Visualization of Stable Heteroclinic Channel-based Movement Primitives

Natasha A. Rouse, *Student Member, IEEE,* and Kathryn A. Daltorio, *Member, IEEE,*

***Abstract*—Movement primitives are a well-established approach to robot motion planning, as they offer a modular basis for robot motion planning. Dynamic movement primitives (DMPs) are a popular control framework based on nonlinear differential equations which, when scaled in time, produce a smooth kinematic movement plan. In this paper, we introduce an adjusted movement primitive framework which replaces the stable equilibria of DMPs with saddle points. These saddle points are linked together in stable heteroclinic channels (SHCs) which are in turn part of dynamical systems called stable heteroclinic networks. SHC-based movement primitives (SMPs) form a framework where the weights of the kernel functions have spatial significance in the task space. In this paper, we show that SMPs and DMPs perform comparably according to a kinematic-based cost function. We also show that SMP kernel weights follow a given trajectory when plotted in the task space. The plotted kernel weights provide an intuitive tool for managing the controller.**

***Index Terms*—Biologically-Inspired Robots, Motion Control**

I. Introduction

DYNAMIC movement primitives (DMPs) are a common robotic control framework [1], [2] which uses modular sub-components—a series of attractor points [3]–[5]—to produce arbitrarily complex motions. The relative strength, timing and growth/decay rates of the attractors are varied to create these custom trajectories [5]. A limitation of DMPs is that the best attractor locations (in both state space and the task space) are increasingly far from the corresponding trajectory which makes it difficult for users to initialize or evaluate a robotic controller using this representation.

DMPs were originally developed based on the biological insight that complex animal behaviors come from the dynamics of neural subsystems. In animals, motor control can be decomposed into motor modules (or muscle synergies) which reflect how neuron and muscle groups combine to accomplish different tasks [6]. DMPs have already been used to mimic humanoid movements [3], animal-inspired movements [7], and have been expanded to initiate repeated motion patterns [1], [4]. Keeping a biologically plausible representation allows insight to be transferred between biology and robotics.

Manuscript received: October 15, 2020; revised January 12, 2021; accepted February 3, 2020.

This paper was recommended for publication by Editor Xinyu Liu upon evaluation of the Associate Editor and Reviewer's comments. This work was supported by NSF Grant #1850168.

N. Rouse & K. Daltorio are with the Department of Mechanical and Aerospace Engineering, Case Western Reserve University, Cleveland OH, 44106 USA. Emails: nar50@case.edu; kathryn.daltorio@case.edu

Digital Object Identifier (DOI): see top of this page

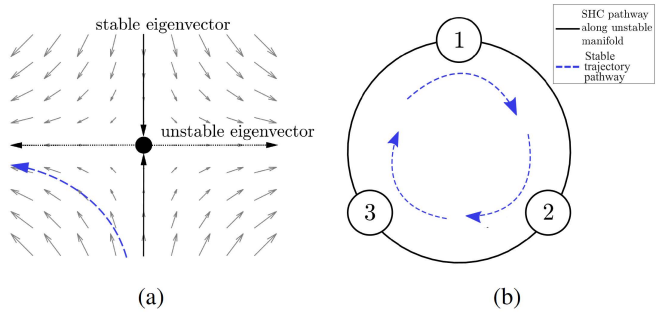


Fig. 1. A saddle equilibrium point (a) can be described with stable and unstable eigenvectors, shown acting along the y- and x-axis respectively, and trajectories (in blue) can be established that approach and depart the neighborhood of the point. If multiple saddle points are combined in this way (b), a stable heteroclinic channel is formed [15].

In recent years, new biological representations have been developed to capture neural dynamics, including attractor-based representations (like DMPs) as well as transient dynamic systems like neural networks [8]–[11] and stable heteroclinic networks [12]. Stable heteroclinic networks are composed of stable heteroclinic channels (SHCs): a series of saddle equilibria where the unstable manifold of one leads onto the stable manifold of another, thus creating pathways between the saddle points (Fig. 1) [13]–[15]. SHCs have already been investigated as a model for neural activation patterns in animals [16]–[18], and have also been used to investigate dynamical state systems [19], [20] as well as apply those systems to robotic movement [14].

In this paper, we will replace the underlying kernels of DMPs, stable attractor points or limit cycles, with the underlying kernels of SHCs, saddle points, and show that they are comparable in both ease of application and similarity of results. This is done by implementing SHC and DMP frameworks for the same tasks (trajectory following) and evaluating the resulting trajectories quantitatively and visually (see Section II).

First, we will demonstrate that this new kernel preserves the learnability of DMPs. Learnability is an essential characteristic of DMPs that lends to their robustness as a framework. DMPs are easily learnable because DMP parameters can be varied without disrupting the overall system stability, therefore various learning algorithms can be applied [21], [22]. In Section II, a batch learning method will be implemented on both frameworks to find the best-fit trajectory according to a user-defined cost function.

Furthermore, we will show that the new representation using

SHC kernels has a specific spatial advantage over DMPs. When learning algorithms are applied to DMPs, they usually result in blackbox optimized systems where users cannot easily evaluate the robotic controller. The SHC representation can be visualized in the task space in a way that follows the trajectory it is producing; this characteristic will be explored in Section III.

II. Methods

The goal of this work is to build a movement primitive framework using stable heteroclinic networks [12]. The saddle points in SHCs can be interpreted as states with pathways joining them. These building blocks are analogous to the attractor points or limit cycles of DMPs. The MATLAB code for this formulation can be found at `https://github.com/NatRouse/DMPandSMP.git`.

A. System Models

For both a dynamic movement primitive (DMP) and an SHC-based movement primitive (SMP), the governing equation of the canonical system is defined as:

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

Where τ is a time-scaling term, y is the relevant system variable (for example, end-effector position or joint angle), α_y and β_y are constants representing the damping and stiffness of the system respectively, g is the "goal" position of the system, and f is the external force being applied to the system by the controller [1]. The difference between the DMP and SMP is f's formulation.

1) DMP System Model: The forcing term for a DMP is defined as follows:

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

Where K is the total number of underlying kernel functions used, ψ_i is the shape each kernel function, w_i is the weight of each kernel, and x is the canonical state of the system. x is given as the solution to the following differential equation:

$$\tau\dot{x} = \alpha_x x \tag{3}$$

Where τ is a time-scaling factor, and α_x is a damping term. The kernel function, ψ_i, is a Gaussian curve defined as:

$$\psi_i = exp\left(\frac{-1}{2\sigma_i^2}(x - c_i)^2\right) \tag{4}$$

Where σ_i determines the width of the ith kernel function, and c_i is the "center" of the kernel function. Together, σ and c define the sequential activation of each kernel function as the canonical system decays to zero [1]. By using a learning algorithm, the weights, w_i are manipulated to produce a forcing term which will cause the system to follow the desired trajectory.

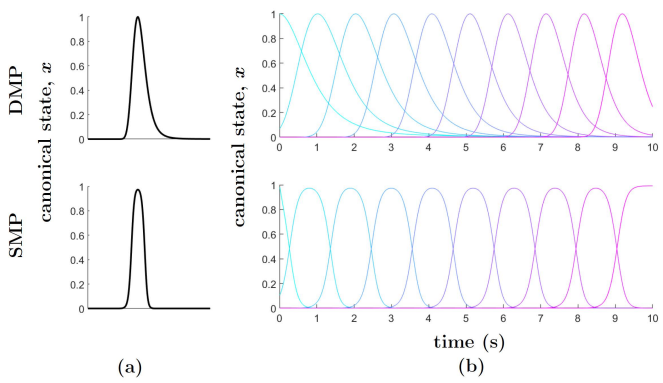


Fig. 2. (a) A single unweighted kernel for a DMP (top) plotted from a Guassian formulation. A single unweighted kernel for an SMP (bottom) plotted from a series of Lotka-Volterra equations. (b) Ten unweighted kernels for both DMPs and SMPs. Lotka-Volterra kernels have a more symmetrical parabolic shape, and Gaussian kernels are less symmetrical with an exponential decay.

2) SMP System Model: In contrast, the governing equations for an SMP are:

$$f(x_i) = \sum_{i=1}^{K} x_i w_i \tag{5}$$

$$\tau dx_i = x_i \left(\alpha_i - \sum_{j=1}^{K} \rho_{ij} x_j\right) dt + \sum_{j=1}^{N} C_{ij} z_j \tag{6}$$

Where:

- x_i is the underlying state vector,
- K is the number of kernel functions being used,
- w_i is the weight of each corresponding kernel function,
- N is the number of sensors,
- α_i and ρ_{ij} are parameters that define the overall behavior of the system,
- C_{ij} is a coupling matrix, and
- z_j is noise.

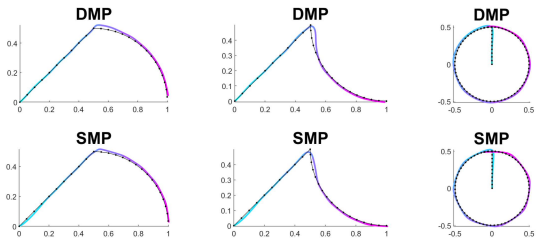
as described by Horchler et al [15]. They are discussed in more depth in Section II-B below.

The canonical state expression in (6) is based on N-dimensional competitive Lotka-Volterra (LV) equations [23]. Fig. 2 shows the visual difference between the waveforms constructed from a Gaussian and LV kernel function. Unlike Gaussian kernels, noise is a critical factor in the use of LV kernels. The noise z_j ensures that the system variable x stays near the SHC saddle point, but not so close that the system remains in static equilibrium [15].

B. Setting System Parameters

The variables in a DMP system that control system behavior are the width, location and weight (magnitude) of the kernel functions (σ, c and w respectively).

The variables that control SMP behavior are the parameters that make up the connection matrix ρ_{ij}, and the noise z_j. The connection matrix is a real, non-symmetric matrix constructed



(a) Convex Trajectory (b) Concave Trajectory (c) Circular Trajectory

Fig. 3. For both SMPs (bottom) and DMPs (top), the resulting trajectories (in color) align well with the desired trajectories (in black) for 10 kernels.

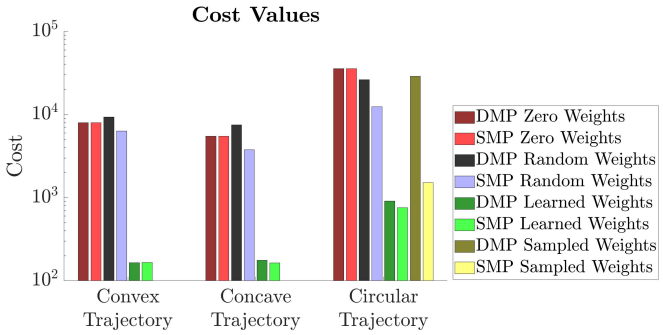


Fig. 4. The alignment of the trajectories is quantified with a cost function (8). The resulting costs are compared according to different cases: Zero Weights—all kernel weights are zero, Random Weights—weights are assigned from a normal distribution with mean=5 and variance=10, Learned Weights—the batch learned weights, and Sampled Weights—weights sampled and scaled from the trajectory. SMP & DMP Learned Weights costs are comparable with each other, and the SMP Sampled Weights cost is comparable with both Learned Weights costs.

with three variables: the growth rate α, the magnitude β, and the saddle value ν. The matrix is constructed as follows:

$$\rho_{ij} = \begin{cases} \alpha_i/\beta_i, & \textit{if i=j} \\ \dfrac{\alpha_i - \alpha_j/\nu_j}{\beta_j}, & \textit{if i=j-1} \\ \dfrac{\alpha_i + \alpha_j}{\beta_j}, & \textit{otherwise} \quad \textit{i,j=1,2,...,K} \end{cases} \tag{7}$$

Where α, β and ν are vectors of length K. Mathematically, α_i controls how fast the *ith* kernel grows in that dimension, β_i is the maximum amplitude of the waveform, and ν_i defines that kernel's insensitivity to input noise. These variables (and the noise) must be varied synchronously to create and maintain functional kernels [15].

The other notable variable in both DMP and SMP systems is the number of underlying kernel functions, *K*, used to perform a task. The complexity of a task determines the required number of kernel functions; in this paper, $K = 10$ so that each kernel and associated trajectory regions can be can be visualized separately.

C. Learning Kernel Weights for Desired Trajectory Shapes

In order to illustrate the usefulness of SMP kernels, three trajectory-following tasks are used. The first two are straight paths connected to convex and concave curves, creating a critical corner in the trajectory, the third trajectory is a circular path (Fig. 3). To produce these trajectories, a batch learning method is applied and the resulting trajectories are evaluated via cost function.

1) Cost Function: The cost function used to evaluate the batch learning method is:

$$\begin{aligned} \text{cost} = 100 * \sum^{DOF} \sqrt{\sum^{t} (y_{\text{learned}} - y_{\text{desired}})^2} \\ + 1000 * \sum^{DOF} \sqrt{(y_{\text{learned (final)}} - g)^2} \\ + \sum^{t} \dot{y} \quad + \sum^{t} \ddot{y} \end{aligned} \tag{8}$$

Where t is the total number of timesteps. Cost values are evaluated for the unforced system (w_i = 0 for all $i = 1 : K$), random weights ($w_i = \mathcal{N}(5, 10)$, an array of normally distributed random variables with a mean of 5, and a variance of 10), the final learned weights, and weights sampled and scaled from the trajectory (which is discussed in Section III-D).

2) Learning Algorithm: There are many approaches to learning DMP weights online, such as locally weighted regression [1], or PI^2 [24]. Here we implement a simple two pass, batch learning method which was faster than and had comparable costs to the two previously mentioned methods. First the canonical state *x* was initialized according to guidelines in [1], [15] & [25]. A desired speed, acceleration and forcing function were calculated at coarse timesteps based on the desired trajectory. These variables were used in vectorized Matlab operations to produce an initial approximation of the weights. This approximation enabled subsequent refinement with finer timesteps.

The batch learned weights are assessed via the cost function in Section II-C1 above. These results are discussed in Section III-B.

III. Results

A. Costs

Learning algorithms can be transferred from DMPs to SMPs. The cost for both processes was evaluated using the batch learning process and cost function described in Section II-C. Fig. 4 shows the costs calculated when all kernel weights are at zero (Zero Weights), randomly assigned weights (Random Weights), and the batch learned weights (Learned Weights). Fig. 4 shows that the cost associated with the batch learned SMP kernel weights is slightly smaller than that of the DMP kernels.

B. Trajectory Following

After learning, the resulting trajectories for both DMPs and SMPs follow the desired trajectory, as shown in Fig. 3. This is because the total forcing function from each representation is similar, despite the difference in formulation ((2) & (5)) (see Fig. 5). The difference comes from the kernel summation. The SMP forcing function can be reconstructed by smoothly

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2021.3061382, IEEE Robotics and Automation Letters

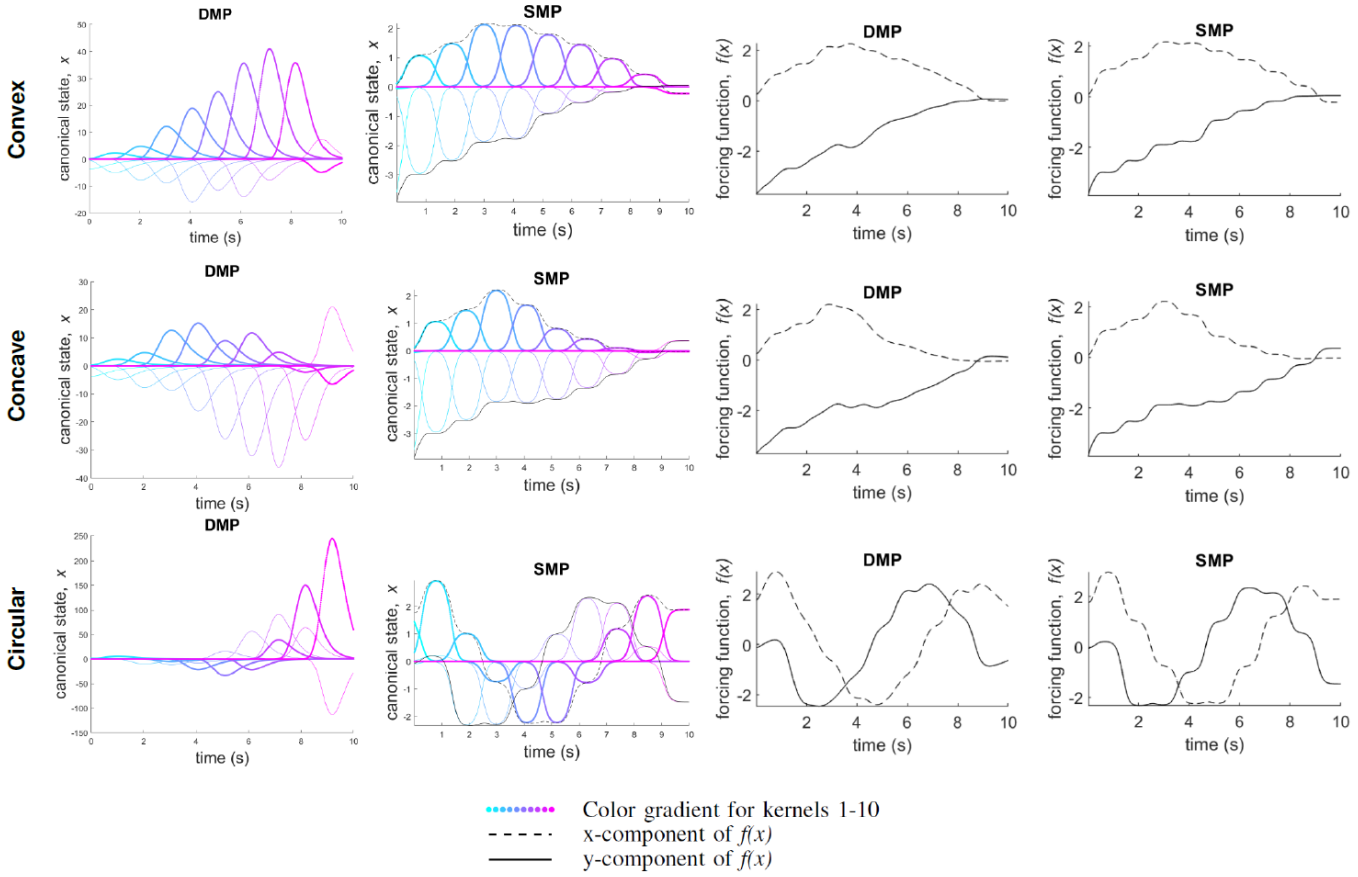


Fig. 5. In the first two columns, the individual kernels (from Fig. 2) are scaled by the learned weights for (a) Convex, (b) Concave and (c) Circular trajectories in state space. In both formulations, the total forcing function (black) is the sum of the weighted kernels (colors). For SMPs, the kernel peaks can be traced to reproduce the corresponding forcing function. In the second two columns, it can be seen that the resulting forcing functions for SMP and DMP are similar, in order to achieve the same desired trajectory.

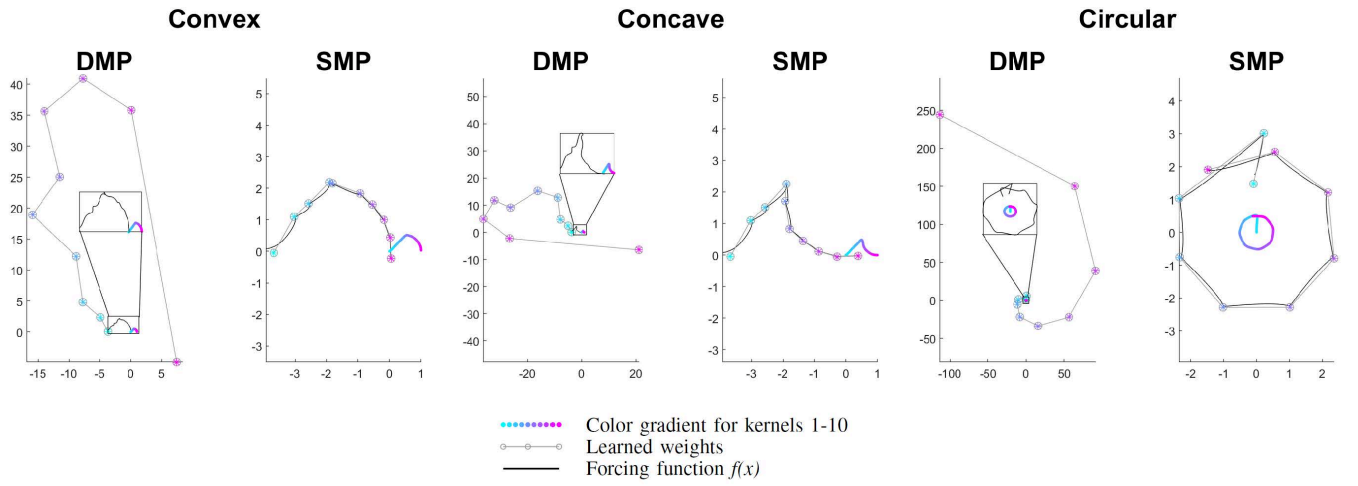


Fig. 6. In Cartesian space, the same forcing function (in black) and trajectory is produced (in color, and see insets for DMPs), but the learned weights (in gray) are much larger for DMPs than for SMPs. In fact, the SMP weights for the x- and y-components of the forcing function can be plotted on the x- and y-axis and correspond to the trajectory. The circular trajectory matches the forcing function, and from inspection SMP weights can be scaled from the trajectory itself. The convex and concave SMP weights can also be scaled from the trajectory with additional translation.

tracing along the peaks of the weighted SMP kernel functions, while the weighted DMP kernels grow over time along the path, with peaks greatly exceeding the required forcing function.

C. Spatial Significance of Kernel Weights

Because SMP kernel weights do not grow exponentially over time (unlike DMP kernels), they follow their respective forcing function more closely. This can be seen when the weights are plotted in state space (Fig. 5) and in the task space (Fig. 6). The spatial significance of SMP kernel weights is clear; in every chosen trajectory, they match the forcing function when it is plotted in the task space. Thus, SMP weights describe the position of the saddle points in a way that is visually comparable with the trajectory. In contrast, DMP kernel positions are not only farther from the trajectory but the shape of the connected path of DMP attractors is distorted compared to the forcing function and desired trajectory.

D. Sampling Weights from Trajectory

When the goal is set to (0,0), as in the circular trajectory, it is possible to approximate kernel weights directly (without learning or optimization) using the desired trajectory path and velocity if the system parameters are known.

We selected weights in this way using:

$$w = \left(\frac{\sqrt{(\alpha_y v)^2 + \left(\frac{\alpha_y \beta_y}{2} - \tau \left(\frac{v^2}{r} \right) \right)^2}}{r} \right) * y_{\text{desired}} \qquad (9)$$

Where v is a desired velocity and r is the radius of curvature of the trajectory. Here, we demonstrate this approach using uniform velocity and averaged curvature.

Fig. 7 shows the sampled weights (red), the forcing function (black), the best weights (gray), and the trajectory produced from these weights (color gradient). When kernel weights for the circular trajectory are chosen using this expression it produces an initial SMP cost that is approximately 12.5% of the cost of the random SMP weights calculated in Fig. 4. This type of scaling is not compatible with DMPs; scaled weights do not have a reduced cost.

Another, more complex, enclosed trajectory was used to validate (9). The heart-shaped trajectory in Fig. 7 was produced, the same scale was applied, and the results are similar to that of the circular trajectory. With best weights selected from the scaled trajectory, the inital SMP cost value was 1.6% of that for DMPs.

IV. Discussion and Conclusion

In this paper, we have shown that DMP kernels can be replaced with another, biologically-relevant kernel based on SHCs. Both DMPs and SMPs were used to produce various reference trajectories (Fig. 3), and a batch learning process was successfully applied to both representations with comparable resulting costs (Fig. 4).

After learning, the best SMP kernel weights were seen to follow the desired trajectory when plotted in the task space,

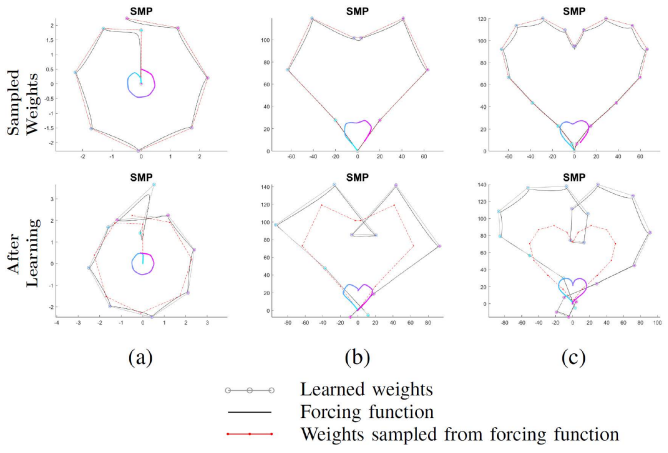


Fig. 7. (a) Circular trajectory produced using 10 kernels. (b) Heart-shaped trajectory produced using 10 kernels. (c) Heart-shaped trajectory produced using 20 kernels. The top row shows trajectories produced by approximating the weights using (9). The bottom row shows trajectories and weights after batch learning & optimization. The sampling approach reduces cost by an order of magnitude. The SMP cost of (b) top is 1.6% of the DMP cost using random weights; a similar difference can be seen for (a) in Fig. 4. This was run using SMPs and resembles the more complex trajectory-following tasks completed by Ijspeert et al in [5]. The weight locations after learning are similar to the sampled weights, but show some adjusted spacing at sharp turns in the trajectory. More kernels are placed at these sharp points to achieve better trajectory-following.

unlike DMPs (Fig. 6). We investigated this result in two ways. First, the underlying kernels and the resulting forcing function were plotted in state space (Fig. 5). The SMP forcing function directly followed its kernels, which suggests that SMP kernels are more closely coupled to the system's behavior than DMP kernels. Second, a process was established for sampling and scaling kernel weights directly from the desired trajectories (9). This was applied to additional trajectories with satisfactory results (Fig. 7).

The close coupling of SMP kernels to their resulting trajectories, and the ease with which they can be sampled could provide a tool for the selection and analysis of SMP kernel weights. This feature may provide a simplified means for moving kernels spatially, thus decreasing the computational cost for learning the best kernel weights for a task (or possibly eliminating that process entirely).

One potential drawback of changing the kernel is in the computational complexity. The DMP framework is $\mathcal{O}(kt)$ where k is the number of kernels and t is the number of timesteps calculated. The SMP framework is $\mathcal{O}(k^2 t)$. However, because the connection matrix (7) is diagonal, only adjacent saddles have significant influence. This would allow for future implementation with $\mathcal{O}(kt)$.

The future advantages of this framework could span beyond visualization. In particular, while DMP attractors each encode incoming trajectory behavior, saddle equilibria encode both incoming and outgoing trajectory behavior. Thus, SMP system dynamics can be more specifically defined spatially, which could enable more complex network topology, such as state-dependent responses to feedback that direct trajectories into separate branches and cycles.

APPENDIX

The variables used in this paper that were common to both frameworks are as follows:

$$\tau = 1, \quad \alpha_y = 4, \quad \beta_y = 1$$

The DMP variables used in this paper are:

$$\alpha_x = 0.5,$$

And both σ and c are logarithmically spaced vectors with the same length as the number of kernel functions.

```
σ = logspace(log10(.3),log10(.002),K)
c = logspace(log10(1),log10(.01),K)
```

The SMP variables used are:

$$\alpha = 10, \quad \beta = 1, \quad \nu = 1.2, \quad \varepsilon = 10^{-9}$$

ACKNOWLEDGMENTS

This project was funded by NSF Grant #1850168. The authors would like to thank Naomi Hourihane for assistance in developing the manuscript.

REFERENCES

[1] S. Schaal, *Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics*. Tokyo: Springer Tokyo, 2006, pp. 261–280. [Online]. Available: https://doi.org/10.1007/4-431-31381-8_23

[2] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," *Springer Tracts in Advanced Robotics*, vol. 15, pp. 561–572, 2005. [Online]. Available: https://link.springer.com/chapter/10.1007/11008941_60

[3] S. Schaal, S. Kotosaka, and D. Sternad, "Nonlinear dynamical systems as movement primitives," *International Conference on Humanoid Robotics Cambridge MA*, vol. 38, no. 2, pp. 117–124, 2001. [Online]. Available: http://www-slab.usc.edu/sschaalhttp://www.erato.atr.co.jp/{∼} kotosakahttp://www.psu.edu/faculty/d/x/dxs48http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.73.9338{&}rep=rep1{&}type=pdf

[4] S. Schaal, J. Peters, and J. Nakanishi, "Control, planning, learning, and imitation with dynamic movement primitives," *Neuroscience*, pp. 1–21, 2003. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Control+,+Planning+,+Learning+,+and+Imitation+with+Dynamic+Movement+Primitives#1

[5] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives : Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, pp. 328–373, 2013. [Online]. Available: http://www-clmc.usc.edu/Resources/Software.http://www.mitpressjournals.org/doi/abs/10.1162/NECO_a_00393

[6] L. H. Ting and H. J. Chiel, *Muscle, Biomechanics, and Implications for Neural Control*. John Wiley & Sons, Ltd, 2017, ch. 12, pp. 365–416. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118873397.ch12

[7] P. M. Wensing and J.-J. Slotine, "Sparse control for dynamic movement primitives," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 10 114 – 10 121, 2017, 20th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896317324102

[8] W. Maass and H. Markram, "On the computational power of circuits of spiking neurons," *Journal of Computer and System Sciences*, vol. 69, no. 4, pp. 593 – 616, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000004000406

[9] R. de Azambuja, F. B. Klein, S. V. Adams, M. F. Stoelen, and A. Cangelosi, "Short-term plasticity in a liquid state machine biomimetic robot arm controller," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 3399–3408.

[10] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," German National Research Center for Information Techology, Tech. Rep., jan 2001. [Online]. Available: https://www.researchgate.net/publication/215385037

[11] J. Kuwabara, K. Nakajima, R. Kang, D. T. Branson, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, "Timing-based control via echo state network for soft robotic arm," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–8.

[12] K. A. Daltorio, A. S. Boxerbaum, A. D. Horchler, K. M. Shaw, H. J. Chiel, and R. D. Quinn, "Efficient worm-like locomotion: slip and control of soft-bodied peristaltic robots," *Bioinspiration & Biomimetics*, vol. 8, no. 3, p. 035003, Aug 2013.

[13] M. Rabinovich, R. Huerta, and G. Laurent, "Neuroscience: Transient dynamics for neural processing," pp. 48–50, Jul 2008. [Online]. Available: http://science.sciencemag.org/content/suppl/2008/07/03/321.5885.48.DC1

[14] K. A. Daltorio, A. D. Horchler, K. M. Shaw, H. J. Chiel, and R. D. Quinn, "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)*, N. Lepora, A. Mura, H. Krapp, P. Verschure, and T. Prescott, Eds., vol. 8064. Springer, Berlin, Heidelberg, 2013, pp. 59–70. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-39802-5_6

[15] A. D. Horchler, K. A. Daltorio, H. J. Chiel, and R. D. Quinn, "Designing responsive pattern generators: stable heteroclinic channel cycles for modeling and control," *Bioinspiration & Biomimetics*, vol. 10, no. 2, p. 026001, Feb 2015.

[16] G. Laurent, M. Stopfer, R. W. Friedrich, M. I. Rabinovich, A. Volkovskii, and H. D. Abarbanel, "Odor encoding as an active, dynamical process: Experiments, computation, and theory," *Annual Review of Neuroscience*, vol. 24, no. 1, pp. 263–297, 2001, pMID: 11283312. [Online]. Available: https://doi.org/10.1146/annurev.neuro.24.1.263

[17] M. Rabinovich, A. Volkovskii, P. Lecanda, R. Huerta, H. D. I. Abarbanel, and G. Laurent, "Dynamical encoding by networks of competing neuron groups: Winnerless competition," *Phys. Rev. Lett.*, vol. 87, p. 068102, Jul 2001. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.87.068102

[18] P. Varona, M. I. Rabinovich, A. I. Selverston, and Y. I. Arshavsky, "Winnerless competition between sensory neurons generates chaos: A possible mechanism for molluscan hunting behavior," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 12, no. 3, pp. 672–677, 2002. [Online]. Available: https://doi.org/10.1063/1.1498155

[19] M. Voit and H. Meyer-Ortmanns, "Dynamical inference of simple heteroclinic networks," *Frontiers in Applied Mathematics and Statistics*, vol. 5, p. 63, Dec 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fams.2019.00063/full

[20] K. M. Shaw, Y. M. Park, H. J. Chiel, and P. J. Thomas, "Phase resetting in an asymptotically phaseless system: On the phase response of limit cycles verging on a heteroclinic orbit," *SIAM Journal on Applied Dynamical Systems*, vol. 11, no. 1, pp. 350–391, Mar 2012.

[21] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Dec 2011, pp. 365–371.

[22] Y. Yuan, Z. Li, T. Zhao, and D. Gan, "Dmp-based motion generation for a walking exoskeleton robot using reinforcement learning," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 5, pp. 3830–3839, May 2020.

[23] C. Bick and M. I. Rabinovich, "On the occurrence of stable heteroclinic channels in lotka–volterra models," *Dynamical Systems*, vol. 25, pp. 110 – 97, 2010.

[24] E. Theodorou, J. Buchli, and S. Schaal, "Learning policy improvements with path integrals," *Journal of Machine Learning Research*, vol. 9, pp. 828–835, Jan 2010. [Online]. Available: http://www-clmc.usc.edu

[25] A. D. Horchler, "SHCTools: Matlab toolbox for simulation, analysis, and design of stable heteroclinic channel networks," Oct 2014. [Online]. Available: http://github.com/horchler/SHCTools