex and it is not possible to construct a model. This calls for data driven control methods like machine learning, which has spread to many fields in the recent years including control theory. However, the success of such algorithms has been dependent on availability of large datasets. Moreover, due to their black box nature, it is challenging to analyze how such algorithms work, which may be crucial in applications where failure is very costly. In this paper, we develop two related novel supervised learning algorithms. The algorithms are powerful enough to control a wide variety of complex underactuated dynamical systems, and yet have a simple and intelligent structure that allows them to work with a sparse data set even in the presence of noise. Our algorithms output a bang-bang (binary) control input by taking in feedback of the state of the dynamical system. The algorithms learn this control input by maximizing a reward function in both short and long time horizons. We demonstrate the versatility of our algorithms by applying them to a diverse range of applications including: switching between bistable states, changing the phase of an oscillator, desynchronizing a population of synchronized coupled oscillators, and stabilizing an unstable fixed point. For most of these applications we are able to reason why our algorithms work by using traditional dynamical systems and control theory. We also compare our learning algorithms with some traditional control algorithms, and reason why our algorithms work better.

1. Introduction

Underactuated dynamical systems are systems with fewer actuators/controls than the dimensionality of the state space of the system. Such systems are ubiquitous in a variety of fields including physics, chemistry, biology and engineering. There have been numerous advances made on controlling such systems, with much of the work in robotics [1–3]. Control in other applications, especially biology [4–7], is on the rise as it provides promising treatment strategies for several disorders such as Parkinson's disease, cardiac arrhythmias, and jet lag. Most of these control methods, both in robotics and biology, are based on traditional model based control theory and optimal control.

Such methods work well when it is possible to model the dynamics of the system accurately, which is very difficult as the systems become complicated, especially in neuroscience applications where the dynamics of a single neuron may change rapidly depending on the response from other neurons in the network. Even if an accurate model could be built to describe the dynamics of such a system, developing a classical model based control for such an underactuated system is a challenging task. If the parameters of the system change with time, or if the model does not describe the dynamics accurately, the theoretical control guarantees like stability and boundedness may not apply in real systems [8,9]. This calls for the development of data driven control algorithms that can learn to control the system without explicitly using a model.

Artificial intelligence algorithms are able to learn to control dynamical systems by using deep neural networks. Such algorithms have been used for a long time [10,11], but with the availability of large data sets, improvement in deep learning architectures, optimization methods, and cheap computation costs, their use is on the rise [12]. However, the success of such algorithms has been largely dependent on availability of large datasets [13], which can be limited in fields like neuroscience where the cost of obtaining human/animal brain data

☆ Implementation of the algorithms in this article is available at https://github.com/bharatmonga/Supervised-learning-algorithms.

* Corresponding author.
E-mail addresses: monga@ucsb.edu (B. Monga), moehlis@engineering.ucsb.edu (J. Moehlis).

https://doi.org/10.1016/j.physd.2020.132621
0167-2789/

is very high. Moreover, the black box nature of such methods makes their analysis difficult. Such analysis would be important in tasks where failure is very costly. Another limitation of such methods is their inability to take advantage of the inherent dynamics of the system to achieve the task, which limits their performance.

All these limitations call for a new machine learning control algorithm that does not rely on large amounts of data, is easy to understand, and can take advantage of the underlying dynamics in achieving the task. In this article, we have developed two related novel supervised learning algorithms based on these three goals. Our algorithms are powerful enough to control a wide variety of complex underactuated dynamical systems, and yet have a simple structure so one can understand how they work using dynamical systems and control theory foundations. Their simple yet intelligent structure also allows them to effectively achieve the control objective by training on a sparse data set, even in the presence of noise. Our algorithms output a bang-bang (binary) control input by taking in feedback of the state of the dynamical system. The algorithms learn this control input by maximizing a reward function in both short and long time horizons. We demonstrate the versatility of our algorithms by applying them to a diverse range of underactuated dynamical systems including: switching between bistable states, changing the phase of an oscillator, desynchronizing a population of synchronized coupled oscillators, and stabilizing an unstable fixed point of a dynamical system. For most of these applications we are able to reason why our algorithms work by using traditional dynamical systems and control theory. We compare our algorithms with traditional control algorithms and reason why our algorithms work better, especially because they learn to take advantage of the underlying system dynamics in achieving the control objective. We carry out a robustness analysis to demonstrate the effectiveness of our algorithms even in the presence of noise.

This article in organized as follows. In Section 2, we develop our supervised learning algorithms to output a binary control. We demonstrate our first supervised learning algorithm by controlling underactuated bistable dynamical systems in Section 3, and compare our algorithm to a fully actuated control. In Section 4, we illustrate the effectiveness of our second supervised learning algorithm by controlling the phase of a single oscillator and comparing the algorithm to a model based optimal control algorithm. We further demonstrate the versatility of our second supervised learning algorithm by using it to desynchronize a population of synchronized coupled oscillators in Section 5. In Section 6, we apply our second algorithm to stabilize an unstable fixed point of an underactuated dynamical system, and compare the algorithm to a model based control algorithm. To demonstrate the applicability of our algorithms in a more realistic setting, we show how their intelligent structure allows them to perform well in the presence of noise. Section 8 summarizes our work and suggests future extensions and tools. Appendix A lists the mathematical models used in this article. In Appendix B, we give background on phase reduction relevant for the model based control comparison and validation of our second learning algorithm in Sections 4 and 5, respectively.

2. Supervised learning algorithms

In this section, we develop our supervised learning algorithms to control a diverse range of underactuated dynamical systems. The development includes two important steps: the first step is generating an appropriate training data set, and the second step is feeding that data set into a binary classifier to control the dynamical system. We call the first step the training algorithm, as it generates the training data by maximizing a reward function. We show in the rest of the paper how choosing an appropriate reward function can be used to take advantage of the inherent dynamics to control a variety of dynamical systems. For the second step, we design a locally weighted binary classifier that takes in the state of the dynamical system as input, and outputs a binary control input in real time. The local nature of our classifier allows it to effectively interpolate between nonlinear boundaries inherent in our data set, and thus plays an important role in the control. The combination of these two steps results in our supervised learning algorithms described below and shown in Fig. 1.

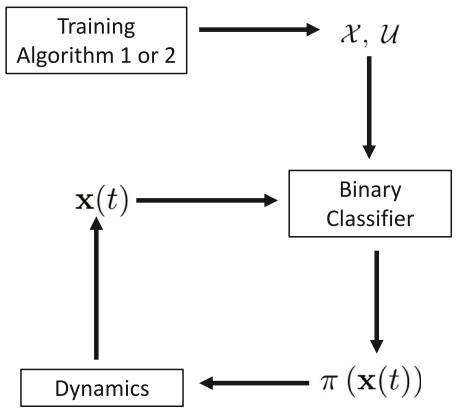

Fig. 1. Flowchart of the two supervised learning algorithms.

We consider an underactuated dynamical system with an additive control input $\pi(\mathbf{x}(t))$ as

$$\frac{d}{dt}\mathbf{x}(t) = F(\mathbf{x}(t)) + [\pi(\mathbf{x}(t)), 0_{n-1}]^T, \qquad \mathbf{x}(t) \in R^n, \tag{1}$$

where 0_{n-1} is an $n-1$ dimensional zero vector. Thus, the control input depends on the full state of the dynamical system, and only directly affects the first state of the dynamical system. The control input is binary in nature having two values $\{u_1 > 0, u_2\}$, which can be chosen differently for different applications. For our first algorithm, we take $u_2 = 0$, and thus the control can be thought of having an "ON" state with the value u_1 and an "OFF" state with value 0. For our second algorithm, we take the control to be a bang bang control with $u_2 = -u_1$. Both algorithms learn the control input $\pi(\mathbf{x}(t))$ as a function of the state to achieve a particular control objective. They do so by learning from the data generated by sampling a model describing the underlying dynamics of the system. To demonstrate our algorithm in this article, we use an analytical model ($F(\mathbf{x}(t))$) which generates our training data. In case a model is not available in an application, one can still use the same algorithms by obtaining training data by direct measurement of the states of the system at different times. Below we describe our supervised learning algorithms in more detail.

2.1. Training Algorithm 1

Our first supervised learning algorithm outputs a "ON" or "OFF" binary control input. The algorithm learns what control input to output to achieve a certain control objective by maximizing a reward function $\mathcal{R}(\mathbf{x}(t))$ which needs to be carefully designed to achieve a control objective in a particular application.

We sample a state $\mathbf{x}(0)$ randomly from the state space of the dynamical model of the system, and evolve the state forward in time for short time Δt with control state OFF. If $\mathcal{R}(\mathbf{x}(\Delta t)) \geq \mathcal{R}(\mathbf{x}(0))$, we set the control policy for state $\mathbf{x}(0)$ as OFF. If it is not, we again evolve the initial state forward for the same time but with control ON, and compare $\mathcal{R}(\mathbf{x}(\Delta t))$ for both control ON and OFF. Whichever control policy maximizes the reward is set for the sampled state $\mathbf{x}(0)$. We repeat the process N times by sampling more initial states randomly. The training algorithm is summarized below:

Training Algorithm 1

Initialize $\mathcal{X}$ as zeros(N,length($\mathbf{x}$)) and $\mathcal{U}$ as zeros(N,1)
for i=1,N **do**
  Randomly sample $\mathbf{x}(0)$
  Compute $\mathbf{x}(\Delta t)$ and $\mathcal{R}(\mathbf{x}(\Delta t))$ with control OFF
  if $\mathcal{R}(\mathbf{x}(\Delta t)) \geq \mathcal{R}(\mathbf{x}(0))$ **then**
    Set policy for $\mathbf{x}(0)$ as OFF
  else
    Compute $\mathbf{x}(\Delta t)$ and $\mathcal{R}(\mathbf{x}(\Delta t))$ with control ON
    if $\mathcal{R}(\mathbf{x}(\Delta t))$ with control ON $> \mathcal{R}(\mathbf{x}(\Delta t))$ with control OFF **then**
      Set policy for $\mathbf{x}(0)$ as ON
    else
      Set policy for $\mathbf{x}(0)$ as OFF
    end if
  end if
  Assign $\mathcal{X}[i,:]$ as $\mathbf{x}(0)$ and $\mathcal{U}[i,:]$ as the policy
end for
return $\mathcal{X}, \mathcal{U}$

Such an algorithm takes advantage of the underlying dynamics by letting the trajectories evolve without any control and only switching "ON" the control when necessary. This makes our algorithm highly energy efficient. Such an algorithm is very suitable for controlling bistable dynamical systems where the objective is for the trajectory to converge to a particular stable state of the system, or to switch from one stable state to another. The control can switch OFF when the trajectory enters the region of attraction of the desired stable state, and let the dynamics take the trajectory to the desired state.

2.2. Training Algorithm 2

Our second supervised learning algorithm outputs a bang-bang control input which can be used to control a variety of dynamical systems, including coupled oscillators. The algorithm learns what control input to output to achieve a certain control objective by maximizing a reward function $\mathcal{R}(\mathbf{x}(\Delta t))$, which needs to be carefully designed to achieve a control objective in a particular application.

We sample a state $\mathbf{x}(0)$ randomly from the state space of the dynamical model of the system, and evolve the state forward in time for short time Δt with both control u_1 and $-u_1$. In both scenarios we let the state evolve further in time with zero control until some event occurs and measure the timing of this event. The reward $\mathcal{R}(\mathbf{x}(\Delta t))$ is dependent on the timing of this event. Whichever policy (u_1 or $-u_1$) maximizes this reward is set for that sampled state. We repeat the process N times by sampling more states randomly. The algorithm is summarized below:

Training Algorithm 2

Initialize $\mathcal{X}$ as zeros(N,length($\mathbf{x}$)) and $\mathcal{U}$ as zeros(N,1)
for i=1,N **do**
  Randomly sample $\mathbf{x}(0)$
  Compute $\mathbf{x}(\Delta t)$ and $\mathcal{R}(\mathbf{x}(\Delta t))$ with control u_1
  Compute $\mathbf{x}(\Delta t)$ and $\mathcal{R}(\mathbf{x}(\Delta t))$ with control -u_1
  if $\mathcal{R}(\mathbf{x}(\Delta t))$ with control $u_1 \geq \mathcal{R}(\mathbf{x}(\Delta t))$ with control -u_1 **then**
    Set policy for $\mathbf{x}(0)$ as u_1
  else
    Set policy for $\mathbf{x}(0)$ as -u_1
  end if
  Assign $\mathcal{X}[i,:]$ as $\mathbf{x}(0)$ and $\mathcal{U}[i,:]$ as the policy
end for
return $\mathcal{X}, \mathcal{U}$

Such control is useful when the objective is to converge to an unstable state of the system, because the control needs to stay "ON" (be non-zero) the whole time even when the control objective has been realized, since the trajectory will go back to the stable state otherwise. Here as well the underlying dynamics of the system play a role in our learning algorithm: to determine the control input for a particular initial state, we let the dynamics evolve the trajectory until an event occurs.

Both of these training algorithms generate data comprising a set of N sampled states of the dynamical system $\mathcal{X}$, and a set of the corresponding control inputs $\mathcal{U}$. However, we need to know the control input $\pi(\mathbf{x}(t))$ as a function of a general trajectory $\mathbf{x}(t)$ of the system, since the trajectory is not restricted to these sampled states. This is achieved with our binary classifier, that takes an input as the state of the dynamical system $\mathbf{x}(t)$ and outputs the corresponding control policy $\pi(\mathbf{x}(t))$ based on this generated data.

2.3. Binary classifier

Our training algorithms generate data comprising a set $\mathcal{X}$ of N sampled states of the dynamical system, and a set of the corresponding control inputs $\mathcal{U}$. Based on this information we build a locally weighted binary classifier that takes an input as the instantaneous state of the dynamical system $\mathbf{x}(t)$, and outputs the corresponding control policy $\pi(\mathbf{x}(t))$ to be applied at that instant.

We assign each element of the set $\mathcal{X}$ with a weight

$$w_i(\mathbf{x}(t)) = \exp\left(-\frac{|\mathbf{x}(t) - \mathcal{X}_i|^2}{2\tau}\right), \quad i = 1, 2, \ldots, N,$$

where $\mathcal{X}_i$ represents the ith sampled state stored in the set $\mathcal{X}$. Thus a sampled state is given a higher weight if its closer to $\mathbf{x}(t)$, and a lower weight if it is further away from $\mathbf{x}(t)$. These weights are normalized so that

$$\sum_{i=1}^{N} w_i(\mathbf{x}(t)) = 1.$$

For the first algorithm, the classifier outputs $\pi(\mathbf{x}(t)) = u_1$ ("ON") if

$$\sum_{i=1}^{N} w_i(\mathbf{x}(t))\mathcal{U}_i > 0.5u_1, \tag{2}$$

and $\pi(\mathbf{x}(t)) = 0$ ("OFF") otherwise. Similarly for the second algorithm, the classifier outputs $\pi(\mathbf{x}(t)) = u_1$ if

$$\sum_{i=1}^{N} w_i(\mathbf{x}(t))\mathcal{U}_i > 0, \tag{3}$$

and $\pi(\mathbf{x}(t)) = -u_1$ otherwise. Here τ is a hyperparameter, which can be thought of as a bandwidth parameter. Smaller τ decreases the influence of faraway data points in local interpolation. Choosing a very small value can lead to overfitting and tight boundaries around the sampled points. On the other hand, a large value can bias the decision boundary in favor of a larger cluster of data points. We run experiments with different τ values and choose the one which enables our classifier to label the training data correctly, and also produces a smooth decision boundary between the clusters. The critical number on the right hand side of Eqs. (2) and (3) can be thought of as another hyperparameter which would influence the decision boundary together with the bandwidth parameter. We keep this parameter fixed for all our case studies, and find that a value of $0.5u_1$ and 0 respectively prove effective in outputting a smooth boundary decision around the cluster of data points for the examples that we consider. Choosing a larger (resp., smaller) critical value would extend the boundary in favor of clusters with labels OFF/negative (resp., ON/positive) control. We do not claim that the chosen hyperparameter values are optimal, but we believe that they are close to optimal for our examples, as the binary classifier is able to correctly label the training data and produce a smooth boundary, and thus is able to achieve the control objectives demonstrated in later sections of the article.

With $\pi(\mathbf{x}(t))$ defined to be the output of this binary classifier, we simulate the dynamical system from (1) starting from a random initial condition and find that our supervised learning algorithms are able to achieve the desired control objectives, while simultaneously maximizing the designed reward functions. The entire algorithm is depicted in the flowchart in Fig. 1.

Note that in our algorithm, the sets $\mathcal{X}$, $\mathcal{U}$ need to be computed only once for a given dynamical system, whereas the control input $\pi(\mathbf{x}(t))$ is computed by the binary classifier at every timestep.

3. Bistable dynamical systems

Bistability is widely found in neural systems [14] and cardiac arrhythmia [15], and is used in digital electronics for storing binary information, in mechanical switches for transitioning between ON and OFF states, and in multivibrators, Schmitt trigger circuits, and even optical systems [16]. It is the key mechanism for understanding several cellular processes including those associated with the onset and treatment of cancer [17]. In this section we apply our first supervised learning algorithm to control underactuated bistable dynamical systems. The control objective is for the trajectory to converge to a particular stable fixed point of the system starting anywhere in the state space (including in the basin of attraction of the other stable state). Such a control objective is relevant for several applications such as biocomputing, gene therapy, and treatment of cancer [18,19], among others.

3.1. Duffing oscillator

With the Duffing oscillator [20,21], we consider the class of bistable dynamical systems having two stable fixed points ($\mathbf{x}^{s1}$, $\mathbf{x}^{s2}$), and an unstable fixed point ($\mathbf{x}^u$). The control objective is for the trajectory to converge to $\mathbf{x}^{s2}$ starting anywhere in the state space. The Duffing oscillator is given as:

$$\dot{x} = y + \pi(\mathbf{x}(t)),$$
$$\dot{y} = x - x^3 - \delta y.$$

For $\delta > 0$, the system has two stable fixed points $\mathbf{x}^{s1} = (-1, 0)$ and $\mathbf{x}^{s2} = (1, 0)$, and an unstable fixed point $\mathbf{x}^u = (0, 0)$, all shown in Fig. 2. We take $\delta = 0.1$ in our simulations.

3.1.1. Learning algorithm

We choose our reward function to be the negative of the Euclidean distance between the current state and the desired state:

$$\mathcal{R}(\mathbf{x}(t)) = -\|\mathbf{x}(t) - \mathbf{x}^{s2}\|. \tag{4}$$

Thus the control will make the trajectory converge towards the desired fixed point while increasing the reward to 0. To converge to $\mathbf{x}^{s2}$ starting anywhere in the state space, we use our learning algorithm to generate a control policy. The ON (resp., OFF) state of the control policy corresponds to a value of $u_1 = 4$ (resp., 0). We randomly sample $N = 50$ points for generating the sets $\mathcal{X}$, $\mathcal{U}$, and choose $\Delta t = 0.001$ and $\tau = 0.4$.

3.1.2. Results

The generated control policy is shown in the left panel of Fig. 2. Blue open circles (resp., black $\times$'s) represent elements of the set $\mathcal{X}$ where the control policy given by elements of the set $\mathcal{U}$ is OFF (resp., ON). The green (resp., red) region is where output $\pi(\mathbf{x}(t))$ of the binary classifier is OFF (resp., ON). A controlled and an uncontrolled trajectory starting from same $\mathbf{x}(0)$ are shown in the right panel of Fig. 2. As can be seen in this figure, the control algorithm gradually converges the trajectory to $\mathbf{x}^{s2}$ by turning the control ON a few times, whereas the uncontrolled trajectory converges to $\mathbf{x}^{s1}$.

The algorithm generates an energy efficient control policy as the policy is OFF 60.41% of the total time it takes to drive the trajectory within a ball of radius of 0.45 in the region of attraction of $\mathbf{x}^{s2}$. We investigate the robustness of our learning algorithm by testing it on 1000 randomly generated initial conditions, and in all the 1000 cases, the control algorithm is able to converge the trajectories to $\mathbf{x}^{s2}$, achieving 100% accuracy. Choosing N is the crucial task in our learning algorithm. We start with a small N and keep increasing it until the algorithm achieves 100% effectiveness. $N = 50$ points turns out to be appropriate for the Duffing oscillator as choosing a lower number of points leads to underfitting, and choosing a higher number of points leads to a higher computational cost.

3.2. Reduced Hodgkin–Huxley model

With the reduced Hodgkin–Huxley model [22–24], we consider the class of bistable dynamical systems having a stable periodic orbit $\mathbf{x}^{s1}(t)$, an unstable periodic orbit $\mathbf{x}^u(t)$, and a stable fixed point $\mathbf{x}^{s2}$. The model is given as

$$\dot{v} = \big(I - g_{Na}(m_\infty(v))^3(0.8 - n)(v - v_{Na}) - g_K n^4(v - v_K) - g_L(v - v_L)\big)/c + \pi(\mathbf{x}(t)),$$
$$\dot{n} = a_n(v)(1 - n) - b_n(v)n,$$

where v is the trans-membrane voltage, and n is the gating variable. I is the baseline current, which we take as 6.69 $\mu A/cm^2$, and $\pi(\mathbf{x}(t))$ represents the applied control policy. For the rest of the parameters, see Appendix A.1. In the absence of control input, the system is bistable having $\mathbf{x}^{s1}(t)$ with period 14.91 ms, $\mathbf{x}^u(t)$ with period 14.33 ms, and $\mathbf{x}^{s2} = (-61.04, 0.38)$, all shown in Fig. 3.

3.2.1. Learning algorithm

The control objective is for the trajectory to converge to the stable fixed point starting anywhere in the state space. Here as well we choose the reward function (4). Without any control input, a trajectory starting outside $\mathbf{x}^u(t)$ will converge to $\mathbf{x}^{s1}(t)$, and a trajectory starting inside $\mathbf{x}^u(t)$ will converge to $\mathbf{x}^{s2}$. To converge to the stable fixed point starting anywhere in the state space, we use our learning algorithm to generate a control policy.

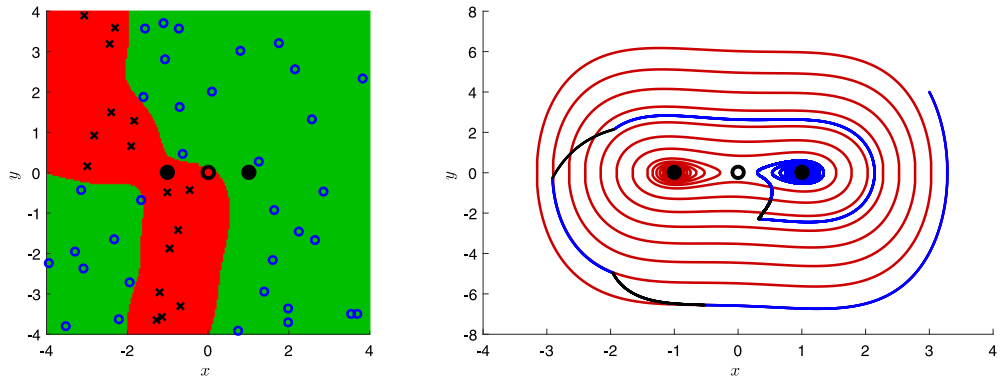

Fig. 2. Duffing Oscillator ($\delta = 0.1$): Solid (resp., open) black circles represent $\mathbf{x}^{s1}$, $\mathbf{x}^{s2}$, (resp., $\mathbf{x}^{u}$). In the left panel, open blue circles (resp., black $\times$'s) represent elements of the set $\mathcal{X}$ where the control policy given by elements of the set $\mathcal{U}$ is OFF (resp., ON). The green (resp., red) region is where the output $\pi(\mathbf{x}(t))$ of the binary classifier is OFF (resp., ON). In the right panel, the trajectory starts in the region of attraction of $\mathbf{x}^{s1}$, and converges to $\mathbf{x}^{s2}$ (resp., $\mathbf{x}^{s1}$) with (resp., without) control. When the control policy is ON (resp., OFF), the trajectory is plotted in black (resp., blue). The uncontrolled trajectory is plotted in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

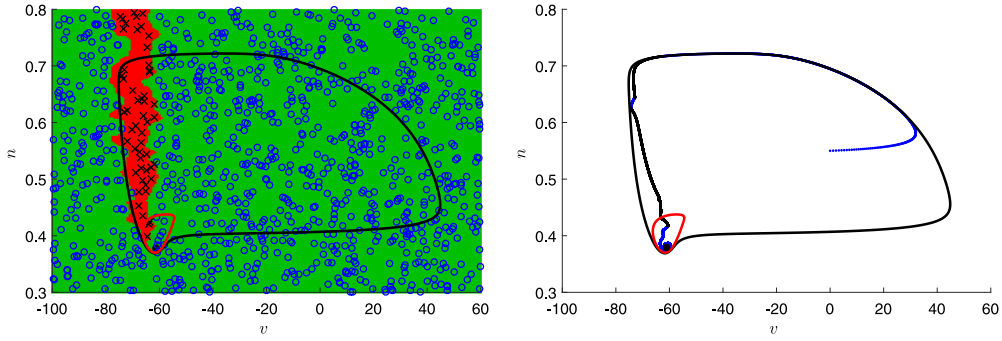

Fig. 3. Reduced Hodgkin–Huxley model: The black and red curves are $\mathbf{x}^{s1}(t)$ and $\mathbf{x}^{u}(t)$, respectively. The black point in the bottom left corner of figure panels is $\mathbf{x}^{s2}$. In the left panel, small black circles (resp., black $\times$'s) represent elements of the set $\mathcal{X}$ where the control policy given by elements of the set $\mathcal{U}$ is OFF (resp., ON). Green (resp., red) regions are where the output $\pi(\mathbf{x}(t))$ of the binary classifier is OFF (resp., ON). In the right panel, the trajectory starts outside $\mathbf{x}^{u}(t)$, and converges to $\mathbf{x}^{s2}$. When the control policy is ON (resp., OFF), the trajectory is plotted in black (resp., blue) color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The ON (resp., OFF) state of the control policy corresponds to a value of $u_c = 15$ (resp., 0). We sample $N = 1000$ points for generating the sets $\mathcal{X}$, $\mathcal{U}$, and choose $\Delta t = 0.001$, and $\tau = 0.001$. Because the two state variables v, n scale differently, it is important to normalize them for calculating the reward function. This is also important for the binary classifier to work effectively, since it is based on the Euclidean norm. To do this, we subtract from each element of the set $\mathcal{X}$ the mean of the set and then divide each element by the variance of the set. We subtract the same mean and divide by the same variance from the state $\mathbf{x}(t)$ that goes in calculating the reward function and also the binary classifier.

3.2.2. Results

The generated control policy is shown in the left panel of Fig. 3. As shown in this figure, the learning algorithm indicates that it is better to have control ON in the left part of the state space in order to maximize the reward function. In all other regions, the learning algorithm indicates that the control policy should be OFF. Since the control policy is ON in only a small region of the state space, we need to sample 1000 points to accurately determine this region. A controlled trajectory using this policy is shown in the right panel of Fig. 3. The learning algorithm is able to converge the trajectory to the stable fixed point $\mathbf{x}^{s2}$ by bypassing the unstable periodic orbit $\mathbf{x}^{u}(t)$. The algorithm generates an energy efficient control policy as the policy is OFF 23.81% of the time it takes for the algorithm to drive the trajectory inside $\mathbf{x}^{u}(t)$. We investigate the robustness of our learning algorithm by testing it on 1000 randomly generated initial conditions, and in all the 1000 cases, the algorithm is able to converge the trajectories to $\mathbf{x}^{s2}$, achieving 100% effectiveness. Note that the learning algorithm has no information about the periodic orbits and fixed points of the system; it only works to maximize the reward function.

3.2.3. Comparison with fully actuated control

To further demonstrate energy efficiency of our learning algorithm, we compare it with a fully actuated feedback control given as

$$\frac{d}{dt}\mathbf{x}(t) = F(\mathbf{x}(t)) + U(\mathbf{x}(t)), \qquad \mathbf{x}(t) \in R^n, \tag{5}$$

$$U(\mathbf{x}(t)) = -F(\mathbf{x}(t)) - 0.2\left(\mathbf{x}(t) - \mathbf{x}^{s2}\right), \tag{6}$$

which also converges the trajectory to the stable fixed point in the same time frame as our learning algorithm. However the energy required by this algorithm calculated as $\int_0^t \|U(\mathbf{x}(t))\|_2^2 dt$ comes out to be more than 3 orders of magnitude larger compared to the energy taken by the control obtained from our learning algorithm. This is because our learning algorithm takes advantage of the natural dynamics of the system to drive the trajectory close to the desired point, and turns the control ON only for a short amount of time when it is really needed. In contrast, the feedback based control is ON the whole time, even when the trajectory reaches inside the unstable periodic orbit.

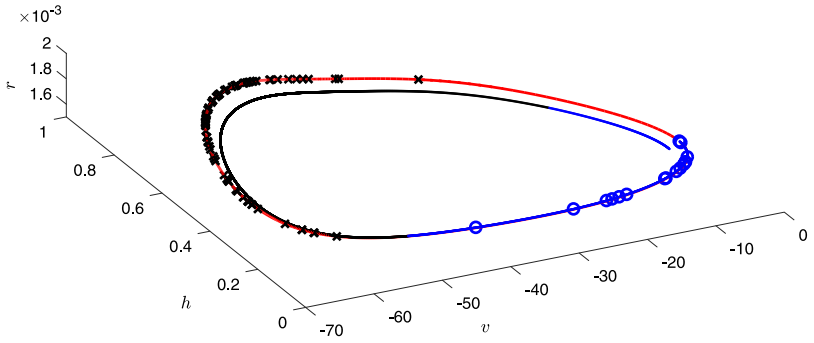

Fig. 4. Thalamic neuron model: Red curve is the stable limit cycle. Small blue circles (resp., black ×'s) represent elements of set $\mathcal{X}$ where control policy given by elements of set $\mathcal{U}$ is $-u_1$ (resp., u_1). The controlled trajectory is plotted in blue (where $\pi(\mathbf{x}(t)) = -u_1$) and black (where $\pi(\mathbf{x}(t)) = u_1$). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4. Phase control of an oscillator

In this section, we use our second algorithm to control a class of underactuated dynamical systems having a stable limit cycle solution $\mathbf{x}^s(t)$. We seek to maximally increase or decrease the phase of the limit cycle solution by using a bang-bang type control input. The motivation behind such a control objective comes from controlling neurons, where one might want a neuron to spike as quickly as possible subject to a constraint on the magnitude of the allowed input current; this constraint can be due to hardware limitations and/or concern that large inputs might cause tissue damage. Thus, instead of thinking in terms of maximally increasing the phase, one can instead think in terms of maximally decreasing the neuron's spike time.

4.1. Model

To demonstrate our algorithm, we consider the 3-dimensional thalamic neuron model [25] for the oscillatory behavior of neurons in the thalamus:

$$\dot{v} = \frac{-I_L(v) - I_{Na}(v,h) - I_K(v,h) - I_T(v,r) + I_b}{C_m} + \pi(\mathbf{x}(t)), \tag{7}$$

$$\dot{h} = \frac{h_\infty(v) - h}{\tau_h(v)}, \tag{8}$$

$$\dot{r} = \frac{r_\infty(v) - r}{\tau_r(v)}, \tag{9}$$

where the state $\mathbf{x}(t)$ is the tuple (v, h, r), v is the transmembrane voltage, and h, r are the gating variables of the neuron. $\pi(\mathbf{x}(t))$ represents the applied current as the control input. For details of the rest of the parameters, see Appendix A.2. With no control input, these parameters give a stable limit cycle $\mathbf{x}^s(t)$ with period $T = 8.40$ ms shown in red in Fig. 4.

4.2. Learning algorithm

Here the control objective is to maximally decrease the spike time of the neuron, meaning we want the oscillation to end sooner than it naturally would. We set the reward function as the negative of the neuron's next spike time (the time when the transmembrane voltage $v(t)$ reaches a maximum):

$$\mathcal{R}(\mathbf{x}(t)) = -t_{spike}. \tag{10}$$

We sample 100 states randomly along the limit cycle and evolve them with both positive and negative control inputs for time $\Delta t = 0.001$, and then evolve them further with zero control input until the neuron spikes. Whichever control input attains the minimal t_{spike} (maximizes the reward function) is selected as control policy for that sampled state. We choose $\tau = 0.01$. Because the state variables v, h, r have different dynamic ranges, we normalize the set $\mathcal{X}$ and the state at every time step similar to in Section 3.2.1.

4.3. Results

The generated control policy, along with the controlled trajectory, is shown in Fig. 4. As seen in this figure, most of the sampled states need to have a positive control in order to maximize the reward function. This is evident from the left panel of Fig. 5 which plots the corresponding control input. Because of the control, the neuron spikes (v reaches its maximum) in $t_{spike} = 7.49$ ms which is 10.82% decrease in its natural spike time of 8.40 ms. Thus our algorithm is able to achieve the control objective while keeping the controlled trajectory close to the stable limit cycle solution (see Fig. 4).

4.4. Model based control comparison

The dynamics of neural oscillations are highly nonlinear and high dimensional, which makes a model based control formulation very challenging. Phase reduction, a model reduction technique valid close to the limit cycle (see Appendix B for more details), can overcome these challenges. The neuron spike time control problem was solved as an optimal control problem in [4, 26] using phase reduction, which also resulted in a bang bang control with control input given as

$$\pi(\mathbf{x}(t)) = -\text{sign}[\mathcal{Z}(\theta)]u_1 \quad \text{for decreasing } t_{spike}, \tag{11}$$

where u_1 is the bound chosen by the user and $\mathcal{Z}(\theta)$ is the phase response curve (see Appendix B for more details) which is a periodic function of θ. Such a control works well, except when the bound u_1 is large, where the controlled trajectory can diverge far away from the limit cycle, decreasing the accuracy of phase reduction and making the control based on phase reduction ineffective. Effectiveness of such a control also relies heavily on accurate measurement of the phase response curve, which may not be possible.

We find that our supervised learning based control outputs a control input very similar to the above model based control, both shown in the left panel of Fig. 5 for $u_1 = 1$. We compute t_{spike} as a function of the bound u_1 and find that our learning based algorithm does slightly better than the model based algorithm in decreasing t_{spike} (shown in the right panel of Fig. 5). Both controls are able to decrease t_{spike} more as u_1 increases.

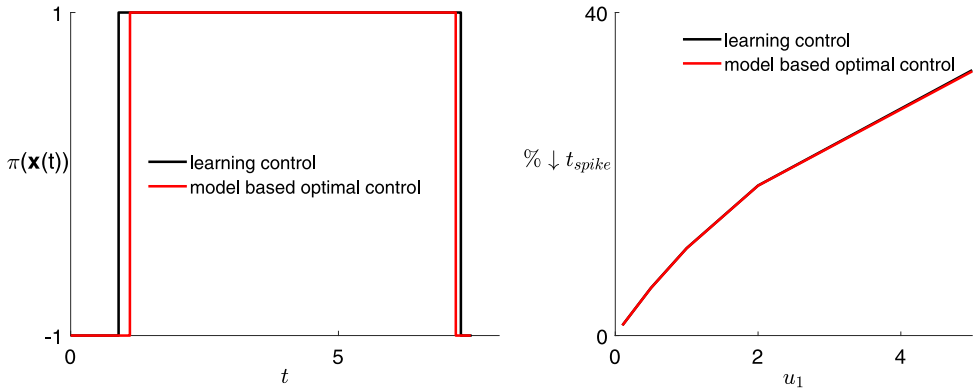

Fig. 5. Thalamic neuron model: The left panel plots the control input $\pi(\mathbf{x}(t))$ for our learning algorithm and optimal control algorithm for $u_1 = 1$. The right panel shows the % decrease in t_{spike} as a function of u_1.

5. Desynchronization of a population of coupled oscillators

Populations of coupled oscillators are ubiquitous in applications from physics, chemistry, biology, and engineering [23,27–29]. The collective behavior of such oscillators varies, and includes synchronization, desynchronization, and clustering in various scenarios. Pathological synchronization of neural oscillations in the thalamus and the subthalamic nucleus (STN) brain region is hypothesized to be one of the causes of motor symptoms for essential and parkinsonian tremor, respectively [30,31]. Deep brain stimulation (DBS), an FDA approved treatment, has proven to alleviate these symptoms [32,33] by stimulating the thalamus or the STN brain regions with a high frequency, (relatively) high energy pulsatile waveform, which has been hypothesized to desynchronize the synchronized neurons; see, e.g., [34,35]. This has motivated researchers to come up with efficient model dependent control techniques [5,36–39] to desynchronize the neural oscillations, but also consume less energy, thus prolonging the battery life of the stimulator and preventing tissue damage or side effects caused by the pulsatile stimuli.

5.1. Model

Inspired by such treatment of parkinsonian and essential tremor, we employ our algorithm to desynchronize an initially synchronized population of M coupled thalamic neural oscillations. We consider the 3-dimensional thalamic neuron model [25] for each individual oscillator with added all-to-all electrotonic coupling:

$$\dot{v}_i = \frac{-I_L(v_i) - I_{Na}(v_i,h_i) - I_K(v_i,h_i) - I_T(v_i,r_i) + I_b + \frac{1}{N}\sum_{j=1}^{M}\alpha_{ij}(v_j - v_i)}{C_m} + \pi(\mathbf{x}(t)), \tag{12}$$

$$\dot{h}_i = \frac{h_\infty(v_i) - h_i}{\tau_h(v_i)}, \tag{13}$$

$$\dot{r}_i = \frac{r_\infty(v_i) - r_i}{\tau_r(v_i)}, \tag{14}$$

where $\mathbf{x}(t)$ represents the full state ($3 \times M$ dimensional) of the oscillator population. Here, $i = 1, \ldots, M$, where M is the total number of oscillators in the neuron population. v_i is the transmembrane voltage, and h_i, r_i are the gating variables of the ith neural oscillator. α_{ij} is the coupling strength between oscillators i and j, which are assumed to be electrotonically coupled [40] with $\alpha_{ij} = 0.01$ for $j \neq i$ and $\alpha_{ii} = 0$ for all i. $\pi(\mathbf{x}(t))$ represents the applied current as the control policy. For details of the rest of the parameters, see Appendix A.2. Note that the same control input $\pi(\mathbf{x}(t))$ is applied to all of the oscillators. With no control input, these parameters give a synchronized oscillator population with period $T = 8.40$ ms.

5.2. Learning algorithm

We index the individual neural oscillators in the order in which they spike, thus neuron 1 spikes first and neuron M spikes last. We set the reward function as the absolute value of spike time difference of neuron 1 and M:

$$\mathcal{R}(\mathbf{x}(t)) = |t_{spike1} - t_{spikeM}|. \tag{15}$$

Since the oscillator population is initially synchronized, this reward is initially a small positive number as all neurons spike very close to each other. We aim to desynchronize the population by maximally increasing this reward function. We consider $M = 51$ oscillators in the synchronized population and sample 51 states along the synchronized oscillation. Since the state of the oscillator population is very large ($3 \times M$), we take the mean across the population to reduce the dimension of our set $\mathcal{X}$. The ith element of the set $\mathcal{X}$ is given as

$$\mathcal{X}_i = \frac{\sum_{j=1}^{M}(v_j, h_j, r_j)}{M}. \tag{16}$$

We evolve the oscillator population with both positive and negative control inputs for time $\Delta t = 0.001$, and evolve them further with zero control input until all neurons in the population spike. Whichever policy attains the maximum reward function is selected for that sampled state $\mathcal{X}_i$.

The binary classifier takes as input the full high dimensional state of the oscillator population. It then computes the mean of the state across the population and compares it with the sampled mean states to output a control policy $\pi(\mathbf{x}(t))$. Because the mean of the states v_j, h_j, r_j scale differently, it is important to normalize them for the binary classifier to work effectively, since it is based on the Euclidean norm. Thus we normalize the set $\mathcal{X}$ and the mean state at every time step similar to in Section 3.2.1. We choose $\tau = 0.01$.

5.3. Results

The generated control policy shown in Fig. 6 gives a positive control input in the bottom left region of oscillation and a negative control in the top right region of the oscillation. The same figure also plots a model based control policy discussed below. Fig. 7 shows the results of desynchronization of a thalamic neuron population by our learning algorithm. As shown in both the left

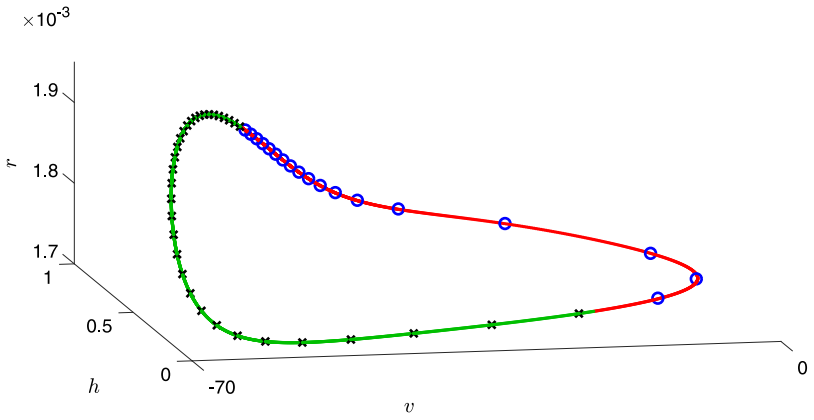

Fig. 6. Thalamic synchronized population oscillation: The closed curve is the synchronized oscillation. Small blue circles (resp., black ×'s) represent elements of the set $\mathcal{X}$ where the control policy given by elements of the set $\mathcal{U}$ is $-u_1$ (resp., u_1). The oscillation is plotted in red (resp., green) where $\mathcal{Z}'(\theta)$ is negative (resp., positive). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

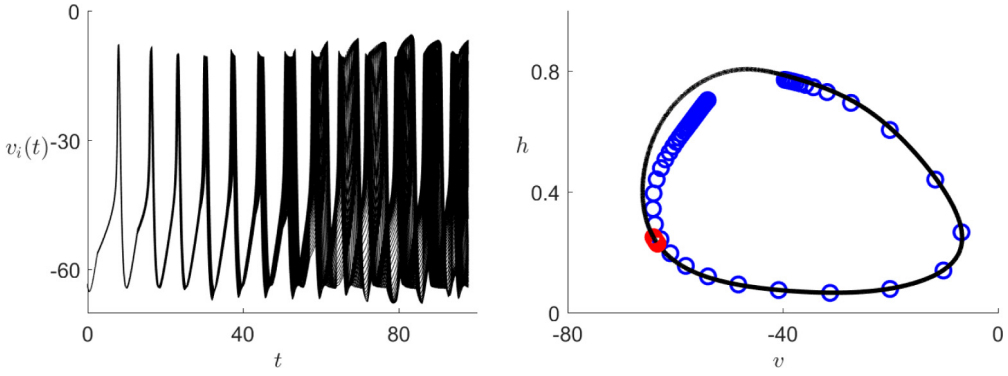

Fig. 7. Desynchronization of thalamic neuron population: Left panel plots the state v_i for $i = 1, \ldots, 51$ neurons as a function of time. Right panel plots the initially synchronized (resp. final desynchronized) neurons as small red (resp., blue) circles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and right panels of the figure, the control policy from our learning algorithm is able to desynchronize an initially synchronized thalamic neuron population in about 90 ms while keeping the oscillators close to the initially synchronized oscillation. It may seem that the population is largely synchronized from the right panel of Fig. 7 but that is not the case. Since the oscillators spend most of their time near the top of the limit cycle, one naturally observes more of them near the top of the limit cycle even though they are evenly spread out in time (and not space). This becomes clear from the left panel of Fig. 7.

5.4. Model based validation of control policy

Here we analyze why the policy predicted by our learning algorithm works. Consider an oscillator population comprised of just 2 oscillators whose dynamics evolve according to phase reduction as

$$\dot{\theta}_1 = \omega + \mathcal{Z}(\theta_1)u(t), \tag{17}$$

$$\dot{\theta}_2 = \omega + \mathcal{Z}(\theta_2)u(t). \tag{18}$$

The dynamics of their phase difference $\phi = \theta_1 - \theta_2$ can be written as (cf, [37])

$$\dot{\phi} = \mathcal{Z}'(\theta)u(t)\phi + \mathcal{O}(\phi^3), \tag{19}$$

where $\theta = 0.5(\theta_1 + \theta_2)$ is the mean of the two oscillators' phases, and $\mathcal{Z}'(\theta)$ is the derivative of the phase response curve with respect to θ. If the oscillators are synchronized then their phase difference $\phi \approx 0$, thus higher order term in Eq. (19) can be ignored and the equation can be rewritten as

$$\dot{\phi} = \mathcal{Z}'(\theta)u(t)\phi. \tag{20}$$

To desynchronize these two synchronized oscillators the coefficient of ϕ in the above equation should be positive. This can be achieved if $u(t)$ is of same sign as $Z'(\theta)$. This is exactly what our policy predicts, as is shown in Fig. 6. The policy predicts the control to be positive in the region of oscillation where $\mathcal{Z}'(\theta)$ is positive, and it predicts the control to be negative in the region of oscillation where $\mathcal{Z}'(\theta)$ is negative, thus explaining why our algorithm is able to desynchronize the oscillator population.

6. Stabilizing an unstable fixed point

In this section we apply our second learning algorithm to stabilize an unstable fixed point of an underactuated dynamical system. This control objective is one of the oldest studied control theory problems that is employed in several fields including robotics, electrochemical systems, and treatment of cardiac arrhythmias [3,41,42]. To demonstrate this, we consider the Lorenz system [43] given as:

$$\dot{x} = \sigma(y - x) + \pi(\mathbf{x}(t)), \tag{21}$$

$$\dot{y} = rx - y - xz, \tag{22}$$

$$\dot{z} = xy - bz. \tag{23}$$

In the absence of control input with parameters $\sigma = 10$, $b = 8/3$, $r = 1.5$, the system is bistable with $\mathbf{x}^{s1} = (-1.15, -1.15, 0.5)$, $\mathbf{x}^{u} = (0, 0, 0)$, and $\mathbf{x}^{s2} = (1.15, 1.15, 0.5)$, all shown in Fig. 8.

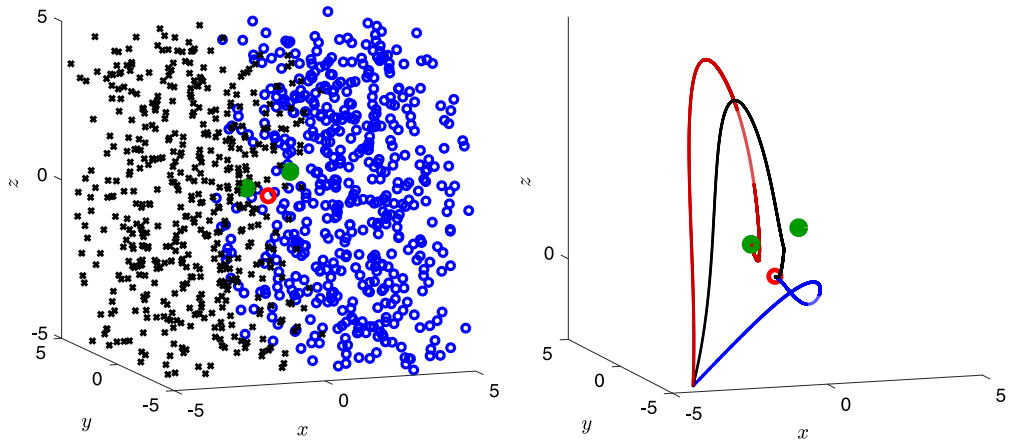

Fig. 8. Lorenz system: Solid green (resp., open red) circles represent $\mathbf{x}^{s1}$, $\mathbf{x}^{s2}$, (resp., $\mathbf{x}^{u}$). In the left panel, open blue circles (resp., black $\times$'s) represent elements of the set $\mathcal{X}$ where the control policy given by elements of the set $\mathcal{U}$ is -5 (resp., 5). In the right panel, the uncontrolled trajectory plotted in red converges to $\mathbf{x}^{s1}$, and the supervised learning (resp., Lyapunov) based control trajectory plotted in black (resp., blue) converges to $\mathbf{x}^{u}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6.1. Learning algorithm

The control objective is for a trajectory to converge to the unstable fixed point $\mathbf{x}^u$ starting anywhere in the state space. We choose our reward function to be the negative of the Euclidean distance between current state and the desired unstable fixed point

$$\mathcal{R}(\mathbf{x}(t)) = -\|\mathbf{x}(t) - \mathbf{x}^u\|. \tag{24}$$

Thus the control will make the trajectory converge towards the desired fixed point while increasing the reward to 0. To converge to $\mathbf{x}^u$ starting anywhere in the state space, we use our learning algorithm to generate a control policy. We take $u_1 = -u_2 = 5$, and sample $N = 1000$ points for generating the sets $\mathcal{X}$, $\mathcal{U}$. We choose $\Delta t = 0.001$ and take the binary classifier parameter $\tau = 5$.

6.2. Results

The generated control policy is shown in the left panel of Fig. 8. Blue open circles (resp., black $\times$'s) represent elements of the set $\mathcal{X}$ where the control policy given by elements of the set $\mathcal{U}$ is -5 (resp., 5). A controlled trajectory using our learning algorithm and an uncontrolled trajectory starting from same $\mathbf{x}(0)$ are shown in the right panel of Fig. 8. The learning algorithm converges the trajectory to $\mathbf{x}^u$, whereas the uncontrolled trajectory converges to $\mathbf{x}^{s1}$. In doing so, the learning based control consumes 150 units of control energy ($\int_0^6 \pi(\mathbf{x}(t))^2\, dt$). We investigate the robustness of our learning algorithm by testing it on 1000 randomly generated initial conditions, and in all the 1000 cases, the learning based control algorithm is able to converge the trajectories within a ball of radius 0.09 units centered at $\mathbf{x}^u$, achieving 100% effectiveness.

6.3. Comparison with Lyapunov based control

To demonstrate energy efficiency of our learning algorithm, we compare it with Lyapunov-based control to stabilize $\mathbf{x}^u$. Consider the following positive definite Lyapunov function

$$V(t) = \frac{1}{2}x(t)^2 + \frac{1}{2}y(t)^2 + \frac{1}{2}z(t)^2. \tag{25}$$

Its time derivative is given as

$$\dot{V}(t) = -2\sigma x(t)^2 - 2y(t)^2 - 2bz(t)^2 + 2x(t)\,(u(t) + (\sigma + r)y(t))\,, \tag{26}$$

where $u(t)$ takes the place of $\pi(\mathbf{x}(t))$ in Eq. (21). Then by taking $u(t) = -(\sigma + r)y(t)$, one gets a negative definite time derivative of the Lyapunov function. Thus by the Lyapunov theorem, this control asymptotically stabilizes the unstable fixed point $\mathbf{x}^u$. The control trajectory based on this control is plotted in blue in the right panel of Fig. 8. As seen in the figure, the Lyapunov-based control is able to converge the trajectory towards the unstable fixed point as well. But in doing so, it consumes 1176.8 units of control energy ($\int_0^6 u(t)^2 dt$), which is almost 8 times the energy consumed by our learning based control. This is partly because our learning based control uses the inherent system dynamics to control the trajectory, as the controlled trajectory seems to stay close to the uncontrolled trajectory. In contrast, the Lyapunov based control drives the trajectory far away before it converges to $\mathbf{x}^u$, thus it ends up consuming much more energy.

7. Robustness to noise

We have demonstrated the effectiveness of our algorithms in several scenarios in which the algorithms were based on data generated from a deterministic dynamical model. However, real data measured from an experimental setup will be noisy. In order for our algorithms to work in an experimental setup it is imperative to investigate their performance when the data is corrupted with noise. We do that by considering the Duffing oscillator in the bistable parameter regime from Section 3.1. The control objective is still for the trajectory to converge to $\mathbf{x}^{s2}$ starting anywhere in the state space.

7.1. Learning algorithm

To replicate the effect of noise in an experimental setup, we use exactly the same parameters as before to generate the sets $\mathcal{X}$, $\mathcal{U}$ and corrupt the set $\mathcal{X}$ by adding Gaussian white noise with mean 0 and standard deviation σ, resulting in the set $\widetilde{\mathcal{X}}$. Thus each element in the dataset will be offset from its true value. We also add Gaussian white noise with the same mean and standard deviation to the state $\mathbf{x}(t)$ resulting in $\widetilde{\mathbf{x}}(t)$, which the binary

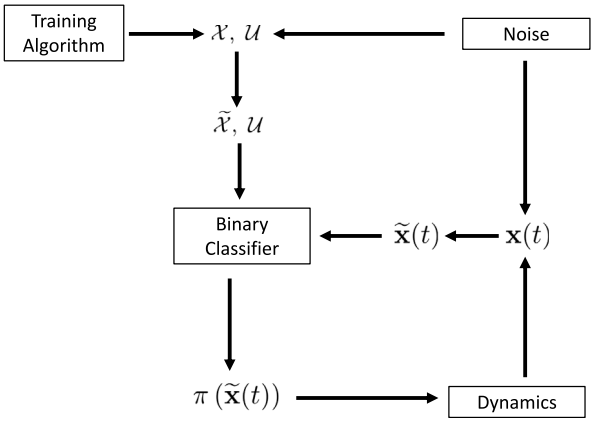

Fig. 9. Flowchart of the Supervised Learning Algorithm with added noise.

classifier takes as input at every time step. This accounts for the noise in estimation of the state by the classifier in a real system. The flowchart from Fig. 1 with added noise is modified and shown in Fig. 9.

7.2. Results

The generated control policy corrupted with noise of standard deviation $\sigma = 0.2$ is shown in the left panel of Fig. 10. Blue open circles (resp., black $\times$'s) represent elements of the noise corrupted set $\widetilde{\mathcal{X}}$ where the control policy given by elements of the set $\mathcal{U}$ is OFF (resp., ON). The green (resp., red) region is where output $\pi(\widetilde{\mathbf{x}}(t))$ of the binary classifier is OFF (resp., ON). The elements of the original set $\mathcal{X}$ are plotted in white to show the shifting of the elements due to the noise. Besides shifting the elements, the addition of white noise blurs the decision boundary between the ON and OFF policy region . A controlled and an uncontrolled trajectory starting from the same $\mathbf{x}(0)$ is shown in the right panel of Fig. 10. As can be seen in this figure, the control algorithm converges the trajectory to $\mathbf{x}^{s2}$ even though it has been corrupted by adding noise both to the training dataset and to the input of the binary classifier. On the other hand, the uncontrolled trajectory converges to $\mathbf{x}^{s1}$. Note that the controlled trajectory follows a different route when compared to the noiseless case from Fig. 2. This is because the noise distorts the decision boundary between ON and OFF states of the policy, resulting in a slightly different path.

7.3. Robustness and noise intensity

We first investigate the robustness of our learning algorithm in the presence of noise by considering noise with intensity $\sigma = 0.2$ and testing it on 1000 randomly generated initial conditions; in all 1000 cases the control algorithm is able to converge the trajectories to $\mathbf{x}^{s2}$, achieving 100% effectiveness. It is natural to ask if effectiveness decreases down from 100% as one increases the noise intensity beyond 0.2. To answer this question, we tested robustness of our algorithm by simulating 1000 randomly generated initial conditions and evolving them under our control policy for various levels of noise intensity σ and bandwidth parameter τ. We found that our algorithm can still achieve very high effectiveness for higher σ, if we adjust the bandwidth parameter τ.

To understand why this is the case, see Fig. 11, where we plot two policies for noise intensity $\sigma = 0.3$ with bandwidth parameter $\tau = 0.1$ and 0.8. Since noise randomly shifts elements of our data set $\mathcal{X}$, it distorts the decision boundary when compared to the decision boundary in the noiseless case (see the left panel of Fig. 2 for a comparison). For small τ values, this leads to significant segmentation of the policy as the output of the binary classifier can vary rapidly with a small change in the input state. This is evident from the left panel of Fig. 11. On the other hand, if we increase the bandwidth parameter, segmentation is significantly reduced, as is evident from the right panel of the same figure. A higher bandwidth parameter leads to less segmentation, because the binary classifier effectively takes into account more neighboring data points in making its decision (equation (2)). In other words, when these points are randomly shifted by the addition of Gaussian white noise, the weighted average from a larger number of data points produces smoother transition between regions, and thus leads to reduced segmentation. On the other hand, if the bandwidth parameter was too small, the binary classifier's decision would be based on only a very few neighboring data points, which, when randomly shifted by Gaussian white noise, can produce a highly segmented and non-smooth policy.

Table 1
Effectiveness of our algorithm as a function of noise intensity σ and bandwidth parameter τ for the Duffing oscillator.

σ	τ				
	0.1	0.4	0.8	1.2	1.6
0.2	100%	100%	100%	100%	100%
0.3	100%	100%	100%	100%	83%
0.4	0%	89.9%	100%	93.2%	64.1%
0.5	0%	78.6%	90.8%	100%	68.7%
0.6	0%	0%	0%	9%	95.6%

But robustness does not keep increasing with increasing τ. A τ value which is too high can actually reduce effectiveness, as depicted in Table 1, because, as previously mentioned, high τ values can wrongly extend the decision boundary in favor of a cluster of data points which is too large. Thus, a carefully chosen moderate value of τ is most effective in assuring robustness in the presence of noise. Another observation from our robustness studies for various levels of noise intensity σ and τ depicted in Table 1 is that we need an increasing value of τ to achieve 100% effectiveness as we increase the noise intensity σ. This is also intuitive to understand, as for a given value of τ, one expects more segmentation in the control policy with increasing σ. Thus, the locally weighted nature of our binary classifier, together with the flexibility to adjust its bandwidth allows our algorithm to be effective in the presence of noise.

While it may not be possible to achieve 100% effectiveness in all control applications, we believe that this example illustrates key considerations in characterizing the robustness of our algorithms to noise. Another possible consideration for achieving high effectiveness could be choosing a higher sample size N, however this would increase the computational cost of our algorithm. Since we are able to achieve very high effectiveness by adjusting the bandwidth parameter while keeping the computational cost the same, we favor this over adjusting N in characterizing the robustness of our algorithms to noise.

8. Conclusion

In this article we have developed two novel supervised learning algorithms to control a range of underactuated dynamical systems. The algorithms output a bang bang (binary) control input to achieve the desired control objectives which maximize a reward function. A simple yet intelligent structure allows the algorithms to be energy efficient as they learn to take advantage of the inherent dynamics. We demonstrated the versatility of our

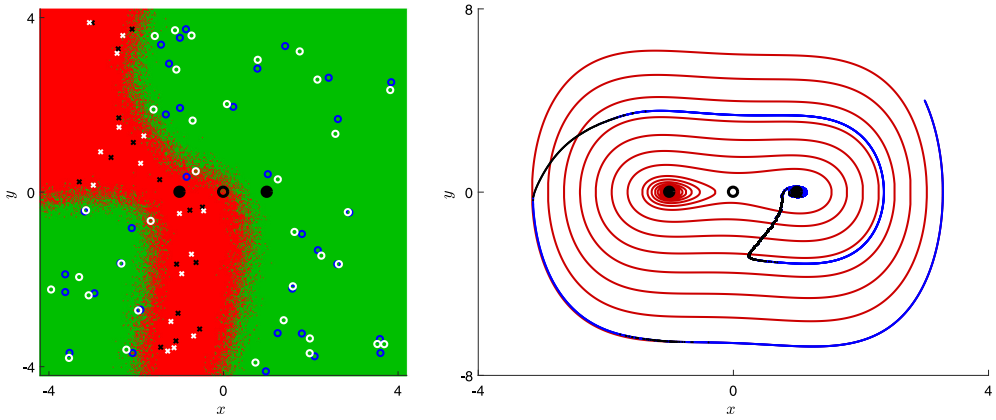

Fig. 10. Duffing Oscillator with noise ($\delta = 0.1$): Solid (resp., open) black circles represent $\mathbf{x}^{s1}$, $\mathbf{x}^{s2}$, (resp., $\mathbf{x}^{u}$). In the left panel, open blue circles (resp., black $\times$'s) represent elements of the set $\widetilde{\mathcal{X}}$ where the control policy given by elements of the set $\mathcal{U}$ is OFF (resp., ON). The green (resp., red) region is where the output $\pi(\widetilde{\mathbf{x}}(t))$ of the binary classifier is OFF (resp., ON). The elements of the original set $\mathcal{X}$ are plotted in white. In the right panel, the trajectory starts in the region of attraction of $\mathbf{x}^{s1}$, and converges to $\mathbf{x}^{s2}$ (resp., $\mathbf{x}^{s1}$) with (resp., without) control. When the control policy is ON (resp., OFF), the trajectory is plotted in black (resp., blue). The uncontrolled trajectory is plotted in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

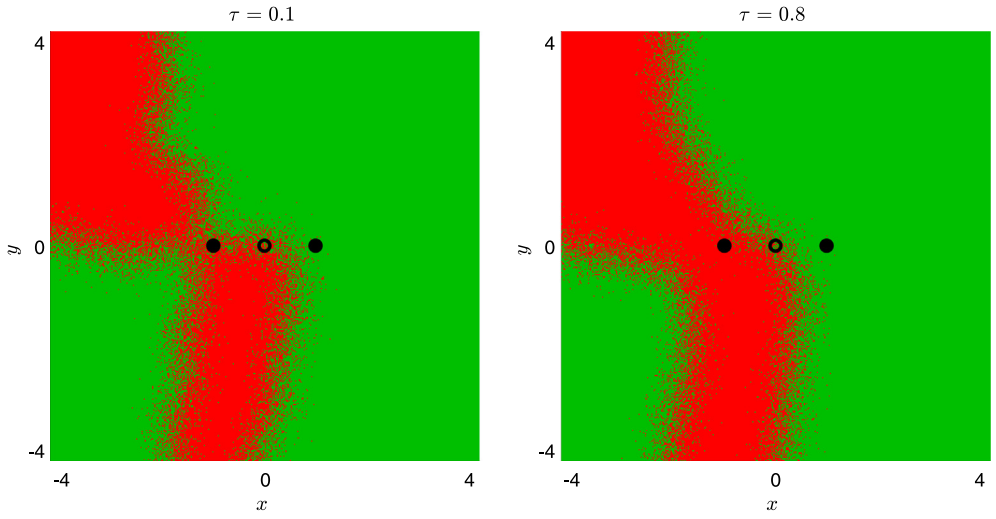

Fig. 11. Duffing Oscillator policy with noise intensity $\sigma = 0.3$.

algorithms by applying them to a diverse range of applications including: switching between bistable states, changing the phase of an oscillator, desynchronizing a population of synchronized coupled oscillators, and stabilizing an unstable fixed point. For most of these applications we were able to reason why our algorithms work by using traditional dynamical systems and control theory. We also compared our algorithms to some traditional nonlinear model control algorithms and showed that our algorithms work better. We also carried out a robustness study to demonstrate the effectiveness of our algorithms even with noisy data.

We simulated various dynamical models to generate data for training our supervised learning algorithms. In an experimental setting such an algorithm can be similarly implemented by stimulating the system with binary control inputs at different states of the system and determining which control input works best for the different sampled states, and ultimately using that information in constructing a binary classifier. The data generated from an experimental setting might be corrupted with noise, thus to demonstrate the potential of our algorithm in a real setting we showed that our algorithm works even in the presence of noise. The unique structure of our binary classifier comes to the rescue by negating the adverse effects of noise. Note that having an additive control input does not restrict our algorithms. Since the algorithms work by maximizing the reward function, the structure of the control input coming into the dynamics does not matter. In the future, we plan to explore how to modify our algorithm if some of the states are not observable, and how to adapt our algorithm for very high dimensional dynamical systems.

CRediT authorship contribution statement

Bharat Monga: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing - original draft, Visualization, Investigation. **Jeff Moehlis:** Supervision, Writing - review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by National Science Foundation Grant No. NSF-1635542.

Appendix A. Models

In this appendix, we give details of the mathematical models used in this article.

A.1. Reduced Hodgkin–Huxley model

Here we give the reduced Hodgkin–Huxley model [22–24] used in Section 3.2:

$$\begin{aligned}\dot{v} &= \big(I - g_{Na}(m_\infty(v))^3(0.8-n)(v-v_{Na}) - g_K n^4(v-v_K) \\ &\quad - g_L(v-v_L)\big)/c + \pi(\mathbf{x}(t)), \\ \dot{n} &= a_n(v)(1-n) - b_n(v)n,\end{aligned}$$

where v is the trans-membrane voltage, and n is the gating variable. I is the baseline current, which we take as 6.69 μA/cm^2, and $\pi(\mathbf{x}(t))$ represents the applied control current.

$$\begin{aligned} a_n(v) &= 0.01(v+55)/(1-\exp(-(v+55)/10)), \\ b_n(v) &= 0.125\exp(-(v+65)/80), \\ a_m(v) &= 0.1(v+40)/(1-\exp(-(v+40)/10)), \\ b_m(v) &= 4\exp(-(v+65)/18), \\ m_\infty(v) &= a_m(v)/(a_m(v)+b_m(v)), \\ c &= 1,\ g_L = 0.3,\ g_{Na} = 120,\ v_{Na} = 50, \\ g_K &= 36\ ,\ v_K = -77,\ v_L = -54.4\ I = 20. \end{aligned}$$

A.2. Thalamic neuron model

The thalamic neuron model is given as

$$\begin{aligned}\dot{v} &= \frac{-I_L(v) - I_{Na}(v,h) - I_K(v,h) - I_T(v,r) + I_b}{C_m} + \pi(\mathbf{x}(t)), \\ \dot{h} &= \frac{h_\infty(v) - h}{\tau_h(v)}, \\ \dot{r} &= \frac{r_\infty(v) - r}{\tau_r(v)}.\end{aligned}$$

where

$$\begin{aligned} h_\infty(v) &= 1/(1+\exp((v+41)/4)), \\ r_\infty(v) &= 1/(1+\exp((v+84)/4)), \\ \alpha_h(v) &= 0.128\exp(-(v+46)/18), \\ \beta_h(v) &= 4/(1+\exp(-(v+23)/5)), \\ \tau_h(v) &= 1/(\alpha_h+\beta_h), \\ \tau_r(v) &= (28+\exp(-(v+25)/10.5)), \\ m_\infty(v) &= 1/(1+\exp(-(v+37)/7)), \\ p_\infty(v) &= 1/(1+\exp(-(v+60)/6.2)), \\ I_L(v) &= g_L(v-e_L), \\ I_{Na}(v,h) &= g_{Na}({m_\infty}^3)h(v-e_{Na}), \\ I_K(v,h) &= g_K((0.75(1-h))^4)(v-e_K), \\ I_T(v,r) &= g_T(p_\infty^2)r(v-e_T), \end{aligned}$$

$$\begin{aligned} &C_m = 1, \quad g_L = 0.05, \quad e_L = -70, \quad g_{Na} = 3, \quad e_{Na} = 50, \\ &g_K = 5, \quad e_K = -90, \quad g_T = 5, \quad e_T = 0, \quad I_b = 5. \end{aligned}$$

Appendix B. Phase reduction

Phase reduction is a classical technique to describe dynamics near a limit cycle. It works by reducing the dimensionality of a dynamical system to a single phase variable θ [28,29]. Consider a general n-dimensional dynamical system given by

$$\frac{d\mathbf{x}(t)}{dt} = F(\mathbf{x}(t)), \qquad \mathbf{x}(t) \in \mathbb{R}^n, \qquad (n \geq 2).$$

Suppose this system has a stable periodic orbit $\mathbf{x}^s(t)$ with period T. For each point $\mathbf{x}^*$ in the basin of attraction of the periodic orbit, there exists a corresponding phase $\theta(\mathbf{x}^*)$ such that

$$\lim_{t\to\infty} \left| \mathbf{x}(t) - \mathbf{x}^s\left(t + \frac{T}{2\pi}\,\theta(\mathbf{x}^*)\right)\right| = 0,$$

where $\mathbf{x}(t)$ is the flow of the initial point $\mathbf{x}^*$ under the given vector field. The function $\theta(\mathbf{x})$ is called the *asymptotic phase* of $\mathbf{x}$, and takes values in $[0, 2\pi)$. For neuroscience applications, we typically take $\theta = 0$ to correspond to the neuron firing an action potential. *Isochrons* are level sets of this phase function, and it is typical to define isochrons so that the phase of a trajectory advances linearly in time both on and off the limit cycle, which implies that

$$\frac{d\theta}{dt} = \frac{2\pi}{T} \equiv \omega$$

in the entire basin of attraction of the limit cycle. Now consider our underactuated system given by Eq. (1). Phase reduction can be used to reduce this system to a one-dimensional system given by [4,44–46]:

$$\dot{\theta} = \omega + \mathcal{Z}(\theta)\pi(\mathbf{x}(t)).$$

Here $\mathcal{Z}(\theta)$ is the first component of the gradient of phase variable θ evaluated on the periodic orbit and is referred to as the *(infinitesimal) phase response curve (PRC)*. It quantifies the effect of an infinitesimal control input on the phase of a limit cycle.

References

[1] I. Fantoni, R. Lozano, Non-linear Control for Underactuated Mechanical Systems, Springer-Verlag London, 2002, http://dx.doi.org/10.1007/978-1-4471-0177-2.

[2] M. Reyhanoglu, A. van der Schaft, N.H. Mcclamroch, I. Kolmanovsky, Dynamics and control of a class of underactuated mechanical systems, IEEE Trans. Automat. Control 44 (9) (1999) 1663–1671, http://dx.doi.org/10.1109/9.788533.

[3] M. Spong, Energy based control of a class of underactuated mechanical systems, IFAC Proc. Vol. 29 (1) (1996) 2828–2832, http://dx.doi.org/10.1016/S1474-6670(17)58105-7, 13th World Congress of IFAC, 1996, San Francisco USA, 30 June - 5 July.

[4] B. Monga, D. Wilson, T. Matchen, J. Moehlis, Phase reduction and phase-based optimal control for biological systems: a tutorial, Biol. Cybernet. 113 (1) (2019) 11–46, http://dx.doi.org/10.1007/s00422-018-0780-z.

[5] P. Tass, A model of desynchronizing deep brain stimulation with a demand-controlled coordinated reset of neural subpopulations, Biol. Cybernet. 89 (2) (2003) 81–88, http://dx.doi.org/10.1007/s00422-003-0425-7.

[6] D. Forger, D. Paydarfar, Starting, stopping, and resetting biological oscillators: in search of optimum perturbations, J. Theoret. Biol. 230 (4) (2004) 521–532, http://dx.doi.org/10.1016/j.jtbi.2004.04.043.

[7] J. Zhang, J. Wen, A. Julius, Optimal circadian rhythm control with light input for rapid entrainment and improved vigilance, in: Proceedings of the 51st IEEE Conference on Decision and Control (CDC), IEEE, 2012, pp. 3007–3012, http://dx.doi.org/10.1109/CDC.2012.6426226.

[8] Z.-S. Hou, Z. Wang, From model-based control to data-driven control: Survey, classification and perspective, Inform. Sci. 235 (2013) 3–35, http://dx.doi.org/10.1016/j.ins.2012.07.014, Data-based Control, Decision, Scheduling and Fault Diagnostics.

[9] B.D.O. Anderson, Failures of adaptive control theory and their resolution, Commun. Inf. Syst. 05 (1) (2005) 1–20, URL https://projecteuclid.org:443/euclid.cis/1149698471.

[10] K. Hunt, D. Sbarbaro, R. bikowski, P. Gawthrop, Neural networks for control systems—A survey, Automatica 28 (6) (1992) 1083–1112, http://dx.doi.org/10.1016/0005-1098(92)90053-I, URL http://www.sciencedirect.com/science/article/pii/000510989290053I.

[11] A.G. Barto, R.S. Sutton, C.W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, IEEE Trans. Syst. Man Cybern. SMC-13 (5) (1983) 834–846, http://dx.doi.org/10.1109/TSMC.1983.6313077.

[12] M.I. Jordan, T.M. Mitchell, Machine learning: Trends, perspectives, and prospects, Science 349 (6245) (2015) 255–260, http://dx.doi.org/10.1126/science.aaa8415, URL https://science.sciencemag.org/content/349/6245/255.

[13] E. Kaiser, J.N. Kutz, S.L. Brunton, Sparse identification of nonlinear dynamics for model predictive control in the low-data limit, Proc. R. Soc. A 474 (2219) (2018) 20180335, http://dx.doi.org/10.1098/rspa.2018.0335.
[14] A. Longtin, A. Bulsara, F. Moss, Time-interval sequences in bistable systems and the noise-induced transmission of information by sensory neurons, Phys. Rev. Lett. 67 (1991) 656–659, http://dx.doi.org/10.1103/PhysRevLett.67.656, URL https://link.aps.org/doi/10.1103/PhysRevLett.67.656.
[15] D.J. Christini, L. Glass, Introduction: Mapping and control of complex cardiac arrhythmias, Chaos 12 (3) (2002) 732–739, http://dx.doi.org/10.1063/1.1504061, arXiv:https://doi.org/10.1063/1.1504061.
[16] C. Bowden, M. Ciftan, H. Robl, Optical Bistability, Springer US, 1981, http://dx.doi.org/10.1007/978-1-4684-3941-0.
[17] R. Lefever, W. Horsthemke, Bistability in fluctuating environments. implications in tumor immunology, Bull. Math. Biol. 41 (4) (1979) 469–490, http://dx.doi.org/10.1007/BF02458325.
[18] T.S. Gardner, C.R. Cantor, J.J. Collins, Construction of a genetic toggle switch in escherichia coli, Nature 403 (2219) (2000) 339–342, http://dx.doi.org/10.1038/35002131.
[19] A. Fiasconaro, A. Ochab-Marcinek, B. Spagnolo, E. Gudowska-Nowak, Monitoring noise-resonant effects in cancer growth influenced by external fluctuations and periodic treatment, Eur. Phys. J. B 65 (3) (2008) 435–442, http://dx.doi.org/10.1140/epjb/e2008-00246-2.
[20] S. Wiggins, Introduction to Applied Nonlinear Dynamical Systems and Chaos, Springer-Verlag New York, New York, 2003, http://dx.doi.org/10.1007/b97481.
[21] T. Kanamaru, Duffing oscillator, Scholarpedia 3 (3) (2008) 6327, http://dx.doi.org/10.4249/scholarpedia.6327, revision #91210.
[22] J. Moehlis, Canards for a reduction of the Hodgkin-Huxley equations, J. Math. Biol. 52 (2) (2006) 141–153, http://dx.doi.org/10.1007/s00285-005-0347-1.
[23] J. Keener, J. Sneyd, Mathematical Physiology, Springer-Verlag New York, Inc., New York, NY, USA, 1998, http://dx.doi.org/10.1007/978-0-387-75847-3_1.
[24] A.L. Hodgkin, A.F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, J. Phys. 117 (4) (1952) 500–544, http://dx.doi.org/10.1113/jphysiol.1952.sp004764.
[25] J. Rubin, D. Terman, High frequency stimulation of the subthalamic nucleus eliminates pathological thalamic rhythmicity in a computational model, J. Comput. Neurosci. 16 (3) (2004) 211–235, http://dx.doi.org/10.1023/B:JCNS.0000025686.47117.67.
[26] A. Nabi, J. Moehlis, Time optimal control of spiking neurons, J. Math. Biol. 64 (2012) 981–1004.
[27] A. Pikovsky, M. Rosenblum, J. Kurths, Synchronization: A Universal Concept in Nonlinear Sciences, University Press, 2003, URL https://books.google.com/books?id=B1hQlwEACAAJ.
[28] Y. Kuramoto, Phase-and center-manifold reductions for large populations of coupled oscillators with application to non-locally coupled systems, Int. J. Bifurcation Chaos 7 (04) (1997) 789–805, http://dx.doi.org/10.1142/S0218127497000595.
[29] A. Winfree, Biological rhythms and the behavior of populations of coupled oscillators, J. Theoret. Biol. 16 (1) (1967) 15–42, http://dx.doi.org/10.1016/0022-5193(67)90051-3.
[30] A. Kane, W. Hutchison, M. Hodaie, A. Lozano, J. Dostrovsky, Enhanced synchronization of thalamic theta band local field potentials in patients with essential tremor, Exp. Neurol. 217 (1) (2009) 171–176, http://dx.doi.org/10.1016/j.expneurol.2009.02.005.
[31] A. Kühn, A. Tsui, T. Aziz, N. Ray, C. Brücke, A. Kupsch, G.-H. Schneider, P. Brown, Pathological synchronisation in the subthalamic nucleus of patients with Parkinson's disease relates to both bradykinesia and rigidity, Exp. Neurol. 215 (2) (2009) 380–387, http://dx.doi.org/10.1016/j.expneurol.2008.11.008.
[32] A. Benabid, S. Chabardes, J. Mitrofanis, P. Pollak, Deep brain stimulation of the subthalamic nucleus for the treatment of Parkinson's disease, Lancet Neurol. 8 (1) (2009) 67–81, http://dx.doi.org/10.1016/S1474-4422(08)70291-6.
[33] A. Benabid, P. Pollak, D. Hoffmann, C. Gervason, M. Hommel, J. Perret, J. De Rougemont, D. Gao, Long-term suppression of tremor by chronic stimulation of the ventral intermediate thalamic nucleus, Lancet 337 (8738) (1991) 403–406, http://dx.doi.org/10.1016/0140-6736(91)91175-T.
[34] C. Wilson, B. Beverlin, T. Netoff, Chaotic desynchronization as the therapeutic mechanism of deep brain stimulation, Front. Syst. Neurosci. 5 (2011) http://dx.doi.org/10.3389/fnsys.2011.00050.
[35] D. Wilson, J. Moehlis, Clustered desynchronization from high-frequency deep brain stimulation, PLoS Comput. Biol. 11 (12) (2015) e1004673, http://dx.doi.org/10.1371/journal.pcbi.1004673.
[36] A. Nabi, M. Mirzadeh, F. Gibou, J. Moehlis, Minimum energy desynchronizing control for coupled neurons, J. Comput. Neurosci. 34 (2) (2013) 259–271, http://dx.doi.org/10.1007/s10827-012-0419-3.
[37] D. Wilson, J. Moehlis, Optimal chaotic desynchronization for neural populations, SIAM J. Appl. Dyn. Syst. 13 (1) (2014) 276, http://dx.doi.org/10.1137/120901702.
[38] B. Monga, G. Froyland, J. Moehlis, Synchronizing and desynchronizing neural populations through phase distribution control, in: 2018 American Control Conference (ACC), 2018, pp. 2808–2813.
[39] B. Monga, J. Moehlis, Phase distribution control of a population of oscillators, Physica D 398 (2019) 115–129, http://dx.doi.org/10.1016/j.physd.2019.06.001.
[40] D. Johnston, S.M.-S. Wu, Foundations of Cellular Neurophysiology, MIT Press, Cambridge, MA, 1995.
[41] K. Pyragas, V. Pyragas, I.Z. Kiss, J.L. Hudson, Adaptive control of unknown unstable steady states of dynamical systems, Phys. Rev. E 70 (2004) 026215, http://dx.doi.org/10.1103/PhysRevE.70.026215.
[42] B. Monga, J. Moehlis, Optimal phase control of biological oscillators using augmented phase reduction, Biol. Cybernet. 113 (1) (2019) 161–178, http://dx.doi.org/10.1007/s00422-018-0764-z.
[43] E.N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. 20 (2) (1963) 130–141, http://dx.doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.
[44] A. Winfree, The Geometry of Biological Time, Second ed., Springer, New York, 2001, http://dx.doi.org/10.1007/978-1-4757-3484-3'.
[45] Y. Kuramoto, Chemical Oscillations, Waves, and Turbulence, Springer, Berlin, 1984.
[46] E. Brown, J. Moehlis, P. Holmes, On the phase reduction and response dynamics of neural oscillator populations, Neural Comp. 16 (2004) 673–715, http://dx.doi.org/10.1162/089976604322860668.