applied sciences

MDPI

*Article*

# Stable Heteroclinic Channel-Based Movement Primitives: Tuning Trajectories Using Saddle Parameters

**Natasha Rouse * and Kathryn Daltorio**

Department of Mechanical and Aerospace Engineering, Case Western Reserve University, Cleveland, OH 44106, USA; kathryn.daltorio@case.edu
* Correspondence: natasha.rouse@case.edu

**Featured Application: The underlying system parameters of a biologically inspired robot control method with an intuitive visualization tool can tune precise parts of a trajectory while maintaining system stability.**

**Abstract:** Dynamic systems which underlie controlled systems are expected to increase in complexity as robots, devices, and connected networks become more intelligent. While classical stable systems converge to a stable point (a sink), another type of stability is to consider a stable path rather than a single point. Such stable paths can be made of saddle points that draw in trajectories from certain regions, and then push the trajectory toward the next saddle point. These chains of saddles are called stable heteroclinic channels (SHCs) and can be used in robotic control to represent time sequences. While we have previously shown that each saddle is visualizable as a trajectory waypoint in phase space, how to increase the fidelity of the trajectory was unclear. In this paper, we hypothesized that the waypoints can be individually modified to locally vary fidelity. Specifically, we expected that increasing the saddle value (ratio of saddle eigenvalues) causes the trajectory to slow to more closely approach a particular saddle. Combined with other parameters that control speed and magnitude, a system expressed with an SHC can be modified locally, point by point, without disrupting the rest of the path, supporting their use in motion primitives. While some combinations can enable a trajectory to better reach into corners, other combinations can rotate, distort, and round the trajectory surrounding the modified saddle. Of the system parameters, the saddle value provides the most predictable tunability across 3 orders of magnitude.

**Keywords:** biologically inspired robots; robust control; optimal control; motion planning



**Citation:** Rouse, N.; Daltorio, K. Stable Heteroclinic Channel-Based Movement Primitives: Tuning Trajectories Using Saddle Parameters. *Appl. Sci.* **2024**, *14*, 2523. https://doi.org/10.3390/app14062523

Academic Editor: Alessandro Gasparetto

Received: 8 February 2024
Revised: 6 March 2024
Accepted: 14 March 2024
Published: 16 March 2024



## 1. Introduction

To develop new versatile, robust, learnable control frameworks for robots, we are seeking ways in which a heteroclinic system can be adjusted *after* learning *while* remaining stable. Most controllers are based around "PID" parameters (Proportional Integrative and Derivative gains) that describe linearized stability around a single stable point (the goal state). Heteroclinic systems orbit not just one, but multiple equilibria, which is useful in controlling periodic motions such as robot locomotion gaits, repetitive motions in manufacturing, or more complex trajectories. In complex, high-dimensional spaces, equilbria can represent holding behaviors, system status patterns, or action sequences. By visualizing these behaviors in Cartesian space [1], we hope to better understand the underlying dynamics. The robustness of such systems comes from their proven stability [2–4] and the ease of use can come from learnability [5–7]. The ability to combine multiple states, which can be thought of as primitive building blocks, enables an inherent versatility to combine and recombine different states. Here, our goal is to increase the versatility further by demonstrating that state-to-state trajectories can be tuned by parameters associated with each state.

Creating behaviors as a composite of stable primitive building blocks has been a powerful tool for controlling robots. For example, Dynamic Movement Primitives (DMPs) have been effective for over twenty years in providing learnable, smooth kinematic control policies [8–12]. DMPs have modular components called kernels, which are generated from stable points that are activated in time sequences. If those stable points are replaced with saddle points, such that the unstable eigenvector of one saddle points to the next saddle, we can refer to them as stable heteroclinic channels (SHCs) [7,13,14]. Using SHCs to generate the kernels results in comparable performance to DMPs, and has the added benefit of visualization [1] and the potential to enable the timing of the saddle-to-saddle transition to be embedded in the dynamic system.

Saddle equilibria-based dynamical systems can help engineers better bridge gaps between neurobiology and biologically inspired artificial intelligence. The spectrum of biologically inspired controllers ranges from high-level, behavior-based controllers like finite state machines, where a robot's desired action is encoded into the control software [15,16], to low-level neuromorphic controllers like neural networks, where neuronal functions are interconnected at scale [17–19]. SHCs are intermediary. Because the connectivity is in the mathematically constructed connection matrix, they can be analyzed. Yet, they are abstract enough that the system dynamics are less prone to the "black-box", unexplainable dynamics seen in high-dimension frameworks [20,21]. For this reason, biologists have used SHCs to model the population behavior of biological neurons because they abstract the dynamics of the system into tractable components [22,23] and engineers are starting to use SHC-based movement primitives (SMPs) to control bio-inspired robots [1,14,24].

Specifically, such bio-inspired controllers make sense in high-dimensional systems where predictive forward models are unavailable. Examples include compliant robots [25,26] in complex, dynamic environments [27–29]. The current alternatives for systems like these are model-based controllers and high-dimensional "black-box" controllers. Model-based controllers are computationally expensive [26,30,31] and thus not optimal for mobile robots with limited on-board computation, and "black-box" controllers lack explainability when the robot's behavior is scrutinized [32,33]. This article builds on the SMP parametric transparency that was introduced in [1], especially for the saddle value, $\nu$, within several orders of magnitude.

This work aims to show that the SMP control framework can locally adapt part of a learned trajectory without compromising the rest of the system. We hypothesize that we can perform the following:

- *Vary the system parameters to change waveform frequencies, magnitudes, and shapes*, which will
- *change the produced trajectory's speed, precision, and/or shape.*

More specifically, we predict the following:

- *The saddle value, $\nu$, is the optimal modifier to prescribe trajectory precision.*

To establish a baseline for readers, we expand on the DMP and SHC frameworks in Section 2. Section 3 contains the SMP equations, the chosen trajectories, and the quantitative evaluation methods for parameter variation. In Section 4, we describe the effects of parameter variation both qualitatively and quantitatively. We compare the system's state space and produced trajectories across the range of a single parameter, as well as the state space and produced trajectories across the collective parameter space. Finally, we use Section 5 to summarize the results and discuss the opportunity costs of each parameter as a trajectory modulation tool.

## 2. Relevant Work

DMPs are a robotic control framework that uses a series of underlying kernels—attractor points or limit cycles—to produce trajectories [10–12,34]. The strength, timing, and growth/decay of the attractors can be varied to create custom trajectories [35]. Over the years, DMPs have been adjusted to better learn rhythmic movements [35,36], to learn from multiple non-identical demonstrations [37], and to learn joint torques along with the kinematic trajectory [38], among

other expansions [11,39,40]. In an effort to improve the versatility of DMPs in online applications, Wang et al. introduced the DMP+ framework in 2016 that can partially update the DMP+ kernel weights to locally adjust a learned trajectory [10]. To update DMP+ kernels, the weights must be relearned according to the user's chosen algorithm, whereas SMP kernel updates can be achieved using the SMP visualization feature [1] and/or the system parameters described in this article. Both SMP update methods maintain the computational complexity of an update step, while the DMP+ method may increase in complexity depending on the desired task.

SHCs are a series of saddle equilibria where the unstable manifold of one equilibrium point leads onto the stable manifold of another; this creates pathways between the saddle points (see Figure 1) [7,13,14]. SHCs have been used as a model for neural activation patterns in animals [23,41,42]. They have also been used to produce and investigate dynamical state systems [5,43], and apply those systems to robotic movement [14,28,44]. In 2015, Horchler et al. described the system parameters alpha $\alpha$, beta $\beta$, and nu $\nu$ for SHCs [7]. In their work, they described alpha as the growth rate of a kernel—how fast the kernel grows in its respective dimension. Beta was described as the kernel magnitude—the maximum amplitude of the waveform. Nu was described as the saddle value—a kernel's insensitivity to external perturbation (modeled as noise). In the SMP system, these variables (and noise) are varied synchronously to create kernels that remain connected in state space, i.e., the connected kernels create a smooth trajectory in the task space. SMPs expand SHCs into a stable, learnable system with a clear transformation from state space into a robot's task space.

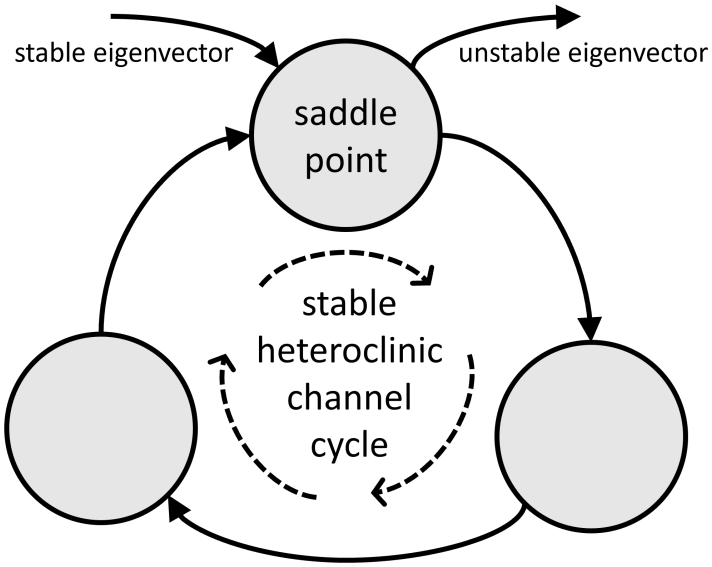


**Figure 1.** Stable heteroclinic channel (SHC) cycle with 3 saddle node points. The cycle is formed by connecting the unstable eigenvector of a saddle onto the stable eigenvector of another saddle. The Lotka–Volterra formulation of SHCs requires at least 3 saddles in the cycle.

Mathematical representations of biological systems are commonly used to develop robot control frameworks [8,18,24,40,45,46]. These frameworks have value in both biological and engineering applications, and characterizing their parameters increases ease-of-use in either application [47–49]. Some biologically inspired, transient dynamic systems, like neural networks, have gained much popularity in recent years, but at larger scales, system analysis, parametrization, and explainability become nearly impossible [26,39,50,51]. Like these other frameworks, SMPs have a biological relevance because of their construction from biologically relevant SHCs. Unlike these frameworks, SMPs are parameterizable because their parameters—from DMPs and from SHCs—have already been described separately in each framework [7,34].

## 3. Methods

The goal of this work is to demonstrate how the SMP system variables modify the system and the trajectory it produces. The trajectory for any variable of interest (e.g., end-effector position or join angle) is produced by controlling a second-order system. The MATLAB 9.14 (version R2023a, computational software by MathWorks Inc.®) code for this formulation can be found at https://github.com/NatRouse/SMP-Characterization.git (accessed 17 August 2023).

### 3.1. System Model

The SMP system model is below. Equation (1) is the governing equation. It produces the final trajectory for the system's variable of interest, $y$, using a forcing function, $f$, from the SHC formulation.

$$\tau\ddot{y} = \alpha_y(\beta_y(g - y) - \dot{y}) + f \tag{1}$$

The forcing function (2) is summed over the number of kernel functions, $K$, used in the system. $K$ is chosen based on the complexity of the system (e.g., actuatable degrees of freedom), and the desired smoothness of the produced trajectory. SMPs are $O(K^2t)$ at their most complex, where $t$ is the number of timesteps calculated [1]. In this work, the kernels are color-matched to show which state space waveform corresponds to each region of the task space trajectory.

$$f(x) = \sum_{i=1}^{K} x_i w_i \tag{2}$$

The canonical state Equation (3) is based on competitive Lotka–Volterra (LV) equations [4].

$$\tau dx_i = x_i \left( \alpha_i - \sum_{j=1}^{K} \rho_{ij} x_j \right) dt + \sum_{j=1}^{N} C_{ij} z_j \tag{3}$$

External perturbation is a critical factor in the use of LV kernels; external perturbation (modeled as Gaussian noise in this work), $z_j$, ensures that the system variable, $x$, passes close to the SHC saddle point, but not so close that the system remains in static equilibrium [7]. The effect of noise on SHCs has been explored in other work, and for practical uses, we can establish a reasonable noise magnitude compared to the rest of the system [13,14,52–55].

All of the variables across the SMP system model are defined in the following table (Table 1).

**Table 1.** SMP system variable definitions.

| *Variable* | *Definition* |
|---|---|
| $y$ | relevant system variable |
| $\tau$ | time-scaling term |
| $\alpha_y$ | system damping |
| $\beta_y$ | system stiffness |
| $g$ | system's "goal" position |
| $f$ | controller force (applied to system) |
| $K$ | total number of kernel functions |
| $w_i$ | kernel function weight |
| $x_i$ | canonical state of the system (for a single kernel) |
| $\alpha_i, \rho_i$ | system behavior parameters |
| $N$ | number of sensors |
| $C_{ij}$ | coupling matrix |
| $z_j$ | noise |

### 3.2. System Parameters

The variables that control SMP behavior are the noise $z_j$, the kernel weights $w_i$, and the parameters that make up the connection matrix $\rho_{ij}$ (seen in (3)). In our previous work [1], we observed the effect of the kernel weights on the system and optimized the weights to make the system follow a desired trajectory. The algorithm design methodology outlined in Figure 2 was used previously and is now used for this article. When the system is at unit scale (magnitude, $\beta = 1$), the kernels can be plotted in the task space using the weights as locations.

In this work, we focus on the connection matrix, $\rho_{ij}$. The connection matrix is a real, non-symmetric matrix constructed from three saddle characteristics: the growth rate $\alpha$, the magnitude $\beta$, and the saddle value $\nu$ [7]. The matrix is constructed as follows:

$$\rho_{ij} = \begin{cases} \alpha_i/\beta_i, & \textit{if i=j} \\ \dfrac{\alpha_i - \alpha_j/\nu_j}{\beta_j}, & \textit{if i=j-1} \\ \dfrac{\alpha_i + \alpha_j}{\beta_j}, & \textit{otherwise} \end{cases} \tag{4}$$

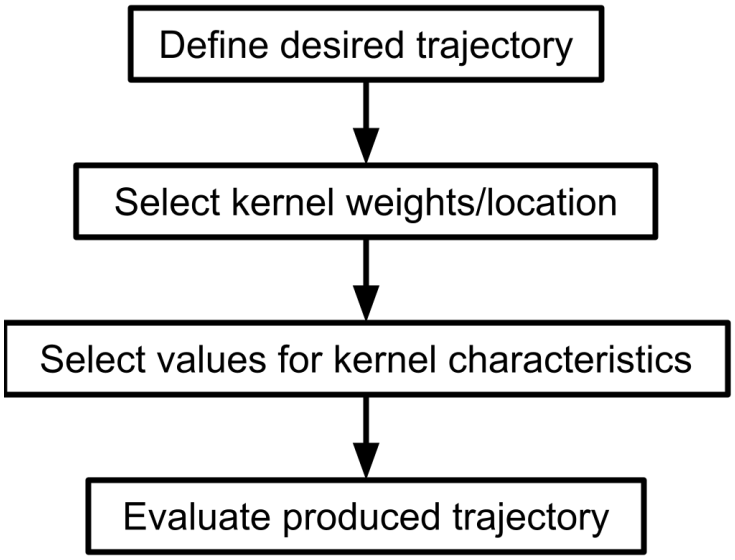


**Figure 2.** SMP design flowchart.

In the SHC system, $\alpha_i$ controls how fast the kernel grows in the $i$th dimension, $\beta_i$ is the maximum amplitude of the waveform, $x_i$, and $\nu_i$ defines the stability of the $i$th saddle with respect to input noise. When designed together, these variables and the external perturbation (modeled here as Gaussian noise) create kernels with smooth, connected pathways [7], which aids in producing a smooth trajectory. In this work, we will observe how $\alpha$, $\beta$, and $\nu$ affect SMPs.

The number of inputs $N$, the noise values $z_j$, and the coupling matrix $C_{ij}$ collectively form the final term in (3). These are selected to create a proportionally small input noise in comparison to the rest of the system. As noted in Section 3.1, this noise is necessary to produce a trajectory along the kernel pathways.

To show the effect of each system parameter on the resulting trajectory, we vary each parameter over a spread of values for a single kernel and for all four kernels. The effect on the trajectory is measured as an area error from the original trajectory (the square in Figure 3). Additionally, since SMP kernels are functions of time, we can measure the time it takes for each kernel to grow and decay. A baseline trial (discussed in Section 3.3) is compared to trials with individually and collectively varied parameter values, and the kernel function waveforms (produced by $x$ from (3)) show the effects.

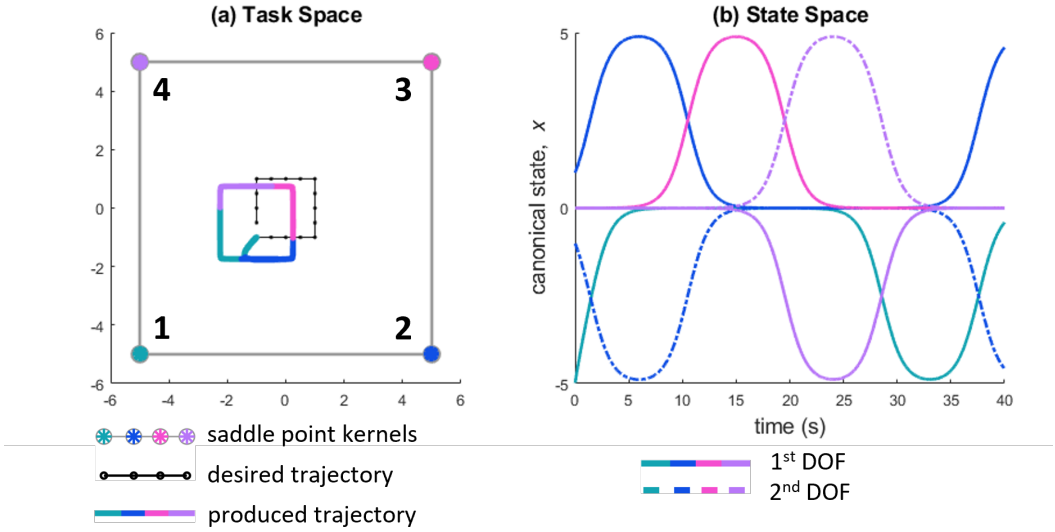


**Figure 3.** (**a**) Saddle point kernels, desired trajectory, and produced trajectory for a square trajectory plotted in the task space. (**b**) The canonical state waveforms of the weighted kernels. The degrees of freedom are the $x$ and $y$ directions in Cartesian space. The initialization waveform of kernel one (the green waveform at time = 0) contributes to the part of the produced trajectory that deviates from the square. For the remainder of this work, the initialization and steady state error will be ignored.

### *3.3. Desired Trajectories*

#### 3.3.1. Square

The square trajectory-following task is defined in Figure 3. The trajectory starts at $(-1,-1)$ and moves counterclockwise around a square. The desired trajectory is the first input to the system. Next, the kernel weights are chosen [1]. $K = 4$ corresponds to each corner of the square. This choice enables each kernel and its associated trajectory region to be visualized separately (see Figure 3).

The baseline SMP system parameters for the square trajectory are $\alpha, \beta, \nu = 1$. These baseline parameters produce the square shown in Figure 3a, which has a runtime of approximately 40 s; all the further parameter trials are compared against this baseline square. The variables from the system model that remain the same across all trials are listed below:

- $\tau = 1$;
- $\alpha_y = 4$;
- $N = 4$;
- $g = (-1, -0.5)$;
- $K = 4$;
- $\beta_y = 1$;
- $z_j = 10^{-9}$.

#### 3.3.2. Number "3" Shape

To explore how the system parameters can be used for trajectory tuning, a number "3" trajectory is reproduced using eight kernels ($K = 8$). The original system parameters for this trajectory are $\alpha = 10$, $\beta = 1$, and $\nu = 1.2$. These are the same parameters used for the complex trajectories in our previous work [1]. This trajectory starts at $(-1,2)$ and ends at $(-0.8,-0.5)$.

### *3.4. Evaluating the Produced Trajectories*

Two trial types are used for the square trajectory:

- Collective parameter change: *all* $\alpha$ or *all* $\beta$ or *all* $\nu$;
- Individual parameter change: a single $\alpha$, $\beta$ or $\nu$.

To evaluate the effects of varying these parameters, we measure the square's time-to-completion and the area error of the produced trajectory. Time-to-completion is the time it takes the system to complete the square trajectory. We determine closure of the square by identifying the first point at which the trajectory crosses itself. The area error is the difference in area between the baseline produced trajectory (Figure 3) and the modified produced trajectory. The area enclosed (in the task space) by the original produced trajectory and each new produced trajectory is found, and their difference is calculated.

$$area\ error = A_{\text{baseline}} - A_{\text{modified}} \tag{5}$$

For the number "3" shape, the distance error is measured as the sum of the shortest Euclidean distances between the desired and modified "3" shapes. The desired "3" is described using 13 waypoints, and this trajectory is used to select the weights of the 8 kernels that are used to run the system. The weight selection process is described in [1].

$$distance\ error = \sum^{DOF} \sqrt{\sum^{t} (y_{\text{modified}} - y_{\text{desired}})^2} \tag{6}$$

## 4. Results

First, each system parameter was halved and doubled from an original value of 1 for all kernels. The effects on the system are summarized in Table 2 and plotted in Figure 4. Next, each system parameter was varied across a range for an individual kernel and collectively for all kernels. The system time and area error were collected for these trials. The results are shown in Figure 5. The results for each parameter are described below. Finally, with the accumulated information on how these variables affect the SMP system and its produced trajectory, we demonstrate that the saddle characteristics can be used to reduce the distance error of the "3"-shaped trajectory.

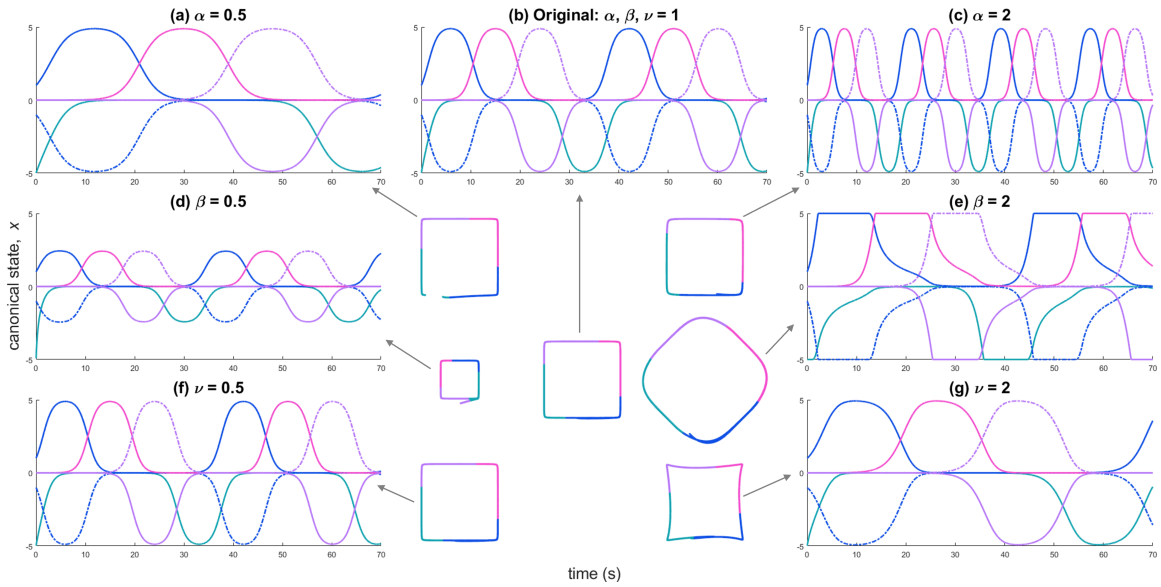


**Figure 4.** State space waveforms of the square trajectory kernels. The canonical state, $x$, as described in (3) is plotted against time. An "original" state (**b**) is set in the middle of the top row. It shows a four-kernel system (square) that has two full activations: each kernel is activated twice in time. Each parameter is reduced by half (**a**,**d**,**f**) or doubled (**c**,**e**,**g**). All of the trials are run for the same amount of time. The effects on the system can be seen in the function waveform frequencies, magnitudes, and shapes, and are described in Table 2.

**Table 2.** The effect on the SMP system (Canonical State Waveform) and the SMP results (Produced Trajectory) for collective parameter changes—all $\alpha$, $\beta$, and $\nu$ values.

| Parameter | | | Canonical State Waveform | | Produced Trajectory | |
|---|---|---|---|---|---|---|
| | | | *Frequency* | *Magnitude* | *Shape* | *Size* |
| *Growth Rate* | $\alpha$ | $(0, \infty)$ | Increases | No effect | Rotation | Reduces |
| *Magnitude* | $\beta$ | $(0, \infty)$ | No effect | Increases | Rotation | Increases |
| *Insensitivity to Noise* | $\nu$ | $[1, \infty)$ | Decreases | No effect | Increased precision around kernel locations | No effect |

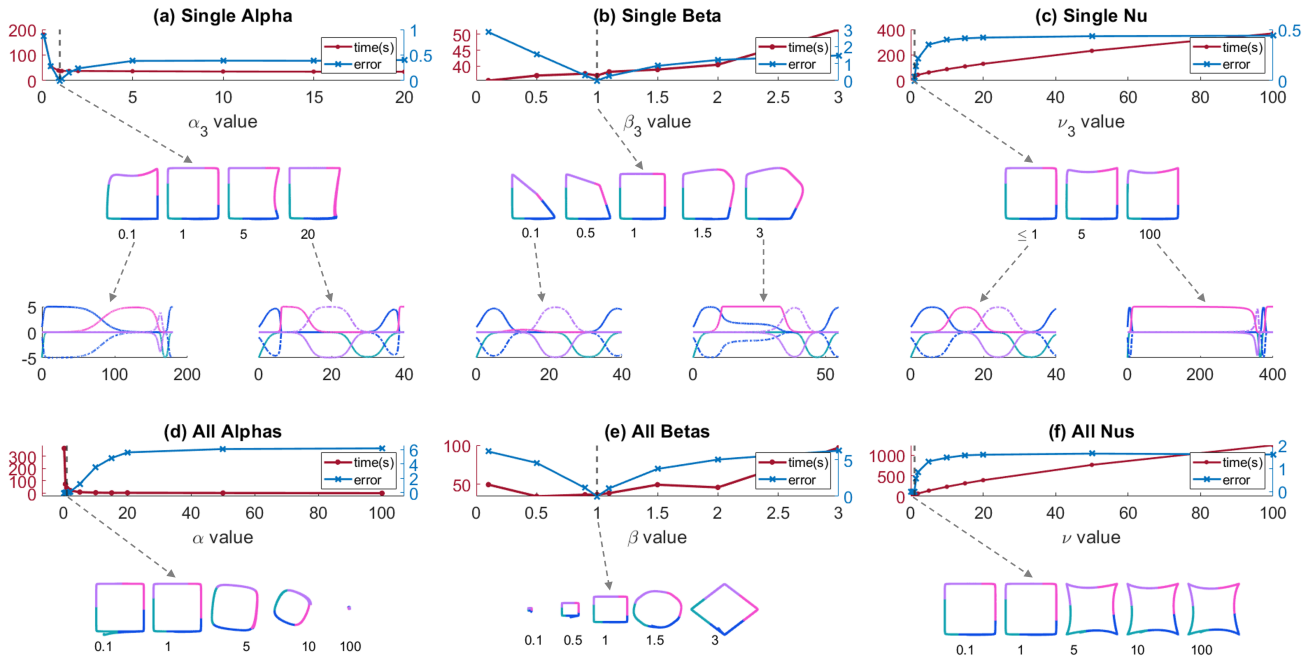


**Figure 5.** Plot of time (red) and area error (blue) vs various parameter values: (**a**) $\alpha_3$, (**b**) $\beta_3$, (**c**) $\nu_3$, (**d**) $\alpha_{all}$, (**e**) $\beta_{all}$, and (**f**) $\nu_{all}$. The produced trajectory for various values is pictured below each plot. Each set of produced trajectories is scaled according to the original (value = 1). The individual parameter trials (**a**–**c**) also have canonical state plots below their produced trajectories. Note that the x-axis, time, varies across the canonical state plots. This agrees with the change in time across the individual parameter trials.

### *4.1. Alpha: Growth Rate*

#### 4.1.1. All Alpha

In the state space, varying $\alpha$ varies the frequency of the canonical state waveform almost proportionally (Figure 4a,c). When $\alpha >> 1$, there is a rotation introduced into the produced trajectory and the trajectory decreases in size (Figure 5d). This can be attributed to the increased frequency of the waveforms. Faster waveforms indicates less time for the kernels to affect the system. The kernels grow so quickly that the trajectory does not fall into the saddle points' neighborhoods in state space. In the task space, this appears as the trajectory passing farther away from the kernels resulting in reduced precision.

#### 4.1.2. Single Alpha

$\alpha_3$ was varied from 0.1 to 20 (Figure 5a) while $\alpha_{i \neq 3} = 1$. According to (3) and (4), the prior and subsequent kernels are affected when a single $\alpha$ is changed.

When $\alpha_3 < 1$, the third kernel's pink waveform rises gradually, slowing the kernel down and creating a wider waveform in state space. The second (prior, blue) kernel's waveform decreases slowly—at a rate similar to the third kernel—while the following

kernel's entire waveform rises and falls quickly. The fourth kernel (purple) activates at a smaller magnitude than the others. In the produced trajectory, this presents itself as a curve away from kernel 4's corner—the fourth kernel's trajectory passes farther away from the kernel than in the unit trial ($\alpha_{all} = 1$).

When $\alpha_3 > 1$, the pink third kernel waveform rises more sharply than the other kernels. To maintain the LV construction, the previous blue kernel's waveform must decrease sharply. This translates across the entire kernel 3 waveform, which activates at a smaller magnitude than the other kernels. In the produced trajectory, this means that the trajectory curves away from the kernel 3 corner.

### *4.2. Beta: Magnitude*

#### 4.2.1. All Beta

In the state space, varying $\beta$ changes the magnitude of the canonical state waveform almost proportionally (Figure 4d,e). When $\beta < 1$, the proportional magnitude translates to the size of the produced trajectory, e.g., at a halved $\beta$, the square's sides are halved. When $\beta > 1$, the waveforms are truncated at the system's maximum—the maximum weight assigned to any kernel. All the waveforms are wider, slower, and decay more slowly than the original trial ($\beta = 1$). Slower and wider waveforms indicate more time spent in each kernel's neighborhood. In the produced trajectory, this translates to a larger, rotated square (Figure 5e). The rotation is likely the truncated tops of the waveforms forming new edges in the produced trajectory.

When a kernel remains activated at its maximum, the trajectory is pulled onto that kernel's stable eigenvector [1] and remains in the kernel's neighbourhood for a longer period of time [7]. In the task space, this presents as a straight line where the corners of the square used to be—the trajectory stays in the kernel's neighborhood instead of approaching and leaving. Each straight line is a new edge in the produced trajectory, thus the entire square appears to rotate.

#### 4.2.2. Single Beta

$\beta_3$ was varied from 0.1 to 3 (Figure 5b) while $\beta_{i\neq 3} = 1$.

When $\beta_3 < 1$, the third kernel's waveform magnitude is smaller than the other kernels, but does not affect the others in any other way. In the produced trajectory, the trajectory skips kernel 3's pink corner, moving almost directly from the kernel 2 corner to the kernel 4 corner.

When $\beta_3 > 1$, the third kernel's waveform is truncated (as in the collective parameter trial). Similar to the collective parameter trial, the truncated waveform causes the trajectory to be pulled into the third kernel's neighborhood faster and longer than the other kernels. This presents itself as a new edge tangential to the kernel location (recall the new edges formed when $\beta_{all} = 2$ in Figure 4e).

### *4.3. Nu: Insensitivity to Noise*

#### 4.3.1. All Nu

Decreasing $\nu$ below 1 showed no change in either the state space or the produced trajectory. When $\nu > 1$, the waveforms become more square; they are not truncated, but they remain near their maximum for an extended period of time before decaying (Figure 4f,g). Unlike the modified $\beta$ waveform activation at maximum, the modified $\nu$ kernels do not remain at their maximum value. Additionally, the rate of both their rise and decay is comparable to the original value ($\nu = 1$). In the produced trajectory, this change presents itself as an increased sharpness in the corners of the square (Figure 5f).

#### 4.3.2. Single Nu

$\nu_3$ was varied from 1 to 100 (Figure 5c) while $\nu_{i\neq 3} = 1$. When $\nu_3 < 1$, there was no change in the state space or the produced trajectory.

When $\nu_3 > 1$, the third waveform increases in width, does not remain at its maximum value, and decays quickly. To maintain LV construction, the following kernel's waveform must rise at a similar rate; kernel 4's waveforms rises and decays quickly, resulting in a smaller waveform. In the produced trajectory, these waveform characteristics present themselves as an increased precision in the kernel 3 corner, a curve along the top edge, and a decreased precision in the kernel 4 corner.

### *4.4. Complex Trajectory Tuning*

With the knowledge of how these parameters affect both the system's canonical state and the produced trajectory, we can now use them to modify the path precision of a trajectory at key points. In Figure 6, we define the trajectory as a number "3" using thirteen waypoints. We initialize our system with eight kernel weights sampled from the trajectory and $\alpha = 10$, $\beta = 1$, and $\nu = 1.2$. These variables originate from our previous work [1], but they could be derived via the process outlined in [7]. With this initialization, we achieve a trajectory that mimics a number three, but could represent the desired shape (Figure 6, top row) by being more precise in the inner point of the shape. The initial distance error (according to Equation (6)) is 7.588.

According to the previous Section 4.3, we can use $\nu$ to change the time that the trajectory spends in a kernel's neighborhood, thus affecting the precision of that portion of the trajectory. We can also use $\alpha$ to produce a similar effect by reducing it below one, but from the trials in Section 4.1 and Figure 5, this can produce some unexpected warping of the trajectory after the modified kernel. Because of this, we will use $\nu$ as our modification tool.

For a new, tuned trajectory, we want to match the curves closest to the inner point of the number "3". Initially, we increase just kernel four's $\nu$ value to $\nu = 2$. This is the kernel that corresponds to the inner point. This improves the trajectory according to the distance error (lowering it from 7.588 to 6.439), but we can see in Figure 6 (middle row) that the produced trajectory can better fit the original desired trajectory. With this in mind, we increase the saddle values for kernels three and five, which are the kernels that surround kernel four—the inner point. Now the trajectory more tightly follows the inner point of the shape (Figure 6, bottom row), reducing the distance error to 5.149. The modifications and error changes are shown in Table 3.

**Table 3.** Saddle value modifications and the resulting distance error values (with percentage decrease from baseline produced trajectory) for the complex number "3" trajectory.

| Modification | Error |
|---|---|
| *Baseline* | 7.588 |
| $\nu_4 = 2$ | 6.439 (15% decrease from baseline ) |
| $\nu_{3,4,5} = 2$ | 5.149 (32% decrease from baseline) |

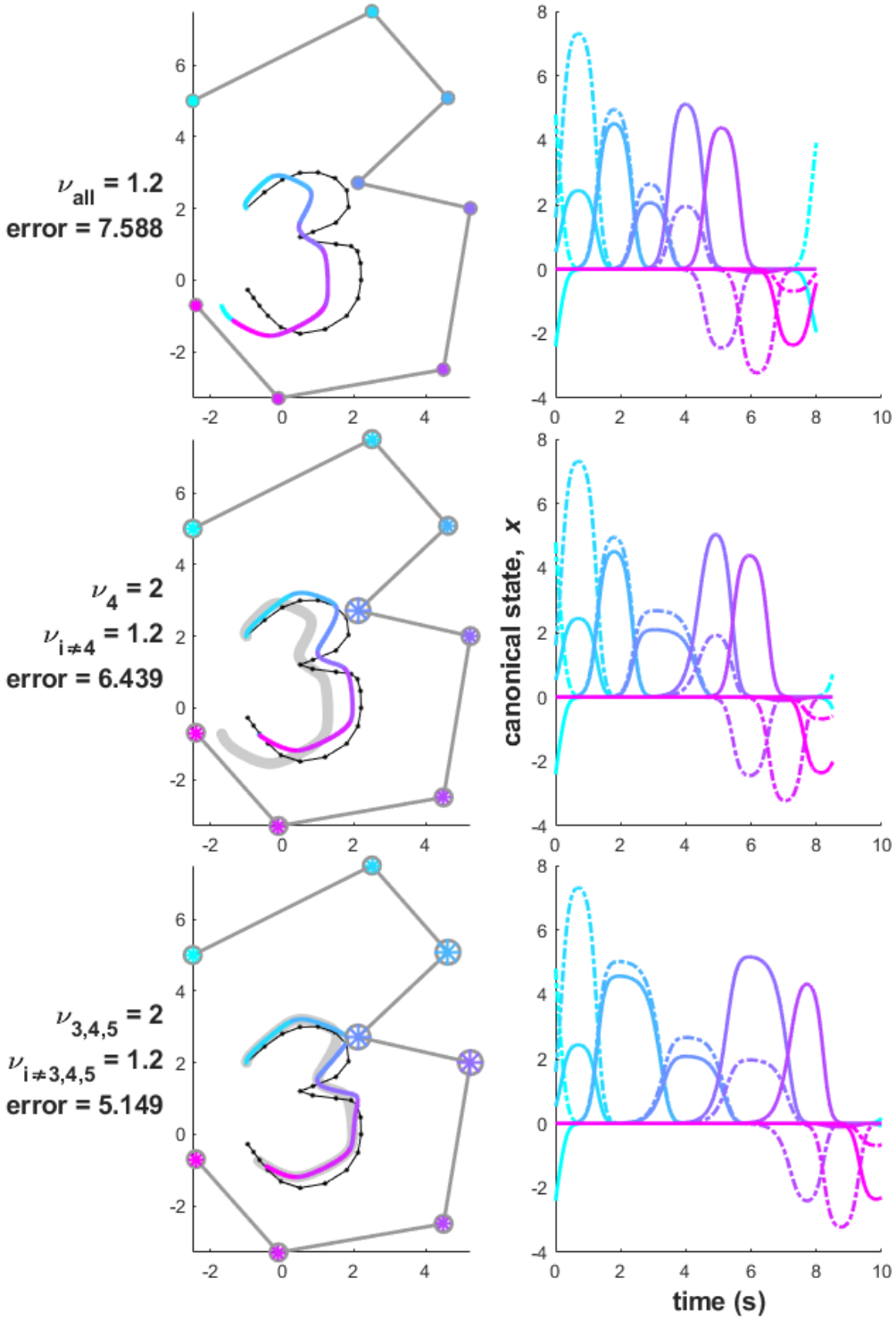


**Figure 6.** "Number three" trajectory. Left column: the produced trajectory, desired trajectory and kernels in the task space. The desired trajectory starts at (−1,2) (dotted black line). The kernel weights inform their location in the task space (colored dots, gray lines), and the marker sizes indicate their saddle values. The produced trajectory (colored line) is unmodified in the top row, and modified according to the labeled saddle values in the middle and bottom rows. The distance error for the new iteration of produced trajectory is listed under each modification. The produced trajectory from each previous trial is shown in gray to show the progression from an unmodified system to a more tuned system. Right column: the kernel waveforms in state space. With each modification, the system time becomes longer and the distance error decreases.

## 5. Discussion

In this work, we characterized the effect of the SMP system parameters $\alpha$, $\beta$, and $\nu$ on both the system and its produced trajectory. Additionally, we used these parameters to tune a localized part of a more complex trajectory without disrupting the rest of the trajectory.

Alpha is a frequency (speed) modifier. As $\alpha$ increases, the kernel takes less time to grow and decay. This decreases the effect of that kernel on the trajectory, causing lower precision in the trajectory areas where a kernel's $\alpha$ value has been increased. In the task space, increasing $\alpha$ increases the trajectory's speed for that kernel's activation.

Beta is a magnitude (scale) modifier. As $\beta$ increases, the magnitude of the kernel's waveform increases until it would exceed the system's maximum. At that point, the waveform is truncated and the produced trajectory's shape around that kernel is changed. In the task space, $\beta$ can be used to under- or over-direct the trajectory past a kernel by decreasing or increasing it, respectively.

Nu is a precision modifier. As $\nu$ increases, the kernel takes more time to grow and decay. This increases the "power" of that kernel on the trajectory, causing a higher precision in those areas in the task space.

The amount of time the system takes to grow/decay towards a kernel determines how closely its associated trajectory will be attracted to that kernel's saddle point. With more time, the trajectory can be executed more closely, e.g., sharper corners for the square trajectory and a more defined inner point for the number "3" trajectory. With less time, the trajectory will be executed more loosely, potentially skipping a kernel's associated portion of the trajectory.

We hypothesized that the kernels can be individually modified to vary the trajectory's fidelity in that location. Specifically, increasing the saddle value $\nu$ would increase the trajectory's precision around that saddle. This was achieved in tuning a number "3" shape. We were able to produce a 32% decrease in the trajectory error by increasing the $\nu$ values of the saddles in the inner point of the shape.

In a limited parameter space, the system variables can be used separately or in tandem to modify the produced trajectory at key points without significantly affecting the surrounding trajectory areas. For example, if a specific kernel becomes higher priority than others, we can increase the activation time for that kernel by increasing its $\beta$ or $\nu$ value, or decreasing its $\alpha$ value. If a kernel becomes lower priority or needs to be avoided, we can decrease its activation time by increasing its $\alpha$ value or decreasing its $\beta$ value. The variable $\beta$ shows a unique feature when it is reduced below 1 for all kernels; it acts as a scaling factor for the whole trajectory.

For practical applications there are three main considerations: speed, precision, and scale. There is a trade off between speed and precision in adjusting $\alpha$ and $\nu$, especially when their values are greater than 1. Due to the visualization feature of SMPs, the scale of a trajectory path can be easily adjusted using $\beta$, and a new path can be initialized quickly. Since the SMP system is stable and robust for this parameter space, all of the system parameters can be learned via a user's chosen optimization algorithm according to the desired task specifications.

In addition to predictable changes at small parameter changes, we observed unexpected changes to the produced trajectories at large parameter changes, such as rotations, distortions, and roundness in corners and sharp turns. Changes in the task space are implied when we look at the state space representations of these trajectories. The state space plots indicate that each kernel waveform is interdependent on its neighbors and that interdependence uniquely affects their activation.

Special consideration should be given to the saddle value $\nu$, which produces the most predictable changes in the produced trajectory over the largest range of values. We were able to vary $\nu$ over 3 orders of magnitude in both the single variable trials and the collective variable trials, and we did not observe any unpredicted changes to the produced trajectory. This points to $\nu$ as the ideal parameter for trajectory tuning (via trajectory precision) for this system.

Several opportunities arise from this work. First, a full characterization of SMP state space may explain the unexpected trajectory changes illustrated in this paper. Additionally, we have not characterized the effect of noise on the SMP system. The SMP framework depends on perturbation to be functional, so an extended mathematical framework could leverage noise as an input to directly introduce sensory information to the system. In this work, we use SMPs to produce a kinematic trajectory plan. As with other modular activation frameworks, SMPs can be used to encode motor activation for different degrees of freedom on a robot instead. Finally, the spatial definition of SMPs enables more complicated network topologies. Each pathway, cycle, or network could encode behaviors—either desired behaviors for a robotic platform or observed behaviors from other, less explainable (less visualizable) control frameworks.

## 6. Conclusions

The conclusions for this article are as follows:

- We tuned state-to-state trajectories of a saddle point-based control system by varying the parameters associated with each state, without jeopardizing the stability of the system at large.
- We reduced the distance error of a complex trajectory (number "3" shape) by 32% by locally tuning the trajectory after it was initialized.
- We identified that the saddle value $\nu$ may be the ideal tool for SMP trajectory tuning because it produces predictable results when it is varied over 3 orders of magnitude.

**Author Contributions:** Conceptualization, N.R. and K.D.; Data curation, N.R.; Formal analysis, N.R. and K.D.; Funding acquisition, K.D.; Investigation, N.R.; Methodology, N.R.; Project administration, K.D.; Resources, K.D.; Software, N.R.; Supervision, K.D.; Validation, N.R.; Writing—original draft, N.R.; Writing—review and editing, N.R. and K.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This project was funded by NSF Grant #2047330.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The original code presented in the study are openly available in GitHub at https://github.com/NatRouse/SMP-Characterization.git (accessed on 5 August 2023). This code was derived from the following publicly-available resource: https://github.com/horchler/SHCTools.git (accessed on 5 August 2023).

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| SMP | Stable heteroclinic channel-based movement primitive |
| DMP | Dynamic movement primitive |
| SHC | Stable heteroclinic channel |
| DOF | Degrees of freedom |

## References

1. Rouse, N.A.; Daltorio, K.A. Visualization of Stable Heteroclinic Channel-Based Movement Primitives. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2343–2348. [CrossRef]
2. Guckenheimer, J.; Holmes, P. Structurally stable heteroclinic cycles. *Math. Proc. Camb. Philos. Soc.* **1988**, *103*, 189–192. [CrossRef]
3. Krupa, M. Robust Heteroclinic Cycles. *J. Nonlinear Sci.* **1997**, *7*, 129–176. [CrossRef]
4. Bick, C.; Rabinovich, M.I. On the occurrence of stable heteroclinic channels in Lotka–Volterra models. *Dyn. Syst.* **2010**, *25*, 110–197. [CrossRef]
5. Voit, M.; Meyer-Ortmanns, H. Dynamical Inference of Simple Heteroclinic Networks. *Front. Appl. Math. Stat.* **2019**, *5*, 63. [CrossRef]

6. Ashwin, P.; Postlethwaite, C. Designing Heteroclinic and Excitable Networks in Phase Space Using Two Populations of Coupled Cells. *J. Nonlinear Sci.* **2016**, *26*, 345–364. [CrossRef]
7. Horchler, A.D.; Daltorio, K.A.; Chiel, H.J.; Quinn, R.D. Designing responsive pattern generators: Stable heteroclinic channel cycles for modeling and control. *Bioinspiration Biomim.* **2015**, *10*, 026001. [CrossRef] [PubMed]
8. Schaal, S.; Kotosaka, S.; Sternad, D. Nonlinear Dynamical Systems as Movement Primitives. *Int. Conf. Humanoid Robot. Camb. MA* **2001**, *38*, 117–124. [CrossRef]
9. Schaal, S.; Peters, J.; Nakanishi, J. Control, planning, learning, and imitation with dynamic movement primitives. *Workshop Bilater. Paradig. Hum. Humanoids IEEE Int. Conf. Intell. Robot. Syst. (IROS 2003)* **2003**, 1–21.
10. Wang, R.; Wu, Y.; Chan, W.L.; Tee, K.P. Dynamic Movement Primitives plus: For enhanced reproduction quality and efficient trajectory modification using truncated kernels and local biases. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Daejeon, Republic of Korea, 9–14 October 2016; pp. 3765–3771. [CrossRef]
11. Koutras, L.; Doulgeri, Z. A novel DMP formulation for global and frame independent spatial scaling in the task space. In Proceedings of the 29th IEEE International Conference on Robot and Human Interactive Communication, RO-MAN 2020, Naples, Italy, 31 August–4 September 2020; pp. 727–732. [CrossRef]
12. Kong, L.H.; He, W.; Chen, W.S.; Zhang, H.; Wang, Y.N. Dynamic Movement Primitives Based Robot Skills Learning. *Mach. Intell. Res.* **2023**, *20*, 396–407. [CrossRef]
13. Rabinovich, M.; Huerta, R.; Laurent, G. Transient dynamics for neural processing. *Science* **2008**, *321*, 48–50. [CrossRef]
14. Daltorio, K.A.; Horchler, A.D.; Shaw, K.M.; Chiel, H.J.; Quinn, R.D. Stable Heteroclinic Channels for Slip Control of a Peristaltic Crawling Robot. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Lepora, N., Mura, A., Krapp, H., Verschure, P., Prescott, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8064, pp. 59–70. [CrossRef]
15. Nogueira, H.S.; Oliveira, F.G.; Pio, J.L. Discrete Movement Control of a Bio-Inspired Multi-Legged Robot. In Proceedings of the 2021 Latin American Robotics Symposium, 2021 Brazilian Symposium on Robotics, and 2021 Workshop on Robotics in Education, LARS-SBR-WRE 2021, Natal, Brazil, 11–15 October 2021; pp. 174–179. [CrossRef]
16. Yang, J.; Wang, X.; Bauer, P. V-Shaped Formation Control for Robotic Swarms Constrained by Field of View. *Appl. Sci.* **2018**, *8*, 2120. [CrossRef]
17. Maass, W. Networks of spiking neurons: The third generation of neural network models. *Neural Netw.* **1997**, *10*, 1659–1671. [CrossRef]
18. Riddle, S.; Nourse, W.R.; Yu, Z.; Thomas, P.J.; Quinn, R.D. A Synthetic Nervous System with Coupled Oscillators Controls Peristaltic Locomotion. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 2022; Volume 13548 LNAI, pp. 249–261. [CrossRef]
19. Szczecinski, N.S.; Goldsmith, C.A.; Nourse, W.R.; Quinn, R.D. A perspective on the neuromorphic control of legged locomotion in past, present, and future insect-like robots. *Neuromorphic Comput. Eng.* **2023**, *3*, 023001. [CrossRef]
20. Sado, F.; Loo, C.K.; Liew, W.S.; Kerzel, M.; Wermter, S. Explainable Goal-driven Agents and Robots—A Comprehensive Review. *ACM Comput. Surv.* **2023**, *55*, 211. [CrossRef]
21. Setchi, R.; Dehkordi, M.B.; Khan, J.S. Explainable Robotics in Human-Robot Interactions. *Procedia Comput. Sci.* **2020**, *176*, 3057–3066. [CrossRef]
22. Shaw, K.M.; Lu, H.; McManus, J.M.; Cullins, M.J.; Chiel, H.J.; Thomas, P.J. Evidence for a central pattern generator built on a heteroclinic channel instead of a limit cycle. In Proceedings of the Computational and Systems Neuroscience 2010, Salt Lake City, UT, USA, 25 February–2 March 2010; Frontiers Media SA; Volume 4. [CrossRef]
23. Shaw, K.M.; Lyttle, D.N.; Gill, J.P.; Cullins, M.J.; Mcmanus, J.M.; Lu, H.; Thomas, P.J.; Chiel, H.J.; Shaw, K.M.; Lyttle, D.N.; et al. The significance of dynamical architecture for adaptive responses to mechanical loads during rhythmic behavior. *J. Comput. Neurosci.* **2015**, *38*, 25–51. [CrossRef] [PubMed]
24. Daltorio, K.A.; Boxerbaum, A.S.; Horchler, A.D.; Shaw, K.M.; Chiel, H.J.; Quinn, R.D. Efficient worm-like locomotion: Slip and control of soft-bodied peristaltic robots. *Bioinspiration Biomim.* **2013**, *8*, 035003. [CrossRef] [PubMed]
25. Riddle, S.; Jackson, C.; Daltorio, K.A.; Quinn, R.D. A Dynamic Simulation of a Compliant Worm Robot Amenable to Neural Control. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 2023; Volume 14157 LNAI, pp. 338–352. [CrossRef]
26. Kuwabara, J.; Nakajima, K.; Kang, R.; Branson, D.T.; Guglielmino, E.; Caldwell, D.G.; Pfeifer, R. Timing-based control via echo state network for soft robotic arm. In Proceedings of the The 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, QLD, Australia, 10–15 June 2012; pp. 1–8. [CrossRef]
27. Graf, N.M.; Grezmak, J.E.; Daltorio, K.A. Get a grip: Inward dactyl motions improve efficiency of sideways-walking gait for an amphibious crab-like robot. *Bioinspiration Biomim.* **2022**, *17*, 066008. [CrossRef]
28. Brecelj, T.; Petrič, T. Application of a Phase State System for Physical Human-Humanoid Robot Collaboration. *Mech. Mach. Sci.* **2023**, *135 MMS*, 89–96. [CrossRef]
29. Picardi, G.; Chellapurath, M.; Iacoponi, S.; Stefanni, S.; Laschi, C.; Calisti, M. Bioinspired underwater legged robot for seabed exploration with low environmental disturbance. *Sci. Robot.* **2020**, *5*, 1012. [CrossRef]
30. Santina, C.D.; Duriez, C.; Rus, D. Model-Based Control of Soft Robots: A Survey of the State of the Art and Open Challenges. *IEEE Control. Syst.* **2023**, *43*, 30–65. [CrossRef]

31. Sun, Y.; Zhang, D.; Liu, Y.; Lueth, T.C. FEM-Based Mechanics Modeling of Bio-Inspired Compliant Mechanisms for Medical Applications. *IEEE Trans. Med. Robot. Bionics* **2020**, *2*, 364–373. [CrossRef]
32. Wachter, S.; Mittelstadt, B.; Floridi, L. Transparent, explainable, and accountable AI for robotics. *Sci. Robot.* **2017**, *2*, 31. [CrossRef]
33. Bundy, A. Preparing for the future of Artificial Intelligence. *AI Soc.* **2016**, *32*, 285–287. [CrossRef]
34. Schaal, S. Dynamic Movement Primitives—A Framework for Motor Control in Humans and Humanoid Robotics. In *Adaptive Motion of Animals and Machines*; Springer: Tokyo, Japan, 2006; pp. 261–280. [CrossRef]
35. Ijspeert, A.J.; Nakanishi, J.; Hoffmann, H.; Pastor, P.; Schaal, S. Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors. *Neural Comput.* **2013**, *25*, 328–373. [CrossRef]
36. Ernesti, J.; Righetti, L.; Do, M.; Asfour, T.; Schaal, S. Encoding of periodic and their transient motions by a single dynamic movement primitive. In Proceedings of the IEEE-RAS International Conference on Humanoid Robots, Osaka, Japan, 29 November–1 December 2012; pp. 57–64. [CrossRef]
37. Paraschos, A.; Daniel, C.; Peters, J.R.; Neumann, G. Probabilistic Movement Primitives. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 2616–2624.
38. Denisa, M.; Gams, A.; Ude, A.; Petric, T. Learning Compliant Movement Primitives Through Demonstration and Statistical Generalization. *IEEE/ASME Trans. Mechatron.* **2016**, *21*, 2581–2594. [CrossRef]
39. Pastor, P.; Righetti, L.; Kalakrishnan, M.; Schaal, S. Online movement adaptation based on previous sensor experiences. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 365–371.
40. Wensing, P.M.; Slotine, J.J. Sparse Control for Dynamic Movement Primitives. *IFAC-PapersOnLine* **2017**, *50*, 10114–10121. [CrossRef]
41. Laurent, G.; Stopfer, M.; Friedrich, R.W.; Rabinovich, M.I.; Volkovskii, A.; Abarbanel, H.D. Odor Encoding as an Active, Dynamical Process: Experiments, Computation, and Theory. *Annu. Rev. Neurosci.* **2001**, *24*, 263–297. [CrossRef]
42. Rabinovich, M.; Volkovskii, A.; Lecanda, P.; Huerta, R.; Abarbanel, H.D.I.; Laurent, G. Dynamical Encoding by Networks of Competing Neuron Groups: Winnerless Competition. *Phys. Rev. Lett.* **2001**, *87*, 068102. [CrossRef]
43. Shaw, K.M.; Park, Y.M.; Chiel, H.J.; Thomas, P.J. Phase resetting in an asymptotically phaseless system: On the phase response of limit cycles verging on a heteroclinic orbit. *SIAM J. Appl. Dyn. Syst.* **2012**, *11*, 350–391. [CrossRef]
44. Brecelj, T.; Petrič, T. Utilizing a Phase State System for Reliable Physical Assistance in Human-Humanoid Robot Collaboration. In Proceedings of the 2023 21st International Conference on Advanced Robotics (ICAR), Abu Dhabi, United Arab Emirates, 5–8 December 2023; pp. 258–263. [CrossRef]
45. Nourse, W.; Quinn, R.D.; Szczecinski, N.S. An Adaptive Frequency Central Pattern Generator for Synthetic Nervous Systems. In *Proceedings of the Biomimetic and Biohybrid Systems, Paris, France, 17–20 July 2018*; Vouloutsi, V., Halloy, J., Mura, A., Mangan, M., Lepora, N., Prescott, T.J., Verschure, P.F., Eds.; Springer: Cham, Switzerland, 2018; pp. 361–364.
46. Azambuja, R.D.; Klein, F.B.; Adams, S.V.; Stoelen, M.F.; Cangelosi, A. Short-term plasticity in a liquid state machine biomimetic robot arm controller. In Proceedings of the International Joint Conference on Neural Networks, Anchorage, AK, USA, 14–19 May 2017; pp. 3399–3408. [CrossRef]
47. Rico Mesa, E.M.; Hernández-Riveros, J.A. Determination of the Central Pattern Generator Parameters by a Neuro-Fuzzy Evolutionary Algorithm. In *Proceedings of the Advances in Emerging Trends and Technologies, Riobamba, Ecuador, 26–30 October 2020*; Botto-Tobar, M., León-Acurio, J., Díaz Cadena, A., Montiel Díaz, P., Eds.; Springer: Cham, Switzerland, 2020; pp. 518–530.
48. Szczecinski, N.S.; Hunt, A.J.; Quinn, R.D. Design process and tools for dynamic neuromechanical models and robot controllers. *Biol. Cybern.* **2017**, *111*, 105–127. [CrossRef]
49. Fitzpatrick, M.N.; Wang, Y.; Thomas, P.J.; Quinn, R.D.; Szczecinski, N.S. Robotics Application of a Method for Analytically Computing Infinitesimal Phase Response Curves. In *Proceedings of the Biomimetic and Biohybrid Systems*; Vouloutsi, V., Mura, A., Tauber, F., Speck, T., Prescott, T.J., Verschure, P.F.M.J., Eds.; Springer: Cham, Switzerland, 2020; pp. 104–115.
50. Jaeger, H. *The "Echo State" Approach to Analysing and Training Recurrent Neural Networks-with an Erratum Note*; Technical Report; German National Research Center for Information Techology: Bonn, Germany, 2001.
51. Yuan, Y.; Li, Z.; Zhao, T.; Gan, D. DMP-Based Motion Generation for a Walking Exoskeleton Robot Using Reinforcement Learning. *IEEE Trans. Ind. Electron.* **2020**, *67*, 3830–3839. [CrossRef]
52. Li, D.; Cross, M.C.; Zhou, C.; Zheng, Z. Quasiperiodic, periodic, and slowing-down states of coupled heteroclinic cycles. *Phys. Rev. E* **2012**, *85*, 016215. [CrossRef] [PubMed]
53. Stone, E.; Holmes, P. Random Perturbations of Heteroclinic Attractors. *SIAM J. Appl. Math.* **1990**, *50*, 726–743. [CrossRef]
54. Jeong, V.; Postlethwaite, C. Effect of noise on residence times of a heteroclinic cycle. *Dyn. Syst.* **2023**, *38*, 79–101. [CrossRef]
55. Ashwin, P.; Postlethwaite, C. Quantifying Noisy Attractors: From Heteroclinic to Excitable Networks. *SIAM J. Appl. Dyn. Syst.* **2016**, *15*, 1989–2016. [CrossRef]