A Robust Task-Level Control Architecture for Learned Dynamical Systems

Eshika Pathak[†] EPATHAK2@ILLINOIS.EDU
Ahmed Aboudonia[*] ABOUDONIA@BERKELEY.EDU
Sandeep Banik[†] BANIKSAN@ILLINOIS.EDU
Naira Hovakimyan[†] NHOVAKIM@ILLINOIS.EDU
[†]*University of Illinois Urbana-Champaign,* [*]*University of California, Berkeley*

Editors: G. Sukhatme, L. Lindemann, S. Tu, A. Wierman, N. Atanasov

Abstract

Dynamical system (DS)-based learning from demonstration (LfD) is a powerful tool for generating motion plans in the operation ('task') space of robotic systems. However, realizing generated motion plans is often compromised by a "task-execution mismatch", where unmodeled dynamics, persistent disturbances, and system latency cause the robot's task-space state to diverge from the desired state. We propose a novel task-level robust control architecture, $\mathcal{L}_1$-augmented Dynamical Systems ($\mathcal{L}_1$-DS), that explicitly handles the task-execution mismatch in tracking a nominal motion plan generated by any DS-based LfD scheme. Our framework augments any DS-based LfD model with a nominal stabilizing controller and an $\mathcal{L}_1$ adaptive controller. Furthermore, we introduce a windowed Dynamic Time Warping (DTW)-based target selector, which enables the nominal stabilizing controller to handle temporal misalignment for improved phase-consistent tracking. We demonstrate the efficacy of our architecture on the LASA and IROS handwriting datasets.

Keywords: Learning from demonstrations; operation/task space; adaptive control; motion plans.

1. Introduction

The execution of complex manipulation skills, such as part assembly in factories, and surface cleaning in homes, are essential for robots operating in the real world. However, programming these skills explicitly is challenging due to their inherent complexity. Learning from Demonstration (LfD) (Billard et al., 2008) has emerged as an effective method, enabling robots to acquire skills from expert demonstrations (e.g., human-guided kinesthetic teaching or teleoperation). One such data-efficient approach within LfD encodes skills as autonomous dynamical systems (DS) (Fu et al., 2026), and learns a function that maps the robot state to a desired velocity (by learning a vector field) in the operational (task) space (Khatib, 2003). The learned DS is then used to generate task-space motion plans. However, during execution, sensing delays, latency, unmodeled low-level dynamics, and persistent external disturbances, often create a "task-execution gap", where the robot's true motion diverges from the generated motion plan. In this work, we address the task-execution gap for trajectory-following tasks by designing stable and robust DS-based motion plans.

In DS-based LfD, various efforts have been devoted to ensuring the stability of the generated DS-based motion plans. Early methods, such as dynamic movement primitives (Pastor et al., 2009; Saveriano et al., 2023), model skills as stable second-order systems with learned time-dependent forcing terms. Time-invariant approaches, such as SEDS (Khansari-Zadeh and Billard, 2011), learn globally asymptotically stable DS to a target point ('attractor') in task-space, but their strict stability constraints often reduce motion reproduction accuracy, as discussed in (Saveriano, 2020; Figueroa

and Billard, 2018). Related work has pursued both expressiveness through deep learning formulations: polynomial DS (Abyaneh and Lin, 2023), Neural ODEs (Sochopoulos et al., 2024; Nawaz et al., 2024), and normalizing flows (Urain et al., 2020); and stability guarantees through methods such as data-driven Lyapunov candidates (Neumann et al., 2013; Khansari-Zadeh and Billard, 2014), contraction theory (Blocher et al., 2017), and diffeomorphic transformations (Neumann and Steil, 2015; Rana et al., 2020). Further extensions include non-Euclidean geometries (Zhang et al., 2022; Duong and Atanasov, 2021), rhythmic motions (Sándor et al., 2015; Nawaz et al., 2024) and multi-step tasks (Li and Figueroa, 2023).

The above methods provide formal stability guarantees for DS-based motion plans (via Lyapunov theory or contraction), and assume a "perfect executor"- a low-level controller that tracks desired task-level motion plan perfectly, even in the presence of disturbances. To account for disturbances, many robust low-level controllers (using methods like adaptive robust feedback linearization (Abdelwahab et al., 2024), backstepping super-twisting (Kali et al., 2021), and adaptive coordination schemes (Khadivar and Billard, 2023)) can, in principle, be developed. Their design, however, typically relies on the availability of a nominal system model, which can be difficult to obtain or utilize as modern robotic systems become increasingly complex (Khatib et al., 2004; Brantner, 2025; Lovett et al., 2025; Taha et al., 2012; He et al., 2020). Furthermore, most commercial robots are "black boxes" that restrict access to low-level control due to certifiability or intellectual property concerns (Bilancia et al., 2023). Our proposed framework enables robust execution without requiring access to low-level control or precise system models.

Recent work in DS-based LfD has sought to augment or embed robustness directly into the learned dynamics. For example, instantaneous state perturbations (e.g. a robot arm being suddenly dragged and dropped off-course to a new position in the task space) can be corrected online via a quadratic program to ensure stable and safe tracking of motion plans (Nawaz et al., 2024) . Other approaches include generating an online modulation policy for sequential single-attractor sub-tasks (Wang et al., 2024), and achieving safety by learning control barrier functions (Saveriano and Lee, 2019; Huang et al., 2025). LAGS-DS (Figueroa and Billard, 2022), on the other hand, learns a DS with global convergence and stiffness-like symmetric attraction around reference trajectories in task-critical regions, but requires manual specification of these regions and is restricted to single-global-attractor tasks. To track a desired motion plan, leveraging the dynamical system nature of the task-level planner, we employ control-theoretic tools and address the "task-execution mismatch".

In this work, we propose $\mathcal{L}_1$-DS, an architecture that augments any learned DS-based LfD model with a nominal stabilizing controller, based on Control Lyapunov Functions (CLFs), and an $\mathcal{L}_1$ adaptive controller to actively handle the task-execution mismatch. Our approach is agnostic to the robot's low-level control stack, addressing the mismatch purely at the task-level. To improve the ability of the nominal stabilizing controller to handle temporal misalignments, we introduce a windowed Dynamic Time Warping (DTW)-based target selector. We empirically validate this framework on the LASA and IROS handwriting datasets, demonstrating the efficacy of $\mathcal{L}_1$-DS under various types of disturbances.

2. Background

2.1. Learning Dynamical Systems from Demonstrations

A continuous-time DS-based motion plan for a robotic system is defined as a function of an unambiguous operational/task-space state $z_{\text{des}}(t) \in \mathbb{R}^d$ at time $t \in \mathbb{R}_{\geq 0}$, which encodes the task-

relevant quantities (e.g., end-effector pose in operation space Khatib and Burdick (1986); Mistry and Righetti (2012); Lee et al. (2021)). The motion plan is characterized as an autonomous nonlinear system: $\dot{z}_{\text{des}}(t) = f(z_{\text{des}}(t))$, with $z_{\text{des}}(t_0) = z_0$, where $f : \mathbb{R}^d \to \mathbb{R}^d$ is an unknown smooth (time-invariant) vector field that encodes the desired specific motion primitives to be reproduced during execution from an initial state x_0 at time $t_0 \leq t$. Given a dataset consisting of state–velocity pairs from N demonstrations, $\mathcal{D} := \{z_i(t_k), \dot{z}_i(t_k)\}_{i,k}$ with $i \in \{1, \ldots, N\}$, and $k \in \{1, \ldots, T_i\}$ (T_i being the length of the i-th demonstration), we learn the parameterized nominal dynamics $f_\theta : \mathbb{R}^d \to \mathbb{R}^d$. To find the optimal parameters θ^*, the estimator f_θ is fit by minimizing a loss over $\mathcal{D}$,

$$\theta^* \in \arg\min_{\theta} \sum_{i=1}^{N} \sum_{k=1}^{T_i} \mathcal{L}(z_i(t_k), \bar{z}_i(t_k), \dot{z}_i(t_k), f_\theta(z_i(t_k))), \text{ subject to } C(\theta) \leq 0,$$

where $\mathcal{L} : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a chosen loss function, and $\bar{z}_i(t_k)$ denotes the state obtained by integrating $\dot{\bar{z}} = f_\theta(\bar{z})$ from t_0 to t_k given $\bar{z}(t_0) = z_i(t_0)$. The optional term $\mathcal{C}(\theta) \leq 0$ represents inequality constraints imposed on the parameters to guarantee desired system properties (such as Lyapunov stability). Methods of modeling and fitting f_θ include, but are not limited to, Khansari-Zadeh and Billard (2011); Figueroa and Billard (2018); Sochopoulos et al. (2024); Nawaz et al. (2024); Figueroa and Billard (2022); Urain et al. (2020), and Saveriano (2020).

2.2. $\mathcal{L}_1$ Adaptive Control

$\mathcal{L}_1$ adaptive control presents an architecture that decouples estimation from control and ensures uniform transient guarantees along with apriori computable robustness margins for a broad class of uncertain systems (Hovakimyan and Cao, 2010). Its effectiveness has been successfully demonstrated in different safety-critical deployments, including NASA's AirStar 5.5% model (Gregory et al., 2010), Calspan's Learjet (Ackerman et al., 2017), unmanned aerial vehicles (Kaminer et al., 2010; Wu et al., 2023) and robotic systems (Pravitra et al., 2020; Cheng et al., 2022; Sung et al., 2024). We briefly review the $\mathcal{L}_1$ adaptive control architecture below. More details can be found in Hovakimyan and Cao (2010).

Consider an uncertain control-affine nonlinear system:

$$\dot{x}(t) = g(x(t)) + h(x(t))u(t) + \delta(t, x(t), u(t)), \quad x(t_0) = x_0, \tag{1}$$

where $x(t) \in \mathbb{R}^d$ is the system state, $u(t) \in \mathbb{R}^m$ is the control input, $g : \mathbb{R}^d \to \mathbb{R}^d$ and $h : \mathbb{R}^d \to \mathbb{R}^{d \times m}$ are known functions, and $\delta : \mathbb{R}_{\geq 0} \times \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^d$ represents the unknown residual uncertainty. The uncertainty is decomposed into two components, namely, matched $\delta_m \in \mathbb{R}^m$ (acting through the control channel $h(x)$) and unmatched $\delta_{um} \in \mathbb{R}^{d-m}$ (acting through its orthogonal complement $h^\perp(x)$), i.e., $\delta(t, x, u) = h(x)\delta_m(t, x, u) + h^\perp(x)\delta_{um}(t, x, u)$. The control objective is to ensure the system state $x(t)$ remains uniformly bounded around a desired reference trajectory $x_{\text{ref}}(t)$, generated by the reference system $\dot{x}_{\text{ref}}(t) = g(x_{\text{ref}}(t)) + h(x_{\text{ref}}(t))u_{\text{nom}}(t)$ with $x_{\text{ref}}(t_0) = x_0$, where $u_{\text{nom}}(t)$ is generated by a nominal controller.

In an ideal, uncertainty-free case ($\delta = 0$), applying $u(t) = u_{\text{nom}}(t)$ to the system yields perfect tracking. In the presence of uncertainty, the $\mathcal{L}_1$ adaptive controller adds an adaptive term $u_a(t)$ such that the total control input $u(t) = u_{\text{nom}}(t) + u_a(t)$ achieves the aforementioned control objective. The $\mathcal{L}_1$ adaptive control architecture consists of three main components:

a. State Predictor: The state predictor equation is given by:

$$\dot{\hat{x}}(t) = g(x(t)) + h(x(t))u(t) + \hat{\delta}(t) + A_s\tilde{x}(t), \quad \hat{x}(t_0) = x_0, \tag{2}$$

where $\tilde{x}(t) = \hat{x}(t) - x(t)$ is the prediction error, A_s is a user-defined Hurwitz matrix, and $\hat{\delta}(t)$ is an estimate of the uncertainty calculated using the adaptation law stated below. A small prediction error $\tilde{x}(t)$ implies that $\hat{\delta}(t)$ accurately estimates the true uncertainty $\delta(t, x(t), u(t))$.

b. Piecewise-Constant Adaptive Law: The uncertainty estimate $\hat{\delta}(t)$ is updated at discrete sampling instants $t_i := iT_s$, $i \in \mathbb{N}$, with sampling period $T_s > 0$. The estimate is held constant between samples:

$$\hat{\delta}(t) = \hat{\delta}(t_i) = -\Phi^{-1}(T_s)\mu(t_i), \quad t \in [t_i, t_{i+1}), \tag{3}$$

where $\Phi(T_s) = A_s^{-1}(e^{A_sT_s} - I)$ and $\mu(t_i) = e^{A_sT_s}\tilde{x}(t_i)$. This update law is designed to cancel the effect of prior prediction errors at the sampling instants, enabling fast and stable estimation.

c. Low-Pass Filtered Control Law: The pseudo-inverse of $h(x(t))$ is used to compute the matched component of the uncertainty $\hat{\delta}_m(t) = h(x(t))^+\hat{\delta}(t)$, which is passed through a strictly proper, stable low-pass filter $C(s)$ to generate the adaptive control input:

$$u_a(s) = -C(s)\hat{\delta}_m(s), \tag{4}$$

where $\hat{\delta}_m(s)$ is the Laplace transform of $\hat{\delta}_m(t)$, and $C(s) = \frac{\omega}{s+\omega}$ is the transfer function of a first-order low-pass filter with $\omega > 0$ as filter bandwidth chosen to satisfy the $\mathcal{L}_1$ small-gain stability condition (Wang and Hovakimyan, 2011). This filter is the key to decoupling adaptation from robustness, as it filters high-frequency estimation noise from entering the control channel and destabilizing the system.

2.3. Control Lyapunov Function (CLF) Quadratic Programs

Consider the control-affine nonlinear system (1) in the absence of uncertainties (i.e., $\delta = 0$). For $x \in \mathcal{X} \subseteq \mathbb{R}^d$, a continuously differentiable function $V : \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a CLF for $x = 0$ if: $V(0) = 0$, $V(x) > 0$ for all $x \neq 0$, and there exists a constant $c > 0$ such that for all $x \in \mathcal{X} \setminus \{0\}$, $\inf_{u \in \mathbb{R}^m} \{L_gV(x) + L_hV(x)u + cV(x)\} \leq 0$, where $L_gV(x), L_hV(x)$ denote the Lie derivatives of V along g and h, respectively (Khalil, 2015). Given a candidate CLF, a control input u_{nom} can be computed, at each time instant, as the solution to the quadratic program (QP): $u_{\text{nom}}(x) = \arg\min_{u \in \mathbb{R}^m} \|u\|_2^2$, s.t. $L_gV(x) + L_hV(x)u + cV(x) \leq 0$, to asymptotically stabilize the uncertainty-free system. This formulation can be extended using control barrier functions (Ames et al., 2019) to enforce safety objectives.

2.4. Dynamic Time Warping (DTW)

DTW (Sakoe and Chiba, 2003) is a technique for measuring the similarity between two temporal sequences that can vary in speed or timing. Given two sequences $A = (a_1, \ldots, a_M)$ and $B = (b_1, \ldots, b_N)$ of lengths M and N, respectively, DTW finds an optimal alignment by constructing a warping path that minimizes the cumulative cost of local alignments. Let $c(i, j) := \|a_i - b_j\|_2$

denote the local distance between elements a_i and b_j. The DTW algorithm employs dynamic programming to compute the cumulative distance matrix $D(i,j)$, defined by the recurrence relation:

$$\begin{aligned} D(0,0) &= 0, \quad D(i,0) = D(0,j) = +\infty \quad \forall i,j > 0 \\ D(i,j) &= c(i,j) + \min\{D(i-1,j), D(i,j-1), D(i-1,j-1)\} \end{aligned} \tag{5}$$

for $i \in \{1,\dots,M\}$ and $j \in \{1,\dots,N\}$. The DTW distance between sequences A and B is given by $\text{DTW}(A,B) = D(M,N)$, and the optimal warping path is recovered through backtracking from (M,N) to $(1,1)$. The warping path $\mathcal{P} = \{(i_k,j_k)\}_{k=1}^{K}$ must satisfy three fundamental constraints: (i) *boundary conditions* requiring $\mathcal{P}$ to begin at $(1,1)$ and end at (M,N); (ii) *monotonicity*, ensuring $i_{k+1} \geq i_k$ and $j_{k+1} \geq j_k$ for all k; and (iii) *continuity*, restricting step transitions to $\{(1,0),(0,1),(1,1)\}$. To prevent pathological alignments and reduce computational complexity, the Sakoe-Chiba band constraint restricts the warping path to regions, where $|i-j| \leq w$ for some window parameter $w \geq 0$. This constraint reduces the time complexity from $O(MN)$ to $O(w \cdot \min\{M,N\})$, and avoids extreme warping distortions.

3. Proposed Control Architecture: $\mathcal{L}_1$-DS

We now present our proposed control architecture, $\mathcal{L}_1$-DS, shown in Figure 1, to generate a stable and robust DS-based motion plan. We denote the robot's task-space state as $z(t) \in \mathbb{R}^d$. This state is expected to follow a nominal target trajectory $z^*(t)$, generated by the nominal learned dynamics $f_\theta : \mathbb{R}^d \to \mathbb{R}^d$ (see Section 3.1), which satisfies the initial value problem:

$$\dot{z}^*(t) = f_\theta(z^*(t)), \quad z^*(t_0) = z_0^\star. \tag{6}$$

During execution, the actual state $z(t)$ will inevitably deviate from the target $z^*(t)$ due to the "task-execution gap" caused by uncertainties such as external disturbances, latency, and modeling errors. For a given radius $\rho > 0$, we define a tube around $z^*(t)$ as $\mathcal{T}(\rho) \triangleq \bigcup_{t \geq t_0} \mathcal{O}(z^*(t),\rho)$, where $\mathcal{O}(z^*(t),\rho) := \{z \in \mathbb{R}^d \mid ||z - z^*(t)|| \leq \rho\}$ (pointwise ball around $z^*(t)$). The primary control objective is to guarantee that the actual state $z(t)$ remains bounded around the nominal target trajectory $z^*(t)$, formally expressed as:

$$z(t) \in \mathcal{O}(z^*(t),\rho), \quad \forall t \geq t_0. \tag{7}$$

The $\mathcal{L}_1$-DS architecture consists of four components at the task-level: (i) the nominal learned dynamics f_θ, defining the nominal motion plan, (ii) a nominal control block to assure the stability of the nominal learned dynamics, (iii) a windowed DTW-based target selector to maintain temporal alignment with respect to the nominal target trajectory, and (iv) an $\mathcal{L}_1$ adaptive control block that actively handles the task-execution mismatch.

3.1. Nominal Learned Dynamics

The nominal dynamics, f_θ, is learned from demonstrations (Section 2.1), and is rolled out to generate the nominal target trajectory $z^*(t)$ (6). To ensure that the control design and stability analysis are well-posed, we define the domain of operation and impose mild regularity assumptions on f_θ.

Assumption A1 (Operating domain and well-posedness)**.** *There exists a compact, forward-invariant set $\mathcal{Z} \subset \mathbb{R}^d$ such that $T(\rho) \subseteq \mathcal{Z}$. The learned dynamics $f_\theta : \mathcal{Z} \to \mathbb{R}^d$ is locally Lipschitz continuous on $\mathcal{Z}$ and bounded: there exists $\Delta_f > 0$ such that $\|f_\theta(z)\| \leq \Delta_f, \forall z \in \mathcal{Z}$.*

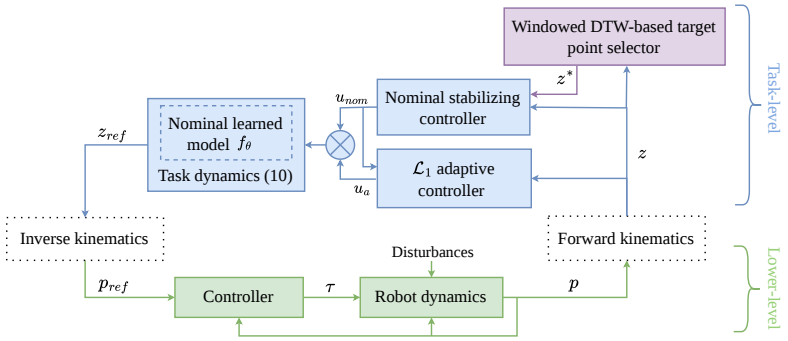

Figure 1: Proposed control architecture. p_{ref} is the reference low-level state (e.g., desired joint positions), τ represents the control inputs (e.g., joint torques) applied to the robot dynamics, and p is the actual measured low-level state (e.g., actual joint positions).

3.2. Nominal Stabilizing Control

The nominal learned dynamics f_θ in (6) generate the nominal target trajectory $z^*(t)$. However, even under ideal conditions with no task-execution gap, convergence from any initial condition $z_0 \neq z_0^\star$ to $z^*(t)$ is not guaranteed unless explicit constraints are imposed on f_θ during training (Figueroa and Billard, 2022; Sochopoulos et al., 2024). To this end, we augment the nominal learned dynamics with a stabilizing control input u_{nom}, forming the regulated learned dynamics:

$$\dot{z}(t) = f_\theta(z(t)) + u_{\text{nom}}(z(t), z^*(t)), \quad z(t_0) = z_0. \tag{8}$$

The objective of u_{nom} is to ensure that the state $z(t)$ exponentially converges to the target trajectory $z^*(t)$. For this purpose, from (6) and (8), we analyze the tracking error $e(t) \triangleq z(t) - z^*(t)$, whose dynamics are given by: $\dot{e}(t) = (f_\theta(z(t)) + u_{\text{nom}}(z(t), z^*(t))) - f_\theta(z^*(t))$, with $e(t_0) = z_0 - z_0^\star$. To render the error dynamics exponentially stable to 0, we follow Section 2.3, and select a C^1 CLF candidate $V : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ such that $V(e) > 0$ for $e \neq 0$ and $V(0) = 0$. Following (Nawaz et al., 2024), we obtain the control input u_{nom} as the solution to the following CLF-QP:

$$u_{\text{nom}}(z(t), z^*(t)) = \arg\min_{u \in \mathbb{R}^d} \quad \frac{1}{2}||u||^2 \text{ s.t. } \nabla V(e(t))^T \left(f_\theta(z(t)) - f_\theta(z^*(t)) + u\right) + cV(e(t)) \leq 0. \tag{9}$$

This approach enables us to superimpose a stabilizing controller onto any learned f_θ. To characterize the stability of the CLF-based control, we formalize the regularity and boundedness conditions typically satisfied by the CLF-based closed-loop system in the following assumption.

Assumption A2 (Nominal exponential stability)**.** *There exist positive constants $\alpha_1, \alpha_2, \lambda$ such that for $e(t) = z(t) - z^*(t)$, with $t \geq t_0$, $z(t) \in \mathcal{T}(\rho)$: (i) the CLF $V(e)$ is quadratically bounded uniformly in time: $\alpha_1||e(t)||^2 \leq V(e(t)) \leq \alpha_2||e(t)||^2$ (ii) the controller u_{nom} ensures an exponential convergence: $\dot{V}(e(t)) \leq -2\lambda V(e(t))$ (Khalil, 2002). (iii) $V(e)$ has bounded gradient on the tube: $||\nabla V(e(t))|| \leq \Delta_b(\rho)$ (written simply as Δ_b for brevity hereafter).*

3.3. Target Point Selection via Windowed DTW

To converge to the target trajectory $z^*(t)$ in (6), the regulated nominal dynamics (8) generates a continuous reference trajectory that needs to be executed by the low-level stack. However, dur-

ing execution, the actual state of the robot $z(t)$ may be phase-shifted relative to the timed target $z^*(t)$ due to disturbances (e.g., suddenly being pushed off-course). If we use a time-indexed target $z^*(t)$ in (9), and the robot is lagging, the error $e(t) = z(t) - z^*(t)$ will be artificially large and "phase-inconsistent." This misalignment might lead to the performance deterioration of the nominal stabilizing controller (9) that takes $z^*(t)$ as an input, resulting in a poor u_{nom} (e.g. trying to "skip" parts of the motion to catch up). Therefore, the nominal controller requires a target selection mechanism to provide a phase-consistent target point at each time instant, denoted as z^*. We propose a windowed DTW-based target selector, detailed in Algorithm 1, and illustrated in Figure 2.

Algorithm 1: Windowed DTW-Based Target Selector

Input : Current state z_t; Execution history $Z_{t-H:t}$;
Target trajectory $Z^*_{1:N}$; Previous index k_{prev};
Forward window W;
Target history H' (default $H' = H$).

Output: New index k_{new}; Target $z^\star = z^*_{k_{\text{new}}}$

1 $Z_{\text{hist}} \leftarrow Z_{t-H:t}$
2 **for** $k = k_{\text{prev}}$ **to** $\min(N,\, k_{\text{prev}} + W)$ **do**
3 $\quad Z^{*(k)} \leftarrow \{z^*_{\max(1,k-H')}, \ldots, z^*_k\}$;
4 $\quad C[k] \leftarrow \text{DTW}(Z_{\text{hist}}, Z^{*(k)})$
5 **end**
6 $k_{\text{new}} \leftarrow \arg\min_k C[k]$; $z^\star \leftarrow z^*_{k_{\text{new}}}$

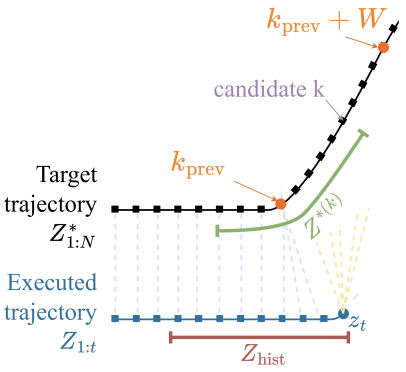

Figure 2: Illustration for how Algorithm 1 chooses the target point.

Recall that the nominal learned dynamics, defined by f_θ, generate a continuous target trajectory $z^*(t)$, as given in (6). For the discrete-time implementation of our controller, we sample this trajectory to create a discrete target sequence $Z^*_{1:N} \triangleq \{z^*_1, \ldots, z^*_N\}$ of length N. At each control step, Algorithm 1 aligns the robot's recent execution history, $Z_{\text{hist}} \triangleq Z_{t-H:t} = \{z_{t-H}, \ldots, z_t\}$, with the target trajectory $Z^*_{1:N}$. To ensure forward progress, the search for z^* is constrained to a forward window of size W (Line 2), starting from the previously selected reference index $k_{\text{prev}} \in \{1, \ldots N\}$. For each point $k \in \{k_{\text{prev}}, ..., k_{\text{prev}} + W\}$ in the forward window (Lines 2-5), the algorithm extracts a candidate target subsequence $Z^{*(k)} \triangleq \{z^*_{\max(1,k-H')}, \ldots, z^*_k\}$ (Line 3) and computes the DTW distance, $C[k]$, between Z_{hist} and $Z^{*(k)}$ (Line 4). After scanning the window, the index k_{new} that yields the minimum DTW distance is selected (Line 6). This index defines the new target point $z^\star = z^*_{k_{\text{new}}}$, which is then used to compute $u_{\text{nom}}(z, z^*)$, and update k_{prev} for the next control step. This selection mechanism, by recomputing the target point based on local geometric similarity, along with the forward-only search, ensures phase-consistent tracking.

3.4. $\mathcal{L}_1$ Adaptive Control Augmentation

3.4.1. Task-Execution Mismatch

The input u_{nom} in (9) is designed to shape the DS flow in the task-space assuming there is no task-execution gap. During execution, an inner stack (low-level controller, actuator/plant dynamics, sensing/communication) attempts to realize the regulated nominal motion plan in (8). Imperfections in this realization due to delays, unmodeled dynamics, and model inaccuracies, cause a deviation

in the actual dynamics of the task state. These imperfections are modeled at the task-level using a discrepancy term $\sigma(z(t))$, such that the task dynamics is realized as:

$$\dot{z}(t) \;=\; f_\theta(z(t)) \;+\; u_{\text{nom}}(z(t), z^\star(t)) \;+\; \sigma(z(t)), \quad z(t_0) = z_0.$$

Due to the nature of the DS-based formulation, the discrepancy term appears as a matched disturbance entering through the same channel as the input u_{nom}, motivating the augmentation of $\mathcal{L}_1$ adaptive control. The properties of matched uncertainty with respect to the dynamics are stated in the following assumption.

Assumption A3 (Bounded uncertainty and derivatives)**.** *The uncertainty $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ is continuous and uniformly bounded by a constant $\Delta_\sigma > 0$, and its derivative is uniformly bounded by constant $L_{\sigma z} \geq 0$, i.e. $||\sigma(z(t))|| \leq \Delta_\sigma$ and $\left|\left|\frac{\partial \sigma}{\partial z}(z(t))\right|\right| \leq L_{\sigma z}$, for all $z(t) \in \mathcal{T}(\rho)$.*

3.4.2. CLOSED-LOOP WITH $\mathcal{L}_1$ ADAPTIVE CONTROL

We now define the closed-loop task-space system by adding the $\mathcal{L}_1$ adaptive augmentation, $u_a(t) \in \mathbb{R}^d$, to the nominal dynamics and the uncertainty term defined in Section 3.4.1:

$$\dot{z}(t) = \underbrace{f_\theta(z(t)) + u_{\text{nom}}(z(t), z^*(t))}_{\text{Regulated Nominal Dynamics}} + \underbrace{u_a(t)}_{\mathcal{L}_1 \text{ Augmentation}} + \underbrace{\sigma(z(t))}_{\text{Uncertainty}}, \quad z(t_0) = z_0. \tag{10}$$

The adaptive control input $u_a(t)$ is generated by using the $\mathcal{L}_1$ adaptive control architecture described in Section 2.2. Equation (10) is expressed in the same form as the general control-affine nonlinear system in (1), under the following correspondence: the task-space state $z(t)$ maps to the system state $x(t)$; the nominal learned dynamics $f_\theta(z(t))$ map to $g(x(t))$; the CLF-based nominal control $u_{\text{nom}}(z(t), z^*(t))$ maps to $u_{\text{nom}}(t)$; and the uncertainty term $\sigma(z(t))$ maps to $\delta(t, x(t), u(t))$. With this mapping, the $\mathcal{L}_1$ adaptive control components (state predictor, piecewise-constant adaptive law, and low-pass filter) from Section 2.2 can be used to estimate the uncertainty $\hat{\sigma}(t) \in \mathbb{R}^d$ and generate the input $u_a(t)$. Hence, following (10), at every time instant, our $\mathcal{L}_1$-DS-based architecture computes the reference point z_{ref}, for the low-level stack to track, by solving, $\dot{z}_{ref}(t) = f_\theta(z(t)) + u_{\text{nom}}(z(t), z^*(t)) + u_a(t)$ with $z_{\text{ref}}(t_0) = z_0$.

Following Wu et al. (2023), using Assumptions A1–A3, by continuity on the compact set $\mathcal{T}(\rho)$, there exist finite constants $\Delta_f, \Delta_{\text{nom}}$ such that: $\|f_\theta(z(t))\| \leq \Delta_f, \|u_{\text{nom}}(z(t), z^\star(t))\| \leq \Delta_{\text{nom}}\ \forall z(t) \in \mathcal{T}(\rho)$. Note, the bound Δ_{nom} is a consequence of the continuity of the mapping $(z, z^*) \mapsto u_{nom}$ over the compact set $\mathcal{T}(\rho)$ rather than an explicit optimization constraint. We also define $\Delta_{\hat{\sigma}}$ as the uniform bound on the uncertainty estimate: $\Delta_{\hat{\sigma}} := \sup_{t \geq t_0} \|\hat{\sigma}(t)\|$. Hence, the bounds on the state derivative in (10) are given by $\|\dot{z}\|_\infty \leq \phi_1 \triangleq \Delta_f + \Delta_{\text{nom}} + \Delta_{\hat{\sigma}} + \Delta_\sigma$. The Huruwitz matrix in (2) is chosen diagonal, $A_s = \text{diag}(\lambda_1, \ldots, \lambda_k)$, where λ_k for $k \in \{1, \ldots d\}$ are negative scalars. Based on the above bounds and constants from Assumptions A1–A3, we define the following constants : $\zeta_1(\omega) \triangleq \frac{\Delta_\sigma}{|2\lambda - \omega|} + \frac{L_{\sigma z}\, \phi_1}{2\lambda\, \omega}$, $\zeta_2(A_s) \triangleq 2\sqrt{d}\, L_{\sigma z}\, \phi_1 \;+\; \sqrt{d}\, \max_k |\lambda_k(A_s)|\, \Delta_\sigma$, $\zeta_3(\omega) \triangleq \Delta_\sigma\, \omega$, and $\zeta_4(\omega, A_s) \triangleq \frac{\Delta_b(\zeta_2(A_s) + \zeta_3(\omega))}{2\lambda}$. For an arbitrarily chosen scalar $\varepsilon > 0$, we define the tube radius in (7) as $\rho \;\triangleq\; \|e(t_0)\| \sqrt{\frac{\alpha_2}{\alpha_1}} \;+\; \varepsilon$, which implies $V(e(t_0)) \triangleq V_0 < \alpha_1 \rho^2$. The $\mathcal{L}_1$ filter bandwidth $\omega > 0$ is chosen large enough, and the sampling time $T_s > 0$ is chosen small enough, satisfying the following conditions:

$$\alpha_1\, \rho^2 > V_0 + \Delta_b\, \zeta_1(\omega), \quad T_s \leq \frac{\alpha_1 \rho^2 - V_0 - \Delta_b \zeta_1(\omega)}{\zeta_4(\omega, A_s)}. \tag{11}$$

Conditions (11) can be met for large ω since $\zeta_1(\omega) = O(\omega^{-1})$ and $V_0 < \alpha_1\rho^2$, which guarantees a valid $T_s > 0$. With the above constants and design conditions in place, we can now formally state the main result guaranteeing that $z(t)$ remains bounded around $z^*(t)$.

Theorem 1 *Consider the dynamical systems in (6) and (10). Let Assumptions A1–A3 and inequalities (11) hold. Then $z(t)$ is bounded around $z^*(t)$, $z(t) \in \mathcal{O}(z^*(t), \rho)$, for all $t \geq t_0$. Furthermore, the closed-loop state $z(t)$ in (10) is uniformly ultimately bounded for all $t \geq t_1 > t_0$, such that*

$$z(t) \in \mathcal{O}(z^*(t), \mu(\omega, T_s, t_1)) \subset \mathcal{O}(z^*(t), \rho),$$

where the ultimate bound is defined as

$$\mu(\omega, T_s, t_1) := \sqrt{\frac{e^{-2\lambda(t_1-t_0)}V_0 \; + \; \Delta_b\zeta_1(\omega) + \zeta_4(\omega, A_s)T_s}{\alpha_1}}.$$

The result follows from Theorem 1 in Wu et al. (2023) and establishes that the proposed $\mathcal{L}_1$-DS architecture guarantees uniform bounds on the error between $z(t)$ and $z^*(t)$.

4. Simulations

Having established the $\mathcal{L}_1$-DS architecture, we now empirically validate its performance using a neural ordinary differential equations (NODE) model trained on the non-periodic LASA, and periodic IROS handwriting datasets (Sochopoulos et al., 2024; Nawaz et al., 2024). We demonstrate the efficacy of $\mathcal{L}_1$-DS in two execution regimes. (i) Perfect Command Following, which simulates a perfect low-level executor, i.e., the input to the forward kinematics block is the same as the output of the inverse kinematics block in Figure 1. In this regime, disturbances are introduced directly into task dynamics via σ in (10) to demonstrate the robustness of the task-level planning layer. (ii) Imperfect Command Following, which models the task-execution mismatch where our low-level plant and controller cannot perfectly realize task-level commands. This regime considers second-order closed-loop plant dynamics at the low-level and is subject to matched (M) and/or unmatched (U) disturbances at the low-level (see Appendix B).

In Table 1, performance is reported using the DTW distance between the executed task-space trajectory $z(t)$ and the nominal target trajectory $z^*(t)$. DTW scores are sensitive to the specific task's (shape's) length and complexity, as each DS reacts differently to the same disturbance type. Therefore, instead of reporting the mean and variance of the DTW across the entire datasets, we normalize the scores to better assess relative improvements. For each disturbance type and shape, we divide the DTW score obtained by the CLF augmented, NODE+CLF (Nawaz et al., 2024), and $\mathcal{L}_1$-DS ($\mathcal{L}_1$-NODE) versions, by the score obtained from its learned nominal model (NODE). This normalization yields scores typically around or below 1.0, where values less than 1.0 indicate a performance improvement attributable to the control augmentation.

On the IROS dataset, NODE+CLF(LE) uses the least-effort-based target selector (Nawaz et al., 2024) designed for periodic tasks, while NODE+CLF(DTW) uses our proposed Algorithm 1 for selecting the target point. We see that the proposed Algorithm 1 improves the performance of the nominal controller (9), which is further improved by the augmentation of $\mathcal{L}_1$ adaptive control. Overall, for both datasets, the $\mathcal{L}_1$-DS architecture improves the tracking performance under the perfect and imperfect execution regimes in the presence of disturbances. In Figure 3, we show sample experiments corresponding to Table 1. Further simulations with a SEDS (Khansari-Zadeh and Billard, 2011) nominal model can be found in Appendix A.

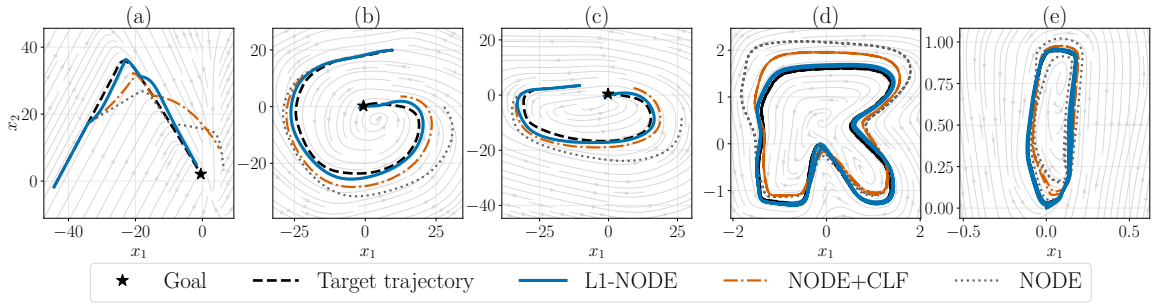

Figure 3: (a) LASA *Angle*, perfect command following regime with step disturbance; (b) LASA *GShape*, imperfect command following regime with unmatched multi-sine disturbance; (c) LASA *DoubleBendedLine*, imperfect command following regime with matched multi-sine disturbance; (d) IROS *RShape*, imperfect command following regime with unmatched constant disturbance; (e) IROS *IShape*, imperfect command following regime with matched multi-sine and unmatched periodic step disturbance.

Table 1: Average Normalized DTW Scores. Lower score is better (1.0 = NODE performance).

	Dataset: LASA		Dataset: IROS		
Regime: Disturbance	NODE+CLF	$\mathcal{L}_1$-**NODE**	NODE+CLF(LE)	NODE+CLF(DTW)	$\mathcal{L}_1$-**NODE**
Perfect: Step	0.627 ± 0.026	$\mathbf{0.118 \pm 0.002}$	0.717 ± 0.043	0.693 ± 0.067	$\mathbf{0.396 \pm 0.022}$
— Imperfect: —					
M. Multi-sine	0.707 ± 0.118	$\mathbf{0.244 \pm 0.023}$	0.720 ± 0.078	0.648 ± 0.147	$\mathbf{0.536 \pm 0.116}$
U. Constant	0.628 ± 0.031	$\mathbf{0.196 \pm 0.010}$	0.435 ± 0.011	0.394 ± 0.011	$\mathbf{0.042 \pm 0.003}$
U. Multi-sine	0.694 ± 0.103	$\mathbf{0.237 \pm 0.019}$	0.585 ± 0.013	0.482 ± 0.020	$\mathbf{0.327 \pm 0.020}$
M. Multi-sine with U. Pulses	0.687 ± 0.065	$\mathbf{0.241 \pm 0.019}$	0.630 ± 0.003	0.482 ± 0.020	$\mathbf{0.212 \pm 0.016}$

5. Conclusion

We introduce $\mathcal{L}_1$-DS, a task-level control architecture designed to address the task-execution mismatch in DS-based motion plans. The proposed architecture leverages the dynamical system nature of the DS-based motion plan formulation to augment control-theoretic methods for stability and robustness. We also design a novel windowed DTW-based target selector for shape-consistent tracking. We successfully demonstrate the efficacy of the proposed architecture on periodic and non-periodic handwriting tasks. Future work includes experimental validations on higher dimensional task spaces and robotic manipulators, robustness to errors in the learned nominal DS-based model, and extensions to multiple skills or skills involving sequential sub-tasks.

6. Acknowledgments

This work is supported by the Air Force Office of Scientific Research Grant FA9550-21-1-0411, the National Aeronautics and Space Administration under Grant 80NSSC22M0070, the National Science Foundation (NSF) under Grants CMMI 2135925, IIS 2331878, and CMMI 24-31216, and the Swiss National Science Foundation under grant agreement P500PT_230223.

References

Mohamed Abdelwahab, Giulio Giacomuzzo, Alberto Dalla Libera, and Ruggero Carli. Adaptive robust controller for handling unknown uncertainty of robotic manipulators. In 2024 IEEE 20th International Conference on Automation Science and Engineering, pages 2992–2997. IEEE, 2024.

Amin Abyaneh and Hsiu-Chin Lin. Learning Lyapunov-stable polynomial dynamical systems through imitation. arXiv preprint arXiv:2310.20605, 2023.

Kasey A Ackerman, Enric Xargay, Ronald Choe, Naira Hovakimyan, M Christopher Cotting, Robert B Jeffrey, Margaret P Blackstun, T Paul Fulkerson, Timothy R Lau, and Shawn S Stephens. Evaluation of an $\mathcal{L}_1$ adaptive flight control law on Calspan's variable-stability Learjet. Journal of Guidance, Control, and Dynamics, 40(4):1051–1060, 2017.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: theory and applications. In 2019 18th European Control Conference, pages 3420–3431. IEEE, 2019.

Pietro Bilancia, Juliana Schmidt, Roberto Raffaeli, Margherita Peruzzini, and Marcello Pellicciari. An overview of industrial robots control and programming approaches. Applied Sciences, 13(4): 2582, 2023.

Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Robot programming by demonstration. In Springer Handbook of Robotics, pages 1371–1394. Springer, 2008.

Caroline Blocher, Matteo Saveriano, and Dongheui Lee. Learning stable dynamical systems using contraction theory. In 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence, pages 124–129. IEEE, 2017.

Gerald Brantner. Humanoid robot acrobatics utilizing complete articulated rigid body dynamics. arXiv preprint arXiv:2508.08258, 2025.

Yikun Cheng, Pan Zhao, Fanxin Wang, Daniel J Block, and Naira Hovakimyan. Improving the robustness of reinforcement learning policies with $\mathcal{L}_1$ adaptive control. IEEE Robotics and Automation Letters, 7(3):6574–6581, 2022.

Thai Duong and Nikolay Atanasov. Hamiltonian-based neural ODE networks on the SE(3) manifold for dynamics learning and control. arXiv preprint arXiv:2106.12782, 2021.

Nadia Figueroa and Aude Billard. A physically-consistent Bayesian non-parametric mixture model for dynamical system learning. In CoRL, pages 927–946, 2018.

Nadia Figueroa and Aude Billard. Locally active globally stable dynamical systems: theory, learning, and experiments. The International Journal of Robotics Research, 41(3):312–347, 2022.

Jiayun Fu, Haotian Huang, Zhehao Jin, Andong Liu, Wen-An Zhang, Li Yu, Weiyong Si, and Chenguang Yang. A survey on learning an autonomous dynamic system for human–robot skills transfer from demonstration. Robotics and Computer-Integrated Manufacturing, 97:103092, 2026.

Irene Gregory, Enric Xargay, Chengyu Cao, and Naira Hovakimyan. Flight test of an $\mathcal{L}_1$ adaptive controller on the NASA AirSTAR flight test vehicle. In AIAA Guidance, Navigation, and Control Conference, page 8015, 2010.

Wei He, Xinxing Mu, Liang Zhang, and Yao Zou. Modeling and trajectory tracking control for flapping-wing micro aerial vehicles. IEEE/CAA Journal of Automatica Sinica, 8(1):148–156, 2020.

Naira Hovakimyan and Chengyu Cao. $\mathcal{L}_1$ adaptive control theory: Guaranteed robustness with fast adaptation. SIAM, 2010.

Haotian Huang, Jiayun Fu, Zhehao Jin, Andong Liu, Wen-An Zhang, and Chenguang Yang. A safety-critical dynamic system framework for high-precision learning from demonstration. IEEE Transactions on Automation Science and Engineering, 2025.

Yassine Kali, Maarouf Saad, and Khalid Benjelloun. Backstepping super-twisting for robotic manipulators with matched and unmatched uncertainties. In 2021 18th International Multi-Conference on Systems, Signals & Devices, pages 1154–1159. IEEE, 2021.

Isaac Kaminer, António Pascoal, Enric Xargay, Naira Hovakimyan, Chengyu Cao, and Vladimir Dobrokhodov. Path following for small unmanned aerial vehicles using $\mathcal{L}_1$ adaptive augmentation of commercial autopilots. Journal of Guidance, Control, and Dynamics, 33(2):550–564, 2010.

Farshad Khadivar and Aude Billard. Adaptive fingers coordination for robust grasp and in-hand manipulation under disturbances and unknown dynamics. IEEE Transactions on Robotics, 39(5): 3350–3367, 2023.

H.K. Khalil. Nonlinear Systems. Pearson Education. Prentice Hall, 2002. ISBN 9780130673893.

H.K. Khalil. Nonlinear Control. Always Learning. Pearson, 2015. ISBN 9780133499261.

S Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. IEEE Transactions on Robotics, 27(5):943–957, 2011.

S Mohammad Khansari-Zadeh and Aude Billard. Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions. Robotics and Autonomous Systems, 62(6):752–765, 2014.

Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE Journal on Robotics and Automation, 3(1):43–53, 2003.

Oussama Khatib and Joel Burdick. Motion and force control of robot manipulators. In Proceedings. 1986 IEEE International Conference on Robotics and Automation, volume 3, pages 1381–1386. IEEE, 1986.

Oussama Khatib, Luis Sentis, Jaeheung Park, and James Warren. Whole-body dynamic behavior and control of human-like robots. International Journal of Humanoid Robotics, 1(01):29–43, 2004.

Yisoo Lee, Sanghyun Kim, Jaeheung Park, Nikos Tsagarakis, and Jinoh Lee. A whole-body control framework based on the operational space formulation under inequality constraints via task-oriented optimization. IEEE Access, 9:39813–39826, 2021.

Tianyu Li and Nadia Figueroa. Task generalization with stability guarantees via elastic dynamical system motion policies. In 7th Annual Conference on Robot Learning, 2023.

Ean Lovett, Maxwell Hammond, Niloufar Seyfi, Amirreza Fahim Golestaneh, Venanzio Cichella, and Caterina Lamuta. A review on sensor technologies, control approaches, and emerging challenges in soft robotics. Advanced Robotics Research, page 202500085, 2025.

Michael Mistry and Ludovic Righetti. Operational space control of constrained and underactuated systems. In Robotics: Science and Systems, volume 7, pages 225–232, 2012.

Farhad Nawaz, Tianyu Li, Nikolai Matni, and Nadia Figueroa. Learning complex motion plans using neural ODEs with safety and stability guarantees. In 2024 IEEE International Conference on Robotics and Automation, pages 17216–17222. IEEE, 2024.

Klaus Neumann and Jochen J Steil. Learning robot motions with stable dynamical systems under diffeomorphic transformations. Robotics and Autonomous Systems, 70:1–15, 2015.

Klaus Neumann, Andre Lemme, and Jochen J Steil. Neural learning of stable dynamical systems based on data-driven Lyapunov candidates. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1216–1222. IEEE, 2013.

Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In 2009 IEEE International Conference on Robotics and Automation, pages 763–768. IEEE, 2009.

Jintasit Pravitra, Kasey A Ackerman, Chengyu Cao, Naira Hovakimyan, and Evangelos A Theodorou. $\mathcal{L}_1$-adaptive MPPI architecture for robust and agile control of multirotors. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 7661–7666. IEEE, 2020.

Muhammad Asif Rana, Anqi Li, Dieter Fox, Byron Boots, Fabio Ramos, and Nathan Ratliff. Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems. In Learning for Dynamics and Control, pages 630–639. PMLR, 2020.

Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing, 26(1):43–49, 2003.

Bulcsú Sándor, Tim Jahn, Laura Martin, and Claudius Gros. The sensorimotor loop as a dynamical system: how regular motion primitives may emerge from self-organized limit cycles. Frontiers in Robotics and AI, 2:31, 2015.

Matteo Saveriano. An energy-based approach to ensure the stability of learned dynamical systems. In 2020 IEEE International Conference on Robotics and Automation, pages 4407–4413. IEEE, 2020.

Matteo Saveriano and Dongheui Lee. Learning barrier functions for constrained motion planning with dynamical systems. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 112–119. IEEE, 2019.

Matteo Saveriano, Fares J Abu-Dakka, Aljaž Kramberger, and Luka Peternel. Dynamic movement primitives in robotics: A tutorial survey. The International Journal of Robotics Research, 42(13): 1133–1184, 2023.

A Sochopoulos, M Gienger, and S Vijayakumar. Learning deep dynamical systems using stable neural ODEs. In RSJ International Conference on Intelligent Robots and Systems, pages 11163–11170, 2024.

Minjun Sung, Sambhu Harimanas Karumanchi, Aditya Gahlawat, and Naira Hovakimyan. Robust model based reinforcement learning using $\mathcal{L}_1$ adaptive control. International Conference on Learning Representations, 2024.

Haithem E Taha, Muhammad R Hajj, and Ali H Nayfeh. Flight dynamics and control of flapping-wing mavs: a review. Nonlinear Dynamics, 70(2):907–939, 2012.

Julen Urain, Michele Ginesi, Davide Tateo, and Jan Peters. Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5231–5237. IEEE, 2020.

Weiyong Wang, Chao Zeng, Hong Zhan, and Chenguang Yang. A novel robust imitation learning framework for complex skills with limited demonstrations. IEEE Transactions on Automation Science and Engineering, 22:3947–3959, 2024.

Xiaofeng Wang and Naira Hovakimyan. $\mathcal{L}_1$ adaptive controller for nonlinear reference systems. In Proceedings of the 2011 American Control Conference, pages 594–599. IEEE, 2011.

Zhuohuan Wu, Sheng Cheng, Pan Zhao, Aditya Gahlawat, Kasey A Ackerman, Arun Lakshmanan, Chengyu Yang, Jiahao Yu, and Naira Hovakimyan. $\mathcal{L}_1$ quad: $\mathcal{L}_1$ adaptive augmentation of geometric control for agile quadrotors with performance guarantees. arXiv preprint arXiv:2302.07208, 2023.

Jiechao Zhang, Hadi Beik Mohammadi, and Leonel Rozo. Learning riemannian stable dynamical systems via diffeomorphisms. In 6th Annual Conference on Robot Learning, 2022.

Appendix A. Simulations with a SEDS model

Table 2: Average Normalized DTW(z, z^*). Lower score is better (1.0 = SEDS performance).

	Dataset: LASA	
Regime: Disturbance	SEDS+CLF	$\mathcal{L}_1$**-SEDS**
Perfect: Pulses	0.080 ± 0.000	$\mathbf{0.014 \pm 0.000}$
— Imperfect: —		
M. Multi-sine	0.256 ± 0.012	$\mathbf{0.011 \pm 0.003}$
U. Constant	0.322 ± 0.051	$\mathbf{0.028 \pm 0.000}$
U. Multi-sine	0.447 ± 0.026	$\mathbf{0.258 \pm 0.008}$
M. Multi-sine with U. Pulses	0.264 ± 0.015	$\mathbf{0.067 \pm 0.001}$

We also validate the efficacy of $\mathcal{L}_1$-DS using a SEDS (Khansari-Zadeh and Billard, 2011) nominal model. SEDS is designed to generate inherently stable DS-based motion plans for convergence to a single target attractor point in the task-space. As discussed in Section 3.3, this does not guarantee convergence to a nominal target trajectory. Therefore, we augment it with a CLF-based controller for a fair comparison for the objective of shape trajectory-tracking.

Appendix B. Simulation Setup

B.1. Datasets and Preprocessing

LASA (non-periodic) shapes. We use 10 randomly chosen LASA shapes (Angle, CShape, DoubleBendedLine, GShape, PShape, Sshape, Sine, WShape, Worm, Snake). All demonstrations are resampled to a common, normalized time grid $t \in [0, 1]$ with $N = 1000$ samples, unless stated otherwise. We use 4 demonstrations for training a NODE for each shape. The initial state for all runs is $[\bar{z}(t_0)]$, the average starting point of the 4 training demonstrations.

IROS (periodic) shapes. For R/I/O/S shapes, demonstrations are resampled to $N = 10{,}000$ points on $t \in [0, 1]$. We use $n_{\text{train}} = 3$ demos for training (if available). The outer simulation step is $\Delta t = 1/(N - 1)$. The initial state is $[\bar{z}(t_0)]$.

B.2. Learned Task Models

Neural ODE (NODE). IROS (NODE). Shapes: RShape, IShape, OShape, SShape. Training: 50k steps; batch 16; base learning rate 5×10^{-4}. MLP architecture: width 128, depth 3. Resampling: $N = 10{,}000$ points. LASA (NODE). MLP architecture: width 128, depth 3. Resampling: $N = 1000$ points.

SEDS (LASA ablations). A stable DS (SEDS) is fitted with K Gaussians: $K = 8$ for *Sshape*, *Sine*, *Snake*, and $K = 6$ for the other LASA shapes. Training and evaluation use $N = 1000$ resampled points.

B.3. Target Selection Policies

Two target selection policies are evaluated:

1. **DTW-based selector.** Forward-only alignment to the NODE reference with local windows $W = 50$ and $H = 40$; initialized at $\bar{z}(t_0)$.

2. **Least-Effort selector.** Look-ahead over a horizon of $N_{\text{LA}} = 35$ steps on Δt. The starting state is $\bar{z}(t_0)$.

B.4. Controller Stack and Simulation Modes

We compare three task-level controllers: NODE/SEDS: tracking using only the learned field f_θ. NODE/SEDS+CLF: NODE/SEDS augmented with a CLF-QP nominal stabilizer. L1-NODE/SEDS: NODE/SEDS+CLF additionally augmented with an $\mathcal{L}_1$ adaptive control input (we use windowed-DTW for target point selection, unless specified).

Two simulation regimes

1. **Perfect command tracking (direct task-space integration).** σ is injected directly in task space. We use two rectangular pulses on the normalized time axis. In Figure 4(a), $\sigma = [\sigma(x_1), \sigma(x_2)]^\top$ are the disturbances along $z(t) = [x_1, x_2]^\top$ axes. This experiment isolates task-level robustness.

2. **Imperfect command tracking (plant + low-level controller).** We model the lower-level plant in task space with position $p(t) = [p_1, p_2]^\top$ and velocity $v(t) = [v_1, v_2]^\top$. Disturbances are split into matched $d_m(t) = [d_{m,1}, d_{m,2}]^\top$ (entering the control channel) and unmatched $d_{um}(t) = [d_{um,1}, d_{um,2}]^\top$ (entering the position–rate channel directly). Per axis $i \in \{1, 2\}$ the double–integrator plant is

$$\dot{p}_i(t) = v_i(t) \; + \; d_{um,i}(t), \tag{12}$$
$$\dot{v}_i(t) = u_i(t) \; + \; d_{m,i}(t), \tag{13}$$

   or, in vector form,

$$\dot{p}(t) = v(t) + d_{um}(t), \qquad \dot{v}(t) = u(t) + d_m(t),$$

   with control input $u(t) = [u_1, u_2]^\top$ produced by a low–level PID acting independently on each axis to track a reference $(p_r(t), v_r(t))$ supplied by the task–level selector:

$$e_i(t) = p_{r,i}(t) - p_i(t), \qquad \dot{e}_i(t) = v_{r,i}(t) - v_i(t), \tag{14}$$
$$u_i(t) = K_p\, e_i(t) + K_d\, \dot{e}_i(t) + K_i \int_0^t e_i(\tau)\, d\tau. \tag{15}$$

   The selector (NODE/CLF/$\mathcal{L}_1$ stack) updates (p_r, v_r) at the outer simulation step Δt, while the plant+PID loop runs $10\times$ faster with inner step $\Delta t_{\text{llc}} = \Delta t/10$. Thus, task-level architecture only shapes (p_r, v_r), and the PID enforces fast, stable tracking on the double–integrator in the presence of unmatched d_{um} and matched d_m disturbances.

B.5. Disturbances

All disturbance signals are plotted on the normalized time grid as shown in Figure 4.

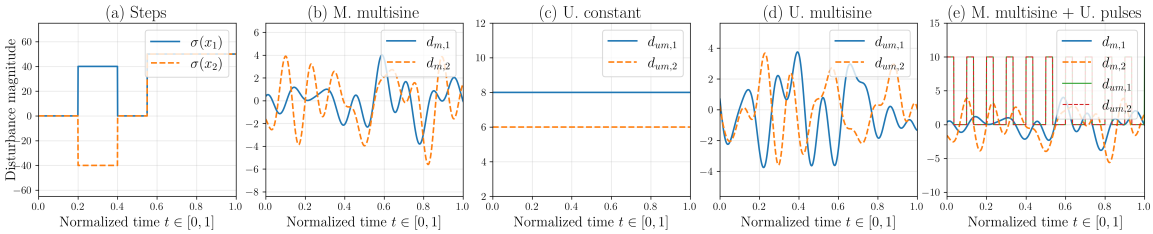

Figure 4: Disturbances used for the LASA dataset experiments. Appropriate magnitude-scaled versions of these were used for the IROS experiments.