Biologically-Inspired Representations for Adaptive Control with Spatial Semantic Pointers

1st Graeme Damberger
Centre for Theoretical Neuroscience
University of Waterloo
Ontario, Canada
graeme.damberger@uwaterloo.ca

2nd Kathryn Simone
Centre for Theoretical Neuroscience
University of Waterloo
Ontario, Canada
kathryn.simone@uwaterloo.ca

3rd Chandan Datta
Project JunoIntelligence
University of Auckland
Auckland, New Zealand
cdat003@aucklanduni.ac.nz

4th Ram Eshwar Kaundinya
Cognigron
University of Groningen
Groningen, Netherlands
0009-0007-7935-4897

5th Juan Escareno
XLIM Research Institute UMR-CNRS 7252
University of Limoges
Limoges, France
juan.escareno-castro@xlim.fr

6th Chris Eliasmith
Centre for Theoretical Neuroscience
University of Waterloo
Ontario, Canada
celiasmith@uwaterloo.ca

2025 Neuro Inspired Computational Elements (NICE) | 979-8-3315-0302-4/25/$31.00 ©2025 IEEE | DOI: 10.1109/NICE65350.2025.11065084

***Abstract*—We explore and evaluate biologically-inspired representations for an adaptive controller using Spatial Semantic Pointers (SSPs). Specifically, we show our method for place-cell-like SSP representations outperforms past methods. Using this representation, we efficiently learn the dynamics of a given plant over its state space. We implement this adaptive controller in a spiking neural network along with a classical sliding mode controller and prove the stability of the overall system despite non-linear plant dynamics. We then simulate the controller on a 3-link arm and demonstrate that the proposed representational method gives a simpler and more systematic way of designing the neural representation of the state space. Compared to previous methods, we show an increase of 1.23-1.25x in tracking accuracy.**

I. Introduction

Control systems are essential for regulating and steering the behaviour of dynamic systems and are relevant to numerous application areas, such as robotics and automation. At its core, controller design concerns formulating a control law that defines the relationship between the system inputs and desired outputs to achieve some performance objective regarding stability, accuracy, and efficiency. Among classical control design techniques, sliding mode control is widely studied due to its robustness towards disturbances and strong performance in non-linear applications. A sliding mode controller applies a feedback-control loop to continuously sample a plant's state to generate a control signal to achieve desired dynamics (1). A sliding variable is introduced in the state space, defining a sliding manifold on which the plant exhibits the desired behaviour. The controller moves the state onto this manifold by controlling the sliding variable to enforce stable plant dynamics. To effectively design this controller, it is assumed that the plant dynamics are known so that the controller can negate their influence. However, given unknown or time-varying dynamics, the controller must adapt. Therefore, an adaptive component can be introduced in parallel to the basic controller as a form of forward control (2). Whereas classically, the adaptive component is structured to estimate the parameters of known dynamics (2), it is also possible to learn an approximation of the unknown dynamics.

Artificial neural networks (ANNs), as universal function approximators (3), provide a convenient parametrization for the approximation of such dynamics. Such structures have become ubiquitous tools in the reinforcement learning paradigm (4), where optimization most often involves back-propagation of error to learn all the parameters of the network simultaneously. However, the best performance is achieved in offline settings (that is, gradient updates are computed from a batch of samples (5)). These optimization approaches are incompatible with the adaptive control problem, where the network must update in real-time. An alternative approach involves structuring the adaptive component so that it can learn an approximation of the dynamics over a fixed set of basis functions with an online learning rule (6). In these schemes, the functional approximation of the network is limited to the parametrization of the output weights over fixed basis functions. As such, the choice in the set of basis functions used is critical to the performance of online learning of the dynamics, as the functions that can be learned may be limited by the basis functions selected. However, a well-chosen set of basis functions will allow for effective learning of arbitrary nonlinear dynamics.

Real brains must deal with a wide array of adaptive control problems in order to support behaviour that allows animals and humans to survive and thrive in changing environments. Unsurprisingly, neuromorphic adaptive control is a highly-active area of research, striving for efficient online learning that can update its model of the system in real-time using event-based processing. DeWolf et al, demonstrated adaptive control of an arm with a simulated network of spiking neurons (6), which was later implemented on neuromorphic hardware (7). In these models, the basis functions for learning are defined by the tuning curves of neurons in the neural network. The input weights, or encoders, of these neurons, are typically drawn from a uniform distribution defined over the domain of the state space (8). A careful selection of encoders that structures the basis functions to improve learning was considered in (9) such that the state space was projected to a higher dimensional hyper-sphere. A key finding was that to learn effectively over space, the encoders must be chosen to sparsely represent the state space. Sparseness – the property of a network that only a small number of neurons are active for a given input - is observed in many biological systems. Hippocampal place cells (10), entorhinal grid cells (11), and Kenyon cells of the insect mushroom body (12) all display significant sparseness.

Spatial Semantic Pointers (SSPs) are a high-dimensional vector representation of lower-dimensional continuous spaces (13; 14; 15), developed within the framework of the Semantic Pointer Architecture (16), a method for building models of biological cognition. SSP feature spaces have been exploited to engineer hidden units with activity patterns that resemble those observed in real brains, such as place cells (17), grid cells (14), (18), and Kenyon cells (19). In a separate line of research, an approach for engineering hidden units that behave like grid cells over a 2D space, in that they exhibit a hexagonally-tiled activity patterns, was developed using Fourier basis functions spaced $120\deg$ apart (20), (21), resulting in a 7-dimensional representation. In previous work, these grid-cell like features were used as an input layer to a deep Q-network, and found to improve sample efficiency on a continuous control problem in a reinforcement learning setting (22). However, to our knowledge, the efficacy of biologically-grounded representations in an adaptive controller has yet to be explored.

Here we introduce a novel, biologically-inspired method for implementing adaptive control that leverages Spatial Semantic Pointers (SSPs)(23) to form a basis representation for learning state based dynamics. By merging this dynamic, brain-inspired adaptive component with a classical sliding mode controller, we build upon past work in neuromorphic control (6) and propose a new biologically inspired method to design the adaptive basis using Spatial Semantic Pointers (SSP). Specifically, we use SSPs to generate place-cell-like, Kenyon-cell-like, and grid-cell-like representations over the state space. Our approach offers a systematic and simpler design process of the state space representation by using spiking neural networks to represent system states in a high-dimensional, sparse manner as an adaptive basis for robust controllers, resulting in several key contributions:

1) We propose the use of SSPs as basis function representations over state space for adaptive control.
2) We implement these SSP representations and a learning rule at the neural level using the Neural Engineering Framework (NEF) (8) to construct a neuromorphic controller that leverages the learning capabilities and power efficiency of spiking neural networks.
3) We prove both the guarantee of stability using SSP representations in a controller, and the convergence of the network's learning process to the desired outcome.
4) Tracking performance is evaluated in a simulation of a 3-link arm and bench-marked against two previously-proposed neuromorphic controllers.
5) We evaluate different methods for representing the online learned model and show that SSPs structured as place cells provide a $1.23-1.25$x performance improvement over past methods.

In the remainder of the paper, we begin by covering the theoretical background and prerequisites in Section II and present the sliding mode dynamics and SSP structure in Section III. We then prove the stability of the dynamics in the context of spiking neurons and introduce the controller structure. In Section IV we present the results of hyper-parameter tuning and compare the performance of our new approach to past work in simulation.

II. THEORETICAL PREREQUISITES

A. Function approximation

In this work, we are concerned with approximating arbitrary dynamics of a system online with a neural network. For real-world systems, it is reasonable to assume that the mathematical model representing the system's dynamics is not perfectly known and, in some cases, may be completely unknown. Additionally, such systems are often exposed to external disruptions that affect their performance. These uncertainties can be categorized into two types: parametric uncertainties, which are typically state-dependent (vanishing), and exogenous disturbances, which are slow/fast time-varying (persistent). For the context of this work, we assume all disturbances are state-based so the switching component of a traditional sliding mode controller can be neglected. However, the inclusion of such a component would be straightforward given a bound on time-based disturbances.

Lemma 1. *Let $\mathcal{F}(\boldsymbol{x}(t))$ be a smooth function whose inputs $\boldsymbol{x}(t)$ belong to the compact set Ω. There exists a neural network (NN) satisfying*

$$\mathcal{F}(\boldsymbol{x}(t)) = \boldsymbol{\omega}^T \boldsymbol{\phi}(\boldsymbol{x}(t)) + \varepsilon \tag{1}$$

where $\boldsymbol{\phi}(\boldsymbol{x}(t)) \in \mathbb{R}^\rho$ represents a suitable set of basis functions spanned by the network, $\boldsymbol{\omega} \in \mathbb{R}^\rho$ the weight vector of the neural network of ρ dimension and ε denotes the approximation error. Let the optimal weight $\boldsymbol{\omega}^\star$

$$\boldsymbol{\omega}^\star := \underset{\boldsymbol{\omega}\in\Omega}{\arg\min} \left\{ \|\mathcal{F}(\boldsymbol{x}) - \boldsymbol{\omega}^T \boldsymbol{\phi}(\boldsymbol{x})\|^2 \right\} \tag{2}$$

where it's upper-bounded by $\|\boldsymbol{\omega}^\star\| \leq \bar{\omega}$, $\|\boldsymbol{\varphi}(\cdot)\| \leq \bar{\phi}$ and $|\varepsilon| \leq \bar{\varepsilon}$, with $\bar{\omega}$, $\bar{\phi}$, and $\bar{\varepsilon}$ are constants $\in \mathbb{R}^+$.

B. Sliding Manifold

The objective is to reach and remain into the manifold s which is written as

$$s = (\frac{d}{dt} + \lambda)^{d-1}, \tag{3}$$

where d is the order of the controlled dynamics and a chosen parameter λ, a positive constant that shapes the convergence speed of the error. To this end, two phases are distinguished:

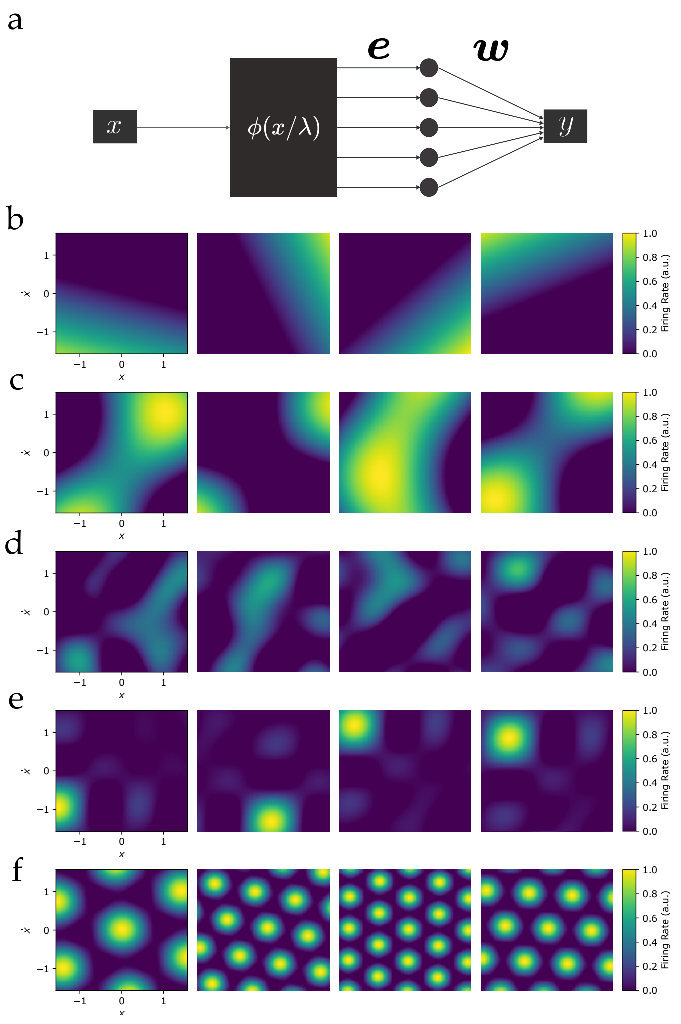


Figure 1: **a** Architecture of the neural network providing the adaptive component of the controller, depicting the input x, SSP projection $\phi(x/\lambda)$, encoders $\boldsymbol{e}$ and weights $\boldsymbol{w}$. Note that in this work only $\boldsymbol{w}$ are learned. **b-f**: Examples of receptive fields of neurons in the hidden layer serving as the basis functions for the adaptive component. Shown are those for the two baselines, Random Encoders (**b**) and the Selected Basis (9) (**c**), as well as the three biologically-inspired basis functions evaluated in this work: Kenyon cells (**d**), place cells (**e**), and grid cells (**f**).

(i) *reaching phase*, where the state trajectories are driven towards the manifold s, and *the sliding phase* where the state trajectories are confined within this manifold by rendering the manifold invariant, i.e.

$$\dot{s} = 0 \tag{4}$$

III. Methods

A. Neural Network Architecture

The architecture of the spiking neural network that realizes adaptive control is shown in Fig.1. The input to the network is the state of the system, $\boldsymbol{x}(t)$, which is projected to a fixed, high-dimensional feature space of *Spatial Semantic Pointers* or *SSP*s, denoted by $\phi(x)$. These SSPs are then provided as inputs to a single hidden layer of neurons. While this network has two sets of weights, we depart from the widely used method involving backpropagation of error to learn both the input and output weights simultaneously. Instead, we select and fix the input weights, which leaves only the output weights to be learned. This scheme avoids many of the non-convex objective issues that arise in a deep neural network, and enables selecting the input weights to mimic the activity patterns observed in real neurons in the brain. We follow the convention of the Neural Engineering Framework and refer to the input weights as the *encoders*, and as the feature space spanned by their activities enables decoding arbitrary functions of the input, we refer to the output weights as the *decoders*. In what follows, we describe the SSP-embedding in detail, and present three encoder selection schemes to generate biologically-inspired basis functions. As baselines, we also implement two previously-characterized adaptive controller networks that do not use the SSP embedding (9).

1) Spatial Semantic Pointers: We develop our model using a locally-smooth embedding of continuous data explored in the Vector Symbolic Algebras (VSAs) literature for cognitive modeling applications (17). VSAs are a family of algebras that bridge connectionist (neural network) and symbolic theories of cognition by suggesting that concepts are represented as high-dimensional vectors that can be manipulated through a set of semantically-meaningful algebraic operations.

The Holographic Reduced Representation (HRR; (24)) VSA is distinguished from most VSAs for its support for representing data defined over continuous feature spaces. In the HRR literature, the method for representing continuous-valued data developed by Plate (25) is referred to as fractional binding (23) or fractional power encoding (26). We follow the convention of Komer et al. (17) and refer to the high-dimensional vectors resulting from these embeddings as Spatial Semantic Pointers (SSPs).

Briefly, input data $\mathbf{X}$ from a continuous domain of any dimensionality m is projected into a vector space of dimensionality d via mapping ϕ:

$$\phi_{\mathbf{X}}(\boldsymbol{\lambda}^{-1}\mathrm{X}) = \mathcal{F}^{-1}\left\{ \mathrm{e}^{\mathrm{i}\Theta\boldsymbol{\lambda}^{-1}\mathbf{X}} \right\} , \tag{5}$$

where $\mathcal{F}^{-1}$ denotes the inverse Fourier transform. The matrix λ is a diagonal, non-negative matrix whose entries $(\lambda_1, ..., \lambda_m)$ are user-specified parameters defining the length scales of the representation for each feature, such that prescaling data by λ ensures that points separated by less than λ along one feature in the domain will have high similarity under the dot product. We refer to Θ as the *phase matrix.* It is a $d \times m$ matrix, constructed of a set of column vectors θ_j, each comprising a collection of frequencies. We impose conjugate symmetry on θ_js to ensure that the inverse Fourier transform is real-valued. Beyond these constraints, the elements of Θ can be selected in many different ways. In this paper, we use two different constructions to define two embeddings of the state space and use the terms Rand SSPs and Hex SSPs (18) when referring to the hypervectors that reside in these two latent spaces.

One way to select the elements of Θ is randomly, such that its elements correspond to frequencies sampled from some power spectral density function. In this work, we draw elements $\theta_{i,j}$, where $i = 1, ..., \frac{d-1}{2}$, from a uniform distribution over the range $[-\pi, \pi]$ with $\theta_{0,j} = 1$. Due to the randomness inherent in this method for constructing Θ, observations embedded in this fashion are referred to as *Random* SSPs (RandSSPs). The projection is mathematically similar to that associated with Random Fourier Features (RFFs) (27) in kernel methods. The dot product between RandSSPs approximates a normalized sinc function (28):

$$k(x, x') = \phi(x) \cdot \phi(x') \tag{6}$$

$$\approx \frac{\sin(\pi|x - x'|)}{(\pi|x - x'|)}. \tag{7}$$

The phase matrix Θ may also be constructed deterministically. grid cell-like features may be constructed from a set of three mutually-orthogonal Fourier basis functions (21), (20). A simplex V, which is a $(d+1) \times d$ matrix, is chosen to minimize mutual dot products among its rows: $\sum_{i}^{d+1} \sum_{j=1, i \neq j}^{d+1} \mathbf{v}_i \cdot \mathbf{v}_j$, where $\mathbf{v}_i$ and $\mathbf{v}_j$ are rows i and j from V, and are unit vectors, and d is the dimensionality of the data. To ensure that the inverse Fourier transform is real-valued, a conjugate-symmetric matrix, $\bar{V}$ is then derived from V, by placing its components in complex-conjugate pairs in Fourier space. For $d = 2$ (2-dimensional space), each pair of conjugate components in $\bar{V} \in \mathbb{C}^7$ defines a plane wave with a corresponding wave vector u, such that the similarity between two points $x, x' \in \mathbb{R}^d$ follows an oscillatory pattern $k(x, x') = e^{iu\cdot(x-x')}$. The plane waves are oriented 120° apart, so the superposition of all three generates a hexagonally-tiled interference pattern.

This hexagonally-tiled interference pattern closely resembles the firing patterns of grid cells in the mammalian medial entorhinal cortex (MEC). However, these cells exhibit spatial tuning with varying orientations and resolutions (11). In this work, we employ HexSSPs (18) proposed by Dumont and Eliasmith (18), which extend this framework to leverage this diversity for improved spatial encoding. Briefly, this approach applies linear transformations (scaling and rotations) to the simplex matrix V, yielding a set of transformed matrices $V_n = s\mathbf{R}V, s \in \mathcal{S}, \mathbf{R} \in \mathcal{R}$, where scaling matrix s and rotation matrix $\mathbf{R}$ control the spatial resolution and orientation, respectively. The full phase matrix is constructed by stacking these transformed matrices:

$$\Theta = \text{stack}\left(\{V_n\} = \{s\mathbf{R}\mathbf{V}\}\right) \tag{8}$$

For 2-dimensional data, the resulting phase matrix Θ defines a high-dimensional mapping $\phi : \mathbb{R}^2 \mapsto \mathbb{R}^{6N+1}$ where, N is the number of simplex transformations. The inner product between vectors in the transformed space induces a superposition of plane wave kernels, providing an accurate basis for spatial encoding (18).

2) Engineered Encoders: The SSP embedding together with the encoders play a crucial role in shaping the activity patterns in the hidden layer. The activity of the n^{th} neuron associated with observation of system state x is determined by:

$$a_n = \text{LIF}(\alpha_n \phi_{\mathbf{X}}(\boldsymbol{\lambda}^{-1}x) \cdot e_n - \xi_n)), \tag{9}$$

where LIF denotes the activation function of a leaky integrate and fire neuron, e_n is the neuron encoder or the preferred direction vector, α_n is the gain, and ξ_n is the intercept of the n^{th} neuron.

Place Cells: If the encoders e_n are selected to be RandSSPs (that is, the preferred direction vectors of the neurons lie within the manifold of the embedding) the controller will learn using a basis of approximate rectified sinc functions. To generate such encoders, we draw samples from a uniform distribution over some bounded region of the state space $\boldsymbol{X}$, and turn these samples into RandSSPs (i.e., SSPs with a constant randomly chosen phase matrix). For simplicity, we set $\alpha_n = 1$ and apply a common $\xi_n = \xi$ across the population. We can select ξ to control the level of expected sparsity in the population. As shown in Figure 1b, these neurons are most active when the state is within a small, localized region within the bounds of the domain. In this paper, we refer to these neurons as place cells, due to the similarity of their receptive fields with neurons of the same name found in the mammalian hippocampus (10).

Kenyon Cells: The place cell representation constrains the region of the state space over which the dynamics can be learned, due to the sampling bounds. Building an effective controller therefore requires knowledge about the regions of the state space the system is likely to visit. To avoid needing such knowledge, another possibility is to sample preferred directions from the surface of the hypersphere: $e_n \sim U(S^{D-1})$. As shown in Fig.1d, this has the effect of reducing the extent to which an individual neuron's receptive field is localized to any particular area, while producing a population of neurons with apparently de-correlated receptive fields. The basis functions resulting from this choice were analyzed by Stewart et al (19). Briefly, these basis functions can be understood by decomposing e_n into a sum of B vectors that are non-orthogonal with the manifold $\phi(x_{b,\parallel})$ and a sum of C vectors that are pseudo-orthogonal to the manifold $\phi(x_{c,\perp})$, with respective scaling factors α_b and α_c: $e_n = \sum_{b=1}^{B} \alpha_b \phi(x_{b,\parallel}) + \sum_{c=1}^{C} \alpha_c \phi(x_{c,\perp})$. A third class of vectors are exactly orthogonal, which are omitted from consideration. Due to the high-dimensional vector representation, the contribution of pseudo-orthogonal components is negligible, and the neural response approximates the sum of contributions of vectors aligned with the manifold:

$$a_n \approx \text{LIF}\left(\phi(x/\lambda) \cdot \sum_{b=1}^{B} \alpha_b \phi(x_{b,\parallel}) - \xi\right) \tag{10}$$

The dot product between vectors in this latent space induces a *sum* of approximate sinc kernels in the domain. As the sinc kernel is defined over an infinite domain, so to is a sum of sinc kernels, yielding a necessarily unbounded representation. Figure 1c shows the effect of this random projection on the activity patterns of neurons over different regions of the state space. The unstructured projection and resulting receptive fields bears some resemblance to early characterizations of Kenyon Cells in the insect mushroom body (29), and we thus name these cells accordingly.

Grid Cells: The hexagonally-tiled interference pattern of grid-cell-like features introduces a structured oscillatory basis that spans the entire state space. Like Kenyon Cells, these basis functions do not constrain the region over which the system can generalize. To produce a neuron with a grid-cell-like firing pattern in a 2-dimensional space while using the HexSSP representation described above, the components of Θ corresponding to a single hexagonal interference pattern must be isolated (18). To achieve this, each neuron is randomly assigned to one of N simplex transformations, s_i and is also assigned a random spatial location, x_i, to introduce a unique phase shift. The encoder vector e_i defining the receptive field of the i^{th} neuron is then computed as

$$e_i = \mathcal{F}^{-1}\{r_i\}, \tag{11}$$

where $\mathcal{F}^{-1}$ denotes the inverse discrete Fourier transform, and r_i is given by:

$$r_i = \begin{cases} 1, & j = 0 \\ e^{i\Theta[j]\cdot x_i}, & j \in J^+ \\ e^{i\Theta[k]\cdot x_i}, & k \in J^- \\ 0, & \text{otherwise} \end{cases} . \tag{12}$$

Here, J^+ and J^- collectively define the indices of the Fourier components in the HexSSP phase matrix Θ associated with the selected simplex s_i.

B. Controller Design

Adapting the techniques derived in (2) which was further applied to spiking neural networks (6), let us consider the second-order dynamics of a n-link manipulator robot in the following form:

$$\ddot{\boldsymbol{q}} = f(\boldsymbol{q}, \dot{\boldsymbol{q}}) + g(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{u}_q, \tag{13}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ stands for the n-link manipulator angular positions, and $\boldsymbol{u}_q$ stands for the control input. The additive uncertainty in the dynamics is defined as $f(\boldsymbol{q}, \dot{\boldsymbol{q}})$, which encompasses both coriolis/centripetal effects and gravitational acceleration. We also define $g(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ as the control effectiveness matrix for a given manipulator. For the sake of simplicity of the subsequent stability analysis, we assume that $f(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is unknown and $g(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is known. Since the control objective is to render the trajectories of the state vector $\boldsymbol{q}$ towards a desired equilibrium point defined by $\bar{\boldsymbol{q}}_{ref} = (\boldsymbol{q}_{ref}, \dot{\boldsymbol{q}}_{ref})^T$, let us propose the sliding surface:

$$\boldsymbol{s}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \dot{\boldsymbol{e}} + \lambda \boldsymbol{e}. \tag{14}$$

with $\boldsymbol{s} \in \mathbb{R}^n$ where $\boldsymbol{e}$ and $\dot{\boldsymbol{e}}$ are the proportional and velocity error of the state with respect to the reference signal vector $\boldsymbol{q}_{ref}$. Specifically, we define $\boldsymbol{e} = \boldsymbol{q}_{ref} - \boldsymbol{q}$ and $\dot{\boldsymbol{e}} = \dot{\boldsymbol{q}}_{ref} - \dot{\boldsymbol{q}}$. We substitute (13) into (14) after taking the derivative with respect to time, finding that the dynamics of the sliding surface $\dot{\boldsymbol{s}}$ can be expanded into:

$$\dot{\boldsymbol{s}} = \ddot{\boldsymbol{q}}_{ref} - f(\boldsymbol{q}, \dot{\boldsymbol{q}}) - g(\boldsymbol{q}, \dot{\boldsymbol{q}})u + \lambda \dot{\boldsymbol{e}}. \tag{15}$$

We desire that the control ensures the error state dynamics reach the sliding manifold. This implies $\dot{\boldsymbol{s}} = 0$, which leads to convergence of $\boldsymbol{e}$ and $\dot{\boldsymbol{e}}$. Therefore, we choose a control signal u in the form:

$$\boldsymbol{u} = g(\boldsymbol{q}, \dot{\boldsymbol{q}})^{-1}(-\hat{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \ddot{\boldsymbol{q}}_{ref} + \lambda \dot{\boldsymbol{e}} + k\boldsymbol{q}), \tag{16}$$

where $k > 0$ is some controller gain, and $\hat{f}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is an approximation of the dynamics $f(\boldsymbol{q}, \dot{\boldsymbol{q}})$. With the control signal 16 we can describe the dynamics of 15 by:

$$\dot{s}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = (\hat{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - f(\boldsymbol{q}, \dot{\boldsymbol{q}})) - ks(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{17}$$

Using the property introduced in (30), and leveraged for spiking neural networks in (6), we can approximate the non-linear dynamics $f(\boldsymbol{q}, \dot{\boldsymbol{q}})$ by letting $\hat{f}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ be a linear combination of N basis functions $\phi(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and corresponding weights $\boldsymbol{\omega} \in \mathbb{R}^N$, such that

$$\hat{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\omega}^T \boldsymbol{\phi}(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{18}$$

We now assume that there exists an optimal set of weights $\omega^* \in \mathbb{R}^N$ such that

$$f(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\omega}^{*T} \boldsymbol{\phi}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \varepsilon,$$

with $\boldsymbol{\varepsilon} \in \mathbb{R}^m$ standing for the approximation error and holding $\|\varepsilon\| \leq \bar{\varepsilon}$, where $\bar{\varepsilon}$ is some converged approximation error upper bound. We then define a weight error term, $\tilde{\boldsymbol{\omega}}$, thus allowing 17 to be reformulated into

$$\dot{\boldsymbol{s}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \tilde{\boldsymbol{\omega}}^T \phi(\boldsymbol{q}, \dot{\boldsymbol{q}}) - ks(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \varepsilon. \tag{19}$$

C. Stability Analysis

In this work, we consider novel representations for adaptive control. However, to our knowledge, stability analysis for these specific bases have yet to be conducted in the literature, and it is therefore unclear if such representations can provide a suitable basis for learning a stable control law. The following proof of stability follows the convention of traditional control theory work, such that we are able to guarantee stability with the given controller. Additionally, for the application of machine learning, we show both that the usage of SSP as encoder representations provides a stable architecture, and that with the given control law we can guarantee convergence.

We now show that both the sliding variable $s(x, \dot{x})$ and the approximation weight error term $\tilde{w}$ are stable using the Lyapunov method. We apply the general structure of 13 in joint space $\boldsymbol{q}^n \in \mathbb{R}^n$ with the goal of controlling the dynamics of the end-effector of the n-link manipulator in operational space $\boldsymbol{x} = [x_1, ..., x_m]^T \in \mathbb{R}^m$, where $\dot{\boldsymbol{x}} = [\dot{x}_1, ..., \dot{x}_m]^T$. We introduce reference dynamics $\boldsymbol{x}_{ref}$ and the Jacobian mapping between operational and joint space:

$$J_{\boldsymbol{x}}(\boldsymbol{q}) = \begin{bmatrix} \frac{\partial x_1}{\partial q_1} & \cdots & \frac{\partial x_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_m}{\partial q_1} & \cdots & \frac{\partial x_m}{\partial q_n} \end{bmatrix}, \tag{20}$$

for a joint state vector of length n and an operational state vector of length m. As a result, 20 allows for the following mapping:

$$\dot{\boldsymbol{x}} = J_{\boldsymbol{x}}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{21}$$

and the inverse of $J_{\boldsymbol{x}}(\boldsymbol{q})$ provides the reverse relationship. Additionally, we can use the Jacobian to map between the dynamics $\ddot{x}$ and $\ddot{q}$ by taking the time derivative of 21:

$$\begin{aligned}\ddot{\boldsymbol{x}} &= \frac{d}{dt}(J_{\boldsymbol{x}}(\boldsymbol{q})\dot{\boldsymbol{q}}) \\ &= \dot{J}_{\boldsymbol{x}}(\boldsymbol{q})\dot{\boldsymbol{q}} + J_{\boldsymbol{x}}(\boldsymbol{x})\ddot{\boldsymbol{q}}.\end{aligned} \tag{22}$$

Therefore, using (13), we can relate the control dependent dynamics in operational and joint spaces by:

$$g(\boldsymbol{x},\dot{\boldsymbol{x}})u_{\boldsymbol{x}} = J_{\boldsymbol{x}}(\boldsymbol{q})g(\boldsymbol{q},\dot{\boldsymbol{q}})u_{\boldsymbol{q}}. \tag{23}$$

Let us consider the sliding surface $\boldsymbol{s}_x = \dot{\boldsymbol{e}}_x + \boldsymbol{e} \in \mathbb{R}^m$ with $\boldsymbol{e}_x = \boldsymbol{x}_{ref} - \boldsymbol{x}$ and $\dot{\boldsymbol{e}}_x = \dot{\boldsymbol{x}}_{ref} - \dot{\boldsymbol{x}}$ representing the translational position and velocity errors, respectively. Now, using (13) and (23) in the corresponding sliding-surface dynamics yields

$$\dot{\boldsymbol{s}} = \ddot{\boldsymbol{x}}_{ref} - (\boldsymbol{f}_x + J_{\boldsymbol{x}}(\boldsymbol{q})g(\boldsymbol{q},\dot{\boldsymbol{q}})u_{\boldsymbol{q}}) + \lambda\dot{\boldsymbol{e}}_x, \tag{24}$$

where $\boldsymbol{f}_x$ is a lumped state-dependent uncertainty written as

$$\boldsymbol{f}_x = \dot{J}_{\boldsymbol{x}}(\boldsymbol{q})\dot{\boldsymbol{q}} + J_x\boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}}). \tag{25}$$

Noting that the control signal must be in joint space to apply a torque to $\boldsymbol{q}$, we choose the control signal as:

$$u_q = g(\boldsymbol{q},\dot{\boldsymbol{q}})^{-1}J_{\boldsymbol{x}}(\boldsymbol{q})^T(-\hat{\boldsymbol{f}}_x + k\boldsymbol{s} + \ddot{\boldsymbol{x}}_{ref} + \lambda\dot{\boldsymbol{e}}), \tag{26}$$

which we can substitute into 24 and simplify the resulting expression into the form defined in 19 with respect to an operational space state variable $\boldsymbol{x}$, giving:

$$\dot{\boldsymbol{s}}(\boldsymbol{x},\dot{\boldsymbol{x}}) = \tilde{\boldsymbol{\omega}}^T\phi(\boldsymbol{x},\dot{\boldsymbol{x}}) - ks(\boldsymbol{x},\dot{\boldsymbol{x}}) + \varepsilon, \tag{27}$$

where $\tilde{\boldsymbol{\omega}}^T\phi(\boldsymbol{x},\dot{\boldsymbol{x}})$ accounts for the mismatch between f_x and $\hat{f}_x$. Let us propose a candidate Lyapunov function $V(\boldsymbol{s},\tilde{\boldsymbol{\omega}})$ to verify the stability of the states $\boldsymbol{s} \in \mathbb{R}^m$ and $\tilde{\boldsymbol{\omega}} = (\tilde{\boldsymbol{\omega}}_1,\ldots,\tilde{\boldsymbol{\omega}}_m)^T$ with $\tilde{\boldsymbol{\omega}}_{(\cdot)} \in \mathbb{R}^\rho$.

$$V(\boldsymbol{s},\tilde{\boldsymbol{\omega}}) = \frac{1}{2}\boldsymbol{s}^T\boldsymbol{s} + \frac{1}{2\gamma}\tilde{\boldsymbol{\omega}}^T\tilde{\boldsymbol{\omega}}, \tag{28}$$

where

$$V(\boldsymbol{s},\tilde{\boldsymbol{\omega}}) \geq 0, \forall \boldsymbol{s} \in \mathbb{R}^m, \tilde{\boldsymbol{\omega}} \in \mathbb{R}^n.$$

We take the derivative of 28 to give:

$$\dot{V}(\boldsymbol{s},\tilde{\boldsymbol{\omega}}) = \boldsymbol{s}^T\dot{\boldsymbol{s}} + \frac{1}{\gamma}\tilde{\boldsymbol{\omega}}^T\dot{\tilde{\boldsymbol{\omega}}}. \tag{29}$$

Since $\boldsymbol{\omega}^*$ is constant and $\dot{\boldsymbol{\omega}}^* = 0$, we can simplify 29 to:

$$\dot{V}(\boldsymbol{s},\tilde{\boldsymbol{\omega}}) = \boldsymbol{s}^T\dot{\boldsymbol{s}} + \frac{1}{\gamma}\tilde{\boldsymbol{\omega}}^T\dot{\boldsymbol{\omega}}. \tag{30}$$

Then by substituting in 27, we can write 30 as:

$$\dot{V}(\boldsymbol{s},\tilde{\boldsymbol{\omega}}) = \boldsymbol{s}^T\tilde{\boldsymbol{\omega}}^T\phi(\boldsymbol{x},\dot{\boldsymbol{x}}) + \frac{1}{\gamma}\tilde{\boldsymbol{\omega}}^T\dot{\boldsymbol{\omega}} - k\boldsymbol{s}^T\boldsymbol{s} + \boldsymbol{s}^T\boldsymbol{\varepsilon} \tag{31}$$

Additionally, regarding the crossed term, let us employ Young's inequality to rewrite (31) as:

$$\dot{V}(\boldsymbol{s},\tilde{\boldsymbol{\omega}}) \leq \tilde{\boldsymbol{\omega}}^T\boldsymbol{s}\phi(\boldsymbol{x},\dot{\boldsymbol{x}}) + \frac{1}{\gamma}\tilde{\boldsymbol{\omega}}^T\dot{\boldsymbol{\omega}} - k\boldsymbol{s}^T\boldsymbol{s} + \frac{\mu}{2}\boldsymbol{s}^T\boldsymbol{s} + \frac{1}{2\mu}\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}, \tag{32}$$

where μ is a constant greater than zero. By using the rule in the form:

$$\dot{\boldsymbol{\omega}} = -\gamma\boldsymbol{s}\phi(\boldsymbol{x},\dot{\boldsymbol{x}}). \tag{33}$$

We can see substituting 33 into 32 simplifies to:

$$\begin{aligned}\dot{V}(\boldsymbol{s},\tilde{\boldsymbol{\omega}}) &\leq -(k-\tfrac{\mu}{2})\boldsymbol{s}^T\boldsymbol{s} + \tfrac{1}{2\mu}\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}, \\ &= -(k-\tfrac{\mu}{2})\|\boldsymbol{s}\|^2 + \tfrac{1}{2\mu}\|\boldsymbol{\varepsilon}\|^2, \\ &\leq -(k-\tfrac{\mu}{2})\|\boldsymbol{s}\|^2 + \tfrac{1}{2\mu}\bar{\varepsilon}^2,\end{aligned} \tag{34}$$

We assume that the controller gain k is chosen to be larger than the arbitrarily small $\frac{\mu}{2}$. Moreover, the presence of the approximation error ε prevents asymptotical stability. Instead, the states are confined within a ball

$$\mathcal{B} = \left\{\boldsymbol{s} \in \mathbb{R}^m : \|\boldsymbol{s}\| \leq \sqrt{\frac{\bar{\varepsilon}^2}{2\mu(k-\frac{\mu}{2})}}\right\} \tag{35}$$

Hence, the $\boldsymbol{s}$ and $\tilde{\boldsymbol{\omega}}$ states are driven into this ball, whose radius can be adjusted using k and μ.

Therefore, with the chosen control signal $\boldsymbol{u}$ and adaptive law $\dot{\boldsymbol{\omega}}$, the state variables $\boldsymbol{s}$ and $\tilde{\boldsymbol{\omega}}$ are proven stable. Thus the unknown dynamics $f(\boldsymbol{x},\dot{\boldsymbol{x}})$ can be negated while driving the tracking error to zero.

We now consider a spiking neuron implementation of the adaptive component by leveraging the NEF to let our basis functions $\phi(\boldsymbol{x},\dot{\boldsymbol{x}})$ be the neuron activity profiles $a(\boldsymbol{x},\dot{\boldsymbol{x}})$ defined in Section III-A1. To be consistent with NEF notation we relabel our weights $\boldsymbol{\omega}$, as decoders, $\boldsymbol{d}$. We then can also use the biologically plausible Prescribed Error Sensitivity (PES) rule (31) for spiking neural networks as the learning rule to update the decoders:

$$\Delta\boldsymbol{d}_i = -\kappa\nu a_i(\boldsymbol{x},\dot{\boldsymbol{x}}), \tag{36}$$

where κ is our learning rate and equal to γ, ν is the learning signal $\boldsymbol{s}$, and $a_i(\boldsymbol{x},\dot{\boldsymbol{x}})$ is the activity of the i^{th} neuron. The PES rule (36) has the same structure as the continuous time gradient descent learning rule 33, where the basis function $\phi(\boldsymbol{x},\dot{\boldsymbol{x}})$ is now the activity of the neurons $a(\boldsymbol{x},\dot{\boldsymbol{x}})$.

D. Controller Variations

Having demonstrated the stability of the controller, we now turn to our main purpose, which is to determine the effect of choosing different neural basis functions for $a_i(\boldsymbol{x},\dot{\boldsymbol{x}})$. To begin, let us consider the controller to be structured as the sum of two parallel control signals:

$$u = u_{sm} + u_{adapt},$$

where the u_{sm} is the sliding mode controller defined as:

$$u_{sm} = g(\boldsymbol{q})^{-1}J_{\boldsymbol{q}}(\boldsymbol{x})^T(k\boldsymbol{s} + \ddot{\boldsymbol{x}}_{ref} + \lambda\dot{\boldsymbol{e}}),$$

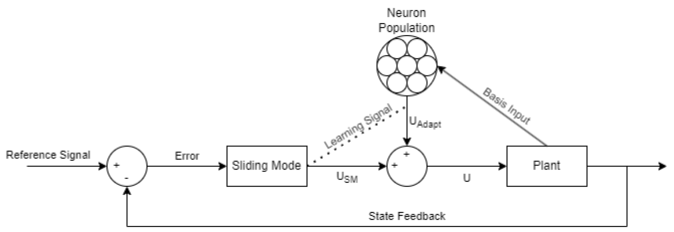


Figure 2: Block diagram of the control feedback loop. The circuit is structured like a standard controller, with the addition of a neural population that gets the state, $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$, as the input and uses the sliding mode output as an error signal to update the decoding weights online.

in equation 26. Additionally, from 26 we define the adaptive control component u_{adapt} as:

$$u_{adapt} = g(\boldsymbol{q})^{-1} J_{\boldsymbol{q}}(\boldsymbol{x})^T(-\hat{f}(\boldsymbol{x}, \dot{\boldsymbol{x}})).$$

The dynamics estimate, $\hat{f}_x$ is updated online using 36 and is equal to the decoded neural activity:

$$\hat{f}_x = \sum_i a_i(\boldsymbol{x}, \dot{\boldsymbol{x}})d$$

We visualize this structure as in the block diagram presented in Figure 2. We consider five formulations of the adaptive controller, alongside an adaptive-free controller as a baseline, which consists of only the sliding-mode controller. Two of the adaptive controllers were introduced by (6) in past work. One of these which we call 'Random', randomly selects encoders uniformly over the D-dimensional hypershere. Neuron intercepts and maximum firing rates are sampled from a uniform distribution over the input space radius and 100-200Hz, respectively. The other past controller (9) improves upon the random adaptive controller by using encoders from the $D+1$ dimensional hyper-sphere. By using encoders in the $D+1$ dimensional hypersphere, the responsive regions in a D-dimensional space of individual neurons can be carefully chosen to be local and uniformly represent state space. We refer to this controller as the Selected basis. We have introduced the remaining bases in Section III-A1 that are reminiscent of place, Kenyon, and grid cells. We refer to these as their respective cell SSP controllers.

IV. EXPERIMENTS

We employ a simulation of a 3-link arm in 2-dimensional space using the general dynamics defined in 13. We simulated the controller and spiking neurons using Nengo (32) interfaced with a Python model of the plant. For each controller we consider the free hyper-parameters for tuning to be the sliding controller gains k and λ, and the adaptive learning rate γ. We let the individual neuron properties, such as the gain α and intercepts ξ, be determined by choosing their intercept, and max firing rate from uniform distributions as described previously.

Table I: Optimization parameters and results over controllers

	k	λ	γ	Error
Non-adaptive	239.3024	1.1319	-	0.0082
Random	198.6770	1.1315	$8.50 * 10^{-6}$	0.0083
Selected	196.6269	1.0981	$7.98 * 10^{-6}$	0.0083
Place SSP	201.7676	1.0478	$9.98 * 10^{-6}$	0.0082
Kenyon SSP	195.2043	1.1268	$4.70 * 10^{-6}$	0.0083
Grid SSP	200.5518	1.1188	8.20*10^{-6}	0.0083

A. Hyperparameter search

We first conduct a hyper-parameter optimization to tune each controller for the specific task of tracking the reference trajectory. We selected a sinusoidal reference trajectory in the form of 37 such that the end effector tracks a circle in operational space:

$$X_{ref}(t) = \begin{bmatrix} x_{ref}(t) \\ y_{ref}(t) \\ \dot{x}_{ref}(t) \\ \dot{y}_{ref}(T) \end{bmatrix} = \begin{bmatrix} x_0 + r\cos\left(\frac{2\pi}{T}t\right) \\ y_0 + r\sin\left(\frac{2\pi}{T}t\right) \\ -r\frac{2\pi}{T}\sin\left(\frac{2\pi}{T}t\right) \\ r\frac{2\pi}{T}\cos\left(\frac{2\pi}{T}t\right) \end{bmatrix}, \tag{37}$$

where r is the radius, T is the period, and x_0 and y_0 are constant biases. For the present experiments, we choose values of 0.1, 5, and 0.1 for the radius, period, and biases, respectively. We evaluate the performance of each simulation using a mean-square error term between the state and reference trajectory 37 in the following form:

$$E = \int (X(t) - X_{ref}(t))^2 dt. \tag{38}$$

We evaluated the $argmin$ of 38 with respect to the space of tuneable hyper-parameters for each controller over 200 simulation trials. We summarize the optimal hyper-parameters found using this method in Table I.

B. Results

We evaluate the controller's performance for adaptation by using the optimal parameters in Table I. We then apply a periodic state-based disturbance $d(x, \dot{x})$ such that the adaptive component $\hat{f}(x, \dot{x})$ must account for the resulting dynamics. We chose 37 as a task with a disturbance over a simulation time of 60 seconds to allow the adaptive component to learn. We switch on the disturbance after one complete period and apply it directly to the arm dynamics in the form of $f(x, \dot{x})$ in Equation 13. We consider two disturbances for testing. The first is a downward force applied to the end-effector to simulate an additional mass being added, and the second is an operational space positional cosine field on the end effector:

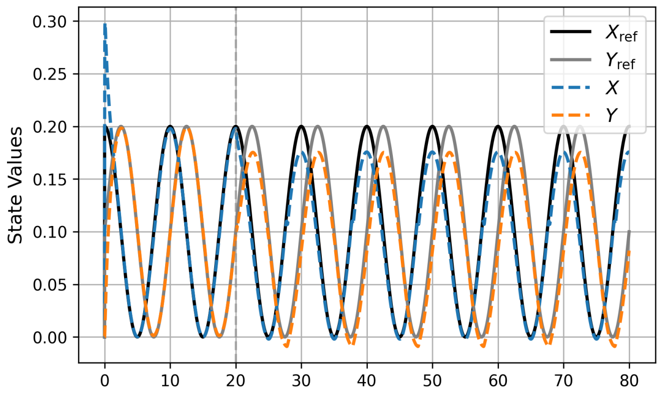
$$F_{dist} = [cos(x), cos(y)]^T. \tag{39}$$

We consider both disturbances on all controllers and evaluate the root mean square error (RMSE) over 5 trials as summarized in Table II with standard deviations included.

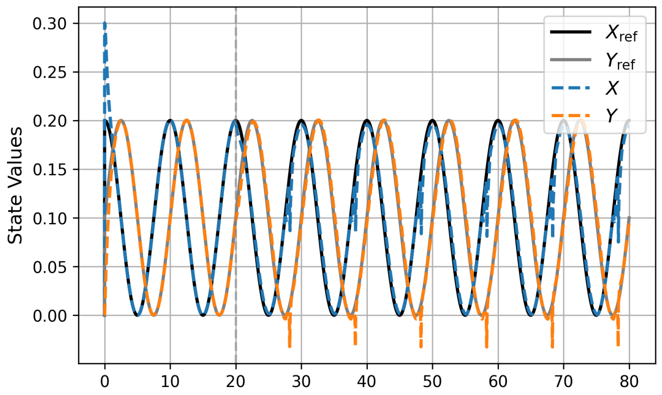
Table II: RMSE results of different controllers

	RMSE	
	Additional Mass Forcing	Cosine Forcing Function
Non-Adaptive	$2.568 * 10^{-2} \pm 0$	$5.679 * 10^{-2} \pm 0$
Random	$2.084 * 10^{-2} \pm 4.09 * 10^{-4}$	$4.290 * 10^{-2} \pm 1.28 * 10^{-3}$
Selected	$1.896 * 10^{-2} \pm 1.92 * 10^{-4}$	$3.684 * 10^{-2} \pm 0.26 * 10^{-3}$
Place Cells	$\mathbf{1.539 * 10^{-2}} \pm 7.26 * 10^{-4}$	$\mathbf{2.939 * 10^{-2}} \pm 0.87 * 10^{-}$
Grid Cells	$1.932 * 10^{-2} \pm 1.02 * 1010^{-4}$	$3.745 * 10^{-2} \pm 0.36 * 10^{-3}$
Kenyon Cells	$2.358 * 10^{-2} \pm 2.24 * 10^{-4}$	$4.818 * 10^{-2} \pm 0.85 * 10^{-3}$

We observe in Table: II that in all cases the place SSP adaptive controller performs the best, with statistical significance. As expected, the Selected controller has good performance as well, however, the place SSP controller has an improvement between $1.23 - 1.25$x. In Figure 3 we show the place SSP state trajectory alongside the Non-adaptive controller. We use the downward force disturbance in this trial, switching it on at $20s$ as denoted by the vertical dashed line. For both controllers, we plot the state and reference trajectories over $80s$. We observe that the place SSP controller clearly performs better at tracking the reference trajectory with a disturbance. We also note the singularities observed in the place SSP trial and suspect that these are the results of singularities that exist in the robots range of motion.



(a) No Adaptive



(b) SSP Adaptive

Figure 3: Simulation results of mass-based disturbance of $\boldsymbol{a}$ no-adaptive and $\boldsymbol{b}$ place cell SSP adaptive controller. The dashed line denotes the time at which the disturbance is switched on.

V. DISCUSSION

In this work, we propose a biologically inspired method for choosing neuron tuning curves in the context of an adaptive controller using spiking neurons in parallel to a classical sliding mode controller. We demonstrate that place cell inspired SSP bases outperform past methods that rely on random encoders in either D or D+1 dimensions. Our proposed controller provides the best performance in Table II with place cells, showing an improvement of $1.23 - 1.25$x over the Selected controller and $1.67 - 1.93$x over the non-adaptive controller. Noticeably, grid cells perform on-par with the Selected controller, and Kenyon cells perform the worst of the adaptive controllers. We attribute this performance increase towards the ability of place cell SSPs to represent spaces both sparsely and uniformly. Additionally, since SSPs are structured as high-dimensional vectors, the use of place cell SSPs provides a nearly orthogonal set of basis functions while tiling the space in a spatially localized manner. This property allows state-based disturbances to be accurately captured. The high-dimensionality of SSP representations suggests that the observed performance improvements should generalize to problems with higher-dimensional state spaces. Notably, Radial basis function networks (RBFNs), first proposed by Broomhead and Lowe (33), are a classical approach to adaptive control. The place-cell representation that we use here achieves a finite approximation of a rectified Sinc kernel using Fourier basis functions. As such, this approach provides a link between the classical and spiking neural network approaches to adaptive control. Future work should assess the effect of this approximation on the learned dynamics, by making an explicit comparison between RBFN and the place-cell adaptive controller we propose here.

An important limitation of the place cell representation is that the region of the state space over which the system can generalize is necessarily limited. That is, if a disturbance were to occur that pushed the system outside of the bounds over which the place cells were defined, the system would be severely limited in its ability to adapt. In such cases, the alternative representations considered here that underperformed on the selected tasks - such as grid cells and Kenyon cells - might have an advantage due to their unbounded representation.

Our method not only outperforms past approaches, but also provides a more systematic approach to the design of an adaptive controller compared to the Selected controller. In contrast to the selected controller, in which encoders must be manually chosen over the D+1 hypersphere, SSP-controllers sample from a higher-dimensional space by simply generating SSPs across the state space. The usage of SSPs as a basis to represent state space in an adaptive controller is a novel approach that could also be integrated into existing SSP architecture (34; 35) for robotic applications. We also suggest that an SSP representation may also allow for the inclusion of dynamic constraints into the control formulation. We leave this exploration, as well as the implementation of the proposed controller on fully neuromorphic hardware, as future work.

References

[1] C. Edwards and S. Spurgeon, *Sliding mode control: theory and applications*. CRC press, 1998.

[2] J.-J. E. Slotine and W. Li, "On the adaptive control of robot manipulators," *The International Journal of Robotics Research*, vol. 6, no. 3, p. 49–59, 1987.

[3] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[4] T. Lillicrap, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[5] I. Goodfellow, *Deep learning*, vol. 196. MIT press, 2016.

[6] T. DeWolf, T. C. Stewart, J.-J. Slotine, and C. Eliasmith, "A spiking neural model of adaptive arm control," *Proceedings of the Royal Society B*, vol. 283, no. 48, 2016.

[7] T. DeWolf, K. Patel, P. Jaworski, R. Leontie, J. Hays, and C. Eliasmith, "Neuromorphic control of a simulated 7-dof arm using loihi," *Neuromorphic Computing and Engineering*, 2023.

[8] C. Eliasmith and C. H. Anderson, *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT Press, 2003.

[9] T. DeWolf, P. Jaworski, and C. Eliasmith, "Nengo and low-power ai hardware for robust embedded neurorobotics," *Frontiers in Neurorobotics*, 2020.

[10] J. O'Keefe and L. Nadel, *The Hippocampus as a Cognitive Map*. Oxford: Clarendon Press, 1978.

[11] T. Hafting, M. Fyhn, S. Molden, M.-B. Moser, and E. I. Moser, "Microstructure of a spatial map in the entorhinal cortex," *Nature*, vol. 436, no. 7052, pp. 801–806, 2005.

[12] G. C. Turner, M. Bazhenov, and G. Laurent, "Olfactory representations by drosophila mushroom body neurons," *Journal of neurophysiology*, vol. 99, no. 2, pp. 734–746, 2008.

[13] T. A. Plate, "Holographic reduced representations," *IEEE Transactions on Neural Networks*, vol. 6, pp. 623–641, 1995.

[14] B. Komer, T. C. Stewart, A. R. Voelker, and C. Eliasmith, "A neural representation of continuous space using fractional binding," in *41st Annual Meeting of the Cognitive Science Society*, (Montreal, QC), Cognitive Science Society, 2019.

[15] B. Komer and C. Eliasmith, "Efficient navigation using a scalable, biologically inspired spatial representation.," in *CogSci*, 2020.

[16] C. Eliasmith, *How to build a brain: A neural architecture for biological cognition*. Oxford University Press, 2013.

[17] B. Komer, T. C. Stewart, A. Voelker, and C. Eliasmith, "A neural representation of continuous space using fractional binding.," in *CogSci*, pp. 2038–2043, 2019.

[18] N. Dumont and C. Eliasmith, "Accurate representation for spatial cognition using grid cells.," in *CogSci*, 2020.

[19] T. C. Stewart, P. M. Furlong, K. Simone, M. Bartlett, and J. Orchard, "Novelty detection, insect olfaction, mismatch negativity, and the representation of probability in the brain,"

[20] H. T. Blair, A. C. Welday, and K. Zhang, "Scale-invariant memory representations emerge from moire interference between grid fields that produce theta oscillations: a computational model," *Journal of Neuroscience*, vol. 27, no. 12, pp. 3211–3229, 2007.

[21] J. Orchard, H. Yang, and X. Ji, "Does the entorhinal cortex use the fourier transform?," *Frontiers in computational neuroscience*, vol. 7, p. 179, 2013.

[22] C. Yu, T. Behrens, and N. Burgess, "Prediction and generalisation over directed actions by grid cells," in *International Conference on Learning Representations*.

[23] T. Lu, A. Voelker, B. Komer, and C. Eliasmith, "Representing spatial relations with fractional binding.," in *CogSci*, pp. 2214–2220, 2019.

[24] T. A. Plate, "Holographic reduced representations," *IEEE Transactions on Neural networks*, vol. 6, no. 3, pp. 623–641, 1995.

[25] T. A. Plate, "Holographic recurrent networks," *Advances in neural information processing systems*, vol. 5, 1992.

[26] E. P. Frady, D. Kleyko, C. J. Kymn, B. A. Olshausen, and F. T. Sommer, "Computing on functions using randomized vector representations (in brief)," in *Proceedings of the 2022 Annual Neuro-Inspired Computational Elements Conference*, pp. 115–122, 2022.

[27] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," *Advances in neural information processing systems*, vol. 20, 2007.

[28] A. R. Voelker, "A short letter on the dot product between rotated fourier transforms," *arXiv preprint arXiv:2007.13462*, 2020.

[29] S. M. Farris and I. Sinakevitch, "Development and evolution of the insect mushroom bodies: towards the understanding of conserved developmental mechanisms in a higher brain center," *Arthropod Structure & Development*, vol. 32, no. 1, pp. 79–101, 2003. Development of the Arthropod Nervous System: a Comparative and Evolutionary Approach.

[30] P. K. Khosla and T. Kanade, "Parameter identification of robot dynamics," in *1985 24th IEEE Conference on Decision and Control*, pp. 1754–1760, 1985.

[31] D. MacNeil and C. Eliasmith, "Fine-tuning and the stability of recurrent neural networks," *PLOS ONE*, vol. 6, pp. 1–16, 09 2011.

[32] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: a Python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, no. 48, pp. 1–13, 2014.

[33] D. Lowe and D. Broomhead, "Multivariable functional interpolation and adaptive networks," *Complex systems*, vol. 2, no. 3, pp. 321–355, 1988.

[34] N. S.-Y. Dumont, P. M. Furlong, J. Orchard, and C. Eliasmith, "Exploiting semantic information in a spiking neural slam system," *Frontiers in Neuroscience*, vol. 17, 2023.

[35] A. R. Voelker, P. Blouw, X. Choo, N. S.-Y. Dumont, T. C. Stewart, and C. Eliasmith, “Simulating and predicting dynamical systems with spatial semantic pointers,” *Neural Computation*, vol. 33, pp. 2033–2067, 07 2021.