Reactive Motion Generation via Phase-varying Neural Potential Functions

Ahmet Tekden[1] Dimitrios Kanoulas[2] Aude Billard[3] Yasemin Bekiroglu[1,2]

***Abstract*— Dynamical systems (DS) methods for Learning-from-Demonstration (LfD) enable stable, continuous policies from few demonstrations. While first-order DS work well when each state maps to a unique velocity, they struggle with tasks involving intersections or repeated state visits. Existing solutions either rely on second-order dynamics, which are sensitive to velocity near intersections, or on open-loop phase variables, which reduce robustness to perturbations. We propose Phase-varying Neural Potential Functions (PNPF), an LfD framework that conditions a potential function on a closed-loop phase inferred from state progression. This enables reactive control while disambiguating repeated states without relying on velocity or time. The learned potential generates local vector fields for stable and adaptive motion. PNPF generalizes across point-to-point, periodic, and 6D tasks, outperforms prior methods on trajectories with intersections, and demonstrates robust real-time performance under disturbances.**

I. Introduction

Learning-from-Demonstration (LfD) enables robots to acquire task policies directly from demonstrations. A central challenge is generating motions that both follow demonstrations and remain reactive to perturbations. Dynamical systems (DS)-based approaches are attractive because they produce stable, continuous policies from few demonstrations [1]–[4]. However, they face a trade-off between modeling capability and reactivity. Time- or phase-dependent methods can represent tasks where the same state is revisited at different stages, but rely on open-loop progression and recover poorly after perturbations [1], [2]. In contrast, state-based DS methods react directly to the current state [3], [5]–[9], but require velocity or higher-order state information to handle intersections, making them sensitive to disturbances and ambiguous when similar position-velocity pairs should lead to different future motions (Fig. 1).

We address this problem with *Phase-varying Neural Potential Functions* (PNPF), a reactive motion-generation framework based on potential functions conditioned on a closed-loop phase variable. The phase is estimated directly from state progression rather than from open-loop time, allowing the system to revisit similar states while remaining reactive to perturbations. PNPF combines two energy functions: (i) a nominal energy, which encodes the desired task behavior and decreases monotonically with task progress; and (ii) a safety energy, which discourages leaving the boundary of demonstrated data, with this boundary estimated in a data-driven manner from the visited states. The nominal energy provides a scalar measure of task progress, enabling closed-loop estimation of phase directly from the state and thereby resolving the ambiguity of repeated state visits without relying on velocity. The safety energy attracts the system back toward the region supported by demonstrations, encouraging recovery when perturbed while preserving compliant behavior within that region. Together, these two terms yield a motion-generation framework that remains reactive while handling point-to-point, periodic, and full 6D tasks.

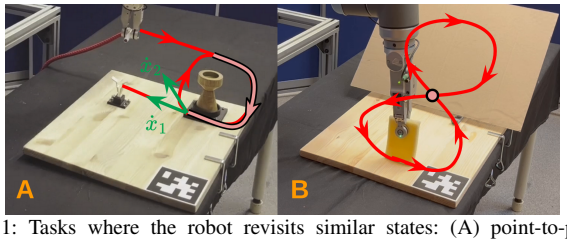

Fig. 1: Tasks where the robot revisits similar states: (A) point-to-point knotting, where identical motion segments branch into different actions $\dot{x}_1, \dot{x}_2$ from the same position–velocity state, and (B) 8-shaped periodic wiping. Our method ensures reactive motion generation in both cases.

Our approach is most closely related to state-based DS methods, movement primitives, and learned potential-function formulations [8], [10], [11]. In contrast to second-order DS methods, PNPF does not rely on velocity to disambiguate repeated states; in contrast to time-dependent motion primitives, it maintains reactivity through closed-loop phase estimation; and in contrast to prior learned potential formulations, it explicitly handles repeated state visits through phase conditioning.

The main contributions are: (i) a reactive motion-generation framework based on phase-varying neural potential functions for point-to-point and periodic tasks; (ii) a closed-loop phase variable that resolves repeated-state ambiguity while preserving reactivity; and (iii) compatibility with configuration-space, task-space, and full 6D motions. We evaluate the method on 2D handwriting and 6D real-robot tasks and show improved robustness and performance over prior reactive motion-generation baselines.

II. Motion Generation with Potential Functions

We represent motion as the gradient flow of a scalar-valued potential function over the state space. Intuitively, the potential defines an energy landscape in which desired task behavior corresponds to descending toward lower-energy regions. This perspective is appealing for reactive motion generation because it directly yields a dynamical system:

This work was supported by Chalmers AI Research Center (CHAIR) and Chalmers Gender Initiative for Excellence (Genie), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. [1]Chalmers University of Technology, Sweden. [2]University College London, UK. [3]Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland. Email: tekden@chalmers.se

[1]This work is based on an accepted paper in IEEE RAL (in press). Supplementary Video: https://youtu.be/9d-7r-kdCsE

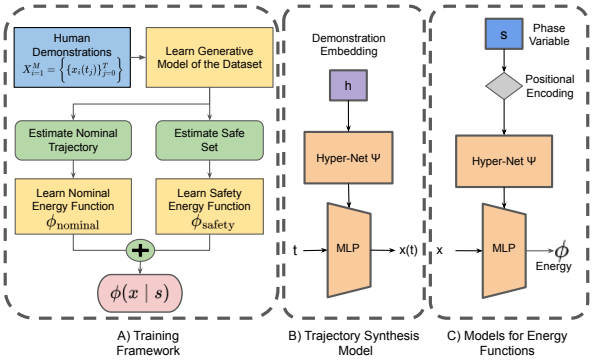

Fig. 2: Overview of PNPF. (A) Construction of the phase-varying potential $\phi(x \mid s)$. (B,C) Network architectures for trajectory synthesis and energy estimation.

rather than predicting trajectories in open loop, the robot continuously updates its motion by following the local energy gradient. Formally, given state $x \in \mathbb{R}^d$ and phase variable s, the policy is

$$\dot{x} = -\nabla_x \phi(x \mid s), \tag{1}$$

where $\phi(x \mid s)$ is the potential function. Conditioning on s allows the same state to correspond to different desired motions at different stages of the task, while preserving the closed-loop nature of the controller. Figure 2 summarizes the proposed architecture, while Figures 3 illustrate the resulting energy landscape and its phase-dependent behavior.

Our key idea is to construct $\phi(x \mid s)$ from two complementary energy functions. The *nominal energy* captures task progression by assigning lower values to states that lie further along the demonstrated motion. The *safety energy* captures whether a state lies within the region supported by demonstrations, and becomes active when the robot moves outside this region. Together, these energies provide an intuitive decomposition of behavior: the nominal term determines *how the task should progress*, while the safety term determines *how the system should recover* when perturbed.

Nominal and Safety Energy Functions: Given M demonstrations $X_{i=1}^{M} = \{x_i(t_0), \ldots, x_i(t_T)\}$, we first learn a continuous trajectory synthesis model that generates trajectories similar to the demonstrations while interpolating between them. These synthesized trajectories serve two purposes. First, they are used to estimate a *nominal trajectory* $x^*(t)$ by selecting the generated trajectory with the lowest total DTW distance to the demonstrations. Second, they define a demonstration-supported region $\mathcal{C}$ by densely sampling the interpolation region of the demonstrations. We represent this region using a signed distance function $\mathrm{SDF}(x)$, where $\mathrm{SDF}(x) \leq 0$ denotes states inside the support region and $\mathrm{SDF}(x) > 0$ denotes states outside.

The safety energy is defined directly from the signed distance field:

$$\phi_{\text{safety}}(x) = \mathrm{relu}(\lambda_{\text{safety}}(\mathrm{SDF}(x))), \tag{2}$$

where λ_{safety} smooths the transition near the boundary. Thus, $\phi_{\text{safety}}(x) = 0$ inside the demonstrated support region and increases outside, causing its gradient to pull the system back toward demonstrated states.

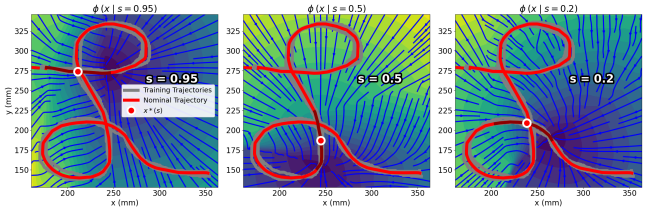

Fig. 3: Phase-varying potential functions. The same spatial state can correspond to different local descent directions at different phases of the task. This allows the system to revisit similar states while maintaining reactive, closed-loop motion generation.

The nominal energy is defined from the remaining arc length of the nominal trajectory. For the discretized nominal trajectory $\{x_j^*\}_{j=0}^{N}$,

$$\phi_{\text{nominal}}(x_k^*) = \sum_{j=k}^{N-1} \|x_{j+1}^* - x_j^*\|. \tag{3}$$

This scalar decreases monotonically along the trajectory and therefore provides a natural notion of task progress. For off-trajectory states, the energy is approximated using the nearest point on x^*.

Phase-Varying Potential Functions: A static potential cannot model tasks that revisit the same state at different times, since a single state cannot simultaneously have two different energy values. To resolve this, we make the potential phase-varying.

Because ϕ_{nominal} decreases along the nominal trajectory, it naturally defines a progress variable:

$$s_k = \phi_{\text{nominal}}(x_k^*). \tag{4}$$

We then construct local nominal and safety energies around the current phase, yielding the phase-varying potential

$$\phi(x \mid s) = \phi_{\text{nominal}}(x \mid s) + k_{\text{safety}}\, \Lambda_{\text{safety}}(s)\, \phi_{\text{safety}}(x \mid s), \tag{5}$$

where $\Lambda_{\text{safety}}(s)$ optionally increases the influence of the safety term later in the task.

The control policy and phase update are

$$\dot{x} = -\alpha \nabla_x \phi(x \mid s), \qquad s \leftarrow \phi_{\text{nominal}}(x' \mid s), \tag{6}$$

where $\alpha > 0$ is a control gain and x' is the next state. Since phase is updated from the current state rather than from time, the system remains reactive under perturbations while still tracking task progression.

The framework can also handle periodic and 6D motion: periodic motions are handled by replacing the scalar phase with its sinusoidal representation, and 6D motions are modeled by expressing orientations in axis-angle form in the tangent space of a reference quaternion.

Neural Field Parameterization: We model $\phi_{\text{nominal}}(x \mid s)$ and $\phi_{\text{safety}}(x \mid s)$ using neural fields, which provide smooth and differentiable energy landscapes. To preserve smooth gradients with respect to x, we use a hypernetwork-conditioned MLP: the phase variable s is position-encoded and used for conditioning, while the state x is provided directly. This yields expressive phase dependence while maintaining smooth vector fields for real-time control.

TABLE I: Method performance across different datasets.

Dataset	Method	DTWD	FD	FP Error	Accuracy
LASA (cm)	PNPF	**1.81**	**3.38**	**0.02**	**1.00**
	CONDOR	2.10	4.14	0.10	**1.00**
	NODE	2.12	4.33	1.82	0.76
LAIR (mm)	PNPF	**4.82**	**11.75**	**5.04**	**1.00**
	CONDOR	7.62	17.43	7.91	0.96
CHAR (mm)	PNPF	**2.96**	**8.44**	**2.00**	**1.00**
	CONDOR	17.67	45.56	3.25	**1.00**
Char-Periodic (mm)	PNPF	**7.77**	**14.30**	NA	NA
	Naive	10.70	24.28	NA	NA

Dataset	Method	DTWD-P (cm)	DTWD-O (rad)	FP Error (cm)	F0 Error (rad)
RoboTasks	PNPF	**2.19**	**0.08**	**0.25**	0.17
	NODE	2.62	0.10	1.51	**0.13**

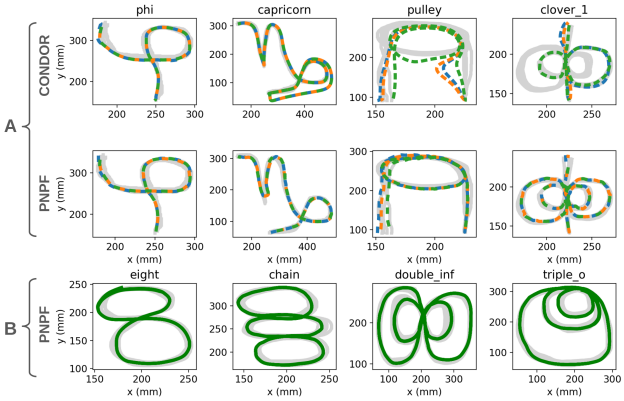

Fig. 4: Comparison with CONDOR. Our method reproduces trajectories more faithfully and maintains correct motion through intersections.

III. Simulation-Based Evaluations

We evaluate the proposed method on LASA (2D), LAIR (2D), and RoboTasks (6D), together with two additional 2D datasets, CHAR and CHAR-Periodic, which contain repeated state visits and intersecting motions that require different continuations. We compare against NODE [9] and CONDOR [8].

Across all datasets, our method achieves the best overall performance (Table I). The largest gains are observed on CHAR, where trajectories revisit similar states but require different onward actions. In this setting, our method reduces DTWD by approximately 80% compared to CONDOR, highlighting the benefit of phase-varying energy landscapes. Qualitative comparisons in Figure 4 show that our method maintains correct motion through intersections, whereas CONDOR often mismodels or skips key segments.

We additionally evaluate reactivity under obstacles and spatial disturbances. As shown in Figure 5, our method recovers smoothly and resumes the task without skipping or repeating motion segments, demonstrating the advantage of estimating phase directly from state progression.

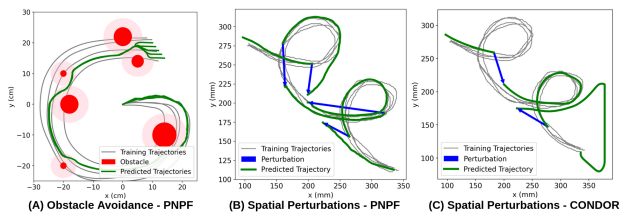

Fig. 5: PNPF remains reactive under obstacles and spatial disturbances, resuming motion without skipping task segments.

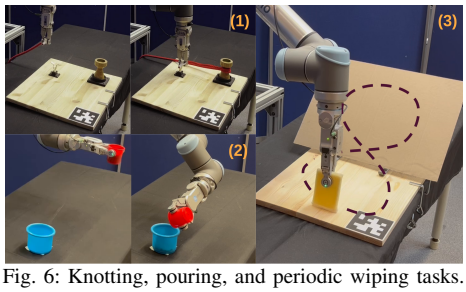

Fig. 6: Knotting, pouring, and periodic wiping tasks.

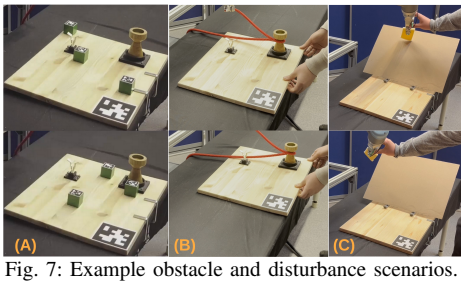

Fig. 7: Example obstacle and disturbance scenarios.

IV. Real-World Experiments

We evaluate the proposed method on three real-robot tasks: knotting, pouring, and periodic wiping (Fig. 6). Demonstrations are collected via kinesthetic teaching, and trajectories are time-aligned using DTW. We record six demonstrations for knotting and pouring, and two for periodic wiping. The first and third tasks are modeled in task space using positions and quaternions, while the pouring task is modeled in joint space. Across all tasks, the model generates successful motions from multiple initial conditions while operating online at up to 50 Hz. For each task, we evaluate 10 different initial configurations, and in the periodic wiping task, multiple motion cycles are generated by initializing the phase accordingly. In all cases, the reproduced motions remain consistent with the demonstrations and successfully complete the intended behavior.

We further evaluate reactivity under environmental and spatial perturbations (Fig. 7). In the knotting task, we place obstacles in the workspace and test two additional disturbance settings: shifting the task frame and manually displacing the robot during execution. We also apply manual perturbations in the periodic wiping task. Across all tested cases, the robot adapts its motion while preserving task consistency and successfully completes the task. Notably, after disturbances that undo partial progress, such as rope slippage during knotting, the system correctly resumes and completes the intended behavior.

V. Conclusion

We introduced Phase-varying Neural Potential Functions (PNPF), a reactive motion-generation framework that combines closed-loop phase estimation with energy-based dynamical systems. By constructing motion from nominal and safety energies, the method models tasks with repeated state visits while remaining robust to perturbations. Across simulated and real-robot evaluations, PNPF improves over reactive dynamical-system baselines and enables stable execution of point-to-point, periodic, and 6D motions.

REFERENCES

[1] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[2] 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, no. 2, pp. 328–373, 2013.

[3] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, 2011.

[4] A. Billard, S. Mirrazavi, and N. Figueroa, *Learning for adaptive and reactive robot control: a dynamical systems approach*. Mit Press, 2022.

[5] N. Figueroa and A. Billard, "A physically-consistent bayesian non-parametric mixture model for dynamical system learning," in *Conf. Robot. Learn. (CoRL)*, 2018, pp. 927–946.

[6] N. Figueroa, S. Faraji, M. Koptev, and A. Billard, "A dynamical system approach for adaptive grasping, navigation and co-manipulation with humanoid robots," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2020, pp. 7676–7682.

[7] F. Khadivar, I. Lauzana, and A. Billard, "Learning dynamical systems with bifurcations," *Robot. Auton. Syst.*, vol. 136, p. 103700, 2021.

[8] R. Pérez-Dattari and J. Kober, "Stable motion primitives via imitation and contrastive learning," *IEEE Trans. Robot.*, 2023.

[9] F. Nawaz, T. Li, N. Matni, and N. Figueroa, "Learning complex motion plans using neural odes with safety and stability guarantees," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024, pp. 17 216–17 222.

[10] S. M. Khansari-Zadeh and O. Khatib, "Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors," *Auton. Robots*, vol. 41, pp. 45–69, 2017.

[11] M. A. Rana *et al.*, "Learning reactive motion policies in multiple task spaces from human demonstrations," in *Conf. Robot. Learn. (CoRL)*, 2020, pp. 1457–1468.