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
Lonrix Ltd.
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

***Abstract*—We present a biologically-inspired representation for an adaptive controller using Spatial Semantic Pointers (SSPs). Specifically we show that Place-cell like SSP representations outperform 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 stability of the overall system with non-linear plant dynamics. We then simulate the controller on a 3-link arm and demonstrate that the proposed adaptive controller gives a simpler and more systematic way of designing the representation than past methods, while offering an increase in performance of $\mathbf{1.23-1.25x}$.**

I. INTRODUCTION

We introduce a novel, biologically-inspired method for implementing adaptive control that leverages Spatial Semantic Pointers (SSPs)[1] to form a basis representation to learn plant dynamics over. By merging this dynamic, brain-inspired adaptive component with a classical sliding mode controller, our approach offers a systematic and simpler design process of the state space representation as an adaptive basis for robust controllers. Our adaptive controller builds on recent advances in neuromorphic computing, using spiking neural networks to represent system states in a high-dimensional, sparse manner [2]. Neuromorphic adaptive control allows for efficient online learning that can update its model of the system in real-time using event-based processing, making it particularly suited for energy-constrained or embedded applications. The proposed model is proven to provide stable dynamics, and its tracking performance is evaluated in simulation of a 3-link arm. Critically, 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, and are a more systematic approach to state space representation then past methods.

A classical sliding mode controller applies a feedback-control loop to continuously sample a plant's state to generate a control signal to achieve desired dynamics. A sliding variable is defined in the state space on the plant's stable manifold (see Section III-B). 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 [3]. This adaptive component can learn an approximation of the dynamics over a set of basis functions with an online learning rule. Additionally, in 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.

The choice in the set of basis functions used is critical to the performance of online learning of the dynamics [4]. Ideally. the basis functions should be able to learn any plausible dynamics function over the state space. We build upon past work in neuromorphic control and propose a new biologically-inspired method to design the adaptive basis using spatial semantic pointers (SSP). Specifically, we use SSPs to generate both place-cell-like (RandSSPs), and grid-cell-like (HexSSPs) representations over the state space. We use these structures to tile the plant's state space with a spiking neural representation that is used as a basis to learn the dynamics for the adaptive controller. We implement these SSP representations and a learning rule at the neural level using the Neural Engineering Framework (NEF) [5] to construct a neuromorphic controller that leverages the learning capabilities and power efficiency of spiking neural networks.

Relevant past work on neuromorphic adaptive control using spiking neurons was conducted in [2], and further built upon in [6], [4] to employ neuromorphic hardware and to carefully choose basis functions for efficient learning. We use this past work to benchmark our current proposal. As well, the concept

of using SSPs to represent a space was introduced in [7] and augmented with grid cells in [8]. We build upon this work by demonstrating a novel application SSP representations to adaptive control.

In the remainder of the paper, we begin by covering the theoretical background and prerequisites in II, and present the sliding mode dynamics and SSP structure in Section III. We then prove 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 performance of our new approach to past work in simulation.

II. THEORETICAL PREREQUISITES

A. Function approximation

For real-world robots, 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).

Here, the objective is to approximate such uncertainties online based on neuro-adaptive control.

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 an neural network (NN) satisfying*

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

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

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

where the upper-bounded by $\|\boldsymbol{\omega}^\star\| \leq \bar{\omega}$, $\|\boldsymbol{\varphi}(\cdot)\| \leq \bar{\varphi}$ and $|\varepsilon| \leq \bar{\varepsilon}$, with $\bar{\omega}$, $\bar{\varphi}$, 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

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

To this end, two phases are distinguished: (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 invariant the manifold, i.e.

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

III. METHODS

A. 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 [7]. 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; [9]) VSA is distinguished from most VSAs for its support by representing data defined over continuous feature spaces. In the HRR literature, the method for representing continuous-valued data developed by Plate [10] is referred to as fractional binding [1] or fractional power encoding [11]. We follow the convention of Komer et al. [7] 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 [8] when referring to the hypervectors that reside in these two latent spaces.

1) Random SSPs: One way to select the elements of Θ is randomly, such that its elements correspond to frequencies sampled from some power spectral density function. For RandSSPs, 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) [12] in kernel methods. The dot product between RandSSPs approximates a normalized sinc function [13]:

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

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

Using this embedding in our controller, and adopting the NEF for representing and computing over vector spaces, the activity of the n^{th} neuron associated with observation of system state x is now defined by:

$$a_n = \text{LIF}(\alpha_n \phi_{\mathbf{X}}(\boldsymbol{\lambda}^{-1} x) \cdot e_n - \xi_n)) \tag{8}$$

Where e_n are the neuron encoder or the preferred direction vector, α_n are the gains, and ξ_n are the intercepts of the n^{th} neuron. 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 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 often 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 [14].

The Place Cell (or RandSSPs) representation constrains the region of the state space over which the control law can be learned 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: $e_n \sim U(S^{D-1})$. To understand the basis function resulting from this choice, we decompose e_n into a sum of B vectors that are non-orthogonal with the manifold $\phi(x_{b,\|})$ 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: $e_n = \sum_{b=1}^{B} \alpha_b \phi(x_{b,\|}) + \sum_{c=1}^{C} \alpha_c \phi(x_{c,\perp})$. A third class of vectors are exactly orthogonal, which we omit from consideration.

Due to the high-dimensional vector representation, the contribution of pseudo-orthogonal components are negligible, and the neural response approximates the sum of contributions of vectors aligned with the manifold:

$$a_n \approx \text{LIF}\left(\phi(x/\lambda) \cdot \sum_{b=1}^{B} \alpha_b \phi(x_{b,\|}) - \xi\right) \tag{9}$$

The dot product between vectors in this latent space induces a sum of approximate sinc kernels in the domain. 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 [15], and we thus name these cells accordingly. An advantage of this representation is that it is unbounded.

2) HexSSPs: For HexSSPs, we begin by constructing a simplex V, which is a $(d+1) \times d$ matrix, selected to minimize

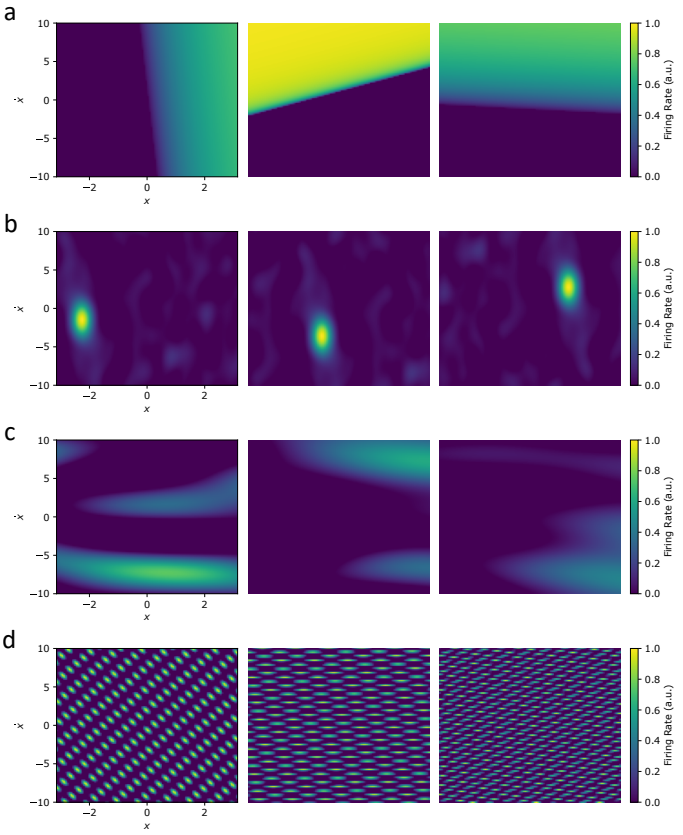

Figure 1: Example receptive fields of neurons adaptive controller's basis functions over state space. **a** Random Encoders (baseline), **b** Place Cells (RandSSPs), **c** Kenyon Cells, and **d** Grid Cells (HexSSPs).

the expression: $\sum_{i}^{d+1} \sum_{j=1, i \neq j}^{d+1} \mathbf{v}_i \cdot \mathbf{v}_j$, where $\mathbf{v}_i$ and $\mathbf{v}_j$ are rows i and j from V, and are unit vectors. In 2D space, the rows of V induce the kernel function, $k(x, x') = \phi(x) \cdot \phi(x')$, which produces plane waves oriented 120° apart, so the combination of all 3 rows generates a hexagonally-tiled interference pattern. This pattern is similar to grid cells in the mammalian medial entorhinal cortex (MEC). The firing patterns of grid neurons in the MEC are characterized by different orientations and sizes [16]. Following the approach taken by Dumont and Eliasmith [8], we apply linear transformations to V to obtain a set of V' to mimic this diversity. Specifically, we stack $\{V'\}$ corresponding to a set of scalings, $\mathcal{S}$, and rotations, $\mathcal{R}$ of V, to obtain an augmented phase matrix Θ:

$$\Theta = \text{stack}\left(\{V'\} = \{s\mathbf{R}\mathbf{V} \mid s \in \mathcal{S}, \mathbf{R} \in \mathcal{R}\}\right) \tag{10}$$

Finally, Θ is augmented to ensure conjugate symmetry [8]. Each of the V in Θ reveals a hexagonal pattern in 2D space akin to grid-cell activity (Fig. 1d). We can now define our HexSSP representations over an ensemble of spiking neurons such that the encoders project the input signal into HexSSP space. The activity of the n^{th} neuron associated with observation of system state x is defined by:

$$a_n(x) = \text{LIF}(\alpha_n x \cdot e_n - \xi_n)), \tag{11}$$

where LIF denotes the activation function of a leaky integrate and fire neuron, e_n is the encoder defining the preferred direction (and equal to a randomly chosen V'), α_n is the gain, and ξ_n is the bias.

B. Controller Design

Adapting the techniques derived in [3] which was further applied to spiking neural networks [2], let us consider the second-order dynamics of a n-link manipulator robot in the following form:

$$\ddot{\boldsymbol{q}} = f(\boldsymbol{q}, \dot{\boldsymbol{q}}) + g(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{u}_q, \tag{12}$$

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:

$$\boldsymbol{s}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \dot{\boldsymbol{e}} + \lambda \boldsymbol{e}. \tag{13}$$

with $\boldsymbol{s} \in \mathbb{R}^n$ where $\boldsymbol{e}$ and $\dot{e}$ 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 (12) into (13) after taking the derivative with respect to time, finding that the dynamics of the sliding surface $\dot{\boldsymbol{s}}$ can be expanded into:

$$\dot{\boldsymbol{s}} = \ddot{\boldsymbol{q}}_{ref} - f(\boldsymbol{q}, \dot{\boldsymbol{q}}) - g(\boldsymbol{q}, \dot{\boldsymbol{q}})u + \lambda \dot{\boldsymbol{e}}. \tag{14}$$

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 e and $\dot{e}$. Therefore, we choose a control signal u in the form:

$$\boldsymbol{u} = g(\boldsymbol{q}, \dot{\boldsymbol{q}})^{-1}(-\hat{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \ddot{\boldsymbol{q}}_{ref} + \lambda \dot{e} + k\boldsymbol{q}), \tag{15}$$

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 15 we can describe the dynamics of 14 by:

$$\dot{s}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = (\hat{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - f(\boldsymbol{q}, \dot{\boldsymbol{q}})) - ks(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{16}$$

Using the property introduced in [17], and leveraged for spiking neural networks in [2], 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

$$\hat{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\omega}^T \boldsymbol{\phi}(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{17}$$

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

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

with $\boldsymbol{\varepsilon} \in \mathbb{R}^m$ standing for the approximation error and holding $\|\boldsymbol{\varepsilon}\| \leq \bar{\varepsilon}$, where $\bar{\varepsilon}$ is some converged approximation error upper bound. We then define a weight error term, $\tilde{\boldsymbol{\omega}}$, thus allowing 16 to be reformulated into

$$\dot{\boldsymbol{s}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \tilde{\boldsymbol{\omega}}^T \phi(\boldsymbol{q}, \dot{\boldsymbol{q}}) - ks(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\varepsilon}. \tag{18}$$

C. Stability Analysis

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 12 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:

$$J_{\boldsymbol{x}}(\boldsymbol{q}) = \begin{bmatrix} \frac{\partial x_1}{\partial q_1} & \cdots & \frac{\partial x_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_m}{\partial q_1} & \cdots & \frac{\partial x_m}{\partial q_n} \end{bmatrix}, \tag{19}$$

for a joint state vector of length n and operational state vector of length m. As a result, 19 allows for the following mapping:

$$\dot{\boldsymbol{x}} = J_{\boldsymbol{x}}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{20}$$

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 20:

$$\begin{aligned} \ddot{\boldsymbol{x}} &= \frac{d}{dt}(J_{\boldsymbol{x}}(\boldsymbol{q})\dot{\boldsymbol{q}}) \\ &= \dot{J}_{\boldsymbol{x}}(\boldsymbol{q})\dot{\boldsymbol{q}} + J_{\boldsymbol{x}}(\boldsymbol{x})\ddot{\boldsymbol{q}}. \end{aligned} \tag{21}$$

Therefore, using (12), we can relate the control dependent dynamics in operational and joint spaces by:

$$g(\boldsymbol{x}, \dot{\boldsymbol{x}})u_{\boldsymbol{x}} = J_{\boldsymbol{x}}(\boldsymbol{q})g(\boldsymbol{q}, \dot{\boldsymbol{q}})u_{\boldsymbol{q}}. \tag{22}$$

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 (12) and (22) in the corresponding sliding-surface dynamics yields

$$\dot{\boldsymbol{s}} = \ddot{\boldsymbol{x}}_{ref} - (\boldsymbol{f}_x + J_{\boldsymbol{x}}(\boldsymbol{q})g(\boldsymbol{q}, \dot{\boldsymbol{q}})u_{\boldsymbol{q}}) + \lambda \dot{\boldsymbol{e}}_x, \tag{23}$$

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

$$\boldsymbol{f}_x = \dot{J}_{\boldsymbol{x}}(\boldsymbol{q})\dot{\boldsymbol{q}} + J_x \boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{24}$$

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

$$u_q = g(\boldsymbol{q}, \dot{\boldsymbol{q}})^{-1} J_{\boldsymbol{x}}(\boldsymbol{q})^T (-\hat{\boldsymbol{f}}_x + k\boldsymbol{s} + \ddot{\boldsymbol{x}}_{ref} + \lambda \dot{\boldsymbol{e}}), \tag{25}$$

which we can substitute into 23 and simplify the resulting expression into the form defined in 18 with respect to an operational space state variable $\boldsymbol{x}$, giving:

$$\dot{\boldsymbol{s}}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \tilde{\boldsymbol{\omega}}^T \phi(\boldsymbol{x}, \dot{\boldsymbol{x}}) - ks(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \boldsymbol{\varepsilon}, \tag{26}$$

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 $\tilde{\boldsymbol{\omega}}_{(\cdot)} \in \mathbb{R}^\rho$.

$$V(\boldsymbol{s}, \tilde{\boldsymbol{\omega}}) = \frac{1}{2}\boldsymbol{s}^T\boldsymbol{s} + \frac{1}{2\gamma}\tilde{\boldsymbol{\omega}}^T\tilde{\boldsymbol{\omega}}, \tag{27}$$

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 27 to give:

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

Since $\boldsymbol{\omega}^*$ is constant and $\dot{\boldsymbol{\omega}}^* = 0$, we can simplify 28 to:

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

Then by substituting in 26, we can write 29 as:

$$\dot{V}(\boldsymbol{s}, \tilde{\boldsymbol{\omega}}) = \boldsymbol{s}^T\tilde{\boldsymbol{\omega}}^T\phi(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \frac{1}{\gamma}\tilde{\boldsymbol{\omega}}^T\dot{\boldsymbol{\omega}} - k\boldsymbol{s}^T\boldsymbol{s} + \boldsymbol{s}^T\boldsymbol{\varepsilon} \tag{30}$$

Additionally, regarding crossed term, let us employ Young's inequality to rewrite (30) as:

$$\dot{V}(\boldsymbol{s}, \tilde{\boldsymbol{\omega}}) \leq \tilde{\boldsymbol{\omega}}^T\boldsymbol{s}\phi(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \frac{1}{\gamma}\tilde{\boldsymbol{\omega}}^T\dot{\boldsymbol{\omega}} - k\boldsymbol{s}^T\boldsymbol{s} + \frac{\mu}{2}\boldsymbol{s}^T\boldsymbol{s} + \frac{1}{2\mu}\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}, \tag{31}$$

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

$$\dot{\boldsymbol{\omega}} = -\gamma\boldsymbol{s}\phi(\boldsymbol{x}, \dot{\boldsymbol{x}}). \tag{32}$$

We can see substituting 32 into 31 simplifies to:

$$\begin{aligned} \dot{V}(\boldsymbol{s}, \tilde{\boldsymbol{\omega}}) &\leq -(k - \tfrac{\mu}{2})\boldsymbol{s}^T\boldsymbol{s} + \tfrac{1}{2\mu}\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}, \\ &= -(k - \tfrac{\mu}{2})\|\boldsymbol{s}\|^2 + \tfrac{1}{2\mu}\|\boldsymbol{\varepsilon}\|^2, \\ &\leq -(k - \tfrac{\mu}{2})\|\boldsymbol{s}\|^2 + \tfrac{1}{2\mu}\bar{\varepsilon}^2, \end{aligned} \tag{33}$$

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 $\boldsymbol{\varepsilon}$ prevents asymptotical stability. Instead the states are confined within a ball

$$\mathcal{B} = \left\{ \boldsymbol{s} \in \mathbb{R}^m : \|\boldsymbol{s}\| \leq \sqrt{\frac{\bar{\varepsilon}^2}{2\mu(k - \frac{\mu}{2})}} \right\} \tag{34}$$

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-A. 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 [18] for spiking neural networks as the learning rule to update the decoders:

$$\Delta\boldsymbol{d}_i = -\kappa\nu a_i(\boldsymbol{x}, \dot{\boldsymbol{x}}), \tag{35}$$

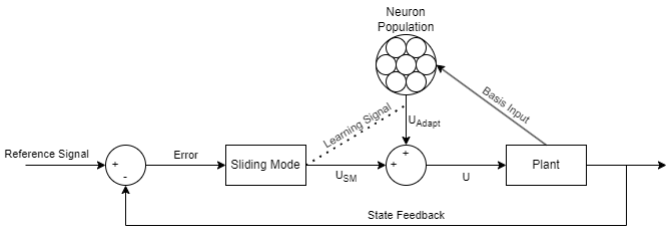

Figure 2: Block diagram of control feedback loop. The circuit is structured like a standard controller, with the addition of a neural population that gets the state, $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$, as the input and uses the sliding mode output as an error signal to update the decoding weights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

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 (35) has the same structure as the continuous time gradient descent learning rule 32, 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:

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

in equation 25. Additionally, from 25 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 35 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 formulation 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 [2] 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 [4] 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

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	197.6379	1.1186	$9.41 * 10^{-6}$	0.0083

to this controller as the Selected basis. We have introduced the remaining bases in Section III-A 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 12. We simulated the controller and spiking neuron's using Nengo [19] 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 encoder, intercept, and max firing rate from uniform distributions as described previously.

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 36 such that the end effector tracks a circle in operational space:

$$X_{ref}(t) = \begin{bmatrix} x_{ref}(t) \\ y_{ref}(t) \\ \dot{x}_{ref}(t) \\ \dot{y}_{ref}(T) \end{bmatrix} = \begin{bmatrix} x_0 + r\cos\left(\frac{2\pi}{T}t\right) \\ y_0 + r\sin\left(\frac{2\pi}{T}t\right) \\ -r\frac{2\pi}{T}\sin\left(\frac{2\pi}{T}t\right) \\ r\frac{2\pi}{T}\cos\left(\frac{2\pi}{T}t\right) \end{bmatrix}, \tag{36}$$

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 36 in the following form:

$$E = \int (X(t) - X_{ref}(t))^2 dt. \tag{37}$$

We evaluated the $argmin$ of 37 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 controllers 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 36 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 12. We consider two disturbances for testing. The first is a downward force applied to the end-effector to simulate a additional mass being added, and the second is an operational space positional cosine field on the end effector:

$$F_{dist} = [cos(x), cos(y)]^T. \tag{38}$$

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

Table II: RMSE results of different controller

	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	$\mathbf{1.539 * 10^{-2}} \pm 7.26 * 10^{-4}$	$\mathbf{2.939 * 10^{-2}} \pm 0.87 * 10^{-3}$
Grid Cells	$2.103 * 10^{-2} \pm 1.27 * 10^{-4}$	$4.109 * 10^{-2} \pm 0.35 * 10^{-3}$
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 that 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.

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 Random 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

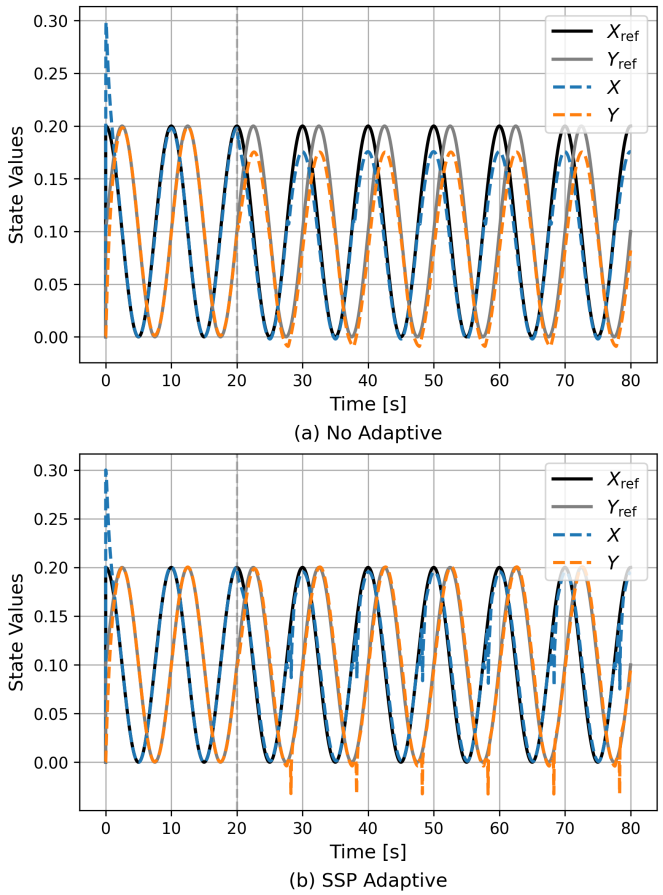

Figure 3: Simulation results of mass-based disturbance of $\boldsymbol{a}$ no-adaptive and $\boldsymbol{b}$ Place cell SSP adaptive controller. Dashed line denotes time as which disturbance is switched on.

suggests that the observed performance improvements should generalize to problems with higher-dimensional state spaces.

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 [20], [21] 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.

VI. CONCLUSION

In this work, we utilize spiking neural networks as basis functions and propose SSPs to represent the state space for an adaptive controller. With our proposed SSP-based adaptive controller, we prove the stability of these bases when operating in parallel to a classical sliding mode controller. We then demonstrate the effectiveness of this approach on simulations of a 3-link arm controller task, and show a performance increase for SSP-based tuning curves of $1.23 - 1.25$x under applied disturbances compared to past methods.

REFERENCES

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

[2] 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.

[3] 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.

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

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

[6] 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.

[7] 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.

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

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

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

[11] 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.

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

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

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

[15] 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.

[16] 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.

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

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

[19] 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.

[20] 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.

[21] 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.