Learning a Flexible Neural Energy Function With a Unique Minimum for Globally Stable and Accurate Demonstration Learning

Zhehao Jin, Weiyong Si, Andong Liu, *Member, IEEE*, Wen-An Zhang, *Member, IEEE*, Li Yu, *Member, IEEE*, and Chenguang Yang, *Senior Member, IEEE*

***Abstract*—Learning a stable autonomous dynamic system (ADS) encoding human motion rules has been shown as an effective way for demonstration learning. However, the stability guarantee may sacrifice the demonstration learning accuracy. This article solves the issue by learning a stability certificate, represented by a neural energy function, on the demonstration set. We propose a polarlike space analysis approach to derive parameter constraints to guarantee the unique-minimum property of the neural energy function, which is essential for it to be a cogent stability certificate. Then, the neural energy function is learned to capture the demonstration preferences via constrained optimization algorithms. With the learned neural energy function, a globally asymptotically stable ADS with predefined position constraint is further formulated. We also quantitatively analyze the generalization ability of the learned ADS by utilizing the substantial flexibility of the neural energy function. The effectiveness of the proposed approach is validated on the LASA dataset and two representative robotic experiments.**

***Index Terms*—Autonomous dynamic system (ADS) learning, demonstration learning, Lyapunov function (LF) learning, motion-skills transfer, optimization.**

Nomenclature

x	Robot position.
$g(x)$	Stable ADS function expected to be learned.
$o(x)$	Original ADS function.
u	Corrected term in $g(x)$.
$V(x)$	Neural energy function.
D	Demonstration set.
M	Number of samples in the demonstration set D.
d_x	Dimension of the robot position.
$f(x)$	Feature of the neural energy function.
d_H	Dimension of the feature $f(x)$.
$z(x)$	Manually designed encoder.
a_k, b_k	Feature parameters of the neural energy function.
ω	Weight parameter of the neural energy function.
r	Length coordinate of a polarlike space element.
θ	Angle coordinate of a polarlike space element.
$h(\theta)$	Direction vector of a polarlike space element.
$V_P(r,\theta)$	Polarlike space form of the neural energy function.
$f_P(r,\theta)$	Polarlike space form of the neural energy feature.
$z_P(r,\theta)$	Polarlike space form of the encoder.
R_+	Set containing all nonnegative real scalars.
R_{++}	Set containing all positive real scalars.
R^n	Set containing all real vectors with dimension n.
$R_+^{n\times n}$	Set containing all semipositive definite real matrices with size $n \times n$.
$R_{++}^{n\times n}$	Set containing all positive-definite real matrices with size $n \times n$.

Manuscript received 2 June 2023; accepted 2 August 2023. This article was recommended for publication by Associate Editor S. L. Smith and Editor Nancy Amato upon evaluation of the reviewers' comments. This work was supported in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LR23F030002 and Grant LD21F030002, in part by the National Natural Science Foundation of China under Grant 61973275, and in part by the Chinese Scholarship Council under Award 202108330357. *(Corresponding author: Andong Liu.)*

Zhehao Jin, Andong Liu, Wen-An Zhang, and Li Yu are with the Zhejiang Provincial United Key Laboratory of Embedded Systems, Department of Information Engineering, Zhejiang University of Technology, Hangzhou 310032, China (e-mail: jzhzjut@outlook.com; lad@zjut.edu.cn; wazhg@hotmail.com; lyu@zjut.edu.cn).

Weiyong Si and Chenguang Yang are with the Bristol Robotics Laboratory, University of the West of England, BS16 1QY Bristol, U.K. (e-mail: weiyong.si@uwe.ac.uk; cyang@ieee.org).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2023.3303011.

Digital Object Identifier 10.1109/TRO.2023.3303011

I. Introduction

With the development of robotic technologies, robots have been widely used in structured factory environments and other unstructured environments such as homes, offices, hospitals, etc. Learning humanlike skills is essential for robots in both structured and unstructured environments, such as humanlike polishing [1], assembly [2], collaboration [3], motion [4], physical interactions [5], [6], [7], etc. Learning from demonstration (LfD) approaches provide effective ways to reliably transfer humanlike skills to robots.

This article focuses on motion-skills transfer, which is basic but essential for robots. Learning a dynamic system (DS) [8] that encodes human motion rules is an effective approach for transferring motion skills. There are basically two types of approaches for learning DS. The first type aims to learn a nonautonomous DS (N-ADS) where the generated trajectory is time-driven. Specifically, the output of the learned N-ADS could be a position/velocity sequence indexed by time steps. The second type learns autonomous DS (ADS), where the state variation of the DS is determined by the current state instead of the time step.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

In principle, any regression algorithm can be used to establish an N-ADS by setting the time index and robot position (velocity) as the input and output, respectively. Most N-ADS learning approaches are conducted under a statistic framework for naturally obtaining learning confidence. In [9], [10], [11], and [12], the time step and demonstration position were assumed to satisfy a Gaussian mixture model (GMM), and the posterior distribution of the position given by the time index was computed by using the Gaussian mixture regression (GMR) algorithm. However, it had been shown in [13] that the generalization capacity of N-ADS learned by the original GMM/GMR could be weak, especially for new scenarios which have not appeared in the demonstration set. In [13], [14], [15], task-parameterized GMM (TP-GMM) was presented to further enhance the generalization capacity by including task-related parameters in the EM optimization procedure. In [16], a learning approach for task parameters was presented to enhance the performance of the TP-GMM. Other popular approaches for learning N-ADS contain probabilistic movement primitives [17], [18] and kernelized movement primitives [19].

A major disadvantage of N-ADS is that it is ill-suited for highly dynamic tasks since the motion generated by the N-ADS does not get feedback from the environment [20]. Unlike N-ADS, humans always determine their behaviors in real-time according to the current environmental state, which is similar to the ADS. Dynamic movement primitives (DMP) [21], [22] might be one of the most popular approaches to learn an ADS. The globally asymptotic stability of the ADS learned by the DMP is ensured by its special structure, i.e., a combination of a stable second-order ADS and a possibly unstable but gradually decaying nonlinear ADS. In [23] and [24], GMM/GMR approaches were integrated into the DMP framework to enhance its learning performance for multiple-demonstration tasks. Although the stability of the DMP could be ensured, its special structure separates each motion dimension and, thus, decreases the reproduction and generalization abilities [25].

Compared with utilizing the special structure, using a stability certificate for the stability ensurement is more prevalent for the ADS learning. A commonly used certificate is called the contraction metric [26], which measures the difference between two trajectories starting from two nearby initial states. A basic work for integrating the contraction theorem and the stable ADS learning was presented in [27], where the ADS was learned by a neural network such that the contraction metric along the ADS is permanently decreasing. Sindhwani et al. [28] presented a nonparametric framework for contracting ADS learning. However, simple forms of the contraction metrics used in [27] and [28] limit the reproduction accuracy for complex demonstration trajectories. In [29], [30], and [31], data-driven contraction metrics were integrated into the learning framework to improve the reproduction accuracy of the learned ADS.

Another stability certificate is called the Lyapunov function (LF), which is a kind of positive-definite, continuously differentiable energy function. Compared with the contraction stability theorems, the Lyapunov stability theorems may attract more attention in the control society since it is more intuitive and developed [8]. It had been pointed out in [26], [28], [29] that the contraction metric can be viewed as a special LF. One of the fundamental works for combing Lyapunov theorems and ADS learning is called stable estimator of dynamical systems (SEDS) [32], where a globally asymptotically stable (GAS) ADS represented by GMM/GMR was learned under some quadratic-energy-function-related stability constraints. In [33], the work in [32] was further extended to the case with an unstatic goal position. Lemme et al. [34] presented LF constraints for a more general neural-parameterized ADS; however, the resulted optimization problem is intractable, and it was approximately solved with a constraints-sampling approach. In [35] and [36], this problem was solved by designing a special neural network activation function.

However, similar to the problem faced in the contraction metric, a quadratic energy function used in the previous approaches limits the reproduction accuracy of the learned ADS. Learning data-driven energy functions consistent with the demonstration trajectories can dramatically improve the reproduction accuracy. In [37], the weight matrix of the quadratic energy function was optimized on the demonstration set to increase the consistency between the energy function and the demonstration set. In [38], a more flexible energy function called "weighted sum of asymmetric quadratic functions (WSAQF)" was learned to further increase the consistency, and the stable ADS was established by using the control Lyapunov approach. In [39], a bridge between learning a stable ADS and a special type of energy functions was built by presenting a diffeomorphic transformation, which resulted in the τ-SEDS algorithm. This approach further highlights the importance and value of the energy function learning. In [40], a fast diffeomorphic matching (FDM) approach was presented to map the complex ADS to a simple stable ADS, and a transformed energy function was simultaneously generated. In [41] and [42], a sum-of-squares (SOS) energy function was presented, and it showed a better performance compared with the WSAQF. Neumann et al. [43] designed a more general neural-parameterized energy function, called neurally-imprinted Lyapunov candidates (NILC), to capture the demonstration preferences, and both simulation and experiment results validated the effectiveness of the NILC. However, the positive-definite property of the NILC can only be approximately ensured, and thus, the stability of the resulted ADS cannot be ensured theoretically. In [44], a special neural-parameterized energy function, referred to as the Lyapunov neural network (LNN), was designed to ensure the positive-definite property. In [45], the generalization error result of the data-driven energy function was analyzed by making bounded assumptions of the true ADS.

However, except for the WSAQF [38] and the energy function transformed by FDM [40], all of the above complex data-driven energy functions may have multiple minima. Undesired minima will introduce spurious attractors for the ADS. Specifically, if an energy function with multiple minima is used as the stability certificate, the trajectory generated by the learned ADS can principally converge to an arbitrary minimum, which may raise safety issues. It should be noted that for a data-driven approach, the model flexibility always positively correlates with the model complexity. Consequently, the WSAQF [38] cannot support the ADS to produce extremely complex trajectories

due to its quadraticlike form. Similarly, the FDM-transformed energy function in [40] does not allow the ADS to generate spiral trajectories due to the topological constraint. It is well known that deep neural networks are very flexible. However, ensuring the unique-minimum property for a general neural network could be very difficult. In recent years, an input-convex neural network (ICNN) [46] was presented to solve the unique-minimum problem by ensuring that the neural network is convex in the input. However, the convexity constraint of the ICNN can severely limit its performance for capturing complex demonstration preferences. Moreover, due to the limited flexibility, both the WSAQF, FDM-transformed energy function and ICNN cannot capture the demonstration preferences on a broad area, which corresponds to the more challenging generalization ability. These issues motivate the main part of the work in this article, i.e., to learn a flexible neural energy function with the unique-minimum property.

In this article, we present an approach to learn a GAS ADS in two steps. First, we learn a flexible neural energy function with unique-minimum, positive-definite, and continuously differentiable properties to serve as a stability certificate. Then, the stable ADS is learned with the guidance of the neural energy function. The main contributions of this article are listed as follows.

1) A flexible neural energy function is designed to be the stability certificate for the ADS to be learned. Specifically, we present a polarlike space analysis approach to derive the neural-parameter constraints, which ensure the unique-minimum, positive-definite, and continuously differentiable properties of the neural energy function.
2) Two neural learning approaches are presented according to whether the feature of the neural network is manually designed. We show that the learning problem can be formulated to a convex form when the feature is manually designed.
3) A GAS ADS learning approach is presented based on the learned neural energy function. Unlike existing ADS learning approaches, we show that the learned ADS can handle position constraints due to the special properties of the neural energy function.
4) We further quantitatively analyze the generalization ability of the learned ADS, i.e., the behaviors of the ADS in areas away from the demonstration set. We show that due to the strong flexibility of the neural energy function, the learned ADS can behave reasonably in these unfamiliar areas.

Validations on the LASA dataset and real robotic experiments are conducted to show the effectiveness of the proposed approach (PA). We also provide a video for intuitive illustrations. Moreover, the source code of the PA is also provided as the supplementary material.

II. Problem Statement

In this work, we focus on learning robot goal-directed motion skills. Goal-directed motions refer to motions with static goal positions. Most robotic tasks can be formulated as a goal-directed motion or a combination of different goal-directed motions. For example, Fig. 1 shows that a robotic picking and placing task can

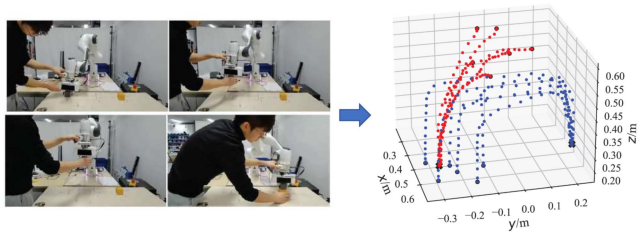


Fig. 1. Picking and placing task that can be completed by combining two goal-directed motions.

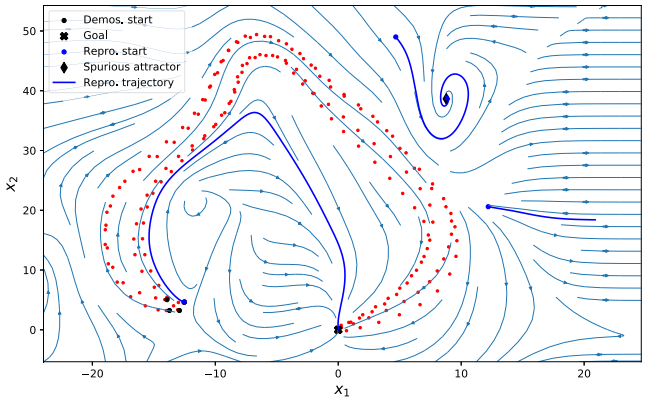


Fig. 2. Original ADS learned on a 2-D demonstration set.

be completed by combining two types of goal-directed motions distinguished by the red and blue colors. This work aims to encode the goal-directed motion as an ADS, denoted as $\dot{x} = g(x)$ where x is the robot position. After obtaining such an ADS, we can reproduce a trajectory for any given initial position by integrating the velocity $\dot{x}$. To be consistent with the goal-directed motion, we must ensure that for any initial position, the trajectory reproduced by the ADS $\dot{x} = g(x)$ will always converge to the goal position x^*. From a control perspective, it means that the ADS $\dot{x} = g(x)$ needs to be GAS.

For each goal-directed motion, we can collect a demonstration set $D = \{x_{t,n}, \dot{x}_{t,n}\}_{t=1,n=1}^{T_n,N}$, where $x_{t,n}$ is the robot position at time step t of nth demonstration trajectory, N is the number of the demonstration trajectories, and T_n is the length of the nth demonstration trajectory. Using the demonstration set D, it is easy to learn an original ADS (O-ADS) $\dot{x} = o(x)$ using any regression algorithm. However, the learned O-ADS is always not GAS. A 2-D example is shown in Fig. 2, where the demonstration set is represented by red circles, and blues lines represent the reproduction trajectories. We can find that when the initial position is away from the set D, there might raise unexpected reproduction trajectories, such as trajectories converging to a spurious attractor or even being divergent. Moreover, even if the initial position is near the set D, we still cannot ensure that the reproduction trajectory will converge to the goal position, although Fig. 2 gives a positive result.

In general, it is necessary to ensure the GAS property of the learned ADS for the safety concern. There are basically two approaches to learn a GAS ADS $\dot{x} = g(x)$ utilizing the

Lyapunov stability theorems. The first one follows the control Lyapunov approach to add a corrected term u into the O-ADS, i.e., $\dot{x} = g(x) = o(x) + u$, and the corrected term u is online computed by solving the convex optimization problem

$$\min_{u} u^T u \tag{1}$$

subject to

$$\left(\frac{\partial V(x)}{\partial x}\right)^T (o(x) + u) \leq -\rho(x) \tag{2}$$

where $V(x)$ and $\rho(x)$ are positive-definite functions. Actually, the function $V(x)$ is an energy function, which is referred to as the LF, and the constraint (2) implies that the energy along the ADS $\dot{x} = g(x)$ always decreases. Hence, the trajectory generated by the ADS $\dot{x} = g(x)$ will converge to a minimum of the energy function $V(x)$.

The second approach directly estimates a stable ADS $\dot{x} = g(x)$ without using the O-ADS, which can be formulated as the following optimization problem:

$$\min_{\Theta} J(\Theta) = \sum_{n=1}^{N} \sum_{t=1}^{T_n} \|g(x_{t,n}, \Theta) - \dot{x}_{t,n}\|_2^2 \tag{3}$$

subject to

$$\left(\frac{\partial V(x)}{\partial x}\right)^T g(x, \Theta) \leq -\rho(x) \quad \forall x \in R^{d_x} \tag{4}$$

where Θ is the learnable parameter. Different from the optimization problem described in (1) and (2), the optimization problem described in (3) and (4) must be solved offline. Since the form (4) contains infinite numbers of constraints, the optimization problem described in (3) and (4) is generally intractable. Existing works handled this issue by utilizing the special structure of the ADS function $g(x)$ to transform the infinite constraints into finite Θ-related constraints. A representative approach is the SEDS presented in [32], where the special structure of the GMM/GMR is utilized.

After checking the forms of these two approaches, we can find that they both introduce an energy-function-related constraint (2) or (4) to ensure the convergence property. These constraints imply that the trajectory reproduced by the learned ADS $\dot{x} = g(x)$ must move from a high-energy-value area to a low-energy-value area. Thus, different energy functions $V(x)$ will lead to totally different qualities of the learned ADS, whether with respect to the reproduction accuracy or the convergence property. A 2-D example is shown in Fig. 3, where the left figure shows the reproduction result when a quadratic energy function is used. As we can see, the reproduction trajectory converges to the goal position well. However, a large number of demonstration points violate the energy-function-related constraint, which are highlighted by the blue crosses. Thus, a low reproduction accuracy will be finally obtained, which is reflected by the fact that the shapes of the reproduction and demonstration trajectories are totally different. The right figure of Fig. 3 shows the reproduction result using a data-driven energy function. We can see that the data-driven energy function can well capture the preferences of the demonstrations, which raises

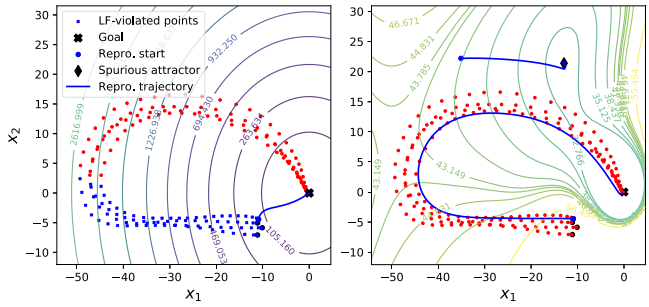


Fig. 3. Different energy function $V(x)$s and their corresponding reproduction results. The left and right figures, respectively, show the results of a simple quadratic energy function and a complex data-driven energy function.

a high reproduction accuracy. However, if the data-driven energy function has some local minima, some spurious attractors will be introduced into the ADS since the trajectory cannot leave from these local minima due to the energy-function-related constraint. An example is shown in the right figure of Fig. 3, where a reproduction trajectory gets stuck in a local minimum. Since we cannot predict the positions of these local minima, directly using a data-driven energy function could raise safety problems in real robotic tasks.

From the above discussion, a natural way to ensure both accuracy and GAS (convergence) properties of the learned ADS is to learn a data-driven energy function $V(x)$ with a unique minimum located at the goal point. Let us imagine that if we have obtained such an energy function $V(x)$, we can naturally solve the intractable optimization problem described in (3) and (4) by setting $g(x, \Theta) = -\frac{\partial V(x)}{\partial x}$. Thus, we state that learning a data-driven energy function with a unique minimum is the critical issue for both the two ADS learning approaches mentioned previously.

However, it is not easy to design a structure for the energy function $V(x)$ if we want to simultaneously hold the unique-minimum property and the flexibility property. An extreme case is if the energy function structure is designed as a quadratic function, i.e., $V(x) = x^T P x$ where P is a positive-definite matrix learned on the demonstration set, the unique-minimum property can be easily ensured while the flexibility will be severely weakened. In this case, the learned energy function could not capture the demonstration preferences well, especially when the demonstration trajectories are too complex to be consistent with a quadratic function (see the left figure of Fig. 3). Another extreme case is that the energy function $V(x)$ is directly represented by a deep neural network, which has high flexibility but may have various unpredictable local minima (see the right figure of Fig. 3).

Compared with the flexibility, the unique-minimum property of the energy function is obviously more important since it is related to the convergence (safety) issue. Thus, one of the main problems considered in this article is learning an energy function $V(x)$ with a unique minimum while increasing its flexibility as much as possible. Specifically, we represent the energy function $V(x)$ as a neural network and derive neural parameter constraints to ensure its unique-minimum

property. Another part of the works in this article is to learn a GAS ADS with the learned neural energy function as the stability certificate. Different from existing works, utilizing some special properties of the designed neural energy function, we can add a position constraint to the learned ADS, which might be valuable for some robotic tasks. Finally, we further quantitatively analyze the generalization ability of the learned ADS by utilizing the flexibility of the neural energy function.

III. Learning a Neural Energy Function With a Unique Minimum

Given a demonstration set $D = \{x_{t,n}, \dot{x}_{t,n}\}_{t=1,n=1}^{t=T_n,N}$, we present the following neural energy function to capture demonstration preferences:

$$\begin{cases} V(x) = V_1(x) - V_1(0) + V_2(x) \\ V_1(x) = \omega^T f(x) \\ f(x) = \left[f_1(x), \dots, f_k(x), \dots, f_{d_H}(x)\right]^T \\ f_k(x) = \varrho(a_k^T z(x) + b_k) \\ z(x) = \left[\|x\|_2^{1+\epsilon} \quad \|x\|_2^{\epsilon} x^T\right]^T \\ V_2(x) = \alpha x^T x \end{cases} \tag{5}$$

where $V_1(x)$ is represented by a learnable neural network with weight parameter $\omega \in R^{d_H}$ and feature $f(x) : R^{d_x} \to R^{d_H}$. The activation function $\varrho(s)$ is chosen to be the well-known "tanh" function, i.e., $\varrho(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$. Function $z(x) : R^{d_x} \to R^{d_x+1}$ is a manually designed encoder, $a_k \in R^{d_x+1}$ and $b_k \in R$ are feature parameters, $\epsilon, \alpha \in R_{++}$ are positive scalars. Function $V_2(x)$ is used to ensure the radially unbounded property of the $V(x)$.

In this section, we will introduce an approach to learn the energy function $V(x)$ such that $V(x)$ only has a unique minimum located at the origin.

A. Deriving Constraints for the Unique-Minimum Property

In this section, we will derive constraints for neural parameters $\{a_k, b_k, \omega\}_{k=1,\dots,d_H}$ such that the energy function $V(x)$ does not have any minimum except for the origin. Instead of deriving constraints in the Cartesian space, we focus on a polarlike space, which can simplify the deriving process. We give the following proposition which enables us to convert any Cartesian space vector into a polarlike space vector.

Proposition 1: For any Cartesian space vector $x \in R^{d_x}, d_x > 1$, there must exist an angle vector $\theta \in R^{d_x-1}$ such that the following equation holds:

$$x = rh(\theta) \tag{6}$$

where $r = \|x\|_2$ is the length of x, and $h(\theta) \in R^{d_x}$ is a direction vector defined as

$$h(\theta) = \begin{bmatrix} \prod_{i=1}^{d_x-1} \sin\theta_i \\ \prod_{i=2}^{d_x-1} \sin\theta_i \cos\theta_1 \\ \prod_{i=3}^{d_x-1} \sin\theta_i \cos\theta_2 \\ \vdots \\ \sin\theta_{d_x-1}\cos\theta_{d_x-2} \\ \cos\theta_{d_x-1} \end{bmatrix}. \tag{7}$$

Proof: We will prove the result by the mathematical induction approach. First, it is obvious that the result is correct in the case $d_x = 2$. The resulted space (r, θ) is the well-known Polar-coordinates space. Then, we assume that the result is further correct for the case $d_x = k-1, k > 3$. What remains is to show that the result also holds for the case $d_x = k$. Since the result is correct for the case $d_x = k-1$, we have

$$x_{1:(k-1)} = r_{k-1} h_{k-1}(\theta) = r_{k-1} \begin{bmatrix} \prod_{i=1}^{k-2} \sin\theta_i \\ \prod_{i=2}^{k-2} \sin\theta_i \cos\theta_1 \\ \prod_{i=3}^{k-2} \sin\theta_i \cos\theta_2 \\ \vdots \\ \sin\theta_{k-2}\cos\theta_{k-3} \\ \cos\theta_{k-2} \end{bmatrix} \tag{8}$$

where $x_{1:(k-1)}$ represents the vector constructed by the first $k-1$ components of x, and $r_{k-1} = \|x_{1:(k-1)}\|_2$. Then, by defining the kth component of x as

$$x_k = r\cos\theta_{k-1} \tag{9}$$

we have

$$r_{k-1} = \sqrt{r^2 - x_k^2} = r\sqrt{(\sin\theta_{k-1})^2}. \tag{10}$$

Note that there must exist a $\theta_{k-1} \in [0, \pi]$ to make (9) hold; thus, we have

$$r_{k-1} = r\sin\theta_{k-1}. \tag{11}$$

Substituting (11) into (8), and combining (9), we have

$$x = rh(\theta) \tag{12}$$

for the case $d_x = k$. ■

Given Proposition 1, we can state that a Cartesian space vector x must have a corresponding polarlike space vector (r, θ). The conversion from the Cartesian space to the polarlike space actually decouples the length and direction components of the Cartesian space vector x. Then, we can rewrite the energy function $V(x)$ in the polarlike space as

$$\begin{cases} V_P(r,\theta) = V_{1,P}(r,\theta) - V_{1,P}(0,\theta) + V_{2,P}(r,\theta) \\ V_{1,P}(r,\theta) = \omega^T f_P(r,\theta) \\ f_P(r,\theta) = \left[f_{1,P}(r,\theta), \dots, f_{k,P}(r,\theta), \dots, f_{d_H,P}(r,\theta)\right]^T \\ f_{k,P}(r,\theta) = \varrho(a_k^T z_P(r,\theta) + b_k) \\ z_P(r,\theta) = r^{1+\epsilon}\left[1 \quad h(\theta)^T\right]^T = r^{1+\epsilon}\overline{z}_P(\theta) \\ V_{2,P}(r,\theta) = \alpha r^2 \end{cases} \tag{13}$$

where the subscript "P" refers to the "polarlike space."

Proposition 2: The energy function $V(x)$ satisfies the condition $\frac{\partial V(x)}{\partial x} \neq 0, \forall x \neq 0$ if its polarlike space form $V_P(r,\theta)$ satisfies the condition $\frac{\partial V_P(r,\theta)}{\partial r} \neq 0, \forall r \neq 0, \forall \theta$.

Proof: By using the chain rule, we have

$$\begin{cases} \frac{\partial V(x)}{\partial x} = \frac{\partial V_P(r,\theta)}{\partial r}\frac{\partial r}{\partial x} + \left(\frac{\partial h(\theta)}{\partial x}\right)^T \frac{\partial V_P(r,\theta)}{\partial h(\theta)} \\ \frac{\partial r}{\partial x} = \frac{\partial \|x\|_2}{\partial x} = \frac{x}{\|x\|_2} = h(\theta) \\ \frac{\partial h(\theta)}{\partial x} = \frac{1}{\|x\|_2} I - \frac{xx^T}{\|x\|_2^3}. \end{cases} \tag{14}$$

Note that if $x \neq 0$, we have

$$\frac{\partial h(\theta)}{\partial x}\frac{\partial r}{\partial x} = \left(\frac{1}{\|x\|_2} I - \frac{xx^T}{\|x\|_2^3}\right)\frac{x}{\|x\|_2} = 0. \tag{15}$$

Thus, vector $\frac{\partial r}{\partial x}$ is in the null space of the matrix $\frac{\partial h(\theta)}{\partial x}$ for any $x \neq 0$, which implies that $\frac{\partial V(x)}{\partial x} = 0$ if and only if $\frac{\partial V_P(r,\theta)}{\partial r}\frac{\partial r}{\partial x} = 0$ and $(\frac{\partial h(\theta)}{\partial x})^T \frac{\partial V_P(r,\theta)}{\partial h(\theta)} = 0$.

Thus, if we can ensure that

$$\frac{\partial V_P(r,\theta)}{\partial r}\frac{\partial r}{\partial x} = \frac{\partial V_P(r,\theta)}{\partial r} h(\theta) \neq 0 \tag{16}$$

we can obtain $\frac{\partial V(x)}{\partial x} \neq 0$. Since

$$\left\| \frac{\partial r}{\partial x} \right\|_2 = \|h(\theta)\|_2 = 1 \quad \forall x \neq 0 \tag{17}$$

we can further state that $\frac{\partial V_P(r,\theta)}{\partial r}\frac{\partial r}{\partial x} \neq 0$ if and only if $\frac{\partial V_P(r,\theta)}{\partial r} \neq 0$, $\forall x \neq 0$. Since the case $\forall x \neq 0$ is equal to the case $\forall r \neq 0, \forall \theta$. ■

There is an intuitive explanation in the Cartesian space for the result proposed in Proposition 2. The condition $\frac{\partial V_P(r,\theta)}{\partial r} \neq 0, \forall r \neq 0, \forall \theta$ implies that the gradient along any ray from the origin (exclude the origin since $r \neq 0$) will not vanish, which raises the result $\frac{\partial V(x)}{\partial x} \neq 0, x \neq 0$.

Proposition 2 allows us to only consider the gradient with respect to the length part r, which is simpler since $\frac{\partial V_P(r,\theta)}{\partial r}$ is just a scalar. In what follows, we can give the kernel result of this work, i.e., deriving constraints of the neural parameters to ensure that the energy function $V(x)$ only has a unique minimum located at the origin.

Proposition 3: The energy function $V(x)$ represented by (5) has a unique minimum value $V(0)$ if neural parameters satisfy the constraints

$$\begin{cases} a_{k,1} > 0 \\ a_{k,1}^2 - \sum_{i=2}^{d_x+1} a_{k,i}^2 > 0 \quad \forall k \in [1, \ldots, d_H] \\ \omega_k > 0 \end{cases} \tag{18}$$

where $a_{k,i}$ is the ith component of the vector a_k, and ω_k is the kth component of the vector ω.

Proof: From the result in Proposition 2, we know that if $\frac{\partial V_P(r,\theta)}{\partial r} \neq 0, \forall r \neq 0, \forall \theta$ holds, we will have $\frac{\partial V(x)}{\partial x} \neq 0, \forall x \neq 0$. We can expand $V_P(r,\theta)$ in (13) as

$$V_P(r,\theta) = \sum_{k=1}^{d_H} \omega_k f_{k,P}(r,\theta) + \alpha r^2 - V_{1,P}(0,\theta). \tag{19}$$

Then, we can compute the gradient $\frac{\partial V_P(r,\theta)}{\partial r}$ as follows:

$$\begin{cases} \frac{\partial V_P(r,\theta)}{\partial r} = \sum_{k=1}^{d_H} \omega_k \frac{\partial f_{k,P}(r,\theta)}{\partial r} + 2\alpha r \\ \frac{\partial f_{k,P}(r,\theta)}{\partial r} = r^{\epsilon}(1+\epsilon)(1-\varrho_k^2) a_k^T \overline{z}_P(\theta) \end{cases} \tag{20}$$

where ϱ_k is the abbreviation for the function $\varrho(a_k^T z_P(r,\theta) + b_k)$. Observing the form of $\frac{\partial f_{k,P}(r,\theta)}{\partial r}$ in (20) we can find that if $a_k^T \overline{z}_P(\theta) > 0, \forall \theta$ holds, we will have $\frac{\partial f_{k,P}(r,\theta)}{\partial r} > 0$ since $1 - \varrho_k^2 > 0$ always holds due to the specific form of the "tanh" activation function.

We then scale the term $a_k^T \overline{z}_P(\theta)$ as follows:

$$a_k^T \overline{z}_P(\theta) = a_{k,1} + a_{k,2:}^T h(\theta) \geq a_{k,1} - \|a_{k,2:}\|_2 \tag{21}$$

where we utilize the property $\|h(\theta)\|_2 = 1$, and $a_{k,2:} \in R^{d_x}$ represents the vector constructed by the last d_x components of the vector a_k. Thus, if the first and second constraints in (18) hold, we have $\frac{\partial f_{k,P}(r,\theta)}{\partial r} > 0$. Then, if $\omega_k > 0$, we can state that

$$\frac{\partial V_P(r,\theta)}{\partial r} > 0 \quad \forall r \neq 0, \forall \theta \tag{22}$$

since $\alpha \in R_{++}$. Thus, $V(x)$ does not have any minimum in the area $x \neq 0$. Moreover, according to (20), we can easily find that $\frac{\partial V_P(r,\theta)}{\partial r}|_{r=0} = 0, \forall \theta$. Combining the fact that the case $r = 0, \forall \theta$ in the polarlike space is the same as the case $x \neq 0$ in the Cartesian space, we can state that $V(0)$ is the unique minimum value of the energy function $V(x)$. ■

To be an effective and smooth energy function, $V(x)$ also needs to be positive-definite and continuously differentiable. Moreover, to be a proper guidance for learning a stable ADS, the gradient of $V(x)$ at the origin should vanish, i.e., $\frac{\partial V(x)}{\partial x}|_{x=0} = 0$. We show that if constraints (18) hold, these properties can be ensured.

Proposition 4: The energy function $V(x)$ represented by (5) is positive-definite, radially unbounded, continuously differentiable, and has the property $\frac{\partial V(x)}{\partial x}|_{x=0} = 0$ if constraints (18) hold.

Proof: In the proof of Proposition 3, we have shown that if constraints (18) hold, $V(0)$ is the unique minimum value, and $\frac{\partial V_P(r,\theta)}{\partial r} > 0, \forall r \neq 0, \forall \theta$ holds. Thus, we have $V(x) > V(0) = 0$ for $\forall x \neq 0$, i.e., the energy function $V(x)$ is positive definite. Moreover, $V(x)$ is also radially unbounded due to the specific form of $V_2(x)$.

To show the differentiability of $V(x)$, we can directly write the gradient $\frac{\partial V(x)}{\partial x}$ as follows:

$$\begin{cases} \frac{\partial V(x)}{\partial x} = \frac{\partial V_1(x)}{\partial x} + 2\alpha x \\ \frac{\partial V_1(x)}{\partial x} = \left(\frac{\partial z(x)}{\partial x}\right)^T \left(\frac{\partial f(z)}{\partial z}\right)^T \omega \\ \left(\frac{\partial f(z)}{\partial z}\right)^T = \begin{bmatrix} \cdots & (1-\varrho_k^2)\, a_k & \cdots \end{bmatrix}, k \in [1, \ldots, d_H] \\ \left(\frac{\partial z(x)}{\partial x}\right)^T = \begin{bmatrix} (1+\epsilon)\|x\|_2^{\epsilon}\frac{x}{\|x\|_2} & \epsilon\|x\|_2^{\epsilon}\frac{xx^T}{\|x\|_2^2} + \|x\|_2^{\epsilon} I \end{bmatrix}. \end{cases} \tag{23}$$

Observing the form of $\frac{\partial z(x)}{\partial x}$ in (23), it seems that $\frac{\partial V(x)}{\partial x}$ cannot be defined at $x = 0$ due to the term $\frac{x}{\|x\|_2}$. However, we can find that $\frac{\partial V(x)}{\partial x} \to 0$ when $x \to 0$ due to the existence of the

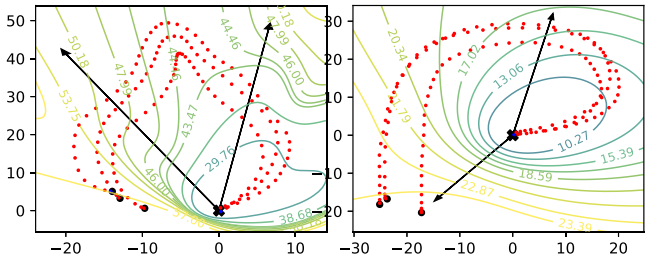


Fig. 4. Two-dimensional examples for the NEUM satisfying constraints (18).

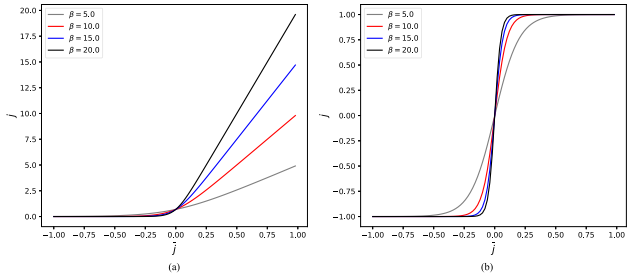


Fig. 5. Activation functions for two learning approaches.

positive scalar ϵ. Thus, we can manually set $\frac{\partial V(x)}{\partial x}|_{x=0} = 0$ without leading any mistake. As a result, $\frac{\partial V(x)}{\partial x}$ is well defined and continuous on R^{d_x}. ■

Arrive here, we can name the energy function (5) satisfying the constraints (18) as "NEUM," which refers to a neural energy function with the unique-minimum property.

In this section, we are committed to find a type of energy function $V(x)$ subject to the constraint $\frac{\partial V_P(r,\theta)}{\partial r} > 0$ for $\forall r \neq 0, \forall \theta$, where $V_P(r,\theta)$ is the polarlike space representation of $V(x)$. There is an intuitive characteristic for such kinds of functions, that is, these functions are monotonically increasing along the rays emitted from the origin. Some 2-D examples are given in Fig. 4 where contours, red points, and black arrows represent the energy function, demonstration set, and rays, respectively. As we can see the energy value always increases along the rays. From the geometrical viewpoint, any ray emitted from the origin always crosses from low-energy contours to high-energy contours. Thus, we claim that such type of energy functions is very flexible since we can arbitrarily change the shapes of contours under the premise of guaranteeing the geometrical constraint.

Another interesting point is that the well-known energy functions WSAQF [38] and ICNN [46] also fall into the investigated energy function type, which could be easily proved. However, both WSAQF and ICNN are special cases of the investigated energy function type, i.e., WSAQF is a quadraticlike function, and ICNN is a convex function. Differently, the proposed NEUM is the general form of the investigated energy function type since it has no other limitations except for the constraint $\frac{\partial V_P(r,\theta)}{\partial r} > 0$ for $\forall r \neq 0, \forall \theta$. Thus, the proposed NEUM is more flexible than the WSAQF and ICNN. This fact will also be validated on the LASA handwriting dataset. Moreover, one should also note that except for increasing the neural network's width d_H, the flexibility of the NEUM can also be enhanced by increasing the depth of the neural network, and the corresponding parameter constraints to ensure the property $\frac{\partial V_P(r,\theta)}{\partial r} > 0$ for $\forall r \neq 0, \forall \theta$ can be easily derived by using a similar polarlike space analysis approach.

B. Learning the Constrained Neural Energy Function

In this section, we develop two approaches to learn the NEUM, i.e., the energy function represented by (5), which satisfies the constraints derived in Proposition 3. In the first approach, the feature $f(x)$ is manually designed to satisfy the constraints (18). The learnable parameter only remains the weight parameter ω. In this case, the learning problem is convex by designing a proper objective function. In the second approach, both feature parameters $\{a_k, b_k\}$ and weight parameter ω are learned by a constrained learning algorithm. In this case, the learning problem is not convex, while the feature $f(x)$ can be automatically fit to the demonstration set.

The purpose is to make the NEUM $V(x)$ be consistent with the demonstration preferences, that is, the demonstration trajectories evolve from high-energy-value areas to low-energy-value areas. Mathematically, this purpose can be described as

$$\left(\frac{\partial V(x_{t,n},\Theta)}{\partial x_{t,n}}\right)^T \dot{x}_{t,n} < 0 \quad \forall (x_{t,n}, \dot{x}_{t,n}) \in D \tag{24}$$

where Θ is the learnable parameter of the NEUM. Note that, in this section, we will use $V(x,\Theta)$ instead of $V(x)$ to denote the NEUM since Θ will be regarded as a variable.

1) First Learning Approach: In this learning approach, we fix the feature parameters $\{a_k, b_k\}$ and only learn the weight parameter ω. Thus, we have $\Theta = \omega$ in this approach. We first determine the objective function of the learning problem. It seems natural to use $\overline{j}_{t,n}$ defined as

$$\overline{j}_{t,n}(\omega) = \left(\frac{\partial V(x_{t,n},\omega)}{\partial x_{t,n}}\right)^T \dot{x}_{t,n} \tag{25}$$

to evaluate the NEUM on a pair of data $(x_{t,n}, \dot{x}_{t,n})$. However, directly using $\overline{j}_{t,n}(\omega)$ as the evaluation function will introduce some misunderstandings for the learning algorithm. For example, the algorithm will try to decrease the norm of the $\frac{\partial V(x_{t,n},\omega)}{\partial x_{t,n}}$ to reduce $\overline{j}_{t,n}(\omega)$, which is obviously meaningless and may lead to a bad parameter ω.

We solve this issue by nesting an activation function on $\overline{j}_{t,n}(\omega)$, which results in the following novel evaluation function:

$$j_{t,n}(\omega) = \zeta(\overline{j}_{t,n}(\omega)) \tag{26}$$

where the activation function is defined as $\zeta(\overline{j}) = \log(e^{\beta\overline{j}} + 1)$, and $\beta \in R_{++}$ is a tunable parameter. Fig. 5(a) gives an intuitive illustration of the activation function $\zeta(\overline{j})$ and its tunable parameter β. The activation function $\zeta(\overline{j})$ with large β implies that we do not consider the magnitude of $\overline{j}_{t,n}(\omega)$ when it is negative.

By defining $J(\omega) = \sum_{t,n} j_{t,n}(\omega)$, we formulate the learning problem as follows:

$$\min_{\omega} J(\omega) \tag{27}$$

subject to

$$\epsilon_1 \leq \omega_k \leq \epsilon_2 \quad \forall k \in [1, d_H] \tag{28}$$

where ϵ_1 and ϵ_2 are any positive scalars, ω_k is the kth component of the weight parameter ω. We show that the above learning problem is convex, which means that we can search the optimal ω for the NEUM by using a convex optimization algorithm.

Proposition 5: The learning problem described in (27) and (28) is convex if energy function $V(x)$ is represented by (5).

Proof: The optimization problem is convex if and only if both the objective function (27) and constraint functions (28) are convex in the parameter ω. The convexity of the constraint functions (28) is obvious since they are linear with the parameter ω. We then show that the objective function (27) is also convex in the parameter ω.

We can expand $\bar{j}(\omega)$ as

$$\bar{j}(\omega) = \left(\omega^T \frac{\partial f(z)}{\partial z}\frac{\partial z(x)}{\partial x} + 2\alpha x^T\right)\dot{x} \tag{29}$$

which is obviously affine in ω, and hence convex in ω. Moreover, the activation function $\zeta(\bar{j})$ is convex and monotonically increasing in $\bar{j}$, we can then claim that $j = \zeta(\bar{j})$ is convex in ω according to the composition rule for preserving convexity [48]. Furthermore, the objective function $J(\omega)$ is a nonnegative weighted sum of j, thus $J(\omega)$ is also convex in ω. ■

2) Second Learning Approach: In this learning approach, both feature parameters $\{a_k, b_k\}$ and weight parameter ω will be learned. Thus, we have the learnable parameter $\Theta = \{a_k, b_k, \omega\}_{k=[1,\ldots,d_H]}$. Similar with the process in the first learning approach, we first determine the objective function of the learning problem. Since, in this approach, the learning problem will not be convex, we have more choices for the objective function. Actually, what we truly care is the angle between the vectors $\frac{\partial V(x,\Theta)}{\partial x}$ and $\dot{x}$, and thus, we design $\bar{j}$ in this learning approach as follows:

$$\bar{j}_{t,n}(\Theta) = \frac{\dot{x}^T \frac{\partial V(x_{t,n,\Theta})}{\partial x_{t,n}}}{\|\dot{x}\|_2 \|\frac{\partial V(x_{t,n,\Theta})}{\partial x_{t,n}}\|_2}. \tag{30}$$

This design eliminates the misunderstanding mentioned in the first learning approach since $\frac{\partial V(x_{t,n,\Theta})}{\partial x_{t,n}}$ is normalized in $\bar{j}$. Moreover, we can also nest an activation function on $\bar{j}$ like what we do in the first learning approach to trade off the accuracy and generalization ability of the NEUM, which results in the following evaluation function:

$$j_{t,n}(\Theta) = \zeta(\bar{j}_{t,n}(\Theta)) \tag{31}$$

where in this approach the activation function is defined as $\zeta(\bar{j}) = \tanh(\beta\bar{j}) = \frac{e^{\beta\bar{j}} - e^{-\beta\bar{j}}}{e^{\beta\bar{j}} + e^{-\beta\bar{j}}}$, and $\beta \in R_{++}$ is a tunable parameter.

Fig. 5(b) gives an intuitive illustration of the activation function and its parameter β. The larger the β is, the more we care about the sign of $\bar{j}$, which corresponds to the accuracy property. On the other hand, if β is small, the algorithm will additionally focus on the angle between the energy function gradient $\frac{\partial V(x,\Theta)}{\partial x}$ and the velocity $\dot{x}$, which relates to the generalization ability of the NEUM. Some 2-D examples illustrating this point are shown in Fig. 6, where red points and contours represent the demonstration set and the learned NEUM, respectively. As we can see, when β takes a large value, i.e., $\beta = 10.0$ in Fig. 6(a), the energy function NEUM only considers the accuracy property, that is the demonstration trajectories always evolve from high-energy-value areas to low-energy-value areas. When β takes a small value as in Fig. 6(b), the learned NEUM additionally aligns its gradient with the demonstration velocities, which encodes the motion preferences in a broader area compared with the large β case. However, we should also note that when the demonstration trajectories are very complex, enhancing the generalization ability may sacrifice the accuracy property.

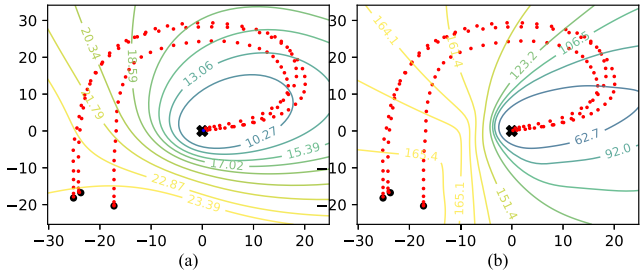


Fig. 6. Two-dimensional examples for the NEUM with different β values. (a). NEUM with $\beta = 10.0$. (b). NEUM with $\beta = 1.0$.

By defining $J(\Theta) = \sum_{t,n} j_{t,n}(\Theta) + L_2\|\Theta\|_2^2$, where $L_2\|\Theta\|_2^2$ is the L_2 regularization term, we formulate the learning problem as follows:

$$\min_{\Theta} J(\Theta) \tag{32}$$

subject to

$$\begin{cases} a_{k,1} > 0 \quad \forall k \in [1, \ldots, d_H] \\ a_{k,1}^2 - \sum_{i=2}^{d_x+1} a_{k,i}^2 > 0 \quad \forall k \in [1, \ldots, d_H] \\ \omega \succ 0. \end{cases} \tag{33}$$

IV. Learning a GAS ADS With Predefined Position Constraint

In this section, we first learn an original ADS $\dot{x} = o(x)$ with a desired equilibrium point, i.e., $o(0) = 0$, using a modified Gaussian process regression (GPR) algorithm [47]. Then, we introduce an approach to formulate a GAS ADS by combining the original ADS and the NEUM learned in the previous section. Except for considering the global stability, we further show that we can add a predefined position constraint to the learned ADS, which may be valuable for some robotic tasks. Moreover, we analyze the generalization ability of the learned ADS in areas away from the demonstration area by utilizing the flexibility of the NEUM.

A. Learning GPR-Based Original ADS With the Desired Equilibrium Point

Given the demonstration set $D = \{x_{t,n}, \dot{x}_{t,n}\}_{t=1,n=1}^{t=T_n,N}$ with size $M = \sum_{n=1}^{N} T_n$, we assume that the observations $\dot{x}_{t,n}$ are generated from a disturbed ADS as follows:

$$\dot{x}_{t,n} = g_T(x) + \phi_{t,n} \tag{34}$$

where $\dot{x} = g_T(x)$ is the true ADS assumed to generate the set, and $\phi \in R^{d_x}$ is the noise signal satisfying the Gaussian distribution $\mathcal{N}(0, \sigma_{no}^2 I)$ with variance $\sigma_{no}^2 I$. The GPR algorithm is utilized to estimate this true ADS. Since the GPR algorithm can only handle 1-D fitting problem, we use d_x GPR systems to estimate the whole true ADS $\dot{x} = g_T(x)$, and the ith, $i = 1, \ldots, d_x$, system is denoted as $\dot{x}_i = g_{T,i}(x)$. The GPR algorithm assumes that the function $g_{T,i}(x)$ follows a GP process as $g_{T,i}(x) \sim \mathcal{GP}(m_i(x), k_i(x, x'))$ where $m_i(x)$ and $k_i(x, x')$ are mean and kernel functions, respectively. In this article, they are defined as

$$\begin{cases} m_i(x) = 0 \\ k_i(x, x') = \sigma_{ker,i}^2 exp\left(-\frac{1}{2}(x - x')^T \Sigma_i^{-1}(x - x')\right) \end{cases} \tag{35}$$

where $\sigma_{ker,i}^2$ and $\Sigma_i \in R_{++}^{d_x \times d_x}$ are hyperparameters, which could be learned by maximizing the log-likelihood on the demonstration set D [47]. Given a query input x^*, the output $\dot{x}_i^* = g_{T,i}(x^*)$ and outputs in the demonstration set D are assumed to follow a multivariate Gaussian distribution

$$\begin{bmatrix} \dot{x}_i^* \\ \vec{x}_i \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_i^* \\ \vec{m}_i \end{bmatrix}, \begin{bmatrix} k_i(x^*, x^*) & \vec{k}_i^T(x^*) \\ \vec{k}_i(x^*) & K_i + \sigma_{no}^2 I \end{bmatrix} \right) \tag{36}$$

where $m_i^* = m_i(x^*)$, $\vec{x}_i \in R^M$ is a vector constructed by stacking ith component of demonstration outputs, $\vec{k}_i(x^*) \in R^M$ is a vector of which the jth component is $k_i(x^*, x_j)$ and x_j is the jth input of the demonstration set, $\vec{m}_i \in R^M$ is a vector of which the jth component is $m_i(x_j)$, and $K_i \in R_+^{M \times M}$ is a semipositive-definite matrix with $K_{i,ab} = k_i(x_a, x_b)$. Then, the posterior distribution $p_i(\dot{x}_i^* | x^*, D)$ will follow a Gaussian distribution $\mathcal{N}(\mu_i(x^*), \sigma_{\text{pos},i}(x^*)^2)$ where

$$\begin{cases} \mu_i(x^*) = m_i(x^*) + \vec{k}_i^T(x^*)(K_i + \sigma_{no}^2 I)^{-1}(\vec{x}_i - \vec{m}_i) \\ \sigma_{\text{pos},i}(x^*)^2 = k_i(x^*, x^*) - \vec{k}_i^T(x^*)(K_i + \sigma_{no}^2 I)^{-1}\vec{k}_i^T(x^*). \end{cases} \tag{37}$$

The estimation of the true ADS can be set as $\dot{x} = \mu(x) = \left[\mu_1(x), \ldots, \mu_{d_x}(x)\right]^T$, which can be used as the original ADS. However, we cannot ensure that the learned original ADS will have a desired equilibrium point. This issue can be solved by introducing a determined input–output pair $(0, 0)$ into the demonstration set, and we denote the novel demonstration set as $\overline{D}$. We can then rewrite (36) as

$$\begin{bmatrix} \dot{x}_i^* \\ \vec{x}_i \\ 0 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_i^* \\ \vec{m}_i \\ 0 \end{bmatrix}, \begin{bmatrix} k_i(x^*, x^*) & \vec{k}_i^T(x^*) & k_i(x^*, 0) \\ \vec{k}_i(x^*) & K_i + \sigma_{no}^2 I & \vec{k}_i(0) \\ k_i(0, x^*) & \vec{k}_i^T(0) & k_i(0, 0) \end{bmatrix} \right) \tag{38}$$

and compute the novel mean function of the posterior as

$$\mu_i'(x^*) = m_i(x^*) + \vec{k}_i'^T(x^*) K_i'^{-1} (\vec{x}_i' - \vec{m}_i') \tag{39}$$

where

$$\begin{cases} \vec{k}_i'^T(x^*) = \begin{bmatrix} \vec{k}_i^T(x^*) & k_i(x^*, 0) \end{bmatrix} \\ K_i' = \begin{bmatrix} K_i + \sigma_{no}^2 I & \vec{k}_i(0) \\ \vec{k}_i^T(0) & k_i(0, 0) \end{bmatrix} \\ \vec{x}_i'^T = \begin{bmatrix} \vec{x}_i^T & 0 \end{bmatrix}, \vec{m}_i'^T = \begin{bmatrix} \vec{m}_i^T & 0 \end{bmatrix}. \end{cases} \tag{40}$$

We can then formulate the original ADS as $\dot{x} = o(x) = \left[\mu_1'(x), \ldots, \mu_{d_x}'(x)\right]^T$. We will show that the original ADS has the desired equilibrium point, i.e., $o(0) = 0$.

Proposition 6: The ADS $\dot{x} = o(x) = \left[\mu_1'(x), \ldots, \mu_{d_x}'(x)\right]^T$, where $\mu_i'(x)$ follows the form (39), satisfies $o(0) = 0$ if the mean and kernel functions of the GPR are designed as (35).

Proof: We first show that the matrix K_i' in (39) is invertible if the kernel function is designed as (35). For any vector $s \in R^{M+1}$, we have

$$s^T K_i' s \geq \sigma_{no}^2 \|s_{1:M}\|_2^2 \geq 0 \tag{41}$$

where $s_{1:M}$ is the vector constructed by the first M components of s. The first inequation is obtained from the positiveness of the kernel function. We then show that the above two equations cannot simultaneously hold if $s \neq 0$. The second equation holds implies that $s_{1:M} = 0$. If $s \neq 0$, we must have $s_{M+1} \neq 0$. However, in this case, the first equation cannot hold since $s_{M+1}^2 k_i(0, 0) > 0$ if the kernel function is designed as (35). Thus, we claim that if the kernel function is designed as (35), K_i' must be invertible.

Then, if K_i' is invertible, we can obtain

$$\begin{cases} det|K_i'| = det|K_i + \sigma_{no}^2 I| \delta \neq 0 \\ \delta = k_i(0, 0) - \vec{k}_i(0)^T (K_i + \sigma_{no}^2 I)^{-1} \vec{k}_i(0). \end{cases} \tag{42}$$

Thus, we have $\delta \neq 0$ since $det|K_i + \sigma_{no}^2 I| > 0$ according to the positiveness property of the kernel function (35). Then, we can analytically compute $K_i'^{-1}$ as

$$\begin{cases} K_i'^{-1} = \begin{bmatrix} \mathcal{K}_{11} & \mathcal{K}_{12} \\ \mathcal{K}_{21} & \mathcal{K}_{22} \end{bmatrix} \\ \mathcal{K}_{11} = (K_i + \sigma_{no}^2 I)^{-1} \left(I + \frac{1}{\delta} \vec{k}_i(0) \vec{k}_i^T(0) (K_i + \sigma_{no}^2 I)^{-1} \right) \\ \mathcal{K}_{12} = -\frac{1}{\delta} (K_i + \sigma_{no}^2 I)^{-1} \vec{k}_i(0) \\ \mathcal{K}_{21} = -\frac{1}{\delta} \vec{k}_i^T(0) (K_i + \sigma_{no}^2 I)^{-1} \\ \mathcal{K}_{22} = \frac{1}{\delta}. \end{cases} \tag{43}$$

Thus, $\mu_i'(0)$ can be computed as

$$\begin{aligned} \mu_i'(0) &= m_i(0) + \left(\vec{k}_i^T(0) \mathcal{K}_{11} + k_i(0, 0) \mathcal{K}_{21} \right) (\vec{x}_i - \vec{m}_i) \\ &= \underbrace{\vec{k}_i^T(0)(K_i + \sigma_{no}^2 I)^{-1}}_{\tau} \\ &\quad + \frac{1}{\delta} \underbrace{\left(\vec{k}_i^T(0)(K_i + \sigma_{no}^2 I)^{-1} \vec{k}_i(0) - k_i(0, 0) \right)}_{-\delta} \tau \\ &= \tau - \tau = 0 \end{aligned} \tag{44}$$

where the second equation is obtained by substituting (43) into

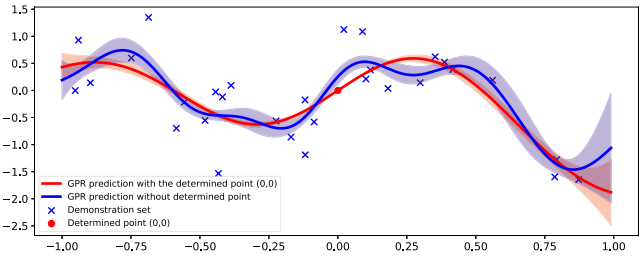


Fig. 7. 1-D example for GPR prediction with the determined point (0,0).

Moreover, it could also be shown that the novel posterior variance function $\sigma'^2_{\text{pos},i}(0) = 0$, which shows the prediction determination. A 1-D example for intuitive illustration is shown in Fig. 7, where blue crosses and red points represent the demonstration set and the determined point $(0, 0)$, respectively. Red and blue lines represent the GPR prediction results with and without the determined point, respectively. Shadow areas correspond to the 95% prediction confidence interval. As we can see, after adding the determined point, the GPR prediction result will cross it with the 100% probability.

B. Learning a GAS ADS With Predefined Position Constraint

In this section, we introduce an approach to formulate a GAS ADS with a predefined position constraint by combining the original ADS and the energy function NEUM. The GAS ADS $\dot{x} = g(x)$ can be obtained by introducing a correct term u into the original ADS as

$$g(x) = o(x) + u \tag{45}$$

where u can be obtained by online solving the following quadratic programming (QP) problem

$$\min_{u} u^T u \tag{46}$$

subject to

$$\begin{cases} (a) & (o(x)+u)^T \frac{\partial V(x)}{\partial x} \leqslant -\rho(x) \\ (b) & (o(x)+u)^T Px \leq 0, x^T Px = r^2_{\text{thres}} \end{cases} \tag{47}$$

where $V(x)$ is the NEUM learned in Section III, $\rho(x)$ is any positive-definite function, P is a positive-definite or semipositive-definite matrix, $r_{\text{thres}} \in R_+$ is a nonnegative scalar. The constraint (47a) is used to ensure the globally asymptotic stability. The left part of constraint (47b) is the time derivative of the term $\frac{1}{2}x^T Px$, and thus, it is a position constraint for the ADS. Specifically, when the condition $x^T Px = r^2_{\text{thres}}$ is activated, constraint (47b) will force a contraction velocity. As a result, when x falls into the region $R_{\text{cons}} = \{x^T Px \leqslant r^2_{\text{thres}}\}$, it will never leave from R_{cons}.

We then show that if the constraints (18) hold, the feasibility of the optimization problem described in (46) and (47) can be ensured for $\forall x \in R^{d_x}$.

Proposition 7: The optimization problem described in (46) and (47) is feasible for $\forall x \in R^{d_x}$ if constraints (18) hold.

Proof: We first consider the case $x = 0$. If constraints (18) hold, we have results $\frac{\partial V(x)}{\partial x} = 0$ and $o(x) = 0$ from Proposition 4 and Proposition 6, respectively. Thus, $u = 0$ is the optimal solution for the optimization problem described in (46) and (47).

We then consider the case $x \neq 0$. We show that there always exists a solution for the inequality system constructed by (47a) and (47b). Constraints (47 a) and (47 b) actually determine two half-spaces, and the only case at which these two half-spaces might have an empty intersection set is $\frac{\partial V(x)}{\partial x} = -Px$. However, it should be noted that

$$\begin{aligned} x^T \frac{\partial V(x)}{\partial x} &= x^T \left(\frac{\partial V_P(r,\theta)}{\partial r} \frac{\partial r}{\partial x} + \left(\frac{\partial h(\theta)}{\partial x} \right)^T \frac{\partial V_P(r,\theta)}{\partial h(\theta)} \right) \\ &= \frac{\partial V_P(r,\theta)}{\partial r} x^T \frac{\partial r}{\partial x} = \frac{\partial V_P(r,\theta)}{\partial r} \|x\|_2 \end{aligned} \tag{48}$$

where the second equation is obtained by using the fact that x is in the null space of the matrix $\frac{\partial h(\theta)}{\partial x}$. If the constraints (18) hold, we can get $\frac{\partial V_P(r,\theta)}{\partial r} > 0$, which has been proved in Proposition 3. Thus, we have $x^T \frac{\partial V(x)}{\partial x} > 0$ for $\forall x \neq 0$. On the other hand, since P is at least semipositive definite, we get $-x^T Px \leq 0$ for $\forall x \neq 0$. Thus, we have $\frac{\partial V(x)}{\partial x} \neq -Px$ for $\forall x \neq 0$. ■

In what follows, we can give the following theorem.

Theorem 1: The ADS (45) is GAS and has a region of attraction $R = \{x^T Px \leqslant r^2_{\text{thres}}\}$ if the following conditions hold.

1) The input u is obtained by online solving the optimization problem described in (46) and (47).
2) Energy function $V(x)$ is represented by (5), and its parameters satisfy the constraints (18).

Proof: The result is obvious based on Proposition 7. ■

Remark 1: It should be noted that if we only consider the stability property, i.e., only constraint (47 a) is considered, we can directly give the analytical form of the input u as

$$u = -\frac{\text{ReLu}\left(o(x)^T \frac{\partial V(x)}{\partial x} + \rho(x)\right)}{\left\| \frac{\partial V(x)}{\partial x} \right\|_2^2} \frac{\partial V(x)}{\partial x} \tag{49}$$

where the activation function $\text{ReLu}(s)$ has the form

$$\text{ReLu}(s) = \begin{cases} s, s > 0 \\ 0, \text{else}. \end{cases} \tag{50}$$

Remark 2: In real applications, the reproduction of the ADS is always achieved in a discrete manner, and thus the condition $x^T Px = r^2_{\text{thres}}$ in the constraint (47b) may not be easily activated. Thus, we soften it as $(1-\eta)r^2_{\text{thres}} <= x^T Px <= (1+\eta)r^2_{\text{thres}}$ in real scenarios.

C. Generalization Ability Analysis

Besides ensuring global stability, we further want to enhance the generalization ability of the learned ADS $\dot{x} = g(x)$. Different from the reproduction accuracy property, the generalization ability is reflected in areas away from the demonstration set. Let us first recall a special property of the GPR with the Gaussian kernel (35), i.e., the posterior mean function $\mu_i(x^*)$ of the GPR will approach to 0 when the query input x^* is away from the dataset. An intuitive 2-D example is shown in Fig. 8(a), where the lighter color indicates the smaller velocity generated by the

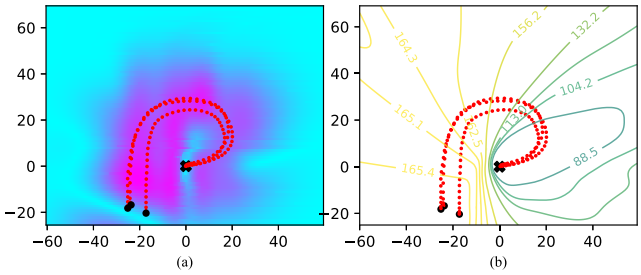


Fig. 8. Intuitive 2-D example for illustrate the generalization ability of the PA. (a) Velocity norm generated by the original ADS. (b) NEUM with $\beta = 1.0$.

original ADS. It shows that the area away from the demonstration set could correspond to an extremely low-velocity norm.

Thus, when we consider the generalization ability, we can assume that the output of $o(x)$ is 0. Then, the stable ADS can be approximated as $\dot{x} = g(x) \approx u$, and the optimization problem to compute u will be

$$\min_{u} u^T u \tag{51}$$

subject to

$$u^T \frac{\partial V(x)}{\partial x} \leqslant -\rho(x). \tag{52}$$

Note that the position constraint (47b) will not be considered in this case since the position constraint will not be activated when x is far away from the demonstration set. Thus, we can directly compute the optimal u as

$$u^* = -\frac{\rho(x)}{\left\| \frac{\partial V(x)}{\partial x} \right\|_2^2} \frac{\partial V(x)}{\partial x}. \tag{53}$$

As a result, in areas away from the demonstration set, we have

$$\dot{x} = g(x) \approx -\frac{\rho(x)}{\left\| \frac{\partial V(x)}{\partial x} \right\|_2^2} \frac{\partial V(x)}{\partial x}. \tag{54}$$

Thus, the velocity generated by the ADS will be approximately proportional to the negative gradient of the energy function NEUM. As discussed in Section III-B, when we choose a small β for the activation function $\zeta(\overline{j})$ in the second learning approach, the NEUM gradient can capture the demonstration preferences in broad areas, which is reflected by Figs. 6(b) and 8(b). Furthermore, if we design function $\rho(x)$ as

$$\rho(x) = \left\| \frac{\partial V(x)}{\partial x} \right\|_2 \overline{v}(x) \tag{55}$$

where $\overline{v}(x)$ is a manually designed function, the influence of the amplitude part of the NEUM gradient will be eliminated, which results in the following approximated ADS:

$$\dot{x} = g(x) \approx -\overline{v}(x) \frac{\frac{\partial V(x)}{\partial x}}{\| \frac{\partial V(x)}{\partial x} \|_2}. \tag{56}$$

The form of (56) implies that in areas away from the demonstration set, the direction of the velocity generated by the ADS will be approximately the same with the negative gradient of the NEUM, and the amplitude of the velocity will be a designable function $\overline{v}(x)$.

V. Validations on the LASA Dataset

In this section, we validate the effectiveness of the PA on the LASA handwriting dataset [49], and the second NEUM learning approach is used. The dataset contains various demonstration cases, which raise different difficulties for the demonstration learning. For the most complex cases with names as "Bended-Line," "DoubleBnededLine," "JShape_2," "Leaf_1," "Leaf_2," and "Snake," the hyperparameters for the NEUM are set as $d_H = 10$ and $\beta = 1.0$. For the remaining cases, we set $d_H = 2$ and $\beta = 1.0$. Functions $\rho(x)$ in (47) is given by

$$\begin{cases} \rho(x) = \left\| \frac{\partial V(x)}{\partial x} \right\|_2 \overline{v}(x) \\ \overline{v}(x) = \frac{v_M}{\gamma} \left(1 - exp\left\{ -\frac{1}{2x_M^2} x^T x \right\} \right) \end{cases} \tag{57}$$

where x_M and v_M are the maximum position and velocity norms in the demonstration set D, respectively. $\gamma \in R_{++}$ is a tunable parameter to determine the convergence rate.

Learning results for the NEUM and the corresponding ADS reproduction results are shown in Fig. 9, where the left and right blocks show the NEUM learning results and ADS reproduction results, respectively. The red points and blue crosses represent the demonstration points satisfying and violating the energy-function-related constraint, respectively. Large black points and crosses are used to, respectively, indicate the initial and goal positions. In this part, the position constraint is not used for the accurate reproduction consideration, that is, r_{thres} in (47b) is set as 0. We use the "trust-constr" optimization algorithm for NEUM learning based on the "scipy.minimize" package with python. The QP problem described in (46) and (47) is solved using the "cvxopt" package.

Let us first observe the NEUM learning results. As expected, the learned NEUMs for all demonstration cases satisfy the unique-minimum property. More specifically, they satisfy the geometrical constraint mentioned in Section III-A, i.e., the energy function value monotonically increases along any rays emitted from the origin. Moreover, we can find that the learned NEUMs can accurately capture the demonstration preferences for all demonstration cases, even though we set $\beta = 1.0$ to highlight the importance of the generalization ability. These results validate that the NEUM structure is very flexible, which allows the NEUM to simultaneously consider the accuracy and generalization properties for most cases.

To give more intuitive comparisons between the NEUM and other existing data-driven energy functions, we propose the following two metrics:

$$\begin{cases} E_1 = 1 + \frac{1}{T} \sum_{t=0}^{T} j_t(\Theta) \\ E_2 = 1 + \frac{1}{T} \sum_{t=0}^{T} \overline{j}_t(\Theta) \end{cases} \tag{58}$$

where T is the length for a demonstration trajectory, $\overline{j}_t(\Theta)$ and $j_t(\Theta)$ are defined as (30) and (31), respectively. We set $\beta = 10.0$

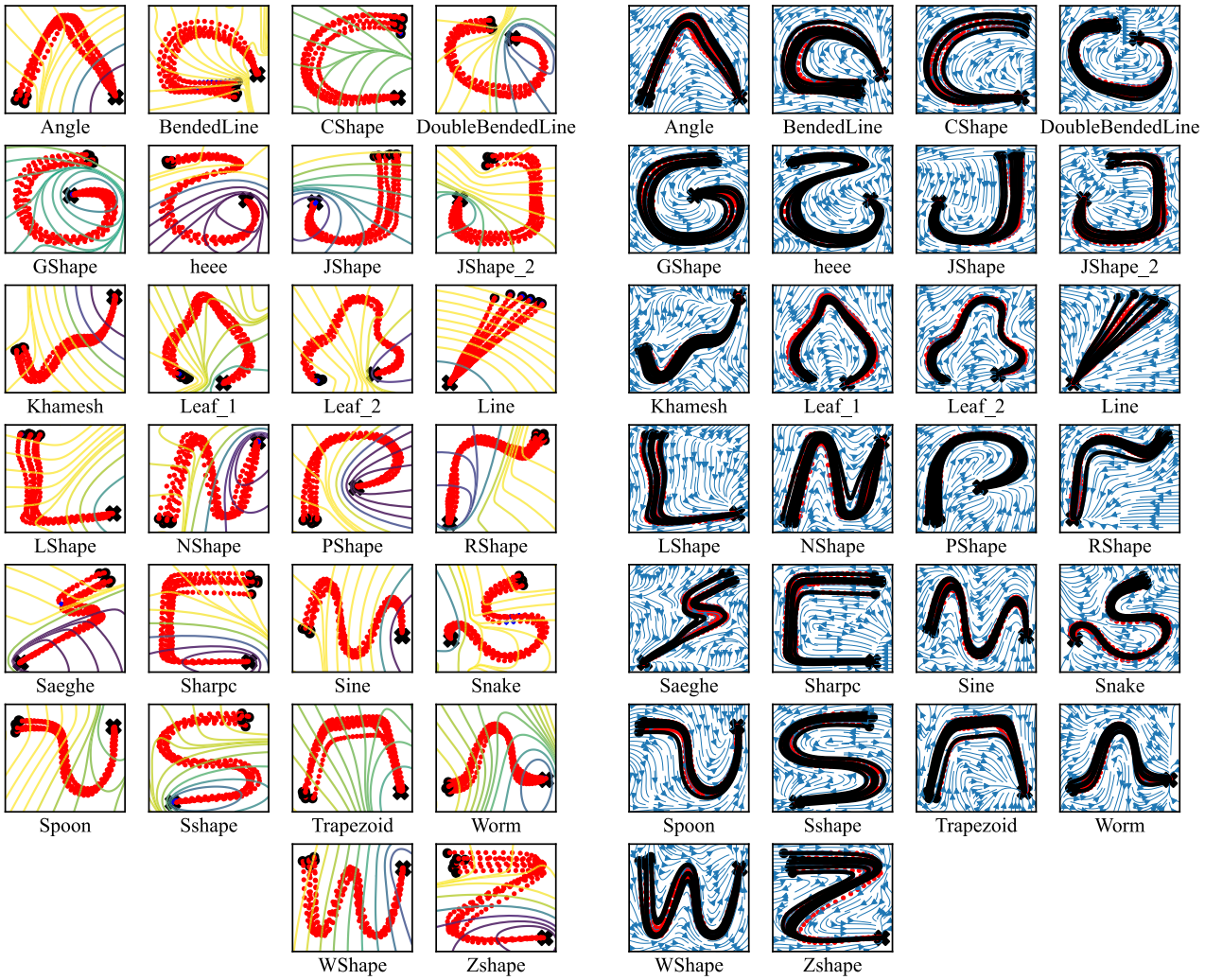


Fig. 9. Learned NEUM and reproduction results for the LASA handwriting dataset [49]. The left block shows the NEUM learning results for the LASA dataset, where red point and blue crosses represent the demonstration points that satisfy and violate the NEUM, respectively. The right block shows the reproduction results, which are represented by the black lines.

for $j_t(\Theta)$, and thus, E_1 is used to measure the accuracy of an energy function, and E_2 additionally measures the generalization ability, i.e., the alignment between the energy function gradient and the demonstration velocity. The comparison results on the six most complex cases are shown in Fig. 10, where the upper and bottom figures, respectively, show the comparison results with respect to the metrics E_1 and E_2. Hyperparameter settings for the compared data-driven energy functions are determined according to their article descriptions. As we can see, the proposed NEUM outperforms all of the existing data-driven energy functions, either with respect to the metric E_1 or E_2. Moreover, we want to highlight that compared with another two neural energy functions NILC and ICNN, the NEUM proposed in this work is much more lightweight. Specifically, as guided in [43] and [46], the NILC and ICNN are, respectively, designed as a three-layer neural network with 100 hidden-layer units, and a four-layer neural network with 100 units for each hidden layer. Comparatively, the neural network in NEUM is just three-layer with 10 hidden-layer units. Thus, we conclude that the NEUM is the most flexible in these existing data-driven energy functions.

Detailed learning results for these data-driven energy functions are shown in Fig. 11, where red points and blue crosses represent demonstration points that satisfy and violate the energy function, respectively. Obviously, we can find that the NEUM is more consistent with these demonstration cases compared with other data-driven energy functions. Moreover, compared with WSAQF and ICNN, which simultaneously have the theoretically unique-minimum guarantee and considerable consistence performance, the NEUM is more flexible, which is reflected by the diversity of contour shapes and the size of the area affected by demonstrations.

Remark 3: Note that for conducting fair comparisons, we use same objective functions for the NEUM, WSAQF, NILC, and ICNN to generate Figs. 10 and 11. The energy function labeled as FDM is obtained by the FDM as mentioned in [40].

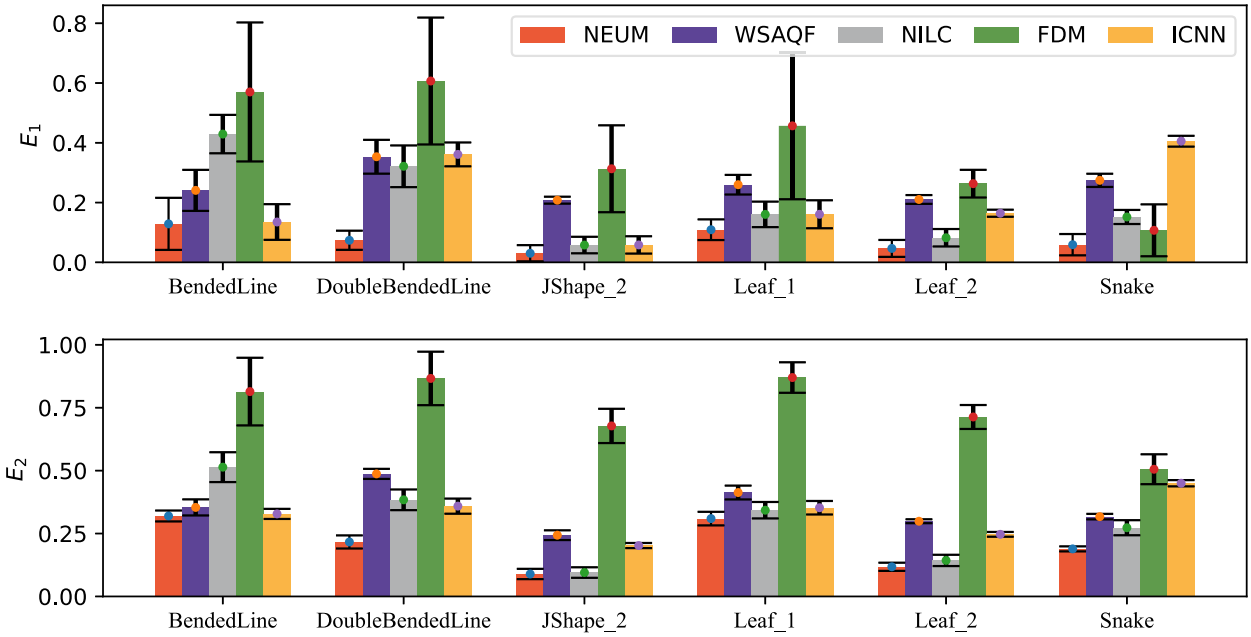


Fig. 10. Quantitative comparison results between the proposed energy function NEUM, WSAQF [38], NILC [43], FDM [40], and ICNN [46] for the six most challenging cases in the LASA handwriting dataset using metrics E_1 and E_2. The upper and bottom figures show the comparison results with respect to the metrics E_1 and E_2 along with one standard deviation, respectively. Hyperparameter settings for the compared approaches are determined according to their article descriptions.

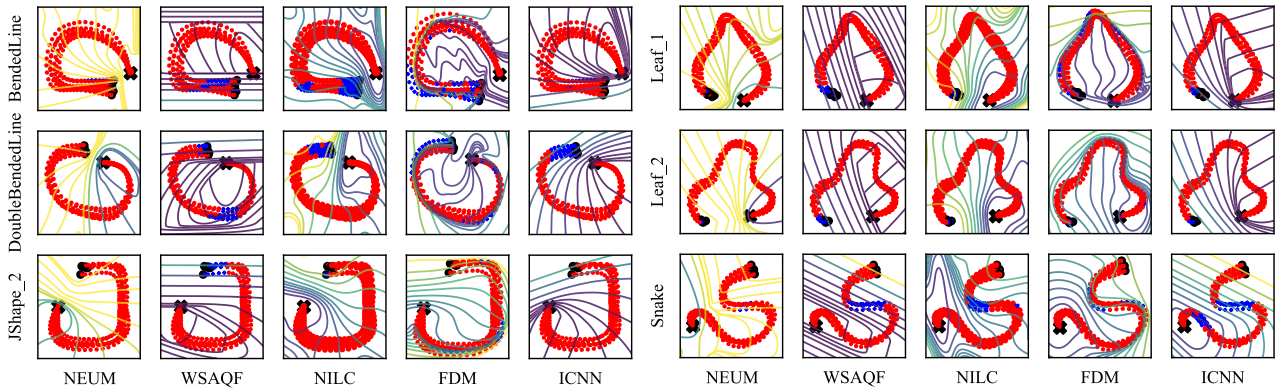


Fig. 11. Learning results of the proposed energy function NEUM, WSAQF [38], NILC [43], FDM [40], and ICNN [46] for the six most challenging cases in the LASA handwriting dataset.

We then compare the proposed demonstration learning approach based on the NEUM with other existing approaches, using the metrics swept error area (SEA) [38] and dynamic time warping distance (DTWD). Note that the DTWD only measures the dissimilarity between the spatial shapes of the demonstration and reproduction trajectories while the SEA penalizes both spatial and temporal dissimilarities [30]. The comparison results are concluded in Table I, where approaches CLF-DM [38], τ-SEDS [39], and CDSP [31] are compared with the PA. To conduct fair comparisons, the optimization algorithms for the CLF-DM, τ-SEDS, and CDSP are both chosen as the "trust-constr" algorithm wrapped in the "scipy.minimize" package. Hyperparameters for these approaches are determined by trials and errors. Moreover, we chose $\beta = 10$ for the WSAQF learning in the CLF-DM approach to highlight the reproduction accuracy.

As we can see in Table I, the PA achieves the best result in almost every case, whether with respect to the SEA or DTWD. For cases "BendedLine," "DoubleBendedLine," "JShape_2," and "Leaf_2," the performances of the PA are similar to these of the CLF-DM since in these cases, energy functions NEUM and WSAQF are both consistent with the demonstration trajectories. Especially, the metrics of the PA and CLF-DM are even the same for cases "JShape_2" and "Leaf_2" since the original ADS need not to be corrected under the energy functions NEUM and WSAQF. However, we should note that in the CLF-DM

TABLE I
QUANTITATIVE COMPARISONS BETWEEN THE PA, CLF-DM [38], τ-SEDS [39], AND CDSP [31] FOR THE SIX MOST CHALLENGING CASES IN THE LASA HANDWRITING DATASET USING METRICS SEA AND DTWD

Cases	SEA, (standard deviation)				DTWD, (standard deviation)			
	PA	CLF-DM	τ-SEDS	CDSP	PA	CLF-DM	τ-SEDS	CDSP
BendedLine	**81.2**, (**58.4**)	76.5, (62.1)	164.6, (124.8)	416.7, (90.8)	**79.2**, (**21.5**)	77.5, (22.4)	89.3, (18.1)	331.1, (55.1)
DoubleBendedLine	**59.4**, (**21.9**)	60.7, (23.1)	177.9, (77.9)	591.1, (43.4)	**57.4**, (**5.0**)	58.4, (5.3)	104.9, (35.8)	451.3, (50.2)
JShape_2	**119.7**, (**67.5**)	119.7, (67.5)	109.3, (44.0)	294.8, (91.3)	**146.6**, (**32.7**)	146.6, (32.7)	137.6, (32.2)	222.2, (36.1)
Leaf_1	**88.8**, (**44.9**)	209.4, (69.6)	324.4, (92.4)	385.0, (64.8)	**80.4**, (**21.4**)	137.0, (50.4)	202.8, (61.2)	303.0, (69.2)
Leaf_2	**86.9**, (**32.3**)	86.9, (32.3)	241.8, (29.9)	354.3, (32.6)	**102.8**, (**9.7**)	102.8, (9.7)	180.1, (16.1)	247.1, (13.0)
Snake	**95.0**, (**45.9**)	222.1, (51.4)	275.5, (68.4)	262.6, (27.0)	**85.2**, (**15.1**)	138.6, (26.9)	175.0, (50.7)	169.8, (21.9)

The boldface values are values generated by the proposed approach.

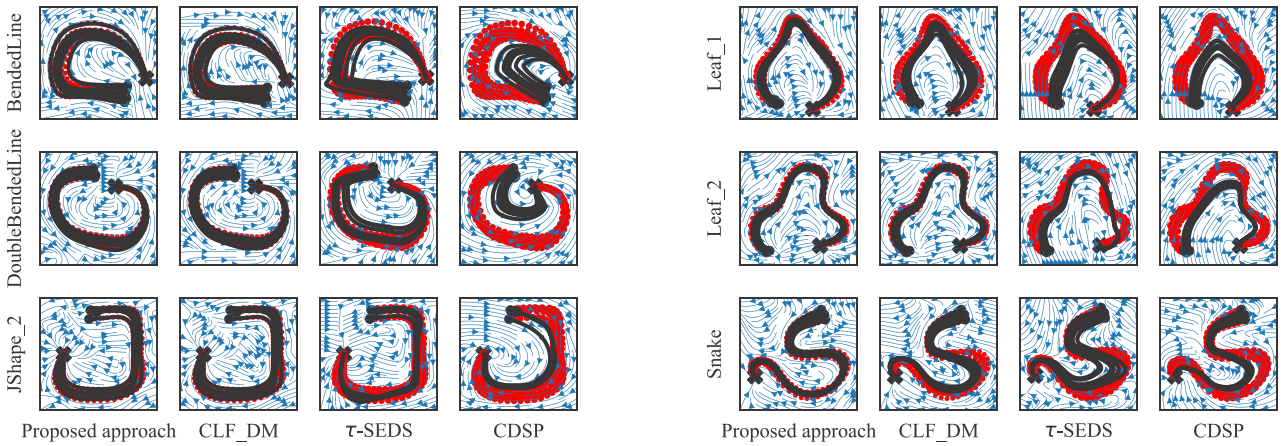


Fig. 12. Learning results of the PA, CLF-DM [38], τ-SEDS [39], and CDSP [31] for the six most challenging cases in the LASA handwriting dataset.

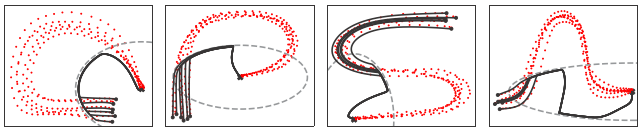


Fig. 13. Learning results of the proposed demonstration learning approach with the position constraint.

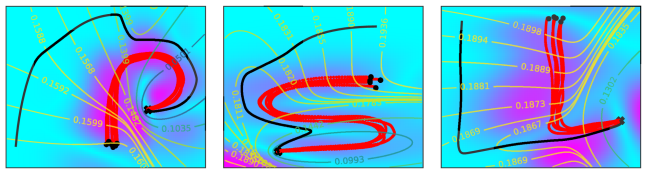


Fig. 14. Results of the generalization ability investigation of the proposed demonstration learning approach.

approach, the β is set as 10 to highlight the accuracy property. Comparatively, the β for NEUM is still set as 1.0 to simultaneously consider the accuracy and generalization ability. We also provide the detailed learning results for these demonstration approaches in Fig. 12. It should be noted that the learning results for these compared approaches might be slightly different from their original articles due to the different optimization algorithms and initial (hyper)parameter settings.

We then validate the position-constraint ability of the PA. The learning results of the PA with the position constraint are shown in Fig. 13, where the position constraint for each case is represented by the gray ellipse. The soften parameter η mentioned in Remark 2 is set as 0.05. As what we expect, once the reproduction trajectory enters the ellipse, it will never be able to leave the ellipse due to the position constraint (47b).

We also investigate the generalization ability of the PA, and the results are shown in Fig. 14, where the deeper background color indicates the larger velocity generated by the original ADS. In this investigation, the position constraint is not considered, and function $\rho(x)$ is given by

$$\begin{cases} \rho(x) = \left\| \frac{\partial V(x)}{\partial x} \right\|_2 \overline{v}(x) \\ \overline{v}(x) = 5. \end{cases} \tag{59}$$

As we can see, in areas away from the demonstration set, the velocity generated by the original ADS will approach zero. According to the analysis in Section IV-C, the generated trajectory will evolve along with the negative gradient of the NEUM with velocity $\overline{v}(x)$. Since the parameter β in the objective function (31) is set as 1.0, the gradient of NEUM can capture the demonstration preferences in a broad area, which results in the strong generalization ability.

We finally investigate the influence of the demonstration trajectory number. Fig. 15(a) and (b) shows the influence of the NEUM learning, where Fig. 15(a) shows the comparison results for one-shot learning (only using the sixth demonstration trajectory "T6") and full learning (all demonstration trajectories are used) with respect to the metric E_2. As we can see,

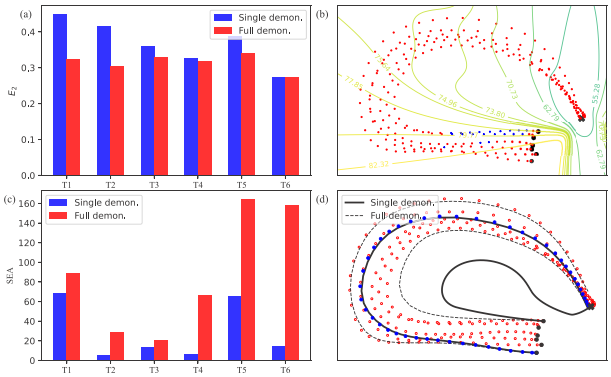


Fig. 15. Results of the PA with different demonstration trajectory numbers. (a) Comparative results between the one-shot NEUM learning and full NEUM learning with respect to the metric E_2. (b) Detailed one-shot NEUM learning result. (c) Comparative results between the one-shot ADS learning and full ADS learning with respect to the metric SEA. (d) Detailed reproduction results of the one-shot ADS learning and full ADS learning.

the NEUMs from one-shot learning and full learning obtain similar scores with respect to the metric E_2, which is caused by the strong generalization ability of the NEUM, and the fact that all demonstration trajectories follow similar patterns. Detailed one-shot NEUM learning result is shown in Fig. 15(b), where red circles and blue crosses represent the demonstration points satisfying and violating the NEUM, respectively. As we can see intuitively, the NEUM from one-shot learning can still capture the overall demonstration preferences in a broad area.

Fig. 15(c) and (d) shows the influence of the demonstration trajectory number on the ADS learning. In Fig. 15(c), we conduct the ADS learning approach for each demonstration trajectory and compare these learning results with the full learning result. An interesting point is that, for each demonstration trajectory, the one-shot learning result is better than the full learning result. This phenomenon is caused by the fact that using multiple demonstration trajectories will raise disturbances when learning the original ADS. An example is shown in Fig. 15(d), where the ADS is learned via the sixth demonstration trajectory "T6" (represented by blue solid circles). As we can see, the one-shot learning is more accurate compared with the full learning when reproducing the trajectory "T6." The reason is that a neighbor demonstration trajectory "T5" dominates the full-learning ADS when reproducing the trajectory "T6," However, we should also note that when using one demonstration trajectory, the generalization ability could be limited due to the less-knowledgeable original ADS, which is also reflected in Fig. 15(d) when we want to reproduce the innermost trajectory "T1." In this case, the full-learning ADS obviously gets the better reproduction result. Thus, as a conclusion, using fewer demonstration trajectories may improve the reproduction accuracy due to the reduced disturbances. However, the lack of demonstration trajectories will limit the generalization ability of the ADS, although this could be relieved in areas where the NEUM can well capture the demonstration preferences, see Fig. 14.

VI. Validations via Robotic Experiments

In this section, we validate the PA via robotic experiments. The first experiment is a robotic assembly task shown in Fig. 16. A human tutor first demonstrates the assembly process, and then, the robot will mimic the human tutor to complete the task.

In this experiment, the parameter settings for NEUM learning are given as $\beta = 1.0$, $d_H = 10$, and $L_2 = 10^{-6}$, function $\rho(x)$ is the same as (57). It should be noted that although the PA needs to online solve the optimization problem described in (46) and (47), the real-time property could be ensured due to the succinct QP formulation. As a result, the sampling period is set as 5 ms in the experiment.

Experiment results are shown in Fig. 17, where we validate the PA in various cases, including task reproduction, task generalization with and without disturbances. Fig. 17(a) and (b) shows the demonstration set and some isosurfaces of the learned energy function NEUM, respectively. Since the β is set as 1.0 here, the NEUM tries to align its gradient with the demonstration trajectories, which finally results in a smooth and generalizable energy function. It should be noted that the learning of NEUM always gets better results in the higher-dimensional cases since the constraints (18) could be less restrictive. As a result, the metrics E_1 and E_2 in this experiment are 4.185×10^{-8} and 0.029, respectively, which are far smaller than these values in Fig. 10.

Fig. 17(c) shows the task reproduction results. In these cases, the initial positions are the same as those in the demonstration processes. As we can see, the reproduction trajectories represented by blue solid lines are almost coincident with the demonstration trajectories, which implies that the PA has high reproduction accuracy.

Fig. 17(d)–(f) shows the different generalization cases of the PA. In Fig. 17(d), the reproduction trajectories' initial positions are different from those of the demonstration set. As we can see, all reproduction trajectories can converge to the goal position. Moreover, these reproduction trajectories also inherit the demonstration characteristic that the last motions should only occur in the x_3 dimension. This characteristic is essential to ensure the successful completion of the assembly task. In Fig. 17(e), the goal positions of the reproduction trajectories are modified to be different from those of the demonstration set. Similar to the case in Fig. 17(d), the reproduction trajectories converge to their goal positions, and simultaneously inherit the vertical motion characteristic at their last motion stages. In Fig. 17(f), we further investigate the robustness of the PA for spatial disturbances. In this case, a human will apply disturbed forces on the robot during robot motions. The disturbed trajectories are represented by the black dashed lines. To allow human interference, the compliant low-level tracking controller presented in our previous work [50] is used. We can find in Fig. 17(f) that although the robot will deviate from the desired path under the disturbed forces, it could recover as soon as possible due to its strong generalization ability. Finally, the reproduction trajectories converge to the goal position with the inherited motion characteristic.

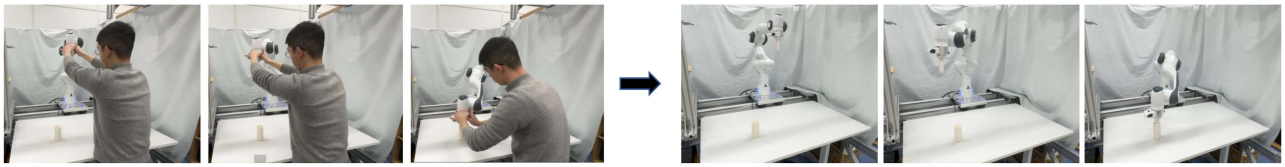


Fig. 16. Robotic assembly experiment.

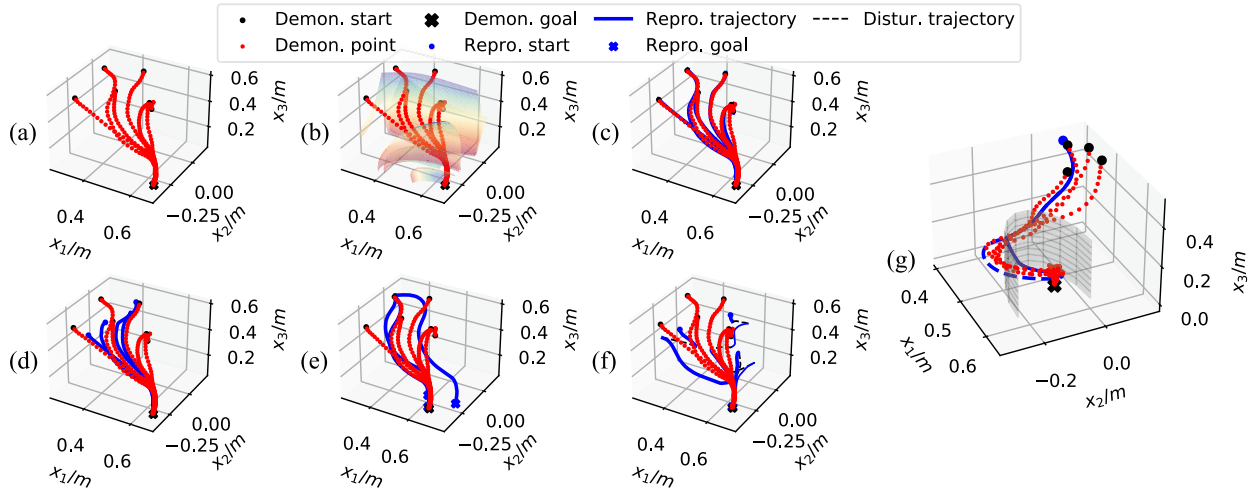


Fig. 17. Experiment results for the robotic assembly task. (a) Demonstration set. (b) Isosurfaces of the learned NEUM. (c) Task reproduction results with the same initial positions as these of the demonstrations. (d) Task generalization with different initial positions from these of the demonstrations. (e) Task generalization with different goal positions from these of the demonstrations. (f) Task generalization under disturbances. (g) Task reproduction with the position constraint.

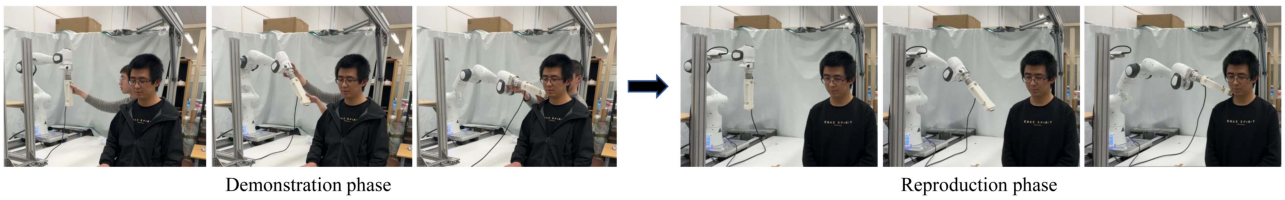


Fig. 18. Robotic ultrasound scanning experiment.

We also investigate the position-constraint ability of the PA in the assembly task. The position constraint is represented by the cylinder in Fig. 17(g). Specifically, the position-constraint parameters are set as $P = \text{diag}(1.0, 1.0, 0.0)$, $r_{\text{thres}} = 0.1\,m$ and $\eta = 0.1$. As we can see, the demonstration trajectories pass through the cylinder, thus violating the position constraint (47b). Without the position constraint, the corresponding reproduction trajectory (represented by the blue dashed line) will also pass through the cylinder. However, when we add the position constraint, the reproduction trajectory (represented by the blue solid line) will not leave the cylinder once it enters the cylinder, which validates the position-constraint ability of the PA.

The second experiment is a robotic ultrasound scanning task shown in Fig. 18. Different from the robotic assembly task, here we conduct the proposed demonstration learning algorithm in the robot joint space, which has dimension 7. All of the parameter settings of the robotic ultrasound scanning task are the same as those of the robotic assembly experiment.

Experiment results are shown in Fig. 19, where the first to seventh figures show the demonstration and reproduction trajectories in each robot joint. As we can see, all of the reproduction trajectories can converge to their goal positions. Moreover, although the demonstration set is very sparse in the 7-D space, the reproduction trajectories can still inherit the demonstration preferences well. For example, for the second and sixth joints, where initial positions are near the demonstration set, the reproduction trajectories almost coincide with the demonstration trajectories. For the fifth joint, although some initial positions are away from the demonstration set, the reproduction trajectories can still inherit the demonstration characteristic, i.e., first holding and then evolving to the goal position.

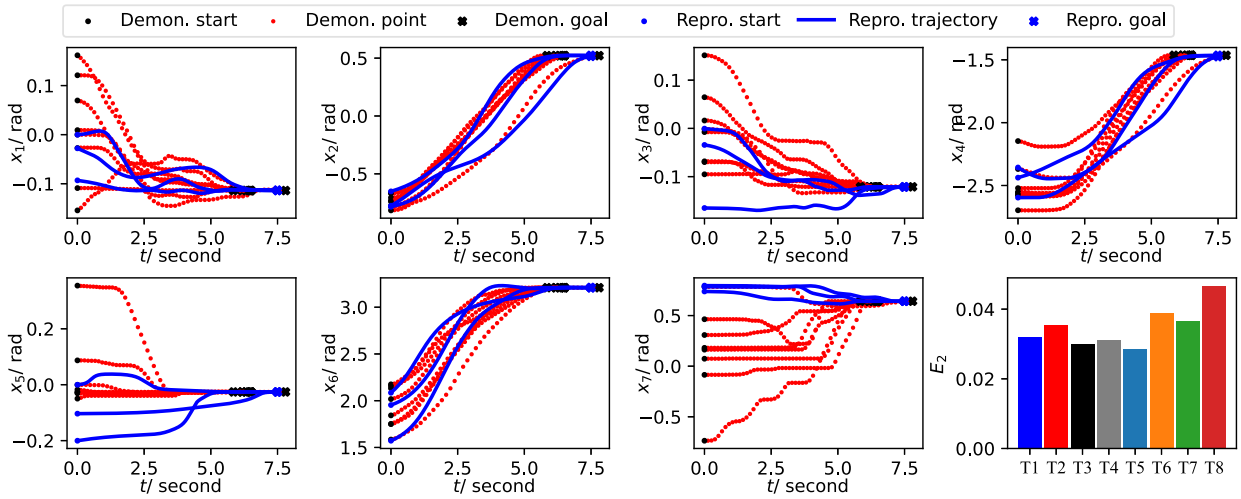


Fig. 19. Experiment results for the robotic ultrasound scanning task.

We also validate the effectiveness of the NEUM learning. Since we cannot plot the NEUM in the high-dimensional case, we instead show the metric E_2 for each demonstration trajectory. The result is shown in the last figure of Fig. 19, where we can find that E_2 takes a very small value for each demonstration trajectory. Thus, we can imagine that the learned NEUM is well consistent with the demonstration set. As a result, this experiment validates the effectiveness of the PA for high-dimensional learning cases.

VII. Conclusion

In this article, we focused on learning an ADS for globally stable and accurate demonstration learning. Specifically, we designed a flexible neural energy function, called NEUM, with the unique-minimum, positive-definite, and continuously differentiable properties. Then, the NEUM is used as the stability certificate for the ADS learning. The kernel contribution is that we proposed a polarlike space analysis approach to derive neural-parameter constraints to ensure the NEUM's unique-minimum, positive-definite, and continuously differentiable properties. We also provided a NEUM learning approach to conveniently consider the accuracy-generalization tradeoff. With the NEUM, the learned ADS can handle position constraints, which might be valuable for some robotic tasks. We further quantitatively analyzed the generalization ability of the ADS by utilizing the flexibility of the NEUM. Finally, we validated the effectiveness of the PA on the LASA dataset and two representative robotic experiments.

References

[1] H. Ochoa and R. Cortesao, "Impedance control architecture for robotic-assisted mold polishing based on human demonstration," *IEEE Trans. Ind. Electron.*, vol. 69, no. 4, pp. 3822–3830, Apr. 2022.

[2] Y. Ma, D. Xu, and F. Qin, "Efficient insertion control for precision assembly based on demonstration learning and reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4492–4502, Jul. 2021.

[3] Y. Yang, Y. Pan, X. Zhu, M. Gao, J. Zhang, and D. Tao, "A human-like dual-forklift collaborative mechanism for container handling," *IEEE Trans. Ind. Electron.*, vol. 68, no. 12, pp. 12871–12880, Dec. 2021.

[4] D. V. Domitilla, M. M. Richard, and P. Pietro, "Decomposition of human motion into dynamics-based primitives with application to drawing tasks," *Automatica*, vol. 39, no. 12, pp. 2085–2098, 2003.

[5] C. Yang, G. Ganesh, S. Haddadin, S. Parusel, A. Albu-Schaeffer, and E. Burdet, "Human-like adaptation of force and impedance in stable and unstable interactions," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 918–930, Oct. 2011.

[6] X. Yu, P. Liu, W. He, Y. Liu, Q. Chen, and L. Ding, "Human-robot variable impedance skills transfer learning based on dynamic movement primitives," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 6463–6470, Jul. 2022.

[7] C. Yang, C. Zeng, C. Fang, W. He, and Z. Li, "A DMPs-based framework for robot learning and generalization of humanlike variable impedance skills," *IEEE/ASME Trans. Mechatron.*, vol. 23, no. 3, pp. 1193–1203, Jun. 2018.

[8] H. K. Khalil, *Nonlinear Control*. London, U.K.: Pearson Press, 2015.

[9] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Trans. Syst., Man, Cybern., Part B. (Cybern.)*, vol. 37, no. 2, pp. 286–298, Apr. 2007.

[10] S. Calinon, F. Guenter, and A. Billard, "On learning the statistical representation of a task and generalizing it to various contexts," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 2978–2983.

[11] S. Calinon and A. Billard, "Statistical learning by imitation of competing constraints in joint space and task space," *Adv. Robot.*, vol. 23, no. 15, pp. 2059–2076, 2009.

[12] M. Muhlig, M. Gienger, S. Hellbach, J. J. Steil, and C. Goerick, "Task-level imitation learning using variance-based movement optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 1177–1184.

[13] S. Calinon, D. Bruno, and D. G. Caldwell, "A task-parameterized probabilistic model with minimum intervention control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 3339–3344.

[14] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, and C. Torras, "Learning physical collaborative robot behaviors from human demonstrations," *IEEE Trans. Robot.*, vol. 32, no. 3, pp. 513–527, Jun. 2016.

[15] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intell. Serv. Robot.*, vol. 9, no. 1, pp. 1–29, 2016.

[16] Y. Huang, J. Silverio, L. Rozo, and D. G. Caldwell, "Generalized task-parameterized skill learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 5667–5474.

[17] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Auton. Robots*, vol. 42, no. 3, pp. 529–551, 2017.

[18] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 2616–2624.

[19] Y. Huang, L. Rozo, J. Silverio, and D. G. Caldwell, "Kernelized movement primitives," *Int. J. Robot. Res.*, vol. 38, no. 7, pp. 833–852, 2019.

[20] S. M. Khansari-Zadeh, K. Kronander, and A. Billard, "Modeling robot discrete movements with state-varying stiffness and damping: A framework for integrated motion generation and impedance control," in *Proc. Robot.: Sci. Syst.*, 2014.

[21] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, 2013.

[22] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 763–768.

[23] A. Hewitt, C. Yang, Y. Li, and R. Cui, "DMP and GMR based teaching by demonstration for a KUKA LBR robot," in *Proc. IEEE Int. Conf. Autom. Comput.*, 2017, pp. 1–6.

[24] C. Yang, C. Chen, W. He, R. Cui, and Z. Li, "Robot learning system based on adaptive neural control and dynamic movement primitives," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 3, pp. 777–787, Mar. 2019.

[25] E. Gribovskaya and A. Billard, "Learning nonlinear multi-variate motion dynamics for real-time position and orientation control of robotic manipulators," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2009, pp. 472–477.

[26] L. Winfried and J. E. Slotine, "On contraction analysis for non-linear systems," *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.

[27] R. Harish and A. Dani, "Learning contracting nonlinear dynamics from human demonstration for robot motion planning," in *Proc. ASME Dyn. Syst. Control Conf.*, 2015, Art. no. V002T27A008.

[28] V. Sindhwani, S. Tu, and S. M. Khansari-Zadeh, "Learning contracting vector fields for stable imitation learning," 2018, *arXiv:1804.04878*.

[29] S. Singh, S. M. Richards, V. Sindhwani, J. E. Slotine, and M. Pavone, "Learning stabilizable nonlinear dynamics with contraction-based regularization," *Int. J. Robot. Res.*, vol. 40, no. 10–11, pp. 1123–1150, 2021.

[30] H. Ravichandar, I. Salehi, and A. Dani, "Learning partially contracting dynamical systems from demonstrations," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 369–378.

[31] H. Ravichandar and A. Dani, "Learning position and orientation dynamics from demonstrations via contraction analysis," *Auton. Robots*, vol. 43, no. 4, pp. 897–912, 2019.

[32] 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, Oct. 2011.

[33] S. M. Khansari-Zadeh, K. Kronander, and A. Billard, "Learning to play minigolf: A dynamical system-based approach," *Adv. Robot.*, vol. 26, no. 17, pp. 1967–1993, 2012.

[34] A. Lemme, K. Neumann, F. Reinhart, and J. J. Steil, "Neurally imprinted stable vector fields," in *Proc. Eur. Symp. Artif. Neural Netw.*, 2013, pp. 327–332.

[35] J. Hu, Z. Yang, Z. Wang, X. Wu, and Y. Ou, "Neural learning of stable dynamical systems based on extreme learning machine," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 306–311.

[36] J. Duan, Y. Ou, J. Hu, Z. Wang, S. Jin, and C. Xu, "Fast and stable learning of dynamical systems based on extreme learning machine," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 6, pp. 1175–1185, Jun. 2019.

[37] C. Jin, A. Liu, S. Liu, and W. Zhang, "A robot skill learning method based on improved stable estimator of dynamical systems," *Acta Automatica Sinica*, vol. 48, no. 7, pp. 1771–1781, 2022.

[38] S. M. Khansari-Zadeh and A. Billard, "Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robot. Auton. Syst.*, vol. 62, no. 4, pp. 752–765, 2014.

[39] K. Neumann and J. J. Steil, "Learning robot motions with stable dynamical systems under diffeomorphic transformations," *Robot. Auton. Syst.*, vol. 70, pp. 1–15, 2015.

[40] P. Nicolas and S. C. Philipp, "Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems," *Syst. Control Lett.*, vol. 96, pp. 51–59, 2016.

[41] J. Umlauft, A. Lederer, and S. Hirche, "Learning stable Gaussian process state space models," in *Proc. Amer. Control Conf.*, 2017, pp. 1499–1504.

[42] L. Pohler, J. Umlauft, and S. Hirche, "Uncertainty-based human motion tracking with stable Gaussian process state space models," *IFACPapersOnLine*, vol. 51, no. 34, pp. 8–14, 2019.

[43] K. Neumann, A. Lemme, and J. J. Steil, "Neural learning of stable dynamical systems based on data-driven Lyapunov candidates," *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 1216–1222.

[44] S. M. Richards, F. Berkenkamp, and A. Krause, "The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Proc. Conf. Robot Learn.*, 2018, pp. 466–476.

[45] N. M. Boffi, S. Tu, N. Matni, J. E. Slotine, and V. Sindhwani, "Learning stability certificates from data," in *Proc. Conf. Robot Learn.*, 2020, pp. 1341–1350.

[46] G. Manek and J. Z. Kolter, "Learning stable deep dynamics models," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 11128–11136.

[47] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[48] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[49] S. M. Khansari-Zadeh and A. Billard, "LASA handwriting data set," 2021. [Online]. Available: https://github.com/justagist/pyLASADataset

[50] Z. Jin, D. Qin, A. Liu, W. Zhang, and L. Yu, "Model predictive variable impedance control of manipulators for adaptive precision-compliance tradeoff," *IEEE/ASME Trans. Mechatron.*, vol. 28, no. 2, pp. 1174–1186, Apr. 2023.

Zhehao Jin received the B.Eng. degree in automation from the Zhejiang University of Technology, Hangzhou, China, in 2018. He is currently working toward the Ph.D. degree in control theory and control engineering with the College of Information Engineering, Zhejiang University of Technology.

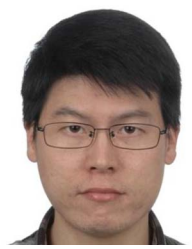
His current research interests include demonstration learning, reinforcement learning, robot-compliant control, and predictive control.

Weiyong Si received the M.S. degree in aerospace engineering from the Beijing Institute of Technology, Beijing, China, in 2018. He is currently working toward the Ph.D. degree in robotics with Bristol Robotics Laboratory, University of the West of England, Bristol, U.K.

His research interests include robot skill learning, teleoperation, and robot control.

Andong Liu (Member, IEEE) received the B.Eng. degree in automation from Guangxi University, Nanning, China, in 2007, and the Ph.D. degree in control theory and control engineering from the Zhejiang University of Technology, Hangzhou, China, in 2014.

He is currently an Associate Professor with the Department of Automation, College of Information Engineering, Zhejiang University of Technology. He was a Postdoctoral Research Fellow with the Department of Electronic Engineering, City University of Hong Kong, from 2016–2018. His research interests include model predictive control, networked control systems, and mobile robots.



Wen-An Zhang (Member, IEEE) received the B.Eng. degree in automation and the Ph.D. degree in control theory and control engineering from the Zhejiang University of Technology, Hangzhou, China, in 2004 and 2010, respectively.

Since 2010, he has been with the Zhejiang University of Technology, where he is currently a Professor with the Department of Automation. He was a Senior Research Associate with the Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, from 2010 to 2011. His current research interests include networked control systems, multisensor information fusion estimation, and robotics.

Dr. Zhang was awarded an Alexander von Humboldt Fellowship during 2011–2012. He has been a Subject Editor for *Optimal Control Applications and Methods* since 2016.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

Li Yu (Member, IEEE) received the B.S. degree in control theory from Nankai University, Tianjin, China, in 1982, and the M.S. and Ph.D. degrees in control theory and control engineering from Zhejiang University, Hangzhou, China, in 1988 and 1999, respectively.

He is currently a Professor with the College of Information Engineering, Zhejiang University of Technology, Hangzhou. He has authored or coauthored three books and more than 200 journal or conference articles. His current research interests include wireless sensor networks, networked control systems, and motion control.

Chenguang Yang (Senior Member, IEEE) received the Ph.D. degree in control engineering from the National University of Singapore, Singapore, in 2010, and the postdoctoral training in human robotics with the Imperial College London, London, U.K, from 2009 to 2010.

His research interests include human–robot interaction and intelligent system design.

Dr. Yang was awarded U.K. EPSRC UKRI Innovation Fellowship and individual EU Marie Curie International Incoming Fellowship. As the lead author, he won the IEEE Transactions on Robotics Best Article Award in 2012 and the IEEE Transactions on Neural Networks and Learning Systems Outstanding Article Award in 2022. He is the Corresponding Co-Chair of IEEE Technical Committee on Collaborative Automation for Flexible Manufacturing (CAFM), a Fellow of Institute of Engineering and Technology (IET), a Fellow of Institution of Mechanical Engineers (IMechE), and a Fellow of British Computer Society (BCS).