Reinforcement Learning of CPG-Regulated Locomotion Controller for a Soft Snake Robot

Xuan Liu, Student Member, IEEE, Cagdas D. Onal, Member, IEEE, and Jie Fu, Member, IEEE

Abstract—Intelligent control of soft robots is challenging due to the nonlinear and difficult-to-model dynamics. One promising model-free approach for soft robot control is reinforcement learning (RL). However, model-free RL methods tend to be computationally expensive and data-inefficient and may not yield natural and smooth locomotion patterns for soft robots. In this work, we develop a bioinspired design of a learning-based goal-tracking controller for a soft snake robot. The controller is composed of two modules: An RL module for learning goal-tracking behaviors given the unmodulated and stochastic dynamics of the robot, and a central pattern generator (CPG) with the Matsuoka oscillators for generating stable and diverse locomotion patterns. We theoretically investigate the maneuverability of Matsuoka CPG’s oscillation bias, frequency, and amplitude for steering control, velocity control, and sim-to-real adaptation of the soft snake robot. Based on this analysis, we proposed a composition of RL and CPG modules such that the RL module regulates the tonic inputs to the CPG system given state feedback from the robot, and the output of the CPG module is then transformed into pressure inputs to pneumatic actuators of the soft snake robot. This design allows the RL agent to naturally learn to entrain the desired locomotion patterns determined by the CPG maneuverability. We validated the optimality and robustness of the control design in both simulation and real experiments, and performed extensive comparisons with state-of-art RL methods to demonstrate the benefit of our bioinspired control design.

Index Terms—Biomimetics, deep reinforcement learning (RL), learning and adaptive systems, neural oscillator, soft robot control.

I. INTRODUCTION

Due to their flexible geometric shapes and deformable materials, soft continuum robots have great potential to perform tasks in dangerous and cluttered environments, including natural disaster relief and pipe inspection [1]. However, planning and control of such robots are challenging, as these robots have infinitely many degrees of freedom in their body links, and soft actuators with stochastic, unknown dynamics, and delayed responses.

In this work, we develop a bioinspired locomotion controller for soft robot snakes to achieve serpentine-like locomotion for set-point tracking tasks. Specifically, we consider utilizing the properties of CPGs, which consists of a special group of neural circuits that are able to generate rhythmic and nonrhythmic activities for organ contractions and body movements in animals. Such activities can be activated, modulated, and reset by neuronal signals mainly from two directions: bottom-up ascending feedback information fromafferent sensory neurons or top-down descending signals from high-level modules including mesencephalic locomotor region [2] and motor cortex [3]. [4].

In the literature, bioinspired control methods have been studied for the control design of rigid robots’ locomotion, including legged [5], [6], [7], [8], [9] and serpentine locomotion [10], [11], [12], [13], [14], [15]. The general approach is to generate motion patterns mimicking animals’ behaviors and then track these trajectories with a closed-loop control design. In [2], the authors developed a trajectory generator for a rigid salamander robot using Kuramoto CPGs and used low-level PD controllers to track the desired motion trajectories generated by the oscillator. Ryu et al. [12] established the velocity control CPG by adapting its frequency parameter with additional linear dynamics. In [14], the authors introduced a control loop that adjusts the oscillation patterns including frequency, amplitude, and phase of the oscillation for adapting to the changes in the terrain. Their results show the advantage of the Hopf oscillator on the direct access to the oscillation patterns for different locomotion purposes. In [16], the Matsuoka oscillator is combined with the amplitude modulation method to realize steering control of a rigid snake robot. However, these approaches have not provided a way to maneuver the oscillation patterns intelligently. Recent work has combined learning-based method and CPG systems for control of rigid robotic systems. Sartoretti et al. [17] proposed a decentralized approach, where each actuator of an articulated rigid snake robot is controlled independently by a neural network (NN) controller learned with an end-to-end RL algorithm. Another recent work [15] employed a spiking neural net under the regulation of reward-modulated spike-timing-dependent plasticity (R-STDP) to map visual information into wave parameters of a phase-amplitude CPG net, which generates desired oscillating patterns to locomote a rigid snake robot chasing a red ball. The similar idea of combining learning and CPG is also investigated in the legged robot. Tran et al. [9] employed a Q-learning selector to make decision on switching among different CPG patterns in a disturbance recovery task during bipedal locomotion. In [18], the authors proposed a CPG-RL method...
that directly learns the neural oscillator’s intrinsic amplitude and frequency, and coordinate the decoupled oscillator network to control the legged locomotion of a quadruped robot.

For a rigid snake robot, an existing literature [2] has introduced a model-based control design combined with CPG for motion planning. Despite the success of bioinspired control with rigid snake robots, the same control scheme may not work as desired for soft snake robots. This is because, in these approaches, the trajectories generated by CPG require high-performance low-level controllers for tracking. The tracking performance cannot be reproduced with soft snake robots due to the nonlinear, delayed, and stochastic dynamical response from the soft actuators.

To this end, we develop a bioinspired learning-based control framework for soft snake robots with two key components: To achieve intelligent and robust goal-tracking with changing goals, we use model-free RL [19], [20] to map the feedback of soft actuators and the goal location, into control commands of a CPG network. The CPG network consists of coupled Matsuoka oscillators [21]. The Matsuoka CPG network acts as a low-level motion controller to generate actuation inputs directly to the soft actuators. The two networks form a variant of cascade control with only one outer-loop, as illustrated in Fig. 1. Comparing to other neural oscillators used in [17], [15], and [18], the Matsuoka oscillator has the following special properties.

1) It is in the class of half-center [22] oscillator model that describes mutually inhibiting mechanism in a pair of neurons. Such mechanism produces alternate activities of flexors and extensors, which can be used to directly control a pair of actuators mimicking antagonistic muscles.

2) It has clear boundary conditions for the parameters such that the neurons can generate free-response oscillation when satisfying the boundary condition [23].

3) On the basis of free-response oscillation, the entrainment property [24], [25] allows the intelligent controller to autonomously regulate the oscillation pattern of the system with forced-response oscillation input.

4) It is a piece-wise linear system with local linearity in certain quadrants.

Based on the aforementioned properties of the Matsuoka oscillator, we showed that several dynamic properties of the Matsuoka oscillators can be leveraged in designing the interconnection between RL and CPG. We have proved that the steering control can be realized by modulating both the amplitudes bias and duty cycles of the neural stimuli inputs of the CPG network, and the velocity control can be realized by tuning the oscillating frequencies of the CPG net. These findings enable us to flexibly control the slithering locomotion with a CPG network given state feedback from the soft snake robot and the control objective.

This article is an extension of our preliminary work [26] that designs a learning-based set-point tracking control for soft snake robots. In comparison to [26], we make the following improvements.

1) Theoretical analysis of steering maneuverability: We analyze the property of the biased oscillation in the Matsuoka oscillator. Using describing function analysis, we show that when the tonic inputs of the Matsuoka oscillator are bounded and satisfy certain constraints, the bias of the output signal becomes linearly related to the tonic inputs. This feature makes the steering control of the snake robot easier to learn for an RL agent.

2) Free-response oscillation constraints (FOC) for sim-to-real transfer: We investigate the transient property of the Matsuoka Oscillator from free-response oscillation to forced-response oscillation. Using this property, we introduce a fixed free-response tonic input signal to help regulate the amplitude and oscillation frequency of the forced tonic inputs, which are generated by the RL policy. The new approach is referred to as free-response oscillation constrained proximal policy optimization option-critics with central pattern generator (FOC-PPOC-CPG). This approach improves the transferability of the RL control policy learned in the simulation to the real robot.

3) Improve reward density with potential field function: We newly introduce a potential-field-based reward shaping to accelerate the learning process.

4) Comprehensive sim-to-real tests and analysis: We added new experiments comparing the learning efficiency and adaptability of the policy between the proposed method and vanilla Proximal Policy Optimization (PPO) [20]. Based on the experimental results for both simulation and reality, we show that our soft snake robot equipped with a properly designed “vertebrate” (the CPG system) can be more easily controlled by the RL agent. Our approach also achieves more reliable locomotion performance under various goal-reaching locomotion tasks that are unseen during the training process.

The rest of this article is organized as follows. Section II provides an overview of the robotic system and the state space representation. Section III presents the design and configuration of the CPG network. Section IV discusses the key properties of the CPG network and the relation to the design of an artificial NN for the RL module. Section V introduces a curriculum and reward design for learning a goal-tracking locomotion controller with a soft snake robot. Section VI presents the experimental validation and evaluation of the controller in both simulated and real snake robots. Finally, Section VII concludes this article.

II. SYSTEM OVERVIEW OF THE SOFT SNAKE ROBOT

As shown in Fig. 2, our soft snake robot is a subtype of WPI-SRS series robot [27]. It consists of 4 pneumatically actuated soft links. The soft links are made of Ecoflex™ 00-30
silicone rubber. Each soft link of the robot has two air chambers mimicking antagonistic muscle (detailed structure of the soft body can be found in [27] and [28]). The links are connected through rigid bodies enclosing the electronic components that are necessary to control the snake robot. Each rigid body contains an ESP32 module (powered by a Lithium-polymer battery) for control command communication and a pair of SMC-S070C-SCG solenoid valves that control the inflation and deflation of the air chambers. Only one chamber on each link is active (pressurized) at a time. In addition, the rigid body components have a pair of one-direction wheels to model the anisotropic friction of real snakes.

The configuration of the robot’s coordinate is shown in Fig. 4. At time $t$, state $h(t) \in \mathbb{R}^2$ is the planar Cartesian position of the center of mass (COM) of the snake’s head, $\rho_g(t) \in \mathbb{R}^2$ is the planar displacement vector pointing from snake’s head COM to the goal position, $d_{\rho}(t) \in \mathbb{R}$ is the distance traveled along the head-to-goal-direction from the initial head COM position, $v(t) \in \mathbb{R}^2$ is the instantaneous planar velocity vector of the snake’s head COM, $\theta_{\rho}(t)$ is the angle between vector $\rho_g(t)$ and vector $v(t)$, and the locomotion speed $v_{\rho}(t) \in \mathbb{R}$ is the length of the projection of $v(t)$ on the head-to-goal-direction. According to [29], the bending curvature of each body link at time $t$ is computed by $\kappa_i(t) = \frac{\delta_i(t)}{l_i(t)}$, for $i = 1, \ldots, 4$, where $\delta_i(t)$ and $l_i(t)$ are the relative bending angle and the length of the middle line of the $i$th soft body link.

In [30], we developed a physics-based simulator that models the inflation and deflation of the air chamber and the resulting deformation of the soft bodies with tetrahedral finite elements. The simulator runs in real time using GPU. We use the simulator for learning the locomotion controller in the soft snake robot, and then apply the learned controller to the real robot.

III. DESIGN OF A CPG NETWORK FOR THE SOFT SNAKE ROBOT Locomotion

In this section, we introduce our CPG network design consisting of interconnected Matsuoka oscillators [23], [24].

A. Primitive Matsuoka CPG

A primitive Matsuoka CPG consists of a pair of mutually inhibited neuron models. The dynamical model of the primitive Matsuoka CPG is given as follows:

$$
K_f \tau_c x^c_i = -x^c_i - ay^c_i - \sum_{j=1}^{N} w_{ji} y_j^f + u^c_i + c
$$

$$
K_f \tau_s y^c_i = z^c_i - y^c_i
$$

$$
K_f \tau_c x^f_i = -x^f_i - az^f_i - \sum_{j=1}^{N} w_{ji} y_j^c + u^f_i + c
$$

$$
K_f \tau_s y^f_i = z^f_i - y^f_i
$$

(1)

where the subscripts $c$ and $f$ represent variables related to the extensor neuron and flexor neuron, respectively. The tuple $(x^c_i, y^c_i)$, $q \in \{e, f\}$ represents the activation state and self-inhibitory state of $i$th neuron, respectively, $z^q_i = g(q^q_i) = \max(0, x^q_i)$ is the output of $i$th neuron, $b \in \mathbb{R}$ is a weight parameter, $u^c_i, u^f_i$ are the forced tonic inputs to the oscillator, and $K_f \in \mathbb{R}$ is the frequency ratio. The set of parameters in the system includes the discharge rate $\tau_c \in \mathbb{R}$, the adaptation rate $\tau_s \in \mathbb{R}$, the mutual inhibition weights between flexor and extensor $a \in \mathbb{R}$, the inhibition weight $w_{ji} \in \mathbb{R}$ representing the coupling strength with the neighboring primitive oscillator, and the free-response oscillation tonic input $c \in \mathbb{R}$ ($c = 0$ in our previous work [26]). In our system, all coupled signals including $x^c_i, y^c_i$, and $z^q_i$ ($q \in \{e, f\}$) are inhibiting signals (negatively weighted), and only the tonic inputs are activating signals (positively weighted). In the current system, we have $N = 4$ primitive Matsuoka CPGs. For simplicity, we introduce a vector

$$
u = \begin{bmatrix} u^c_1, u^c_2, u^c_3, u^c_4, u^f_1, u^f_2, u^f_3, u^f_4 \end{bmatrix}^T
$$
(2)

to represent all tonic inputs to the CPG net.

B. Structure of the Matsuoka CPG Network for the Soft Snake Robot

Extending from a primitive Matsuoka CPG system to the multilinked snake robot, we construct a CPG network shown on the right of Fig. 3. The network includes four linearly coupled primitive Matsuoka oscillators. It is an inverted, double-sized version of Network VIII introduced in Matsuoka’s paper [23]. The network includes four pairs of nodes. Each pair of nodes (e.g., the two nodes colored green/yellow) in a row represents a primitive Matsuoka CPG (1). The edges correspond to the coupling relations among the nodes. In this graph, all the edges with hollowed endpoints are positive activating signals, while the others with solid endpoints are negative inhibiting signals. The oscillators are numbered 1 to 4 from head to tail of the robot. In order to build the connection between the CPG network and robot actuators, we define the output of the $i$th primitive Matsuoka CPG as

$$
\psi_i = a_\psi z_i = a_\psi \left( z^c_i - z^f_i \right)
$$
(3)

where $a_\psi$ is a ratio coefficient of $z_i$. Given the bounded input bounded output (BIBO) stability of the Matsuoka CPG net [21],

\[\text{The maximum function is noted as } g(\cdot) = \max(0, \cdot) \text{ in this article.} \]
the outputs $\psi = [\psi_1, \psi_2, \psi_3, \psi_4]^T$ from the primitive oscillators can be limited within $[-1, 1]$ by adjusting the ratio $a_\psi$. We let $\psi_i = 1$ for the full inflation of the $i$th extensor actuator and zero inflation of the $i$th flexor actuator, and vice versa for $\psi_i = -1$.

The actual pressure input to the $i$th chamber is $\lambda_i \cdot \psi_i$, where $\lambda_i$ is the maximal pressure input of each actuator. The primitive oscillator with green nodes controls the oscillation of the head joint. This head oscillator also contributes as a rhythm initiator in the oscillating system, followed by the rest parts oscillating with different phase delays in sequence. Fig. 3 shows all activating signals to the CPG network.

### C. Configuring the Matsuoka CPG Network

To determine the hyperparameters in the CPG network that generate a more efficient locomotion pattern, we employed a genetic programming (GP) algorithm similar to [31]. In this step, all tonic inputs are assigned with value 1 for the simplicity of fitness evaluation.

We define the fitness function—the optimization criteria—in GP as $F(v_{g,T}, \theta_{g,T}, d_{g,T}) = a_1|v_{g,T}| - a_2|\theta_{g,T}| + a_3|d_{g,T}|$, where $g$ indicates a fixed goal initiated in the heading direction of the snake robot. $T$ indicates the terminating time of fitness evaluation for each trial, and all coefficients $a_1, a_2, a_3 \in \mathbb{R}^+$ are constants.

To achieve stable and synchronized oscillations of the whole system, the following constraint must be satisfied [21]:

$$ (\tau_a - \tau_r)^2 < 4\tau_r\tau_a b $$

where $\tau_a, \tau_r, b > 0$. To satisfy this constraint, we can set the value of $b$ much greater than both $\tau_a$ and $\tau_r$, or make the absolute difference $|\tau_r - \tau_a|$ sufficiently small.

In other words, this fitness function is a weighted sum over the robot’s instantaneous speed, deviation angle, and total traveled distance on a fixed straight line at the terminating time $T$. In this scenario, a better-fitted configuration is supposed to maintain oscillating locomotion and reaches faster locomotion speed $|v_{g,T}|$ along the original heading direction at time $T$. In addition, the locomotion pattern is required to have a smaller deviation between the robot’s heading direction and the goal direction (with a small $|\theta_{g,T}|$), and with overall a longer traveled distance along the robot’s heading direction ($|d_{g,T}|$).

The desired parameter configuration found by GP is given in Table II in Appendix A.

### IV. MANEUVERABILITY ANALYSIS AND DESIGN OF THE LEARNING-BASED CONTROLLER WITH THE Matsuoka CPG NETWORK

When provided with equally constant tonic inputs, the designed Matsuoka CPG system can generate stable oscillation patterns to efficiently drive the soft snake robot slithering forward. However, a single CPG network cannot achieve intelligent locomotion and goal-tracking behaviors with potentially time-varying goals. For an intelligent controller, the free turning and accelerating (or decelerating) behaviors are the fundamental skills to realize autonomous locomotion in the goal-tracking tasks. In this article, we denote these two maneuverability demands as—steering control and velocity control (see Fig. 5). The later parts will focus on investigating the properties of the
Matsuoka CPG system to prove that it is controllable from both steering and velocity control perspectives. We design a proper connection between RL actions and controllable coefficients of the Matsuoka CPG system so that both steering and velocity control of the snake robot can be efficiently learned by the RL agent.

For steering control, we prove that the bias of tonic inputs is linearly proportional to the bias of the CPG output in both amplitude and duty cycle dimensions. This property inspires a rule that transforms the action outputs of the RL policy into the tonic inputs of the CPG system.

Next, we excavate two mechanisms that are helpful for velocity control. First, we show that the frequency ratio coefficient $K_f$ allows the RL agent to tune the locomotion velocity by directly adjusting the oscillation frequency. Second, by introducing the FOC, we provide a way to adjust the converging amplitude of the oscillation driven by the RL agent. With experiments, we show that the FOC is very helpful for reducing performance drop in the sim-to-real problem.

A. Steering Control With Imbalanced Tonic Inputs

Most existing methods based on CPG realize steering by either directly adding a displacement [2] to the output of the CPG system, or using a secondary system such as an artificial NN to compose the weighted outputs from multiple CPG systems [5]. In this section, we present a different approach based on the maneuverability of the Matsuoka oscillator—tuning tonic inputs to realize the biased wave patterns of CPG outputs for steering the slithering locomotion of the soft snake robot.  

For the RL controller to steer our snake robot smoothly through the Matsuoka CPG system, we need to find a clever way to make the steering dynamics easy to learn for the RL algorithm. In other words, the relation between tonic inputs and the output bias of each primitive Matsuoka oscillator in the CPG network needs to be simple and clear. In the original design of the Matsuoka oscillator, the flexor and extensor tonic inputs are independent of each other. This setting not only increases the dimension of action space for the RL agent but also makes the relationship between tonic inputs and the output bias more complicated. To simplify this problem, we first introduce a new relation defined as *complementation* to reform the relation between $u^c$ and $u^f$.

**Definition 1 (Complementation):** For two real signals $u(t)$ and $v(t)$, and a known bounded range $D : [a, b]$ where $D \subseteq \mathbb{R}$, we say $u(t)$ and $v(t)$ are complementary to each other in range $D$ when $u(t), v(t) \in D$ for all $t \in \mathbb{R}$ and $u(t) + v(t) \equiv b - a$.

Another important definition for this section is a relation between two periodic signals named *entrainment* based on the related theory in [24] and [25].

**Definition 2 (Entrainment):** Given a neural oscillator system with its natural frequency $\omega_n > 0$. If the neural oscillator’s output is synchronized to the coupled input with frequency $\omega$, then this system is entrained with the coupled input signal. The relation between the neural oscillator’s output and the coupled input signal is called *entrainment*. If the two signals are perfectly entrained, they are supposed to have the same oscillation amplitude and bias in addition to the synchronized oscillation frequency.

From our previous work [26], we have observed in experiment that the steering bias of a primitive Matsuoka oscillator is proportional to the amplitude of $u^c$ when $u^c$ and $u^f$ are complementary within the range $[0, 1]$. This key observation inspires us a dimension reduction technique to the input space of the CPG net: Instead of controlling $u^c_i, u^f_i$ for $i = 1, \ldots, n$ for a

| Metrics                             | Vanilla PPO | PPO-C/CPG | FOC-PPO-C/CPG |
|-------------------------------------|-------------|-----------|---------------|
| Simulated average speed (m/s)       | 0.14        | 0.137     | 0.135         |
| Simulated success rate              | 0.95        | 0.99      | 0.98          |
| Real average speed (m/s)            | 0.027 (4.807%) | 0.063 (4.54%) | 0.121 (4.11%) |
| Real success rate                   | 0.5 (4.21%) | 0.82 (4.17%) | 0.9 (4.81%)   |

The bold entities indicate the best values of each performance metrics among the compared methods.

![Fig. 5. Overview of the maneuverability of Serpentine locomotion with the Matsuoka oscillator.](image)

---

3 The fact that the biased wave output of the Matsuoka CPG system could cause the turning behavior of the snake robot comes from a previous work [16], which shows that the steering angle of a slithering snake robot on the planner ground can be linearly controlled by the bias of the oscillatory output of the Matsuoka oscillator as the command signal of joint actuators.
n-link snake robot, we only need to control $u^i$ for $i = 1, \ldots, n$ and let $u^i = 1 - u^j$. As the tonic inputs have to be positive in Matsuoka oscillators, we define a four dimension action vector $\alpha = [\alpha_1, \alpha_2, \alpha_3, \alpha_4]^T \in \mathbb{R}^4$ and map $\alpha$ to tonic input vector $\bar{\alpha}$ as follows:

$$u^i = \frac{1}{1 + e^{-\alpha_i}}, \quad \text{and} \quad u^j = 1 - u^i, \quad \text{for} \quad i = 1, \ldots, 4. \quad (5)$$

This mapping bounds the tonic input within $[0, 1]$. The reduced input dimension enables a more efficient policy search in RL.

Based on this design, we show that there are certain combinations of tonic inputs in a Matsuoka oscillator that are capable of generating imbalanced output trajectories and, therefore, result in the turning behavior of the robot. We present three possible cases of the forced tonic inputs that could maneuver the turning behavior of the snake robot.

1) The two tonic inputs are different constants.
2) The two tonic inputs are wave functions with imbalanced duty cycles.
3) The two tonic inputs are wave functions with imbalanced duty cycles, and both wave functions are added by different constant offsets.

It is noted that the third case is a linear combination of the first two. As a result, as long as the first two cases are proved to share the same property, the third one naturally holds. Next, we provide the frequency domain analysis of the Matsuoka oscillator to explain why the first two cases of tonic inputs enable imbalanced oscillation for the turning behavior.

Steering with biased amplitude of constant tonic inputs: To show that a pair of constant tonic inputs with different bias values can result in a biased oscillating output trajectory, we need to find out the relation between the bias of the output $z$ and the bias of tonic inputs, when the tonic inputs are constant and complementary in $[0, 1]$. In this situation, a primitive Matsuoka oscillator needs to be a zero damping harmonic system to maintain limit cycle oscillation. When the system has zero damping, the ratio between the amplitudes of state $x^q$ and output $z^q$ for $q \in \{e, f\}$, referred to as $K_n$, is obtained from a second-order linear ordinary differential equation ([B.9] in Appendix B-C) derived from (1)

$$K_n = \frac{\tau_e + \tau_a}{\tau_a a}, \quad (6)$$

where $\tau_e$ and $\tau_a$ are the discharge rate and the adaptation rate in (1), and parameter $a$ is the mutual inhibition weight between flexor and extensor of a primitive Matsuoka oscillator. The derivation of (6) can be found in Appendix B-C.

When the Matsuoka oscillator’s output only consists of free-response oscillation, we can establish the following relation between the output bias $z$ and the tonic input bias $u$.

Proposition 1: If a primitive Matsuoka oscillator satisfies the following three conditions:
1) the dynamical model of the primitive Matsuoka oscillator is harmonic;  
2) the tonic inputs $u^e$ and $u^f$ are square wave functions and complementary to each other;  
3) states $x^e$ and $x^f$ are perfectly entrained;  
then the oscillation bias of outputs $z$ and the bias of inputs $u$ satisfies the following linear relationship:

$$\text{bias}(z) = \frac{K_n}{b - a + 2} \text{bias}(u) \quad (7)$$

where $z = z^e - z^f$, $u = u^e - u^f$, and the coefficient $K_n$ satisfies $K_n = (\tau_e + \tau_a)/(\tau_a a)$.

Proof: See Appendix C-A.

Equation (7) suggests that there is a linear relationship between bias($u$) and bias($z$) in a primitive Matsuoka oscillator. We further validate this conclusion through the numerical simulation of a single primitive Matsuoka oscillator. We calculate the mean oscillation bias (numerical average\(^4\)) of the simulated state output $z$ and compare it with the estimated bias based on (7) (linear reference). Fig. 6 shows the curve of bias($z$) varies with bias($u$) $\in [-1, 1]$ in a primitive Matsuoka oscillator.

Fig. 6 and theoretical analysis (see Appendix C-B) show that for bias($u$) $\in [\frac{-2a}{a+b+1} - 1, 1 - \frac{2a}{a+b+1}]$, the linear relationship mentioned in Proposition 1 is applicable to the data of bias($z$) and bias($u$) collected by simulating the original Matsuoka system in (1). It is also observed that the applicable range of bias($u$) for Proposition 1 to hold increases with $K_n$. As shown in Fig. 7, when bias($u$) $\in [-1, \frac{-2a}{a+b+1} - 1] \cup [1 - \frac{2a}{a+b+1}, 1]$, the original Matsuoka system stops oscillating, which means the system stays at a set point equilibrium. In this case, bias($z$) and bias($u$) follow another linear relationship:

$$\text{bias}(z) = \begin{cases} \frac{\text{bias}(u) - (1 + 2c)}{2(1 + b)} & \text{if bias}(u) \in [-1, \frac{-2a}{a+b+1} - 1] \\ \frac{\text{bias}(u) + (1 + 2c)}{2(1 + b)} & \text{if bias}(u) \in [1 - \frac{2a}{a+b+1}, 1] \end{cases} \quad (8)$$

The derivation of the aforementioned relationship is provided in Appendix C-B.

In the next paragraph, we show that there is also a linear relationship between bias($z$) and bias($u$) of the Matsuoka oscillator when $u^e$ and $u^f$ are periodical signals with biased duty cycles.

Steering With the Biased Duty Cycle of Periodic Tonic Inputs: We show a different approach to control the steering of the snake robot given that both $u^e$ and $u^f$ are square wave functions and are complementary to each other.

Proposition 2: If a primitive Matsuoka oscillator satisfies the following three conditions:
1) the dynamical model of the primitive Matsuoka oscillator is harmonic;  
2) the tonic inputs $u^e$ and $u^f$ are square wave signals and are complementary to each other;  
3) $u^e$ is entrained with $z^e$, and $u^f$ is entrained with $z^f$;  
then the oscillation bias of $z$ and the bias of $u$ satisfies the following linear relationship:

$$\text{bias}(z) = \frac{1 + 2m}{b - a + 2} \text{bias}(u) \quad (9)$$

where $z = z^e - z^f$, $u = u^e - u^f$, and

$$m = \frac{1}{\pi} \sqrt{\frac{2K_n - 1}{2K_n - 1 + \frac{2a}{a+b+1}} \sin^{-1}(K_n)}$$

\(4\)Based on Fourier series analysis, given a continuous real-valued $P$-periodic function $z(t)$, the constant term of its Fourier series has the form $\frac{1}{P} \int_P z(t)dt.$
Fig. 6. Relation between oscillation bias and extensor tonic input \( u_e \) when setting different \( a \) values to obtain (a) \( K_n = 0.19 \). (b) \( K_n = 0.39 \). (c) \( K_n = 0.53 \). (d) \( K_n = 0.66 \). (e) \( K_n = 0.79 \).

Fig. 7. Relation between oscillation amplitude and duty cycle bias.

is a constant coefficient (\( r \) indicates amplitude of state \( x \)).

**Proof:** See Appendix C-C.

The simulated results also supports Proposition 2 when the Matsuoka system is taking periodic tonic inputs with biased duty cycles. Fig. 8 shows that with various \( K_n \) values, the linear relationship in (9) fits well with the curve between bias(\( u \)) and bias(\( z \)) collected by simulating the original Matsuoka system in (1).

Combining the conclusions in Propositions 1 and 2, we make the following remark.

**Remark 1:** If a primitive Matsuoka oscillator has periodical tonic input signals \( u_e \) and \( u_f \) that are complementary to each other, with imbalanced duty cycles and both wave functions are added by different constant offsets, then bias(\( z \)) is linearly related to the bias(\( u \)), where \( z = z_e - z_f \), \( u = u_e - u_f \).

Propositions 1 and 2 show that the oscillation bias of the Matsuoka CPG system is easy to maneuver through the biased tonic input signals. Since the oscillation bias is the key to steering in the snake’s slithering locomotion, these two propositions provide us insight to the design of RL module so as to improve the efficiency in learning the steering behavior of the snake robot.

**B. Velocity Control With Frequency Modulation**

Generally, the linear velocity of serpentine locomotion is affected by the snake’s oscillation amplitude and frequency. In this section, we show that the amplitude and frequency can be controlled by two coefficients of the Matsuoka CPG system to change the locomotion velocity of the soft snake robot.
First, the following relation between the frequency ratio $K_f$ and the natural frequency $\omega_i$ of the $i$th oscillator is established in [24, (5),(6)]

$$\tilde{\omega}_i \propto \frac{1}{\sqrt{K_f}}, \ i \in \{1, 2, 3, 4\}. \quad (10)$$

Second, the oscillating amplitude $\tilde{A}_i$ of the $i$th oscillator is linearly proportional to the amplitude of free-response oscillation tonic input $c$ when $c > 0$ and $u^{c}, u^{f}$ are constants [24], that is

$$\tilde{A}_i \propto c, \ i \in \{1, 2, 3, 4\}. \quad (11)$$

Equations (10) and (11) show that the frequency and amplitude of the Matsuoka CPG system are independently influenced by the frequency ratio $K_f$ and the free-response oscillation tonic inputs $c$. Therefore, these two coefficients can be considered major factors for the Matsuoka CPG system to control the velocity of the soft snake robot’s locomotion. In Fig. 9, we collect 2500 uniform samples within the region $c \in [0.4, 0.8]$, and $K_f \in [0.45, 1.05]$ and record the velocities generated in the simulator. We observe that with a fixed $c$, the average velocity increase monotonically with the frequency ratio $K_f$. We also observe that with the same $K_f$, the change of $c$ does not affect the locomotion velocity significantly. While with different values of $c$, the efficiency of $K_f$ in affecting the locomotion velocity is different. This means that we can mainly use $K_f$ to adjust the locomotion velocity, but the value of $c$ needs to be carefully selected. Given this analysis, we use $K_f$ to control the velocity of the robot. It is noted that the frequency ratio $K_f$ only influences the strength but not the direction of the vector field of the Matsuoka CPG system. Thus, modulating $K_f$ would not affect the stability of the whole CPG system.

C. Modulating Forced-Response Oscillation Amplitude With Free-Response Oscillation Tonic Input Constraint

Equation (11) shows that the free-response oscillation tonic input $c$ could affect the output amplitude of the Matsuoka oscillator when $u^{c}$ and $u^{f}$ are constants. We further discover that a positive value of the free response tonic input $c$ could set a threshold for the amplitude of the force-response tonic inputs $u^{c}$ and $u^{f}$, such that they need to pass this amplitude threshold in order to control the oscillation of the CPG system.

In the experiment section, we show that this property of $c$ can significantly improve the sim-to-real performance of our control framework.

In our previous work [26], when $c = 0$, there is no free-response oscillation in the system. When a Matsuoka oscillator has no free-response oscillation pattern, its output oscillation amplitude and bias are only controlled by the forced input signal given by the control tonic inputs $u^{c}$ and $u^{f}$. When the inertia in the simulated learning environment is high and the contact friction force is low, the RL agent learns to generate the forced-response oscillation tonic inputs with very small amplitude to keep a more stable heading direction during the locomotion. However, if we need the RL control policy to be able to initiate the CPG oscillation with an increased amplitude on the real robot (e.g., for traversing a terrain with higher friction resistance), the learned policy would not meet the requirement.

When $c \neq 0$, we conclude that in the Matsuoka oscillator, the amplitude $A_0$ of the force-response tonic inputs $u^{c}$ and $u^{f}$ must satisfy the inequality $A_0 > A_0$ to completely entrain with the output $z$ ($A_0$ is the entrainment threshold for $u^{c}$ and $u^{f}$ to synchronize the output $z$ of the Matsuoka oscillator [25]).
equation of $A_0$ is given as follows:

$$A_0(c, \omega) = \frac{c}{\sqrt{\tau_2 \omega^2 + 1} \left(1 - \frac{1}{\tau_f} \right)}$$  \hspace{1cm} (12)

where $A_0 > 0$ is the free-response oscillation amplitude and

$$\omega_n = \frac{1}{\tau_f K_f} \sqrt{\frac{(\tau_r + \tau_n)b}{\tau_r a} - 1}$$

is the free-response oscillation frequency [24]. The detailed derivation of $A_0$ is provided in Appendix B-D. According to [24, (30)], we have

$$A_0(c, \omega) \approx \frac{c}{\sqrt{\tau_2 \omega^2 + 1} (2K_n - 1 + \frac{2}{a} (a + b) \sin^{-1}(K_n))}.$$  \hspace{1cm} (13)

In (13), if $c = 0$, $A_0 \equiv 0$. In this case, there is no limiting threshold for the control policy to entrain the CPG output $z$ with $u^e$, $u^f$. When $\omega$ is fixed and $c > 0$, then the threshold $A_0 > 0$ and $A_0$ increases with $c$. Notice that $A_0 > 0$ must be satisfied for the free-response oscillation of the Matsuoka system be attenuated by the system damping. This also means the force-response tonic inputs $u^e$, $u^f$ entrain the CPG output $z$. In this scenario, the control policy needs to increase $A_0$ to control the CPG system effectively. It is also noted that $A_0 \rightarrow 0$ as $\omega \rightarrow \omega_n$, therefore, there are two ways for the RL agent to realize the entrainment status: one is keeping the oscillation frequency $\omega$ close to the free-response oscillation frequency $\omega_n$, and the other is increasing the value of $A_0$ to make $A_0 > A_0$. Therefore, the combination of the two directions can encourage the intelligent controller to produce force-response tonic inputs that can not only approach desired oscillating amplitude, but also pursue frequency resonance with the original CPG system. Based on this special property of the Matsuoka oscillator, we propose a new method—FOC-PPOC-CPG to enforce better entrainment between RL control signals and the CPG states.

According to the relation between $A_0$ and $c$, we can use $c$ to keep the oscillation amplitude of the Matsuoka oscillator at different levels. One previous work [32] has shown that the oscillation amplitude of the Matsuoka oscillator can be used to improve the slithering locomotion velocity of a rigid snake robot in different environments with different friction coefficients. Hence, we can use $c$ to adapt the body undulation amplitude of the soft snake robot to different environments with various contact properties. With this approach, we can improve the sim-to-real performance of an RL snake controller by tuning its signal amplitude, instead of relying on the environment-based methods, such as domain randomization (DR) [33] or other data augmentation techniques, which are computationally expensive.

In the later part of this article, our experiment results (see Section VI-E) verify the merit of $c$ in improving the sim-to-real performance of our snake locomotion controller.

**D. NN Controller**

We have now determined the encoded input vector of the CPG net to be vector $\alpha$ (tonic inputs) and frequency ratio $K_f$. This input vector of the CPG is the output vector of the NN controller. The input to the NN controller is the state feedback of the robot, given by $s = [\|\rho_b\|, v_g, \dot{\theta}_y, \theta_y, \kappa_1, \kappa_2, \kappa_3, \kappa_4]^T \in \mathbb{R}^8$ (see Fig. 4). Next, we present the design of the NN controller.

The key insight for the design of the NN controller is that the robot needs not to change velocity very often for smooth locomotion. This means the updates for tonic inputs and the frequency ratio can be set to be at two different time scales. With this insight, we adopt a hierarchical reinforcement learning method called the option framework [34, 35] to learn the optimal controller. The controller uses the tonic inputs as low-level primitive actions and frequency ratio as high-level options of the CPG net. The low-level primitive actions are computed at every time step. The high-level option changes infrequently. Specifically, each option is defined by $\{I, \pi_y : S \rightarrow \{y\} \times \mathbb{R}^4, \beta_y\}$ where $I = S$ is a set of initial states, and $\pi_y$ is the intraoption policy. By letting $I = S$, we allow the frequency ratio to be changed at any state in the system. Variable $y \in [0, 1]$ is a value of frequency ratio, and $\beta_y : S \rightarrow [0, 1]$ is the termination function such that $\beta_y(s)$ is the probability of changing from the current frequency ratio to another frequency ratio.

The options share the same NN for their intro-option policies and the same NN for termination functions. However, these NNs for intro-option policies take different frequency ratios. The set of parameters to be learned by policy search include parameters for intraoption policy function approximation, parameters for termination function approximation, and parameters for high-level policy function approximation (for determining the next option/frequency ratio). Proximal policy optimization critics (PPOC) in the OpenAI baselines [36] is employed as the policy search in the RL module.

Let us now review the control architecture in Fig. 3. We have a multilayer perceptron (MLP) NN with two hidden layers to approximate the optimal control policy that controls the inputs of the CPG net in (1). The output layer of MLP is composed of action $\alpha$ (green nodes), option in terms of frequency ratio (pink node), and the terminating probability (blue node) for that option. The input of NN consists of a state vector (yellow nodes) and its output from the last time step. The purpose of this design is to let the actor network learn the unknown dynamics of the system by tracking the past actions in one or multiple steps [5, 37, 38]. Given the BIBO stability of the Matsuoka CPG net [21] and that of the soft snake robots, we ensure that the closed-loop robot system with the FOC-PPOC-CPG controller is BIBO stable. Combining with (5) which enforces a limited range for all tonic inputs, this control scheme is guaranteed to generate bounded inputs, which lead to bounded outputs in the system.

**V. CURRICULUM AND REWARD DESIGN FOR EFFICIENT LEARNING-BASED CONTROL**

In this section, we introduce the design of the curriculum and reward function for efficiently learning a goal-tracking controller given the proposed FOC-PPOC-CPG scheme.
Combining with the definition of goal-reaching tasks and their corresponding level setups, the reward at every time step is defined as

$$R(v_g, \theta_g) = c_v v_g + c_g U + c_g \cos \theta_g \sum_{k=0}^{i} \frac{1}{r_k} I(||\rho_g|| < r_k)$$

(14)

where $c_v, c_g \in \mathbb{R}^+$ are constant weights, $v_g$ is the length of the projection of the snake’s head COM velocity $v$ on the head-to-goal-direction, $\rho_g$ is the linear displacement vector between the head COM of the robot and the goal position, $\theta_g$ is the angle between vector $v$ to vector $\rho_g$ in Fig. 4, $r_k$ defines the goal range in task-level $k$, for $k = 0, \ldots, i$, and $I(\rho_g < r_k)$ is an indicator function that outputs one if the robot head is within the goal range for task-level $k$.

This reward trades off two objectives. The first term, weighted by $c_v$, encourages high locomotion velocity toward the goal. The second term, weighted by $c_g$, rewards the learner based on the position of the robot to the goal, and the level of the curriculum the learner has achieved for the goal-reaching task. For every task, if the robot has not entered the goal range, it receives a potential field reward only. When the robot enters the goal range in task-level $i$, it receives a summation of rewards $1/r_k$ for all $k \leq i$ (the closer to the goal the higher this summation), shaped by the approaching angle $\theta_g$ (the closer the angle to zero, the higher the reward).

If the agent reaches the goal defined by the current task-level, a new goal is randomly sampled in the current or next level (if the current level is completed). There are two failing situations, where the desired goal is resampled and updated. The first situation is starving, which happens when the robot stops moving for a certain amount of time, referred to as the starvation time. The second case is missing the goal, which happens when the robot keeps heading in the wrong direction as opposed to moving toward the goal ($v_g(t)$ being negative) for a certain amount of time.

### VI. Experimental Evaluation

In this section, we evaluate the proposed method in both simulation and real environments. We first introduce the experimental setup to explain how the data of the robot is collected during locomotion, as well as the training configuration for the RL algorithm. Then, we compare the properties of control signals between our method and the vanilla PPO as locomotion controllers for goal-reaching tasks in simulation. In the comparison analysis, we highlight the performance drop of each method from simulation to real to extrapolate the advantage of our method. Last, we further test and analyze the sim-to-real robustness in difficult goal-reaching tasks that are never seen at the training stage, and the real robot performance against disturbance.

#### A. Experimental Setup

1) Environment Sensing and Data Collection: The states of the real snake robot are captured by a single web camera hanging...
on the ceiling of the experiment room. The robot body detection is realized by using Aruco—a library in OpenCV for QR codes detection and localization [41]. These QR codes are printed and attached to the rigid bodies of the snake robot and the goal position. Fig. 11 shows the experiment setup for the real snake robot goal-reaching tasks. Once the QR codes on the robot bodies and the goal(s) are detected, their pixel-wise coordinate vectors are calculated with distortion corrected. Given the camera calibration information, we can translate the pixel data of all QR codes into the real world 2-D coordinates, and then transform it into positional information and the body posture of the robot. The control policy function running on the desktop computer receives the observation states, generates the control commands and passes them through WiFi communication. The ESP32 chips on the snake bodies translate the commands into pulsewidth modulation signals to activate or deactivate the valves [28], [30] on the snake robot.

2) Reinforcement Learning Configuration: We use a four-layered NN with 128×128 hidden layer neurons as a general configuration for the actor and critic networks of all RL methods mentioned in this section. The back-propagation of the critic net was done with Adam optimizer and a step size of 5×10^{-4}. For data collection of each trial trajectory, the starvation time for the failing condition is 60 ms. The missing goal criterion is triggered whenever \( v_g(t) \) (the velocity on the goal-direction, see Fig. 4) stays negative for over 60 time steps. In order to compensate for the mismatch between the simulation and the real environment, most notably the friction coefficients, we employ a DR technique [33], in which a subset of physical parameters are sampled from several uniform distributions. The range of distributions of DR parameters used for training are in Table IV (see Appendix A).

For PPOC-CPG and FOC-PPOC-CPG, we first train the policy net with fixed options (at this moment, the termination probability is always 0, and a fixed frequency ratio \( K_f = 1.0 \) is used). When both the task-level and the reward cannot increase anymore, we allow the learning algorithm to change the option, i.e., pick a different frequency ratio \( K_f \) along with termination function \( \beta \), and keep training the policy until the highest level in the curriculum is passed.

In the PPOC-CPG method, the value of the free-response tonic input \( c \) is equivalently considered zero since it is not formally introduced in the previous control design [26]. According to the definition of \( A_0 \) (see (12)), the amplitudes of both \( u^e \) and \( u^f \) need to be greater than \( A_0 \) in order to dominate in controlling the outputs of the Matsuoka CPG system. The value of \( A_0 \) should not be greater than the upper bound of \( u^e \) and \( u^f \), which is 1 defined by (5). Among a group of candidates ranging from 0.25 to 2, we choose \( c = 0.75 \) as our FOC for the FOC-PPOC-CPG controller. This value is valid for our system because when we set \( c = 0.75 \) and all other coefficients of the CPG network (see Table II) to (12), the result shows \( A_0 \in [0.24, 0.34] \subset [0, 1] \), with \( \omega \in [3.77, 5.02] \). It is noted that the range of \( \omega \) here is calculated from multiple sampled sequences of \( u^e \) and \( u^f \) recorded in the real snake goal-reaching tasks. Since we are testing the sim-to-real performance, all methods involved in this comparison are trained in the simulator for sufficiently long iterations (12,500 episodes) to ensure convergence. Each method is trained ten times with different random seeds and the controller with the best performance is selected to be tested on the real robot. All curriculum parameters (see Table III) and DR parameters (see Table IV) are fixed for all three methods involved.

The whole training process of each method runs on 4 simulated soft snake robots in parallel on a workstation equipped with an Intel Core i7-9700 K, 32 GB of RAM, and one NVIDIA RTX2080 Super GPU.

B. Verification of Steering Property of PPOC-CPG

We use a simulated experiment to show that our FOC-PPOC-CPG control policy has learned the turning behavior with the biased tonic input signal, and the Matsuoka CPG system can linearly map the biased tonic input to the biased actuation signal as Propositions 1 and 2 predicted. In the experiment, we test the converged FOC-PPOC-CPG policy on multiple set-point goals placed in certain directions (−90°, −70°, −60°, −45°, −30°) with a fixed distance (1 m), which approximately represent the desired turning angles of the locomotion tasks. For each goal position, we carry out 5 trials and record the tonic inputs data and CPG output data of the head CPG node of the soft snake robot. The reason for choosing the head node is because this node’s behavior best reflects the desired steering direction of the RL agent. Fig. 12(a) shows a violin plot of the tonic input bias and the CPG output bias for different turning angles [the bias signals are calculated by (C.10)]. It is observed from Fig. 12(a) that both bias signals are monotonically related to the desired turning angle (initial goal-direction). Fig. 12(b) shows the linear
 regression result based on all data points. We can observe a clear linear relationship between bias\(_z\) and bias\(_u\) of the head CPG node (with the coefficient of determination equal to 0.978, a value closer to 1 indicate higher linearity). This result provides stronger support for Propositions 1 and 2.

C. Control Signal Comparison Between PPOC-CPG and Vanilla PPO

First, we compare PPOC-CPG and vanilla PPO in terms of the smoothness of the control input learned in simulation. We train both PPOC-CPG and vanilla PPO in the same environment until convergence. Fig. 13 shows segments of the control signal \(\psi_1\) generated by the vanilla PPO controller and PPOC-CPG controller controlling the simulated soft snake robot in a straight line goal-tracking task. From Fig. 13(a), it is observed that the signal generated by the vanilla PPO policy oscillates at a relatively higher frequency (about 10 Hz on average) with irregular oscillation patterns. Such kind of control signals are not feasible for the actuators in reality. This is because the inflation and deflation of soft air chambers on the snake robot have a certain delay so that the soft pneumatic actuators are not able to track fast oscillating signals. On the other end, the curve in Fig. 13(d) shows that the agent trained with PPOC-CPG can converge to a stable limit cycle trajectory at a relatively lower but more natural frequency (1.6 Hz) for serpentine locomotion. Our approach shows its advantage of being able to generate smoother oscillatory control signals even when the inputs to the CPG system are discontinuous. Fig. 13 also compares the phase plane portraits recorded at different time stages of the two learning methods. From Fig. 13(b) and (c), we observe that the oscillating signal generated by vanilla PPO policy performs irregular oscillation in the first 300 time steps, and cannot converge to a stable limit cycle when it evolves to time step 700. While in Fig. 13(e) and (f), despite a little deviation from the first 300 time steps, the outputs of the CPG network eventually converge to a stable limit cycle within 700 time steps. This experiment shows that the CPG system is capable of stabilizing the oscillation pattern in simple locomotion tasks for the soft snake robot.

D. Comparison of our Reward Design and a Sparse Reward Function

We compare the learning process of the revised reward function with our previous one that only rewards the agent for goal reaching events [26] (for each case we record 5 learning trials). In average, the agent with dense reward is able to reach and converge to level-12, while the agent with sparse reward only converges to level-8 (see Table III). The calculated results in Fig. 14 show that the system trained with dense reward function outperforms that with a sparse reward design.

In the following section (Section VI-E), these methods are compared in the real robot to demonstrate the advantage of the proposed PPOC-CPG control.

E. Sim-to-Real Performance of FOC-PPOC-CPG

1) Performance Comparison With Original PPOC-CPG and Vanilla PPO: Since FOC-PPOC-CPG is designed for improving the sim-to-real transfer learning performance of the PPOC-CPG
Fig. 15. Sample comparison of trajectories generated by Vanilla PPO policy, PPOC-CPG policy, and FOC-PPOC-CPG policy in reality.

Fig. 16. Sample way-point trajectories followed by improved PPOC-CPG controller in simulation and real in (a) zigzag and (b) square.

Fig. 17. Disturbance recovery for goal-reaching task followed by FOC-PPOC-CPG controller in real experiments. (a) xy plane trajectory, (b) control signals for the actuators, and (c) video snapshot of recorded robot motion.
method, we first compare the sim-to-real performance of the FOC-PPOC-CPG with the original PPOC-CPG and Vanilla PPO in single goal-reaching tasks. For the real robot tests, all three controllers trained by the simulator are directly applied without further training. We test the controllers by setting goals in three directions (mid, left, and right) with fixed angles, distances, and an accuracy radius of \( r = 0.175 \text{m} \). Each direction takes 10 trials for all three methods in both simulation and reality.

To evaluate the sim-to-real performance, we calculate the average locomotion speed \( (v_g) \) and the success rate for goal-reaching tasks collected from both simulation and real experiments. According to Section IV-C, the contact resistance forces in the simulator are smaller than in the real environment, when applying the RL control policy learned in the simulator directly to the real robot, the performance of the real robot is often worse than the simulated agent. In the rows of real robot evaluations in Table I, we use down-arrows and percentage values to show the extent of performance drop compared to the simulating performance with the same method. From Table I, it is observed that although the Vanilla PPO controller learns the best locomotion speed in the simulator at the cost of goal-reaching accuracy, its locomotion pattern cannot fit the real robot well. The real robot experiences a drastic drop in performance on both locomotion speed (80.7%) and success rate (42.1%). For the original PPOC-CPG, though it has achieved an overall better performance than Vanilla PPO, its sim-to-real performance drop is still relatively high, with a 54% speed drop and 17% of accuracy drop. After adding the FOC to the CPG system, the new policy reaches almost the same performance as the original PPOC-CPG in the simulator. In Section IV-C, we have shown that the free-response oscillation tonic input \( \gamma > 0 \) could help maintain the oscillation amplitude of the control signal of FOC-PPOC-CPG during the learning process. It is noticed that the maintained amplitude of the control signals does not improve the locomotion speed and goal-reaching accuracy at the training stage in the simulation. However, when the learned policy of FOC-PPOC-CPG is applied to the real robot without further training, it performs significantly better than the previous two methods in both locomotion speed and success rate.

Fig. 15 shows a more intuitive result by comparing sample trajectories of the abovementioned three methods in different goal-reaching tasks performed on the real robot. The trajectories show that the robot controlled by Vanilla PPO policy moves much slower than the other two. And, it moves in a less symmetric way for the left and right turning tasks. While the original PPOC-CPG and FOC-PPOC-CPG show similar symmetry properties in the trajectory shapes, the difference is that the controller trained with FOC-PPOC-CPG moves almost as fast as that trained with PPOC-CPG. This comparison is presented in the video:\textsuperscript{5} “PPO Learning methods comparison.mp4.”

Since PPO is an on-policy RL algorithm and has been established for many years, we also use a more up-to-date off-policy RL algorithm—twin delayed deep deterministic policy gradient \textsuperscript{42} to replace the role of PPO in our framework, and train it with a shorter learning period (2000 episodes) to verify the generality of our approach. The results and a brief discussion can be viewed in our Supplementary document.\textsuperscript{6}

2) Performance in Reaching Unseen Goals: We also investigate the sim-to-real performance of FOC-PPOC-CPG in harder goal-reaching tasks. Fig. 16 compares the head trajectories in Cartesian space for two different setups of way-point goals. The testing trajectories include a square turning trajectory for testing consecutive sharp turning in the same direction [see Fig. 16(b)], and a zigzag trajectory for testing continuous sharp turning in opposite directions [see Fig. 16(a)]. Both way-point goal series have sharper turning angles than the highest level in the training curriculum in Table III. Videos “Half square trajectory sim2real.mp4” and “Zigzag trajectory sim2real.mp4” provide the dynamic view of Fig. 16(a) and (b), respectively. From the example videos, it is observed that in both trajectories, the speed drop of the real robot is still around 10%, which is not worse than single goal-reaching tasks in Table I. It is noted that in both Fig. 16(a) and (b), it takes the real robot longer distances to make the sharp turning. This is also due to the larger ground resistance forces in reality.

3) Robustness to External Disturbance: We also test the FOC-PPOC-CPG controller’s ability to keep tracking the desired target when the robot is disturbed by an external pushing force. Fig. 17(a) and video “Disturbance recovery.mp4” show an example trajectory of a disturbed goal-reaching task. It is observed from Fig. 17(c) that the FOC-PPOC-CPG controller reacts accordingly to its situation during the locomotion. When the deviation between the robot’s head and the goal is relatively smaller before the disturbance (before 4.1 s), the robot gently oscillates and adjusts its turning direction gradually towards the goal-direction. At around 9.04 s, when the robot is pushed away from its desired direction, one can observe a clear redirection to the left-hand side of the robot’s heading direction. The FOC-PPOC-CPG is able to adjust and make sharp turning to return to the correct direction and still reach the goal without wasting too much time on the recovery.

VII. CONCLUSION

This article develops a bioinspired controller for learning agile serpentine locomotion with a CPG net mimicking the central nervous system of natural snakes. The contribution of this article is two-fold: First, we investigate the properties of the Matsuoka oscillator for achieving diverse locomotion skills in a soft snake robot. Second, we construct an FOC-PPOC-CPG net that uses a CPG net to actuate the soft snake robot, and an NN to efficiently learn a closed-loop near-optimal control policy that utilizes different oscillation patterns in the CPG net. This learning-based control scheme shows promising results in goal-reaching tasks in soft snake robots.

This control scheme can be applicable to a range of biomimetic motion control for robotic systems and may require different designs of the CPG network given insights from the corresponding biological systems. We have been investigating the generality of the proposed control scheme on different robotic systems.

\textsuperscript{5} All videos in this article can be viewed from http://shorturl.at/cgms1

\textsuperscript{6} The Supplementary document is available at https://shorturl.at/ntAKM
and obtained promising early results (see Supplementary document). Our current research focuses on introducing sensory inputs into the CPG system, which enables reactive responses to contact forces with the external environment and generates an obstacle-aided locomotion controller for the soft snake robot. It is also interesting to investigate distributed control designs that can scale to high-dimensional soft snake robot or other biomimetic robotic systems.

**APPENDIX A**

**DATA**

This section includes the parameter configuration of the Matsuoka CPG network and the hyperparameter setting of DR for the experiment.

### TABLE II

**PARAMETER CONFIGURATION OF THE MATSUOKA CPG NET CONTROLLER FOR THE SOFT SNAKE ROBOT**

| Parameters            | Symbols | Values  |
|-----------------------|---------|---------|
| Amplitude ratio       | $a_0$   | 2.0935  |
| +Self-inhibition weight| $b$     | 10.0355 |
| +Discharge rate        | $\tau_i$| 0.7096  |
| +Adaptation rate       | $\tau_a$| 1.7728  |
| Period ratio           | $K_f$   | 1.0     |
| Mutual inhibition weights| $a_k$  | 4.6062  |
| Coupling weights       | $w_{ij}$| 8.8669  |

The bold entities indicate the key parameters to determine whether the oscillation exists in a Matsuoka oscillator.

### TABLE III

**CURRICULUM SETTINGS**

| Levels | Distance range (m) | Turning angles (°) | Goal radius (m) |
|--------|--------------------|-------------------|-----------------|
| 1      | 1.2–1.5            | −10–10            | 0.5             |
| 2      | 1.2–1.5            | −10–10            | 0.4             |
| 3      | 1.2–1.5            | −15–15            | 0.3             |
| 4      | 1.2–1.5            | −20–20            | 0.25            |
| 5      | 1.2–1.5            | −30–30            | 0.2             |
| 6      | 1.0–1.5            | −40–40            | 0.18            |
| 7      | 1.0–1.5            | −45–45            | 0.15            |
| 8      | 1.0–1.5            | −50–50            | 0.12            |
| 9      | 0.9–1.5            | −60–60            | 0.09            |
| 10     | 0.9–1.5            | −70–70            | 0.06            |
| 11     | 0.9–1.5            | −80–80            | 0.05            |
| 12     | 0.8–1.5            | −80–80            | 0.05            |

### TABLE IV

**DR PARAMETERS**

| Parameter                  | Low  | High |
|----------------------------|------|------|
| Ground friction coefficient | 0.1  | 1.5  |
| Wheel friction coefficient  | 0.05 | 0.10 |
| Rigid body mass (kg)       | 0.035| 0.075|
| Tail mass (kg)             | 0.065| 0.085|
| Head mass (kg)             | 0.075| 0.125|
| Max link pressure (psi)    | 5    | 12   |
| Gravity angle (rad)        | −0.001| 0.001|

**APPENDIX B**

**FOUNDATION**

### A. Describing Function Analysis of the Matsuoka Oscillator

According to Fourier theory, we denote the main sinusoidal and constant component in Fourier expansion of the vanilla state $x(t)$ as

$$x(t) = A \cos(\omega t) + d$$

$$= A \cos(\omega t) + r$$  \hspace{1cm} (B.1)

where $r = d/A, r \in R$ is the ratio of bias to the amplitude of the signal. We assume $x_f(t)$ only contains cosine term for simplicity. And, because this article only discusses amplitude and bias properties of the signals, such simplification will not affect the following derivations. We use $z_f(t) = g(x_f(t)) = \epsilon(t)$ to represent the main sinusoidal property of $z(t) = g(x(t)) = \max (x(t), 0)$. In a single period $[-\frac{\pi}{\omega}, \frac{\pi}{\omega}]$

$$g(x_f(t)) = \begin{cases} 
0 & \text{elsewhere} \\
A \cos(\omega t) + r & t \in \left[ -\arccos \left( -\frac{r}{\omega} \right), \arccos \left( -\frac{r}{\omega} \right) \right] 
\end{cases}$$

Using Fourier expansion, the output state $z_f(t)$ can also be expressed as

$$g(x_f(t)) = g(A \cos(\omega t) + r)$$

$$= A g(\cos(\omega t) + r)$$

$$= A(K(r) \cos(\omega t) + L(r)) + \epsilon(t)$$

$$= z_f(t) + \epsilon(t) \quad (n \geq 1)$$  \hspace{1cm} (B.2)

such that

$$z_f(t) = A(K(r) \cos(\omega t) + L(r))$$  \hspace{1cm} (B.3)

where

$$K(r) = \begin{cases} 
0 & (r < -1) \\
\frac{1}{\pi} (r \sqrt{1 - r^2} - \cos^{-1}(r)) + 1 & (-1 \leq r \leq 1) \\
1 & (r > 1) 
\end{cases}$$  \hspace{1cm} (B.4)

and

$$L(r) = \begin{cases} 
0 & (r < -1) \\
\frac{1}{\pi} (\sqrt{1 - r^2} - \cos^{-1}(r)) + r & (-1 \leq r \leq 1) \\
1 & (r > 1) 
\end{cases}$$  \hspace{1cm} (B.5)

The derivation of $K(r)$ and $L(r)$ are based on Fourier series analysis (see Appendix B-B). Both $K(r)$ and $L(r)$ are constrained by $-1 \leq r \leq 1$ for $x_f(t)$ to be nonnegative in the period $[-\frac{\pi}{\omega}, \frac{\pi}{\omega}]$.

Function $\epsilon(t)$ is the summation of all remaining high-frequency terms in the Fourier expansion of $z_f(t)$.

When $t \in \left[ -\arccos \left( -\frac{r}{\omega} \right), \arccos \left( -\frac{r}{\omega} \right) \right]$, $z_f(t) = x_f(t)$, we have

$$\epsilon(t) = x_f(t) - A \{ K(r) \cos(\omega t) + L(r) \}$$

$$= -\frac{A}{\pi} \left\{ \left[ r \sqrt{1 - r^2} - \arccos(\frac{r}{\omega}) \right] \cos(\omega t) + \sqrt{1 - r^2} - r \arccos(\frac{r}{\omega}) \right\}$$

When $t \in \left[ -\frac{\pi}{\omega}, -\arccos \left( -\frac{r}{\omega} \right) \right] \cup \left[ \arccos \left( -\frac{r}{\omega} \right), \frac{\pi}{\omega} \right]$, $z_f(t) = 0$, we have

$$\epsilon(t) = 0 - A \{ K(r) \cos(\omega t) + L(r) \}$$
\[ \begin{align*}
\epsilon(t) &= \begin{cases}
0, & t \in [0, 0.2055] \\
\arccos(-r), & t \in \left[ -\frac{\omega}{\pi}, -\frac{\arccos(-r)}{\omega} \right] \\
\arccos(-r) - \frac{\omega}{\pi}, & t \in \left[ -\frac{\pi}{\omega}, -\frac{\arccos(-r)}{\omega} \right] 
\end{cases}
\end{align*} \]

Then, we can numerically calculate the bound of \( \epsilon(t) \) for certain \( A \) and \( \omega \). For instance, if \( A = 1 \) and \( \omega = 1 \), we have

\[ \epsilon(t) \in [0, 0.2055] \text{ when } t \in [0, 0.2055] \]

\[ \epsilon(t) \in \left[ -\frac{\omega}{\pi}, -\frac{\arccos(-r)}{\omega} \right] \text{ when } t \in \left[ -\frac{\pi}{\omega}, -\frac{\arccos(-r)}{\omega} \right] \]

**B. Calculation of \( K(r) \) and \( L(r) \)**

Given \( x_{rf}(t) = A_{ix}(\cos(\omega t) + r) \) as an even function, the general Fourier expansion of \( x_{rf}(t) = g(x_{rf}(t)) \) is

\[ z_{rf}(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(n\omega t) \]

\[ = \frac{1}{2}a_0 + a_1 \cos(\omega t) + \epsilon(t). \]  

(B.6)

where

\[ a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} g(A(\cos(\omega t) + r)) dt \]

\[ a_1 = \frac{1}{\pi} \int_{-\pi}^{\pi} g(A(\cos(\omega t) + r)) \cos(\omega t) dt. \]

In this case, both the bias \( a_0 \) and the amplitude \( a_1 \) become functions of \( r \). Combining with (B.2), we use \( AK(r) \) to represent \( a_1 \) and \( AL(r) \) to represent \( a_0 \), which are calculated as follows:

\[ K(r) = \frac{a_1}{A} = \frac{\omega}{\pi} \int_{-\pi/\omega}^{\pi/\omega} g((\cos(\omega t) + r)) \cos(\omega t) dt. \]

Let \( t = \omega \tau \), we have

\[ K(r) = \frac{1}{\pi} \int_{-\pi}^{\pi} g((\cos(t) + r)) \cos(t) dt \]

\[ = \frac{1}{\pi} \int_{-\pi}^{\pi} \left( \cos(t) + r \right) \cos(t) dt \]

\[ = \frac{1}{\pi} \left( r \sqrt{1 - r^2} - \cos^{-1}(r) \right) + 1 \quad \left( -1 \leq r \leq 1 \right) \]

and

\[ L(r) = \frac{a_0}{A} = \frac{1}{\pi} \int_{-\pi}^{\pi} g(\cos(t) + r) dt \]

\[ = \frac{1}{\pi} \int_{-\pi}^{\pi} \cos(t) + r \cos(t) dt \]

\[ = \frac{1}{\pi} \left( r \sqrt{1 - r^2} - \cos^{-1}(r) \right) + r \quad \left( -1 \leq r \leq 1 \right). \]

**C. Derivation of \( K_n \)**

Based on (1), we first set \( x_i(t) = x_i^{e}(t) - x_i^{f}(t) \), \( y_i(t) = y_i^{e}(t) - y_i^{f}(t) \), \( z_i(t) = z_i^{e}(t) - z_i^{f}(t) \), \( u_i(t) = u_i^{e}(t) - u_i^{f}(t) \). Then, by taking subtraction between flexor and extensor in (1) and neglect phase related coupling terms from other primitive CPGs, we have

\[ K_f \tau_r \frac{d}{dt} \left( x_i^{e} - x_i^{f} \right) = -\left( x_i^{e} - x_i^{f} \right) - a \left( z_i^{e} - z_i^{f} \right) \]

\[ - b \left( y_i^{e} - y_i^{f} \right) + \left( u_i^{e} - u_i^{f} \right) \]

\[ K_f \tau_a \frac{d}{dt} \left( y_i^{e} - y_i^{f} \right) = \left( z_i^{e} - z_i^{f} \right) - \left( y_i^{e} - y_i^{f} \right) \]

which can be simplified to

\[ K_f \tau_r \frac{d}{dt} x_i = -x_i + a z_i - b y_i + u_i \]

\[ K_f \tau_a \frac{d}{dt} y_i = z_i - y_i. \]  

(B.7)

If \( x_i^{e} \) and \( x_i^{f} \) satisfy the perfect entrainment assumption, such that the amplitude \( A x_i^{e} = A x_i^{f} = A_x \), and the bias \( r x_i^{e} = r x_i^{f} = r_x \), and the phase delay between \( x_i^{e} \) and \( x_i^{f} \) is \( \frac{\pi}{2} \) (half of the period). Then, we have \( z_{rf} = K(r_x) x_{rf} \). Similar to the notation in Appendix B-A, the marker \( F_r \) indicates the fundamental sinusoidal and constant component in Fourier expansion of the corresponding variable. Let \( K_f = 1 \), (B.7) can be further simplified to

\[ K_r \tau_r \frac{d}{dt} x_i = a K(r_x) x_i - b y_i + u_i \]

\[ K_f \tau_a \frac{d}{dt} y_i = K(r_x) x_i. \]  

(B.8)

Next, an ordinary differential equation can be obtained by merging the two equations in (B.8) as

\[ \tau_r \tau_a \frac{d^2}{dt^2} x_i + (\tau_r + \tau_a - \tau_a a K(r_x)) \frac{d}{dt} x_i \]

\[ + ((b - a) K(r_x) + 1) x_i = \frac{d}{dt} u_i + u_i. \]  

(B.9)

When the system is harmonic, the coefficient of the first-order derivative of (B.9) becomes zero, then

\[ K(r_x) = \frac{\tau_r + \tau_a}{\tau_r a} \triangleq K_n. \]  

(B.10)

Coefficient \( K_n \) is a special case of \( K(r_x) \) in the harmonic Matsuno oscillator.

**D. Amplitude Threshold of Transition From Free Oscillation to Forced Entrainment**

In order to extract the free-response oscillation component, let \( \tilde{u} = u - 1, \tilde{c} = c + 1 \) then (1) is equivalent to

\[ K_f \tau_r \tilde{x}_i^{e} = -x_i^{e} - a z_i^{f} - b y_i^{f} - \sum_{j=1}^{N} w_{ij} y_j^{f} + \tilde{u}_i + \tilde{c} \]

\[ K_f \tau_a \tilde{y}_i^{e} = z_i^{e} - y_i^{f} \]
\[ K_f \tau_s \dot{x}_i^l = -x_i^l - a z_i^l - b y_i^l - \sum_{j=1}^{N} w_{ji} y_j^l + \ddot{u}_i^l + \ddot{c} \]
\[ K_f \tau_a \dot{y}_i^l = z_i^l - y_i^l. \]  
(B.11)

Since \( u_i^q \in [0, 1] \) (for \( q \in \{ e, f \} \)) and \( c \geq 0 \), we have \( \dot{u}_i^q \in [-1, 0] \) (for \( q \in \{ e, f \} \)), and \( c \geq 1 \). Now \( \ddot{c} \) becomes the only positive term in the primitive Matsuoka system in (B.11). According to Matsuoka’s derivation in [24, 26], from (B.11), the free-response oscillation amplitude of the Matsuoka oscillator can be written as

\[ A_n = \frac{\ddot{c}}{K^{-1}(K_n) + (a + b)L(K^{-1}(K_n))}. \]  
(B.12)

Assume the fundamental harmonic component of the vanilla action signal \( \alpha_x \) generated by RL policy has the following form:

\[ \alpha_{F_x} = A \cos(\omega t). \]

Then, substitute \( \alpha_{F_x} \) into (5), we have

\[ u_{F_x}^e \approx \frac{1}{1 + e^{-A \cos(\omega t)}}. \]  
(B.13)

And

\[ \ddot{u}_{F_x}^e \approx \frac{1}{1 + e^{-A \cos(\omega t)}} - 1. \]  
(B.14)

Because the sigmoid function in \( \ddot{u}_{F_x}^e \) is monotonically increasing with \( \alpha_{F_x} \), the frequency of \( \ddot{u}_{F_x}^e \) is the same as the frequency of \( \alpha_{F_x} \). The amplitude of \( \ddot{u}_{F_x}^e \) is

\[ A_u = \frac{\max_t (\ddot{u}_{F_x}^e(t)) - \min_t (\ddot{u}_{F_x}^e(t))}{2} = \frac{1}{2} e^A - 1. \]  
(B.15)

And the bias of \( \ddot{u}_{F_x}^e \) can be calculated as

\[ \bar{r}_u = \frac{\max_t (\ddot{u}_{F_x}^e(t)) + \min_t (\ddot{u}_{F_x}^e(t))}{2} = -\frac{1}{2} \bar{r}_a. \]  
(B.16)

It is noted that \( \ddot{u}_{F_x}^e \) and \( \ddot{u}_{F_x}^f \) are complementary to each other by Definition 1. Thus, \( \ddot{u}_{F_x}^e \) and \( \ddot{u}_{F_x}^f \) share the same amplitude and bias.

By taking time average of all variables in (B.11) and ignoring the coupling term from other primitive Matsuoka oscillator nodes, we have the equation of the amplitude of the inner state \( x_{F_x}^q \) (\( q \in \{ e, f \} \)) as

\[ A_x \mathcal{F}(r_x) + (a + b)L(r_x) = \ddot{c} + \bar{r}_u \approx \ddot{c} - \frac{1}{2}. \]  
(B.17)

Next, since (B.11) can be reduced to

\[ \tau_r \frac{d}{dt} x_{F_x} + x_{F_x} = a K(r_x) x_{F_x} - b y_{F_x} + \ddot{u}_{F_x}, \]
\[ \tau_a \frac{d}{dt} y_{F_x} + y_{F_x} = K(r_x) x_{F_x}, \]  
(B.18)

where \( \ddot{u}_{F_x} = \ddot{u}_{F_x}^e - \ddot{u}_{F_x}^f \). We derive the describing function from \( u_{F_x}(t) \) to \( x_{F_x}(t) \). Applying the Laplace transform to (B.18), we have

\[ G(s, A) = \frac{s}{\tau_r s + 1 - K(r_x)(a - \frac{b}{\tau_a s + 1})}. \]

More precisely, the frequency transfer function is

\[ G(\omega, A) = \frac{1}{1 + (\tau_r \tau_a \omega_n^2 - 1) \frac{K(r_x)}{K_n} + \tau_r \tau_a \omega^2 + (K_n - K(r_x)) \tau_a \omega}. \]  
(B.19)

\[ \omega_n = \frac{1}{\sqrt{\frac{\tau_r + \tau_a}{\tau_a}} - 1}. \]  
(B.20)

where \( \omega_n \) is the frequency of the imaginary component of \( \ddot{c} \). The gain in the frequency transfer function is given by \( |G(\omega, A)| \). Since the amplitude of \( x_{F_x}(t) \) is twice of \( A_x \), and the amplitude of \( u_{F_x}(t) \) is twice of \( A_u \), we have the relation between \( A_x \) and \( A_u \) expressed as

\[ A_x = |G(\omega, A)| A_u = |G(\omega, A)| A_u. \]  
(B.22)

Given (B.9), \( (\tau_r + \tau_a - \tau_r a K(r_x)) \) is the coefficient of first-order differential variable, also known as damping coefficient. When \( K(r_x) = K_n = \tau_r + \tau_a \), the original oscillation system is harmonic. For the damped oscillation system, the damping coefficient should be positive such that \( K(r_x) < K_n \), or equivalently, \( \frac{K(r_x)}{K_n} < 1 \). In this situation, there will be only forced-response oscillation, and all free-response oscillations diminish due to the positive damping. From (B.4) and (B.5), we know both \( K(\tau) \) and \( L(\tau) \) are monotonic, and therefore, \( K^{-1}(\tau) \) and \( L^{-1}(\tau) \) are monotonic as well. When \( K(r_x) < K_n \)

\[ A_n = \frac{\ddot{c}}{K^{-1}(K_n) + (a + b)L(K^{-1}(K_n))}, \]
\[ \leq \frac{\ddot{c}}{r_x + (a + b)L(r_x)} \]  
(B.23)

that is

\[ r_x + (a + b) L(r_x) < \frac{\ddot{c}}{A_n}. \]  
(B.24)

From the other end, let \( K_x = K(r_x) \), we have

\[ A_x = |G(\omega, A)| A_u \]
\[ = \frac{A_u \sqrt{\tau_a^2 + 1}}{\sqrt{1 + (\tau_r \tau_a \omega_n^2 - 1) \frac{K(r_x)}{K_n} + (K_n - K(r_x))^2 \tau_a^2 \omega^2}} \]
\[ \triangleq \frac{A_u \sqrt{\tau_a^2 + 1}}{\sqrt{1 + (\tau_r \tau_a \omega_n^2 - 1) U + (K_n - K(r_x))^2 \tau_a^2 \omega^2}} \]  
(B.25)

where \( U \triangleq \frac{K(r_x)}{K_n} \), and \( U \subseteq [0, 1] \). Next, define a function \( Q(U) \) as

\[ Q(U) \triangleq \left[ (\tau_r \tau_a \omega_n^2 - 1) - U \right]^2 - K_n^2 \frac{1 - U^2}{\tau_a^2 \omega_n^2} \]
\[ + K_n^2 \frac{1 - U^2}{\tau_a^2 \omega_n^2} \]  
(B.26)

When \( \omega > \omega_n \) and \( \tau_r \tau_a \omega_n^2 - 1 > 0 \), or \( \omega < \omega_n \) and \( \tau_r \tau_a \omega_n^2 - 1 < 0 \)

\[ Q_{\min}(U) = Q(1) = \tau_r^2 \tau_a^2 (\omega^2 - \omega_n^2)^2. \]  
(B.27)
Thus, when \( U \subseteq (0, 1) \) is satisfied, we have
\[
A_e < A_u \sqrt{\frac{\tau_2^2 \omega^2 + 1}{\tau_r \tau_n |\omega|^2 - \omega^2}}. \tag{B.28}
\]
Combining (B.28), (B.24), and (B.17), we have
\[
A_u \sqrt{\frac{\tau_2^2 \omega^2 + 1}{\tau_r \tau_n |\omega|^2 - \omega^2}} \frac{\dot{c}}{A_n} > \frac{\dot{c}}{2} > \frac{\dot{c}}{} - 1. \tag{B.29}
\]
Thus, we have
\[
A_u > \frac{\frac{\dot{c}}{} - 1}{\tau_r \tau_n |\omega|^2 - \omega^2} A_n = \frac{c}{\frac{\dot{c}}{} + 1} \triangleq A_0(c, \omega). \tag{B.30}
\]
Substitute \( A_n \) in the aforementioned equation with its approximation [24, (30)], we have
\[
A_0(c, \omega) \approx \frac{c}{\sqrt{\frac{\tau_2^2 \omega^2 + 1}{\tau_r \tau_n |\omega|^2 - \omega^2}}} (2K_n - 1 + \frac{2}{\pi} (a + b) \sin^{-1}(K_n)) \tag{B.31}
\]
Since \( c \geq 0 \), when \( \omega \) is fixed, \( A_0 \) linearly increases with \( c \).

**APPENDIX C**

**THEORY**

**A. Proof of Proposition 1**

**Proof:** As seen in (B.9), when \( u_i^c \) and \( u_i^f \) of the \( i \)th oscillator satisfy constant constraints in Problem 1, the tonic inputs become time-invariant, such that \( \frac{d}{dt} u_i(t) = 0 \). If the oscillation is harmonic \((K(\tau_x) = K_a)\), then (B.9) can be rewritten as
\[
\tau_r \tau_n \frac{d^2}{dt^2} x_i + (\tau_r + \tau_n - \tau_n a K_n) \frac{d}{dt} x_i + ((b - a)K_n + 1)x_i = 2u_i^c - 1 \tag{C.1}
\]
Then, the abovementioned equation can be interpreted as a nonhomogeneous spring-damper system with a constant load. By setting
\[
\dot{x}_i \triangleq x_i - \frac{2u_i^c - 1}{(b - a)K_n + 1}
\]
and substitute \( x_i \) with \( \dot{x}_i \) in (C.1), we can obtain its homogeneous form as
\[
\tau_r \tau_n \frac{d^2}{dt^2} \dot{x}_i + (\tau_r + \tau_n - \tau_n a K_n) \frac{d}{dt} \dot{x}_i + ((b - a)K_n + 1)\dot{x}_i = 0. \tag{C.2}
\]
Here, \( \dot{x}_i \) is the unbiased variable of \( x_i \), and thus, the bias of \( x_i \) naturally becomes
\[
\text{bias}(x_i) = \frac{2u_i^c - 1}{(b - a)K_n + 1} = \frac{1}{(b - a)K_n + 1} \text{bias}(u_i). \tag{C.3}
\]
Since \( z_i \) and \( x_i \) are entrained (Definition 2), \( z_i = z_i^e - z_i^f = g(x_i^e) - g(x_i^f) = K_n x_i \), we have
\[
\text{bias}(z_i) = K_n \text{bias}(x_i) = \frac{K_n}{(b - a)K_n + 1} \text{bias}(u_i). \tag{C.4}
\]
\square

**B. Applicable Range of Proposition 1**

Let \( x_i^e < 0, x_i^f > 0 \), from (1), we have
\[
z_i^e = \max(x_i^e, 0) = 0, z_i^f = \max(x_i^f, 0) = x_i^f.
\]
Thus
\[
z_i = z_i^e - z_i^f = -x_i^f.
\]
Since \( u_i^e \) and \( u_i^f \) are constants in Proposition 1, we have
\[
x_i^e + ax_i^f = u_i^e + c \tag{C.5}
\]
\[
x_i^f + bx_i^f = u_i^f + c. \tag{C.6}
\]
Let \( u_i = u_i^e - u_i^f, x_i = x_i^e - x_i^f \), the aforementioned two equations can be reduced to
\[
u_i = x_i^e + (1 + b - a)z_i. \tag{C.7}
\]
According to Definition 1, \( u_i^e + u_i^f = 1 \), then we have
\[
u_i^e = \frac{1 + u_i}{2}, u_i^f = \frac{1 - u_i}{2}.
\]
Substitute \( x_i^e \) in (C.5) with (C.6), and then substitute \( u_i^e, u_i^f \) with \( u_i \), we have
\[
x_i^e = \frac{1 + b + a}{2(1 + b)} u_i + \frac{1 + b - a}{1 + b} \left( \frac{1}{2} + c \right). \tag{C.8}
\]
Substitute the aforementioned equation of \( x_i^e \) to (C.7) to obtain
\[
u_i = \frac{1 + b + a}{2(1 + b)} u_i + \frac{1 + b - a}{1 + b} + (1 + b - a)z_i
\]
which can be rearranged to
\[
z_i = \frac{1}{2(1 + b)} u_i - \frac{1}{1 + b} \left( \frac{1}{2} + c \right), \quad (c \geq 0).
\]
Similarly, for the case when \( x_i^e < 0, x_i^f > 0 \), we have
\[
z_i = \frac{1}{2(1 + b)} u_i + \frac{1}{1 + b} \left( \frac{1}{2} + c \right), \quad (c \geq 0).
\]
Since \( z_i \) and \( u_i \) are both constants, \( \text{bias}(z_i) = z_i \) and \( \text{bias}(u_i) = u_i \). In summary, we have
\[
\text{bias}(z_i) = \left\{ \begin{array}{ll}
\frac{1}{2(1 + b)} \text{bias}(u_i) - \frac{1}{2(1 + b)} \left( \frac{1}{2} + c \right) & \left( x_i^e < 0, x_i^f > 0 \right) \\
\frac{1}{2(1 + b)} \text{bias}(u_i) + \frac{1}{2(1 + b)} \left( \frac{1}{2} + c \right) & \left( x_i^e > 0, x_i^f < 0 \right)
\end{array} \right. \tag{C.9}
\]

The derivation in this section shows that, when the value of \( u_i^e \) and \( u_i^f \) causes the Matsuda system fall into a quadrant such that \( x_i^e x_i^f < 0 \), the system converges to a set point equilibrium. At this moment the conclusion in Proposition 1 is not applicable to the system. The system should instead follow the relation described in (C.9).

The boundary case is at \( x_i^e = 0, x_i^f > 0 \) or \( x_i^e = 0, x_i^f > 0 \). For \( x_i^e = 0, x_i^f > 0 \), substitute \( x_i^e = 0 \) to (C.5) and (C.7), we can obtain the equation
\[
\text{bias}(u_i) = u_i = \frac{2a}{a + b + 1} - 1.
\]
Similarly when \( x_i^f > 0 \), \( x_i^e = 0 \), we have
\[
\text{bias}(u_i) = u_i = 1 - \frac{2a}{a + b + 1}.
\]

\section{Proof of Proposition 2}

\textbf{Proof:} For simplicity we denote \( A_q \triangleq A_{x^q} \) and \( r_q \triangleq r_{x^q} \) for \( q \in \{e, f\} \). Instead of looking into the relation between \( u_i \) and \( z_i \), we focus on the bias between the two states.

According to the \textit{perfect entrainment assumption} \cite{24} and Definition 2, let \( u_i \) be resonant to \( z_i \). We define the duty cycle of a wave function as \( D(\cdot) \). Let the period of \( z_i \) be \( T = 2\pi \) (a different value of \( T \) would not affect the result of calculation), based on the Fourier expansion, the bias of \( u_i \) can be expressed as
\[
\text{bias}(u_i) = \frac{1}{T} \int_{-T/2}^{T/2} u_i(t) dt = \frac{1}{2\pi} \int_{-\pi}^{\pi} u_i(t) dt = \frac{2}{2\pi} \int_{-\pi}^{\pi} u_i(t) dt - 1 = 2D(u_i^*) - 1. \quad (C.10)
\]

Because the bias terms of \( x_i \) and \( u_i \) are time-invariant, from (B.7), we can extract the bias component to form a new equation as follows:
\[
\text{bias}(x_i) = a \cdot \text{bias}(z_i) - b \cdot \text{bias}(y_i) + \text{bias}(u_i) \\
\text{bias}(y_i) = \text{bias}(z_i). \quad (C.11)
\]

Assume \( x_i \) can be approximated by its main sinusoidal component and the period of both \( x_i \) and \( z_i \) is represented by \( T \). From (B.1) and (B.2), we have
\[
\text{bias}(x_i) = \frac{1}{T} \int_{-T/2}^{T/2} x_i dt = \frac{1}{T} \int_{-T/2}^{T/2} \left( x_i^e - x_i^f \right) dt = \frac{1}{T} \int_{-T/2}^{T/2} A^e \cos(\omega t) + r^e - A^f \cos(\omega t) + r^f dt \\
= A^e r^e - A^f r^f
\]
\[
\text{bias}(z_i) = \frac{1}{T} \int_{-T/2}^{T/2} z_i dt = \frac{1}{T} \int_{-T/2}^{T/2} \left( z_i^e - z_i^f \right) dt = \frac{1}{T} \int_{-T/2}^{T/2} \left( A^e (K (r^e) \cos(\omega t) + L (r^e)) - A^f (K (r^f) \cos(\omega t) + L (r^f)) \right) dt \\
= A^e \left( L (r^e) - \frac{1}{\pi} \right) - A^f \left( L (r^f) - \frac{1}{\pi} \right) + \frac{1}{\pi} \left( A^e - A^f \right).
\]

Apply Taylor expansion on \( L(r) \) (Appendix B-B) at \( r = 0 \), we have
\[
L(r) = \frac{1}{\pi} + \frac{r}{2} + o(r), r \in (-1, 1).
\]
Then, we have
\[
\text{bias}(z_i) = \frac{1}{2} A^e r^e - \frac{1}{2} A^f r^f + \frac{1}{\pi} \left( A^e - A^f \right) = \frac{1}{2} \text{bias}(x_i) + \frac{1}{\pi} \left( A^e - A^f \right). \quad (C.12)
\]

According to \cite{24}, the amplitude \( A^q \) (for \( q \in \{e, f\} \)) has the form
\[
A^q = \frac{\text{bias}(u^q)}{r^q + (a + b)L(r^q)} + c
\]

When the system is harmonic, according to \cite{24}, (30), we have
\[
\text{bias}(u_i) = \frac{\text{bias}(u^q)}{r^q + (a + b)L(r^q)} \approx \frac{\text{bias}(u_i)}{2K_n - 1 + \frac{1}{2}(a + b)\sin^{-1}(K_n)}.
\]

Let \( m = \frac{1}{2}K_n^{-1} + \frac{1}{2(a + b)\sin^{-1}(K_n)} \), (C.12) can be rewritten as
\[
\text{bias}(z_i) = \frac{1}{2} \text{bias}(x_i) + mbias(u_i). \quad (C.15)
\]

Substitute bias \( (z_i) \) in (C.11) with (C.15), we can obtain the pure relation between bias \( (x_i) \) and bias \( (u_i) \) as
\[
(1 - m(b - a))\text{bias}(u_i) = \left( \frac{1}{2} (b - a) + 1 \right) \text{bias}(x_i). \quad (C.16)
\]
In this case, the relation between bias \( (z_i) \) and bias \( (u_i) \) can be expressed as
\[
\text{bias}(z_i) = \frac{1 - m(b - a)}{b - a + 2} \text{bias}(u_i) + mbias(u_i) = \frac{1 + 2m}{b - a + 2} \text{bias}(u_i). \quad (C.17)
\]

\section{References}

\begin{thebibliography}{99}
\bibitem{1} C. Majidji, “Soft robotics: A perspective—Current trends and prospects for the future,” \textit{Soft Robot.}, vol. 1, no. 1, pp. 5–11, 2014.
\bibitem{2} A. J. Ijspeert, “Central pattern generators for locomotion control in animals and robots: A review,” \textit{Neural Netw.}, vol. 21, no. 4, pp. 642–653, 2008.
\bibitem{3} A. Roberts, S. Soffe, E. Wolf, M. Yoshida, and F.-Y. Zhao, “Central circuits controlling locomotion in young frog tadpoles,” \textit{Ann. New York Acad. Sci.}, vol. 860, no. 1, pp. 19–24, 1999.
\bibitem{4} R. Yuste, J. MacLean, I. Smith, and A. Lansner, “The cortex as a central pattern generator,” \textit{Nature Rev. Neurosci.}, vol. 6, no. 6, pp. 477–483, 2005.
\bibitem{5} T. Mori, Y. Nakamura, M.-A. Sato, and S. Ishii, “Reinforcement learning for CPG-driven biped robot,” in \textit{Proc. AAAI Conf. Artif. Intell.}, vol. 2004, no. 4, pp. 623–630.
\bibitem{6} G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, “Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot,” \textit{Int. J. Robot. Res.}, vol. 27, no. 2, pp. 213–228, 2008.
\bibitem{7} J. Nassour, P. Hénaff, F. Benouezdou, and G. Cheng, “Multi-layered multi-pattern CPG for adaptive locomotion of humanoid robots,” \textit{Biol. Cybern.}, vol. 108, no. 3, pp. 291–303, 2014.
\bibitem{8} F. Dreladini, N. Air-Bouzid, and A. Ijspeert, “CPG-based control of humanoid robot locomotion,” \textit{Humanoid Robotics: A Reference}, A. Goswami, P. Vadakkepat, Eds., Springer, 2017.
\bibitem{9} D. H. Tran, F. Hamker, and J. Nassour, “A humanoid robot learns to recover perturbation during swinging motion,” \textit{IEEE Trans. Syst., Man, Cybern. Syst.}, vol. 50, no. 10, pp. 3701–3712, Oct. 2020.
\bibitem{10} A. Crespi, A. Badertscher, A. Guignard, and A. J. Ijspeert, “Swimming and crawling with an amphibious snake robot,” in \textit{Proc. IEEE Int. Conf. Robot. Autom.}, 2005, pp. 3024–3028.
\bibitem{11} A. Crespi and A. J. Ijspeert, “Online optimization of swimming and crawling in an amphibious snake robot,” \textit{IEEE Trans. Robot.}, vol. 24, no. 1, pp. 75–87, Feb. 2008.
\end{thebibliography}
A. J. Ijspeert, J. Hallam, and D. Willshaw, “Evolving swimming controllers,” *Biorob. Behav.*, vol. 31, no. 12, 2017, Art. no. 053001.

Z. Wang, Q. Gao, and H. Zhao, “CPG-inspired locomotion control for a snake robot basing on nonlinear oscillators,” *J. Intell. Robot. Syst.*, vol. 85, no. 2, pp. 209–227, 2017.

Z. Bing, Z. Jiang, L. Cheng, C. Cai, K. Huang, and A. Knoll, “End to end learning of a multi-layered SNN based on R-STD for a target tracking snake-like robot,” in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 9645–9651.

X. Wu and S. Ma, “Neurally controlled steering for collision-free behavior of a snake robot,” *IEEE Trans. Control. Syst. Technol.*, vol. 21, no. 6, pp. 2443–2449, Nov. 2013.

G. Sartoretti, W. Paivine, Y. Shi, Y. Wu, and H. Choset, “Distributed learning of decentralized control policies for articulated mobile robots,” *IEEE Trans. Robot.*, vol. 35, no. 5, pp. 1109–1122, Oct. 2019.

G. Bellegarda and A. Ijspeert, “CPG-RL: Learning central pattern generators for quadruped locomotion,” *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 12547–12554, Oct. 2022.

R. S. Sutton, D. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” *Adv. Neural Inf. Process. Syst.*, MIT Press, vol. 12, 2000.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017, arXiv:1707.06347.

K. Matsuoka, “Sustained oscillations generated by mutually inhibiting neurons with adaptation,” *Biol. Cybern.*, vol. 56, no. 5/6, pp. 367–376, 1985.

T. G. Brown, “The intrinsic factors in the act of progression in the mammal,” *Proc. Roy. Soc. B: Biol. Sci.*, vol. 84, no. 572, pp. 308–319, 1911.

K. Matsuoka, “Mechanisms of frequency and pattern control in the neural rhythm generators,” *Biol. Cybern.*, vol. 56, no. 5/6, pp. 345–353, 1987.

K. Matsuoka, “Analysis of a neural oscillator,” *Biol. Cybern.*, vol. 104, no. 4/5, pp. 297–304, 2011.

K. Matsuoka, “Frequency responses of a neural oscillator,” 2013. [Online]. Available: https://matsuokai.jimdofree.com/app/download/7896851691/Frequency_Response_jimdo.pdf?u=1378121835

X. Liu, R. Gasoto, Z. Jiang, C. Onal, and J. Fu, “Learning to locomote with artificial neural-network and CPG-based control in a soft snake robot,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 7758–7765.

M. Luo et al., “Motion planning and iterative learning control of a modular soft robotic snake,” *Front. Robot. AI*, vol. 7, 2020, Art. no. 599242.

M. Luo et al., “Toward modular soft robotics: Propropriocutavre curvature sensing and sliding-mode control of soft bidirectional bending modules,” *Soft Robot.*, vol. 4, no. 2, pp. 117–125, 2017.

M. Luo, M. Agelli, and C. D. Onal, “Theoretical modeling and experimental analysis of a pressure-operated soft robotic snake,” *Soft Robot.*, vol. 1, no. 2, pp. 136–146, 2014.

R. Gasoto et al., “A validated physical model for real-time simulation of soft robotic snakes,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 6272–6279.

A. J. Ijspeert, J. Hallam, and D. Willshaw, “Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology,” *Adaptive Behav.*, vol. 7, no. 2, pp. 151–172, 1999.

G. Wang, X. Chen, and S.-K. Han, “Central pattern generator and feedforward neural network-based self-adaptive gait control for a crab-like robot locomoting on complex terrain under two reflex mechanisms,” *Int. J. Adv. Robot. Syst.*, vol. 14, no. 4, 2017, doi: 10.1177/1729981417725440.

J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *Proc. Int. Conf. Intell. Robots Syst.*, 2017, pp. 23–30.

R. S. Sutton, D. Precup, and S. Singh, “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning,” *Artif. Intell.*, vol. 112, no. 1/2, pp. 181–211, 1999.

D. Precup, “Temporal abstraction in reinforcement learning,” Ph.D. dissert., Univ. Massachusetts Amherst, Amherst, MA, USA, 2000.

P. Dhariwal et al., “OpenAI baselines,” 2017. [Online]. Available: https://github.com/openai/baselines

X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization,” in *Proc. Int. Conf. Robot. Autom.*, 2018, pp. 1–8.