Learning an autonomous dynamic system to encode complex periodic human motion skills

Jiayun Fu, Andong Liu, *Member, IEEE*, Wenan Zhang, *Senior Member, IEEE*, Weiyong Si, Haohui Huang, Yue Wang, *Senior Member, IEEE*, Chenguang Yang, *Fellow, IEEE*

***Abstract*—Autonomous dynamic systems (ADS)-based encoding of human motion skills has been widely demonstrated to exhibit high transfer efficiency in goal-directed tasks; however, significant gaps remain in the study of skill transfer for periodic motions, particularly complex periodic motions. In this letter, we propose a novel method for directly learning complex periodic movements using Lyapunov functions (LF), which is fundamentally different from all existing methods that learn periodic movements through limit cycle mapping. First, we introduce a polar coordinate transformation to decouple trajectory and angle modulation, enabling the radially increasing Lyapunov energy function to acquire non-radial expansion capabilities. Then, based on this, we learn a data-driven Lyapunov energy function by solving a dual objective optimization problem that combines geometric alignment and orbit parallelism constraints, aligning one of its horizontal surfaces with a periodic human demonstration trajectory. Finally, ADS is learned by sequentially solving LF-related constrained optimization problems. By designing appropriate constraint functions, we can ensure that the trajectories generated by ADS converge to an LF-level surface whose shape resembles that of periodic human demonstration trajectories. Both simulations and real robot experiments validate the effectiveness of the proposed method.**

***Index Terms*—Demonstration learning, complex periodic motion skills transfer, autonomous dynamic system, neural network**

I. Introduction

With the rapid development of robotics technology, the application scenarios for robots have expanded from structured environments to unstructured environments, such as offices, hospitals, and restaurants—complex settings where conditions are constantly changing. In these dynamic and unpredictable environments, traditional offline programming methods often fail to meet the operational requirements of robots, as they cannot anticipate all possible scenarios. Learning from Demonstration (LfD) [1] offers a promising alternative by enabling robots to acquire adaptive skills through imitation of human operations [2], [3].

This letter focuses on modeling complex periodic operational skills, which are essential in tasks such as medical-surgical robot operation and assembly. Such skills should exhibit inherent human motion characteristics—continuity, smoothness, and boundedness—properties naturally captured by stable dynamic systems (DS) due to their ability to generate bounded and continuous outputs [4].

Over the past decade, several DS-based methods for learning periodic motion have emerged, which can be categorized into two types. The first type learns non-autonomous DS, where motion is driven explicitly by time or phase. Representative methods include Dynamic Movement Primitives (DMP) [5], [6], which use rhythm-based functions to generate human-like periodic trajectories. Similarly, Probabilistic Movement Primitives (ProMP) [7] and Kernelized Movement Primitives (KMP) [8], [9] model periodic motion via rhythm functions and kernel methods, respectively. However, their reliance on time/phase inputs limits their spatial generalization. Unlike the non-autonomous DS, the second type methods ensure the stability of learning ADS by using the Lyapunov stability theorem. In early ADS methods for learning periodic motion, they primarily rely on limit cycle implicit mapping for learning, that is, diffeomorphism methods. For instance, ImitationFlow [10] establishes a diffeomorphism from potential attractor limit cycles to complex periodic motions. Jin et al. [11] simplified the mapping of limit cycles and simple periodic motions into point-to-point correspondences, but this one-to-one mapping method is only applicable to simple periodic motion scenarios. The Neural Liénard System [12] proposed by Zhang et al. builds upon the core ideas of ImitationFlow, utilizing a reversible neural network (NVP) [13], [14] to learn the diffeomorphism mapping between Liénard-type limit cycle dynamic systems and complex periodic motion, while maintaining the system's stable convergence. Similarly, Stable Diffeomorphic Diagrammatic Teaching (SDDT) [15] employs a 3D framework with predefined limit cycles as base dynamics, transformed via diffeomorphisms.

It is worth noting that existing methods rely on predefined, simple limit cycles or transformation mappings, and no general method capable of directly learning complex periodic motion has yet emerged. The core challenges facing direct learning

Manuscript received xx, xx, xxxx; revised xx, xx, xxxx. This work was supported by the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars of China under Grant LR23F030002 and Program of China Scholarship Council under Grant No. 202508330111. (Corresponding author: Andong Liu.)

J. Fu, A. Liu and W. Zhang are with the College of Information Engineering, Zhejiang University of Technology, and Zhejiang Key Laboratory of Intelligent Perception and Control for Complex Systems, Hangzhou, 310023, P.R. China; wazhang@zjut.edu.cn).

W. Si is with the School of Computer Science and Electronic Engineering, University of Essex, CO4 3SQ Colchester, U.K. (e-mail: w.si@essex.ac.uk).

H. Huang is with the School of Automation, Guangdong University of Technology, Guangzhou 528300, China. (e-mail: ac_huang@gdut.edu.cn.

Y. Wang is wtih the College of Control Science and Engineering, Zhejiang University, Hangzhou, 310027 China. (e-mail: ywang24@zju.edu.cn).

C. Yang is with the Department of Computer Science, University of Liverpool, L69 3BX Liverpool, U.K. (e-mail: cyang@ieee.org).

methods are: 1) the difficulty of constructing Lyapunov stability functions, 2) the complexity of nonlinear periodic dynamic systems, and 3) the balance between global stability and local motion accuracy.

In this letter, we propose a novel direct LF learning-based method for learning an ADS of complex periodic behavior, ensuring generated trajectories stably converge to the desired periodic motion. The main contributions are listed as follows:

1) We propose two coordinate transformations applicable to the energy function, modulating the equipotential curves of the energy function in terms of length and angle, respectively. This dual transformation mechanism breaks through the energy radial growth limitation of LF, enabling it to adapt in non-radial directions.
2) We propose a dual-objective optimization loss function based on the Lyapunov energy function, which achieves adaptive coding of complex periodic motion by simultaneously optimizing periodic geometric constraints and orbit parallelism conditions.
3) Unlike periodic motion methods such as DMPs and KMPs, our approach is based on autonomous dynamic systems, providing inherent stability guarantees.

The proposed method is validated by various experiments conducted on the 7-DOF Franka Panda robot. The source code and datasets are publicly available at: https://github.com/fjyggg/Complex-periodic-skiil-learning.

II. Problem Statement

This letter aims to directly learn a Lyapunov energy function that closely aligns with complex periodic demonstrations, which will guide ADS learning, ensuring that for any initial state, the trajectories generated by the learned ADS will converge to the demonstrated region and evolve periodically following the demonstration trajectories. Unlike the diffeomorphism method [16], [17], this method does not rely on mapping transformations but instead achieves motion imitation and stability assurance through direct learning of the Lyapunov energy function.

The considered complex periodic demonstration learning problem is illustrated in Fig. 1. As described in [11], [16], directly constructing a Lyapunov function via limit cycle mapping struggles to meet the requirements of complex periodic tasks. Issues such as one-to-many mapping and center-point drift can catastrophically degrade subsequent ADS learning. As shown in Fig. 1(a), taking the simplest quadratic Lyapunov function $x^\top x$ as an example, the one-to-many mapping problem can also be understood as the issue of Lyapunov energy values violating the transition from low to high energy values along complex periodic trajectories, while ideally, the Lyapunov function values on periodic motion should remain consistent. Additionally, the trajectory preprocessing caused by data normalization often leads to the center point of the periodic demonstration trajectory deviating from the true trajectory center, which results in an unreasonable Lyapunov function design. Therefore, accurately locating the center point and ensuring the consistency of the energy function have become critical issues that need to be addressed.

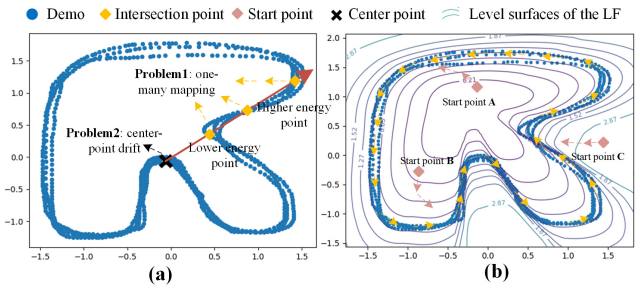

Fig. 1. The problems faced and Introduction of the periodic demonstrations.

The above problem requires that the Lyapunov function $V(x)$ to be learned have the following properties:

1) $V(x)$ is positive definite;
2) $V(x) = c$ for all $x \in S = \{x|V(x) = c\}$;
3) $\frac{\partial V(x)}{\partial x} \neq 0$ for all $x \neq 0$.

where c is any positive constant. S is the convergent level set. The property 1) ensures the LF to be a valid LF, which will be used to ensure the stability of the ADS learned in the next subsection. The property 2) is used to determine the converged set mentioned above. The property 3) must be ensured for the ADS learning. In the following section, we will present a detailed methodology for constructing such a Lyapunov function and demonstrate how it guides the learning process of the ADS.

III. Methodology

Assume we have a demonstration set $D = \{x_i^n, \dot{x}_i^n\}$, where $x_i^n \in \mathbb{R}^{d_x}$ represents the robot position at the ith time step of the nth demonstration trajectory. The proposed algorithm can be divided into two stages. First, an LF is learned such that one of its level surfaces is consistent with the complex periodic demonstration set, that is, $V(x) = c$ approximately holds for all positions x in the demonstration set D. Then, an ADS is learned, such that its generated trajectory will converge to the set $S = \{x|V(x) = c\}$. In Subsections III A and B, we introduce the learning approach for the 2-dimensional cases. In the Subsection III D, the learning approach is extended to the 3-dimensional case. The overall framework of the proposed method are shown in Fig. 2.

A. Leaning Lyapunov function

In this subsection, we aim to learn a Lyapunov function that possesses the aforementioned properties while addressing the problems in Section II. We propose a improved neural energy function based on NEUM [18] to capture demonstration preferences

$$\begin{cases} V(x) = z^T z \\ z = f_T(y) \\ y = f_R(x - x_c) \\ f_T(y) = \lambda(y) \cdot y \\ \lambda(y) = \omega(\overrightarrow{r}_y)\sigma(y) - \omega(\overrightarrow{0})\sigma(0) + \eta\|y\|^2 \\ \sigma(y) = [\sigma_1(y), \ldots, \sigma_k(y), \ldots \sigma_{dx}(y)]^\top \\ \sigma_k(y) = \rho(\alpha_i(\overrightarrow{r}_y)r_y + \beta_i(\overrightarrow{r}_y)) \end{cases} \tag{1}$$

where $r_y = ||y||$, $\overrightarrow{r}_y = \frac{y}{\|y\|}$ respectively represent the magnitude and direction of state y. x_c represents the central

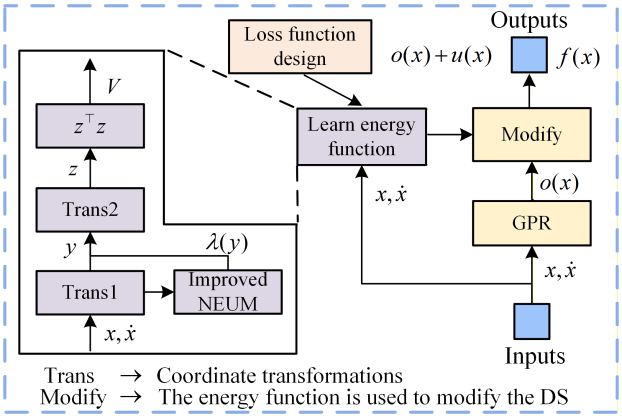

Fig. 2. The overall results of the proposed method and the schematic diagram of the entire dynamic system.

parameter to be learned. $f_T : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$, $f_R : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ respectively represent two different coordinate transformations. z represents the state of the system in the learned and transformed coordinate space. η is a positive scalars. The activation function $\rho(s)$ is chosen to be the well-known "tanh" function, i.e., $\rho(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$. $\omega_i(\overrightarrow{r}_y)$, $\alpha_i(\overrightarrow{r}_y)$, and $\beta_i(\overrightarrow{r}_y)$ represent the i-th component of $\omega(\overrightarrow{r}_y)$, $\alpha(\overrightarrow{r}_y)$, and $\beta(\overrightarrow{r}_y)$. As in NEUM, $\omega(\overrightarrow{r}_y)$, $\alpha(\overrightarrow{r}_y)$, and $\beta(\overrightarrow{r}_y)$ are set as constant value optimization parameters. In order to obtain stability conditions, the parameters here need to satisfy $\alpha_i > 0$, $w_i > 0$. In this paper, we set ω, α, and β as outputs of three fully connected neural networks (FCNN) and can be defined by the equation:

$$\begin{cases} \omega(\overrightarrow{r}_y) = NN_1(\overrightarrow{r}_y, \Theta_1), \omega \in \mathbb{R}^{d_x} \\ \alpha(\overrightarrow{r}_y) = NN_2(\overrightarrow{r}_y, \Theta_2), \alpha \in \mathbb{R}^{d_x} \\ \beta(\overrightarrow{r}_y) = NN_3(\overrightarrow{r}_y, \Theta_3), \beta \in \mathbb{R}^{d_x} \end{cases} \tag{2}$$

where Θ_1, Θ_2, and Θ_3 are trainable parameters, NN_1 and NN_2 represent the outputs of two arbitrary neural networks activated by the softplus activation function $\delta(s) = log(1 + e^s)$. This activation function ensures that the output is strictly greater than zero while maintaining a smooth gradient.

Up to here, we can observe that the original Lyapunov function $\lambda(y)$ in Eq. (1) already possesses the core functionality similar to NEUM in a simplified form. That is, it constructs an Lyapunov energy function where the energy value monotonically increases along the radial direction. However, since NEUM is essentially an end-to-end learning method for Lyapunov functions, it struggles to satisfy the properties required for periodic trajectories, as well as the challenges discussed in Section II.

To address this, we introduce two different coordinate transformations, $f_T(y)$, $f_R(x)$, based on $\lambda(y)$. As described in the question, the one-to-many mapping problem faced by complex periodic trajectories is essentially a problem where NEUM-type radial increase Lyapunov energy functions fail to learn Lyapunov energy functions with radial variations. Here, $f_T(y)$ is used to construct radial increase Lyapunov energy functions, while $f_R(x)$ is introduced to break the limitations inherent in these radially increasing Lyapunov energy functions. It should be noted that $f_T(y)$ is composed of the radially increasing function $\lambda(y)$ multiplied by y, which resembles scaling in polar coordinates. This formulation does not affect the learning of the radially increasing Lyapunov energy function since both terms exhibit coordinated growth and decay.

In order to break the constraints imposed by the radially increasing Lyapunov energy function, we applied a scaling transformation and a rotational transformation between the original input x and the input y (using two dimensions as an example), defined as follows:

$$\begin{cases} y = f_R(\bar{x}) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \bar{x} \\ \theta = NN_4(\|\bar{x}\|_2^2, \bar{x}, \Theta_4) \end{cases} \tag{3}$$

where $\bar{x} = x - x_c$, and θ is represented by an arbitrary neural network, the parameterization of θ enhances the expressive ability of the rotational modulation. Θ_4 is trainable parameters.

The final Lyapunov energy function $V(x)$ is defined in a quadratic form. After constructing the function architecture, we need to consider how to design the loss function so that the learned Lyapunov function meets the periodicity requirements and solves the problem described in Section II. We note that traditional end-to-end loss functions typically focus on the angle between vectors $\frac{\partial V(x)}{\partial x}$ and $\dot{x}$, enforcing it to be as close to 180° as possible [18]. However, in periodic dynamic systems, we instead want the angle between the vectors to be 90°, as this ensures that the energy level sets of the Lyapunov function align as closely as possible with the periodic trajectories.

Based on this insight, we define the loss function as follows:

$$\bar{J}_1(\Theta) = \left(\frac{\dot{x}^\top \frac{\partial V(x)}{\partial x}}{\|\dot{x}\|_2 \left\| \frac{\partial V(x)}{\partial x} \right\|_2} \right)^2 \tag{4}$$

where Θ represents the training parameters set of $\{\Theta_1, \Theta_2, \Theta_3, \Theta_4\}$. Additionally, we can incorporate activation functions nested within $\bar{J}_1$, similar to the approach in [7], to adjust the degree of the accuracy concern, which results in the following evaluation function

$$J_1(\Theta) = \zeta(\bar{J}_1(\Theta)) \tag{5}$$

where in this approach the activation function is defined as $\zeta(\bar{j}) = \tanh(\gamma \bar{j}) = \frac{e^{\gamma \bar{j}} - e^{-\gamma \bar{j}}}{e^{\gamma \bar{j}} + e^{-\gamma \bar{j}}}$ where $\gamma \in R_{++}$ is a tunable parameter. The larger γ is, the more we care about the sign of $\bar{j}$, which corresponds to the accuracy property. On the other hand, if γ is small, the algorithm will additionally focus on the angle between the Lyapunov energy function gradient and the velocity, which may relate to the algorithm's generalization ability.

Then, in order to make the learned Lyapunov energy function satisfy the second property $V(x) = c$ for all $x \in S = \{x | V(x) = c\}$, we define a second loss function:

$$\begin{cases} J_2 = J_c + J_t \\ J_c = \frac{1}{N} \sum_{i=1}^{N} \|V(x_i) - c\|_2^2 \\ J_t = \frac{1}{N-1} \sum_{i=2}^{N} [(V(x_i) - V(x_{i-1})) - (\|x_i\| - \|x_{i-1}\|)]^2 \end{cases} \tag{6}$$

where J_c and J_t represent the track loss and trend matching loss, respectively. c is a positive constant value. J_c is used to ensure that the learned Lyapunov energy function satisfies the

second property as much as possible. J_t is used to strengthen the trend-matching degree of the learned Lyapunov energy function.

Remark: Although J_c alone enforces $V(x) \approx c$ on demonstrations, it fails to constrain the trend of energy variation, potentially causing gradient misalignment with periodic motion. As coordinate transformations break the radial constraint, the target trajectory lies on a complex-shaped contour $V(x) = c$, where energy is constant yet state positions vary widely. The introduction of J_t helps to utilize the temporal information between adjacent states (x_i, x_{i-1}) to adjust the energy change trend, guiding the learned energy function to tend towards a geometry consistent with the exhibited periodic motion.

The total loss J is ultimately as follows:

$$J = J_1 + \mu_1 J_2 + \mu_2 J_\theta \tag{7}$$

where μ_1 and μ_2 are positive values. J_θ is the regularization loss of the parameter. In this paper, we determined the optimal interval for selecting μ_1 through parameter scanning experiments, with the optimal interval being $\mu_1 \in [0.3, 0.8]$. The parameter scanning results are shown in the Tab. II.

Although the proposed architecture can effectively learn complex periodic trajectories, the center point bias problem in traditional normalization methods may still cause the learned Lyapunov energy function to exhibit unexpected behavior. Through analysis, it was found that the center of periodic trajectories is typically located at the dynamic position of the center of mass of a three-dimensional object rather than a fixed single point. Therefore, we propose parameterizing the center point as a learnable variable, enabling the system to adaptively adjust the trajectory center. The parameters to be learned are ultimately $\Theta = \{\Theta_1, \Theta_2, \Theta_3, \Theta_4, x_c\}$. We constructed the proposed structure using PyTorch and trained it using the Adam optimizer. In the following section, we will introduce a stability analysis of the proposed framework.

B. Stability analysis

In this subsection, we provide proof of the overall stability of the system. Prior to this, we proved that an appropriate Lyapunov function must satisfy desired properties in the problem statement, where property 2) is satisfied by the loss function (6). Therefore, the focus of this section is to argue these conditions.

Proposition 1: The proposed Lyapunov energy function $V(x)$ satisfies 1) $V(x)$ is positive definite and 2) it is radially unbounded.

Proof: Firstly, $V(x) = (f_T(f_R(x - x_c)))^\top (f_T(f_R(x - x_c)))$ are semi-positive definite, which is obvious because our Lyapunov energy function is derived from a quadratic form. By observing the forms of f_T and f_R, we can find that it is easy to obtain $f_T(0) = 0$ and $f_R(0) = 0$. Then, we can prove that $V(0) = 0$ by substituing $x = x_c$ to $V(x)$. Therefore, $V(0) = 0$ hold if only if $x = x_c$. When $x \to \infty$, $f_R(x - x_c) = \infty$, and $f_T(y) = \infty$, it is clear that V is radially unbounded.

Proposition 2 (Non-zero Gradient): The Lyapunov energy function $V(x) = (f_T(f_R(x - x_c)))^\top (f_T(f_R(x - x_c)))$ has the property $\frac{\partial V}{\partial x} \neq 0$ if $x \neq x_c$.

TABLE I
IMPACT OF μ_1 ON LYAPUNOV ENERGY FUNCTION PERFORMANCE

μ_1 **range**	J_1 **behavior**	J_2 **behavior**	**Energy function behavior**
$[0, 0.1)$	Fast conv.	Oscillatory	Orbital divergence
$[0.1, 1.0]$	Gradual descent	Stable conv.	Small error, conv. fast
$(1.0, 5.0]$	Slow descent	Overfitting	Stiff orbit, big error
> 5.0	Gradient vanish	Extreme minimalism	Stability loss

Proof: According to the Eq. (1), $V(x)$ can be rewritten as follows:

$$V(x) = (f_T(y))^\top (f_T(y)) = \lambda^2(y) \|y\|_2^2 = \lambda^2(y) r_y^2 \tag{8}$$

According to the chain rule of differentiation, we have:

$$\frac{\partial V}{\partial y} = 2\lambda(y) \frac{\partial \lambda(y)}{\partial y} r_y^2 + 2\lambda(y)^2 \frac{\partial r_y}{\partial y} \tag{9}$$

Since $\lambda(y)$ is a neural network parameterization of the NEUM form, it retains the original properties of the NEUM, namely 1) $\lambda(y) > 0$, and 2) $\frac{\partial \lambda(y)}{\partial y} > 0$ when $y \neq 0$. The detailed derivation process can be found in [18]. We can further deduce that $\frac{\partial r_y}{\partial y} = \frac{\partial \|y\|}{\partial y} = \frac{y}{\|y\|} \neq 0, \forall y \neq 0$. Thus, all terms on the right side of Eq. (9) are greater than 0, we can obtain $\frac{\partial V}{\partial y} > 0, \forall y \neq 0$.

Since $\frac{\partial y}{\partial x} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$ is full rank, we have $y = Rx \neq 0 \Rightarrow x \neq x_c$. We can further obtain that $\frac{\partial V}{\partial x} = \frac{\partial V}{\partial y} \frac{\partial y}{\partial x} \neq 0$ if $x \neq x_c$, which completes the proof.

Based on the proof above, we can derive the Lyapunov energy function (1) satisfies the required properties.

C. Learning an ADS with a predefined Lyapunov function

In this subsection, we aim to learn an ADS, of which the generated trajectory will converge to the set S. Based on the demonstration set D, we can first learn an original ADS, denoted as $\dot{x} = o(x)$, using any regression algorithm, such as Gaussian Process Regression (GPR) [19], [20]. This subsection omits the details of learning the original DS, as it is trivial. However, we cannot guarantee that the trajectories generated by the original ADS will converge to the set S, and issues such as divergence or getting stuck in local minima often arise. In this subsection, we modify the original ADS to $\dot{x} = o(x) + u(x)$, where the complementary input $u(x)$ is used to correct the original ADS so that the modified ADS can generate the desired trajectory.

The correct input $u(x)$ [3] can be computed by solving the following optimization problem

$$\min_u u^\top u \tag{10}$$

subject to

$$\begin{cases} (i) \left(\frac{\partial V(x)}{\partial x}\right)^\top (o(x) + u(x)) = \frac{-d}{c} V(x) + d \\ (ii) \, \|o(x) + u(x)\| \leq \overline{v}(x) \end{cases} \tag{11}$$

where d is a positive constant, c is the same constant in property 2), and $\overline{v}(x)$ is a function of which the output is positive. The first constraint in Eq. (11) is used to ensure

that ADS converges to the desired periodic behavior. We can rewrite the first constraint in constrain (11) as follows:

$$\dot{V} = \frac{-d}{c}V + d \tag{12}$$

For convenience, the variable x is omitted. Multiplying $e^{\frac{b}{c}\tau}$ on both sides of the above equation yields

$$e^{\frac{d}{c}\tau}\dot{V} + \frac{d}{c}e^{\frac{d}{c}\tau}V = e^{\frac{d}{c}\tau} \tag{13}$$

By integrating the above equation over the time interval $\tau \in [0, t]$, we can obtain

$$e^{\frac{d}{c}t}V = e^{\frac{d}{c}t}c - c + V_0 \tag{14}$$

where V_0 is the initial value of the Lyapunov energy function $V(x)$. Multiplying both sides of Eq. (14) by $e^{\frac{d}{c}t}$, we can obtain

$$V = c + e^{\frac{-d}{c}t}(V_0 - c) \tag{15}$$

As a result, the Lyapunov function V asymptotically approaches c. From Eq. (15), it follows that when $V_0 \neq 0$, V remains non-zero $\forall t > 0$. The secondary condition in (11) serves as a velocity saturation constraint for ADS safety compliance. In addition, the ratio of $\frac{d}{c}$ directly affects the convergence speed and generalization characteristics of dynamic systems: while a smaller $\frac{d}{c}$ preserves periodic generalization trends, it substantially slows convergence; and vice versa.

D. Extending to the 3-dimensional case

In 3D space, loss functions designed for the 2D case may no longer be applicable. This is because the set S forms a closed surface, not just a closed curve. In the 2D case, the loss function, by constraining the trajectory to align with the contour lines of the Lyapunov energy function (i.e., a 1D closed curve), can directly guarantee that the trajectory moves along a periodic path. However, in 3D space, the contour surface is a 2D surface, and simply constraining the trajectory to this surface does not guarantee that it moves along a specific closed path—the trajectory may drift arbitrarily on the surface and fail to follow the desired periodic trajectory.

To address this challenge, we propose a hierarchical extension strategy that decomposes the three-dimensional problem into a two-dimensional periodic subsystem and a tracking controller for tracking the third dimension. Let us redefine $x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$ and $\bar{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$. Firstly, we can obtain the 2D ADS $\bar{x} = o(\bar{x}) + u(\bar{x})$ by Eq. (10) and (11) mentioned in the above Section III. C. Then, we can establish the relationship between the predicted x_3^* and $\bar{x}$, where the function $f(\cdot)$ can be determined by using any regression algorithm. In this work, we adopt GPR. Then, we determined the dynamics of the third dimension as follows

$$\dot{x}_3 = -k_{x_3}(x_3 - f(\bar{x})) + \dot{f}(\bar{x}) \tag{16}$$

where k_{x_3} is a positive constant. It is obvious the third dimension x_3 is also periodic under the dynamic (16). Through mathematical transformation, (16) can be expressed as $\dot{e} = -k_{x_3}e$, where $e = x_3 - f(\bar{x})$. If $k_{x_3} > 0$, we can ensure that e will decrease to 0, that is, x_c will converge to $f(\bar{x})$. Since $\bar{x} = o(\bar{x}) + u(\bar{x})$ is periodic, then x_3 is also periodic.

Algorithm 1 Complex periodic ADS algorithms

1: Collecting the demonstration set D.
2: Setting values of constant parameters and c.
3: Off-line learning the LF $V(\cdot)$ according to (1).
4: Off-line learning the original ADS $o(\cdot)$ using GPR.
5: Setting values of constant parameters d and k_{x_3}.
6: Off-line learning the function $f(\cdot)$ using GPR.
7: **for** $t = t_0$ to $\cdots$ **do**
8: Observing the robot position x.
9: Computing ut by solving the optimization problem described in (10) and (11).
10: Computing 2D ADS $\bar{x}_t = o(\bar{x}_t) + u(\bar{x}_t)$.
11: Computing $x_{3,t}^{\cdot}$ according to (16).
12: Sending the desired velocity $\bar{x}_t = \begin{bmatrix} \dot{x}_t & \dot{x}_{3,t} \end{bmatrix}^\top$ to the low-level velocity tracking controller
13: **end for**

At the end of this section, we have included the pseudocode for the entire algorithm, as shown in Algorithm 1.

IV. EXPERIMENTS

In this section, we test our method using two datasets, evaluating it sequentially from simple periodic trajectories to complex periodic trajectories. Subsequently, we further validate our method through a series of real-world robot experiments.

A. Parameter settings

In this section, we validate the effectiveness of the proposed algorithm on the two datasets (i.e., the dataset in [11] and the "IROS" dataset in [17]). A total of eight trajectories are included, namely "2-D circle", "2-D rectangle", "2-D star1" , "2-D star2", "RShape" and "SShape". We utilize Eq. (1) as the Lyapunov function. NN_1, NN_2 are defined as multilayer perceptrons (MLPs) with two hidden layers, where each hidden layer contains 10 neurons and the output layer consists of 2 neurons, and the output is activated by an exponential function. NN_3 are defined as MLPs with two hidden layers, where each hidden layer contains 10 neurons, and the output layer consists of 2 neurons. NN_4 is set as MLP with two hidden layers, each containing 10 neurons, with a normalization layer added after the first hidden layer. We constructed the proposed structure using PyTorch and trained it using the Adam optimizer. The number of training epochs was set to 2000, with a learning rate of 0.01. All experiments were conducted on a laptop equipped with an Intel Core i9-12900K CPU and an NVIDIA GeForce RTX 3060 GPU.

B. Simulation

We first consider the 2-dimensional case, with a total of eight periodic simulation trajectories. Some related parameters are set as $\mu_1 = 0.3$, $\mu_2 = 0.001$, $c = 1.0$, where $\mu_1$can also be adjusted in the optimal interval. According to Eq. (15), the ratio between parameters d anb c determines the convergence rate and generalization ability of the ADS, and we set $d = 3$,

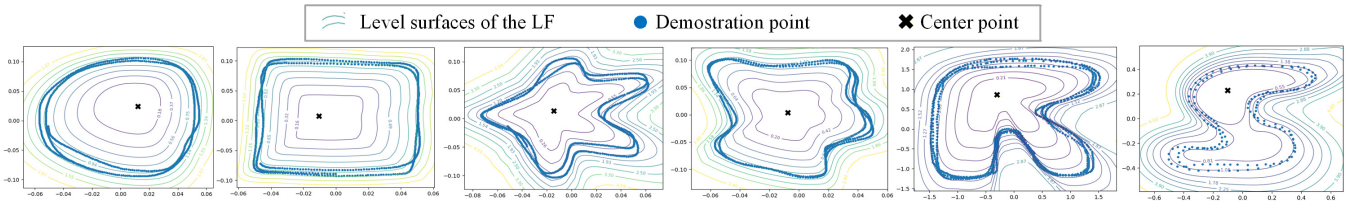

Fig. 3. The Lyapunov energy function results for the eight 2-dimensional simulation experiment. From left to right and top to bottom, the shapes are “2-D circle”, “2-D rectangle”, “2-D star1”, “2-D star2”, “RShape”, and “SShape”.

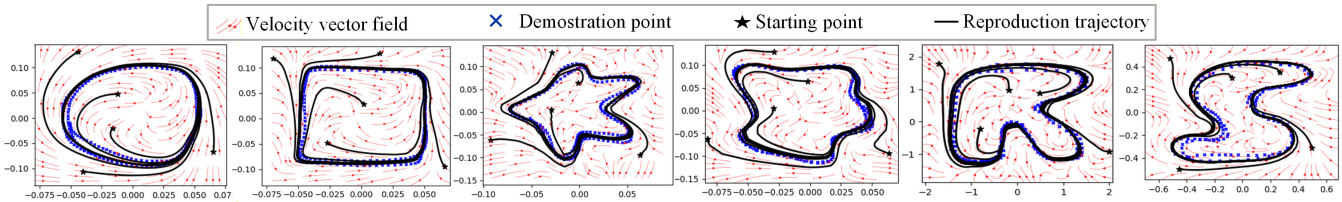

Fig. 4. The Vector fields for the eight 2-dimensional simulation experiment. From left to right and top to bottom, the shapes are “2-D circle”, “2-D rectangle”, “2-D star1”, “2-D star2”, “RShape”, and “SShape”.

at which case the learned ADS will nearly be convergent in 3s. For some simple periodic trajectories, such as “2-D circle”, “2-D rectangle” and “2-D star1”, we set $\gamma = 1.5$, while for some complex ones, such as “RShape” and “SShape”, we set $\gamma = 2$.

The results of the Lyapunov energy function in two dimensions are shown in Fig. 3, where the blue points represent the sampled demonstration points and the black crosses represent the centers of the learned periodic trajectories. In experiments ranging from easy to difficult, it can be seen that the LF learned by the proposed method can accurately capture the shape of periodic motion. To give a more intuitive illustration of the global stability of the learned ADSs, we plot their vector fields in Fig. 4, where the black trajectories are ideally generated by the learned ADS $\bar{x} = o(\bar{x}) + u(\bar{x})$. As we can see, for each simulation and any start point, the trajectory will gradually evolve to the demonstration area and then be periodic.

To verify the contributions of each component of the loss function in our proposed framework (bi-objective loss J_2 (J_t and J_c) and learnable centroid x_c), we conducted ablation experiments on the ”RShape” trajectory. The baseline model is our proposed full method (PA-Full). The results are shown in Fig. 5. It can be observed that without J_t loss, the learned energy function is not significantly different from that in the PA-Full case, showing only a small gap. The choice of different fixed center point has a significant impact on the learned energy function. Poor centroid selection may lead to poor energy function learning or even failure, while adaptive center point learning will learn an optimal center point. Furthermore, the ablation experiments with J_2 loss and polar coordinate transformation are meaningless in themselves because J_2 discards stability condition 2, and the output of the improved NEUM framework is a vector. We also evaluated the robustness of the proposed method under different noise levels, as shown in Fig. 5. (h-j). As can be seen from the figure, our method maintains good consistency with the learned energy function and exhibits slow performance degradation. Even at a large noise level ($\sigma = 0.05$), the generated energy still maintains the

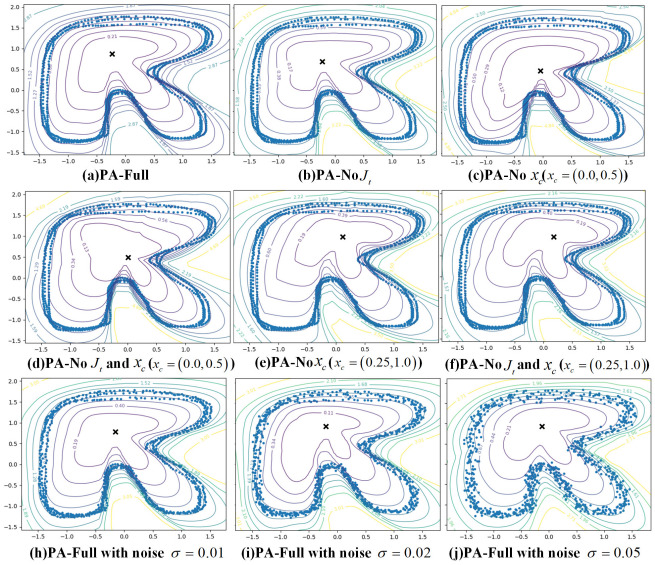

Fig. 5. The results of ablation experiments and robustness assessments under different noise levels.

correct periodic shape. This demonstrates the strong robustness of the proposed method.

In addition, we compared the proposed method with various state-of-the-art methods, such as the rhythm DMP [11] method and the Gaussian Process (GP) [21] based periodic learning method. The parameter settings of DMP have been carefully adjusted to ensure that it can run well in experiments. The comparison results are shown in Fig. 6. Due to the limitations of the GP method on single point-to-point mapping and fixed center points, it cannot handle complex periodicity. Therefore, comparisons were made on trajectory shapes ”Rshape2-D star2” (Fig. 5(a)) and ”Rshape” (Fig. 5(b)). We can observe that the reproduction trajectory of DMP has a significant error compared to the demonstration trajectory, while the GP and the proposed algorithm (PA) can almost match the demonstration trajectory. In Tab. II, we provide a more intuitive comparison of accuracy metrics Root Mean Square Error

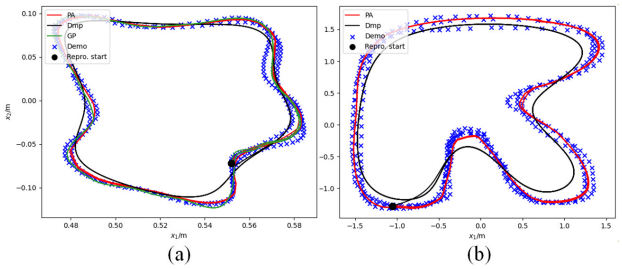

Fig. 6. Comparison results between PA, DMP, and GP methods. The left subfigure (a) is "2-D star2" trajectory, and the right subfigure (b) is "RShape" trajectory.

TABLE II
COMPARISON OF ACCURACY AND TRAINING TIME RESULTS FOR DIFFERENT METHODS.

	2-D star2		Rshape		Sshape	
Method	RMSE	SEA	RMSE	SEA	RMSE	SEA
DMP [11]	0.01856	0.005095	0.4195	4.505	0.3756	4.023
GP [21]	0.01247	0.002541	–	–	–	–
PA	**0.01097**	**0.002385**	**0.2585**	**1.801**	**0.2273**	**1.625**

Method	$V(x)$ time(s)	ODS time(s)	$u(x)$ time(ms)
2-D star2	23.69	10.27	0.2391
RShape	38.37	9.31	0.2392
Sshape	29.83	10.12	0.2407

*The training rounds for the above experiments were all 2000 epochs.

TABLE III
COMPARISON OF PA WITH THE STATE-OF-THE-ART

Method	Stability Guar.	General. Ability	Complex Periodic	High-DoF Scalab.	Training Time
DMP [11]	×	weak	✓	✓	fast
GP [21]	✓	strong	×	✓	fast
NVP [12]	✓	weak	✓	×	slow
SDDT [15]	✓	weak	✓	×	slow
PA	✓	**strong**	✓	✓	**medium**

(RMSE, measuring average pointwise deviation) [3] and Swept Error Area (SEA, simultaneously measuring the temporal and spatial differences between two trajectories) [4] in three two-dimensional scenarios. We can find that PA is superior to the DMP and GP methods. Tab. II also reports the time required to learn $V(x)$, ODS, and $u(x)$, with the most computationally intensive stage being the learning of the Lyapunov energy function. Training time varies with the complexity and length of the demonstration trajectory. Once training is complete, the learned ADS can generate trajectories in real time. The optimization of correcting the input $u(x)$ is modeled as a QP problem, requiring significantly less time than the typical control cycle (e.g., 1 millisecond) of modern robotic systems such as the Franka Panda. For broader validation, we extended our comparison framework to include neural Liénard systems [12] and SDDT [15] methods, with comprehensive results detailed in Tab. III.

C. Real robot experiment

In this section, the effectiveness of the proposed algorithm is validated through writing trajectory experiments on the Franka Panda robot. The experimental setup and expert demonstration process is shown in Fig. 7, where a human first drags the robot to perform a periodic motion demonstration, and then the robot is expected to reproduce these actions. The collected data includes four sets of trajectory data (position and velocity), namely "Z", "J", "U" and "T" trajectories, where "Z" and "J" are 2D data, while "U" and "T" are 3D data, used to validate the algorithm's effectiveness in a 3D scenario.

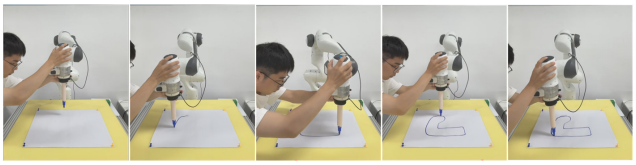

Fig. 7. The experimental setup and expert demonstration process (from left to right), using "Jshape" as an example.

The experimental results for the 2-dimensional case are shown in Fig. 8, where cross points are sampled demonstration points, black trajectories are ideally generated by the learned ADS $\bar{x} = o(\bar{x}) + u(\bar{x})$ (obtained by simulations), and the red trajectories are generated in the real experiments. As shown in Fig. 8, in both experiments, the data-driven LF learned by the proposed framework accurately captures the shape of periodic motion due to its learning ability. Both the simulated trajectory (black) and the experimental trajectory (red) converge to the demonstration area and then evolve periodically like the demonstration behavior. Therefore, the value of V in the real robot experiment will eventually hover around c, as shown in Fig. 8. (f). However, if the robot ideally tracks the required speed (in the simulation scenario), the value of V will perfectly converge to the value c, as shown in Fig. 8. (e). The 2D periodic trajectory robot is reproduced as shown in Fig. 10. (a). The experimental results for the 3-dimensional case are shown in Fig. 9. (a) and (b), where each row corresponds to the experimental results in three dimensions. As expected, the proposed method can be successfully extended to three-dimensional situations. For dimensions x_1 to x_2, the mechanism for generating periodic motion is the same as in the previous 2-dimensional experiments. Once the motion in dimensions x_1 to x_2 converges, the expected motion in dimension x_3 will also be periodic. The 3D periodic trajectory robot is reproduced as shown in Fig. 10. (b). To quantitatively evaluate the performance of the 3D extension, we compare the PA with DMP. Fig. 9. (a) and (b) show that the DMP tracking performance is significantly affected by the periodic parameter setting, while our method can maintain a high degree of consistency with the demonstration trajectory without fine-tuning. Quantitative analysis (Fig. 9. (c)) further shows that our method is significantly lower than DMP in terms of SEA, verifying the effectiveness and accuracy advantage of the proposed 3D extension framework.

The experimental results show that the proposed method is capable of learning complex periodic motion trajectories.

V. CONCLUSION

In this letter, we propose a novel direct Lyapunov functions learning approach of complex periodic motions, fundamentally

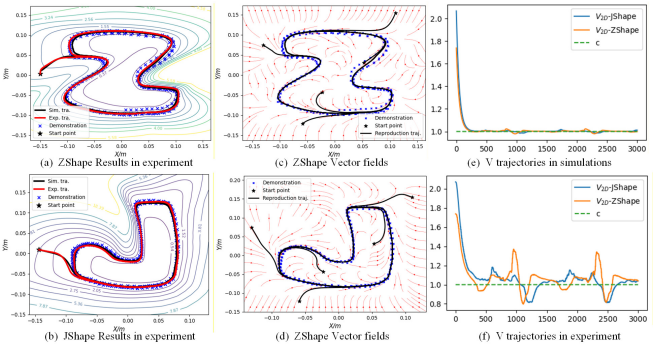

Fig. 8. (a)(b): Simulation and experimental results of two-dimensional experiments on the trajectories of "Z" and "J". (c)(d): Vector fields for the 2-dimensional experiments. (e)(f): LF value trajectories for 2-dimensional experiments and simulations.

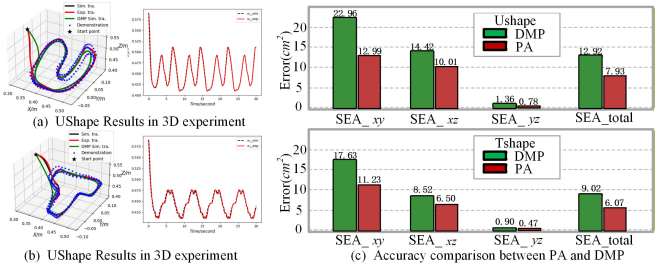

Fig. 9. Simulation, experimental, and comparative results of three-dimensional experiments on "U"-shaped and "T"-shaped trajectories.

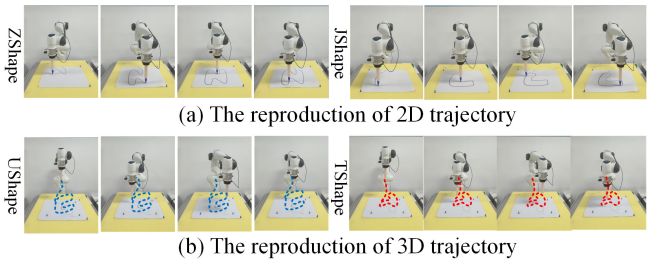

Fig. 10. The real periodic trajectory robot reproduction.

distinct from all existing limit cycle mapping methods, and construct a stable dynamical system using this function. Firstly, we propose two polar coordinate transformations that break through the radial energy growth limitation of LF, enabling it to adapt non radially. Subsequently, we learn the ADS for complex periodic motions by solving the convex optimization problem associated with the LF. Furthermore, we successfully extend the proposed method from two-dimensional to three-dimensional cases. Experimental results demonstrate that our approach can effectively encode complex periodic motions while maintaining guaranteed stability.

While the proposed method performs well on various complex periodic motion tasks, it still has some limitations. First, despite introducing neural networks to enhance expressive power, the method still struggles to accurately model extremely complex or highly unstructured periodic trajectories (such as intersecting or high-frequency oscillating trajectories). Furthermore, the method involves multiple hyperparameters (μ_1, μ_2, γ), the tuning of which significantly impacts performance, and a fully adaptive parameter selection mechanism has not yet been achieved.

REFERENCES

[1] A. G. Billard, S. Calinon, and R. Dillmann, "Learning from humans," *Springer handbook of robotics*, pp. 1995–2014, 2016.

[2] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, no. 1, pp. 297–330, 2020.

[3] J. Fu, H. Huang, Z. Jin, A. Liu, W.-A. Zhang, L. Yu, W. Si, and C. Yang, "A survey on learning an autonomous dynamic system for human–robot skills transfer from demonstration," *Robotics and Computer-Integrated Manufacturing*, vol. 97, p. 103092, 2026.

[4] Khansari-Zadeh, S Mohammad and Billard, Aude, "Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014.

[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, no. 2, pp. 328–373, 2013.

[6] C. Yang, C. Chen, W. He, R. Cui, and Z. Li, "Robot learning system based on adaptive neural control and dynamic movement primitives," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 3, pp. 777–787, 2019.

[7] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," *Advances in neural information processing systems*, vol. 26, 2013.

[8] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 833–852, 2019.

[9] A. Liu, J. Fu, S. Zhan, Z. Jin, and W.-a. Zhang, "A policy searched-based optimization algorithm for obstacle avoidance in robot manipulators," *IEEE Transactions on Industrial Electronics*, vol. 71, no. 9, pp. 11 262–11 271, 2024.

[10] J. Urain, M. Ginesi, D. Tateo, and J. Peters, "Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5231–5237.

[11] Z. Jin, A. Liu, W.-A. Zhang, L. Yu, and C. Yang, "Learning an autonomous dynamic system to encode periodic human motion skills," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 4, pp. 7757–7763, 2025.

[12] H. Zhang, L. Cheng, Y. Zhang, and Y. Wang, "Neural liénard system: learning periodic manipulation skills through dynamical systems," *Science China Information Sciences*, vol. 67, no. 12, pp. 1–16, 2024.

[13] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[14] M. A. Rana, A. Li, D. Fox, B. Boots, F. Ramos, and N. Ratliff, "Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 630–639.

[15] W. Zhi, H. Tang, T. Zhang, and M. Johnson-Roberson, "Teaching periodic stable robot motion generation via sketch," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 1154–1161, 2025.

[16] H. Zhang, L. Cheng, Y. Zhang, and Y. Wang, "Neural liénard system: learning periodic manipulation skills through dynamical systems," *Science China Information Sciences*, vol. 67, no. 12, pp. 1–16, 2024.

[17] J. Urain, M. Ginesi, D. Tateo, and J. Peters, "Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5231–5237.

[18] Z. Jin, W. Si, A. Liu, W.-A. Zhang, L. Yu, and C. Yang, "Learning a flexible neural energy function with a unique minimum for globally stable and accurate demonstration learning," *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4520–4538, 2023.

[19] M. Seeger, "Gaussian processes for machine learning," *International journal of neural systems*, vol. 14, no. 02, pp. 69–106, 2004.

[20] J. Fu, Z. Jin, A. Liu, W.-A. Zhang, and L. Yu, "Non-parametric gaussian process movement primitive with via-point constraint for effective and safe robot skill learning," *Neurocomputing*, vol. 589, p. 127711, 2024.

[21] C. Yang, C. Chen, W. He, R. Cui, and Z. Li, "Robot learning system based on adaptive neural control and dynamic movement primitives," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 3, pp. 777–787, 2019.