Learning Complex Motion Plans using Neural ODEs with Safety and Stability Guarantees

Farhad Nawaz[1*], Tianyu Li[1], Nikolai Matni[1] and Nadia Figueroa[1]

Abstract— We propose a Dynamical System (DS) approach to learn complex, possibly periodic motion plans from kinesthetic demonstrations using Neural Ordinary Differential Equations (NODE). To ensure reactivity and robustness to disturbances, we propose a novel approach that selects a target point at each time step for the robot to follow, by combining tools from control theory and the target trajectory generated by the learned NODE. A correction term to the NODE model is computed online by solving a quadratic program that guarantees stability and safety using control Lyapunov functions and control barrier functions, respectively. Our approach outperforms baseline DS learning techniques on the LASA handwriting dataset and complex periodic trajectories. It is also validated on the Franka Emika robot arm to produce stable motions for wiping and stirring tasks that do not have a single attractor, while being robust to perturbations and safe around humans and obstacles. The project's web-page is `https://sites.google.com/view/lfd-neural-ode/home`.

I. INTRODUCTION

We are witnessing an increased use of robotic manipulators for everyday tasks in homes, offices, and factories, all of which involve human interaction. In such cases, robots should be be able to perform not just simple pick-and-place tasks [1], but also complex motions such as wiping a surface, stirring a pan, or peeling vegetables [2]. The major challenges in these settings are: (a) learning complex tasks from few observations (b) reactive and compliant motion planning in a dynamic environment and (c) remaining safe and robust to unexpected disturbances.

Learning from Demonstrations (LfD) is a widely used framework that enables transfer of skills to robots from demonstrations of desired tasks [2]–[4]. Typically, the observations are robot trajectories obtained through kinesthetic teaching, wherein humans passively guide the robot through the nominal motion. This approach has been shown to avoid the correspondence problem and reduce the simulation to reality gap [5]. In this setting, demonstrations are costly to collect: hence, it is essential to learn motion plans from as few demonstrations as possible, while still being robust, safe, and reactive in a dynamic environment.

There are multiple approaches for learning from demonstrations. Inverse Reinforcement Learning (IRL) and Behavior Cloning (BC) are popular methodologies to imitate motion [6]–[8] by optimizing an underlying task dependent objective function. IRL and BC require the demonstrator to explore the task space for learning an accurate motion policy that generalizes well: collecting the large amounts

[1]Farhad Nawaz, Tianyu Li, Nikolai Matni and Nadia Figueroa are with the GRASP Lab, University of Pennsylvania, PA 19104, USA.
*Corresponding author: `farhadn@seas.upenn.edu`

of data needed for these approaches is not feasible when the demonstrator is a human. Popular imitation algorithms such as DAgger [9] and deep imitation learning [10] are also data-hungry (typically requiring hundreds or thousands of demonstrations), and further provide no stability or safety guarantees. Another imitation method based on Gaussian Processes (GPs) is proposed in [11] that relies on time inputs and way-points to roll out desired trajectories. Such a framework cannot accommodate delays in accomplishing a task, e.g., as caused by a human moving the robot, and also cannot guarantee safety or stability.

We therefore root our methods in Dynamical System (DS) based motion planning approaches [12]–[15] that learn a global vector field instead of discrete way-points, and have shown to generate adaptable motion plans with minimal demonstrations. Prior work on LfD using DS [3], [13], [16] have demonstrated success in providing a powerful framework that can learn stable robotic motions while being compliant and reactive to human interactions online. Stable Estimator of Dynamical Systems (SEDS) [3] is a LfD method that learns globally stable dynamical systems with respect to a goal point using Gaussian mixture models and quadratic Lyapunov functions. An important limitation of SEDS is that it can only model trajectories whose distance to a single target point decreases monotonically in time. Another method based on SEDS is presented in [13] via a Linear Parameter Varying (LPV) re-formulation that learns more complex hand-drawn trajectories than SEDS, but is limited to motions that converge to a *single attractor (target)*.

In contrast, for robots to perform a variety of everyday tasks, we must be able to model complex periodic motions such as wiping a surface or stirring a pan. We propose to learn DS based motion plans that converge to a *target trajectory* which encodes the desired complex periodic motions, rather than to a single target point. Stability is then guaranteed at task execution with respect to the error dynamics between the robot and the target trajectory, while remaining safe to unforeseen obstacles. To the best of our knowledge, no prior work except for the approach proposed in [17] learns stable periodic motions for robotic tasks which uses normalizing flows. However, the approach described in [17] requires prior knowledge of whether the demonstrations depict single attractor dynamics or limit cycle dynamics. Our approach relies on no prior knowledge about the demonstrations, and captures the invariant features of complex target trajectories using the rich model class of Neural Ordinary Differential Equations (NODEs). We then guarantee stability (in the error dynamics) and safety at task execution by augmenting the

nominal motion plan with corrective inputs computed using Control Lyapunov Functions (CLFs) and Control Barrier Functions (CBFs), respectively [18].

Contributions In summary, our contributions are given below.

1) We employ a NODE based offline learning method that captures the invariant features of complex nonlinear and periodic motions, such as wiping and stirring, using only a few demonstrations.
2) We propose a modular motion planner which generates a DS-based reactive motion policy by solving a Quadratic Program (QP) online at high frequency (1 KHz) that augments the nominal NODE-based plan with corrective terms based on CLFs and CBFs to guarantee stability and safety, respectively.
3) We define a novel look-ahead strategy that chooses a target point at every time step for the robot to follow a target trajectory instead of a single target point.
4) We show significant performance improvements over existing methods on the LASA handwriting data set [3], and on periodic 2D and 3D trajectories. We further validate our approach on complex nonlinear and periodic motions with the Franka Emika robot arm.

II. PROBLEM FORMULATION

A DS based motion plan for a robotic manipulator is defined in terms of the state variable $x \in \mathbb{R}^d$, where x is the robot's end-effector Cartesian state relevant to the task. The motion plan is formulated as an autonomous DS

$$\dot{x} = f(x), \tag{1}$$

where $f(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ is a nonlinear continuous function. The demonstrations from kinesthetic teaching is given by $\mathcal{D} := \{x_i(t_1), x_i(t_2), \ldots, x_i(t_T)\}_{i=1}^M$, where, $x_i(t_k)$ is the state of the robot at time t_k, for the ith demonstration. We have a total of M demonstrations. The discrete points in each demonstration are sampled at time $\{t_1, t_2, \ldots, t_T\}$. We assume that the training data trajectories $\mathcal{D}$ approximate an *unknown nominal target trajectory* $z^*(t)$ that encodes the task of the robot such as wiping, stirring, scooping, etc. Our aim is to design a vector field $f(x)$ using the demonstrations $\mathcal{D}$ such that $x(t)$ follows the target trajectory $z^*(t)$. Previous work [3], [13] in the DS-based motion planning framework has considered convergence only to a single target. We consider convergence to a trajectory $z^*(t)$ that can represent more complex, e.g., highly-nonlinear and periodic motions as shown in Fig. 1.

Under nominal circumstances, i.e., in the absence of disturbances or obstacles, the target trajectory $z^*(t)$ should be viewed as the reference for the low level controller to track. However, during deployment, the robot might not always follow the target trajectory because of disturbances, obstacles, time delays, etc. For example, consider the scenario in Fig. 1a, where the target trajectory encodes a periodic wiping motion, and the vector field is an unconstrained DS model (1) learned from demonstrations $\mathcal{D}$. As shown in Fig. 1a, if the robot is perturbed by a disturbance during deployment to

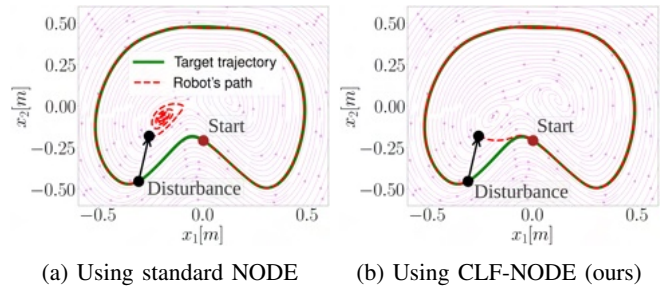(a) Using standard NODE   (b) Using CLF-NODE (ours)

Fig. 1: Illustrative example of a spurious attractor when the robot's path is guided by a DS-based motion plan in the presence of a disturbance (a) using NODE to encode the motion plan and (b) using the corrected CLF-NODE.

a region where there is no training data, the learned model commands the robot to a spurious attractor. However, the desired behaviour is to continue tracking the target trajectory so that the robot wipes the necessary space as shown in Fig. 1b. Since we focus on stability of the error dynamics, the robot's path converge back to the target trajectory, instead of a single goal point [3], [13], which enables the robot to perform more complex and periodic motions. Ensuring robustness to spatial and temporal perturbations is critical for deploying robots in human-centric environments, as disturbances can arise due to obstacles unseen in demonstrations, intentional or adversarial disturbances caused by humans [12], [19], and time delays in the controller [11]. This leads to our formal problem statement:

Given a set $\mathcal{D}$, design a vector field $f(x)$ for the dynamical system (1), such that it generates safe and stable motion plans at deployment for scenarios possibly not seen in the demonstrations, while ensuring that the robot's trajectory $x(t)$ converges to the target trajectory $z^*(t)$.

III. PROPOSED APPROACH

We parameterize the vector field (1) of the motion plan as

$$\dot{x} = \hat{f}(x) + u(x), \tag{2}$$

where, $\hat{f}(x)$ is used to encode the nominal system behavior, and $u(x)$ is used to enforce safety and disturbance rejection properties. We learn the nominal system $\hat{f}(x)$ from demonstrations $\mathcal{D}$, and compute a correction term $u(x)$ based on control theoretic tools so that the goals of stability and safety are met in a composable, modular, and data efficient way. Similar to our objectives, the method proposed in [16] generates motion plans that not only converge to a global goal, but also has local stiff behaviours in regions around the target trajectory. Yet, they still lack in representing complex motions that are periodic, have high curvature and partial divergence. We use a NODE model to represent the complex nominal motion and a modular approach similar to [20], but, we define a CLF with respect to a target trajectory rather than a single goal point that is assumed in [20]. CBFs [18], [21] are widely used in the low level controller to enforce safety of dynamical systems, and we adopt them to generate safe motion plans for robotic manipulation.
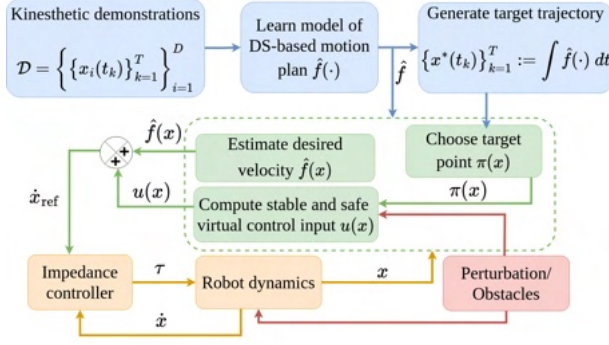
Fig. 2: The control flow of our proposed pipeline. The state of the robot is x, the low level control input are the joint torques τ, and the desired velocity in state space is $\dot{x}_{\text{ref}}$.

A schematic of the proposed control flow is presented in Fig. 2. The blue blocks represent the offline learning component and the green blocks are the online computation modules. We use a neural network parameterized model $\hat{f}$ that we learn from demonstrations $\mathcal{D}$, but any other model class could be used within our proposed modular framework. Starting from an initial condition, we integrate $\hat{f}(x)$ for the same time span of the task given in demonstrations to generate a target trajectory $x^*(t)$ that approximates the unknown nominal target trajectory $z^*(t)$. At deployment, given an observation of the current state x, our planner chooses the target point $\pi(x)$ that the robot should follow using the pre-computed target trajectory $x^*(t)$. The ability to select a target point without relying on time input, unlike prior BC and GP-based methods [9], [11], enhances reactive motion planning by eliminating time delay issues in the control pipeline. We estimate the nominal desired velocity $\hat{f}(x)$ using our learned model. However, as illustrated in Fig. 1, the generated motion plan from $\hat{f}$ is neither guaranteed to be stable nor safe. Hence, we compute a *virtual control input* $u(x)$ as an additive correction term that generates the reference motion plan $\dot{x}$ using (2) so that the trajectory generated by $\dot{x}$ converges to the target trajectory even in the presence of disturbances and unsafe regions such as obstacles. We denote the reference velocity for the low level controller as $\dot{x}_{\text{ref}}$, which in general may be different from the real velocity $\dot{x}$ of the robot. The reference velocity $\dot{x}_{\text{ref}}$ is given as input to the impedance controller [22] that computes the low level control input τ (joint torques) for the physical robotic system. We emphasize that the virtual control input $u(x)$ is different from the low-level control inputs τ given in Fig. 2, and $u(x)$ is a component of the motion planning DS (2).

IV. LEARNING NOMINAL MOTION PLANS USING NEURAL ODEs

We propose a neural network parameterized function class to learn the dynamics of the motion offline from demonstrations. Although existing work [16] learns stable dynamics for motions that converge only to a single target, we aim to learn more complex trajectories that not necessarily converge to a single target. Since neural networks have demonstrated high capability for accurate time-series modeling [23], we base

our approach on Neural ODEs [24], which are the continuous limit (in depth) of ResNets [25]. We parameterize our models of nominal target trajectories as:

$$\frac{d\hat{x}(t)}{dt} = f_\theta(\hat{x}(t)), \tag{3}$$

where $f_\theta(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ is a neural network with parameters θ, and $\hat{x}(t) \in \mathbb{R}^d$ is the state variable predicted by $f_\theta(\cdot)$ at time t. In the forward pass, given the integration time interval $[a, b]$ and an initial point $\hat{x}(a)$, the model outputs the trajectory $\hat{x}(t)$ for $t \in [a, b]$. The trajectory is obtained by solving the differential equation in (3) using a general-purpose differential equation solver based on fifth-order Runge-Kutta [26]. We set $f_\theta(\cdot)$ to be a Multi-Layer Perceptron (MLP), where the inputs and outputs are in $\mathbb{R}^d$. We consider the supervised learning setup with training data $\mathcal{D}$ and solve the empirical risk minimization problem

$$\min_\theta \frac{1}{MT} \sum_{i=1}^{M} \sum_{k=1}^{T} \left\| x_i(t_k) - \hat{x}_i(t_k) \right\|_2^2, \tag{4}$$

to learn the parameters θ, where the predictions of the state $\hat{x}_i(t_k)$ are obtained by integrating (3) with initial condition $\hat{x}_i(t_1) = x_i(t_1)$, $\forall\ i \in \{1, 2, \ldots, M\}$.[1] In contrast to previous work [16], [20] which learns a map $\hat{f}(\cdot)$ using labeled data $\{x(t), \dot{x}(t)\}$, we do not assume access to velocity measurements as they are often not easily collected and/or noisy [28], [29]. Fitting a map to noisy measurements lead to aggressive trajectories at inference that are not desirable for the low-level controller. From our results presented in Fig. 1 and Section VI, we observe that the NODE model generates smooth trajectories utilizing only state variables $x(t)$ to learn f_θ and not their derivatives $\dot{x}(t)$. While such a NODE-based vector field will behave reliably near the training data, unanticipated disturbances or obstacles during deployment might deviate the robot to regions of the state-space where the learned vector field is unreliable, as shown in Fig. 1a. We present a method that computes a correction term to ensure robust and safe tracking of the learned target trajectory.

V. ENFORCING STABILITY AND SAFETY VIA VIRTUAL CONTROL INPUTS

Safety and stability with respect to obstacles and changes in the environment, as illustrated in Fig. 1, are central for tasks in Human-Robot Interaction. Artificial Potential Fields (APFs) [30], [31] have been widely used for obstacle avoidance, but are prone to oscillations and are slow in real-time [30]. A DS based modulation strategy is presented in [32], [33] to avoid obstacles that are limited to convex shapes. We use the CBF based approach that can handle dynamic and multiple non-convex obstacles without sacrificing real-time performance, and is a generalization of both the modulation strategy [32] and APFs [30]. Prior work on CLFs [20] for motion generation of robotic manipulators focused only on point-to-point reaching motions, but we integrate them into our learned NODE model for complex

---

[1] A binomial checkpoint method is used during training for solving (4) as implemented in [27].

periodic motions and present a unified framework that guarantees both stability and safety in real-time.

A. Preliminaries on Safety and Stability

We begin with a review of control theoretic tools that provide sufficient conditions for stability and safety of nonlinear control affine dynamical systems:

$$\dot{x} = g(x) + h(x)u, \tag{5}$$

where, $x \in \mathcal{X} \subset \mathbb{R}^d$ and $u \in \mathcal{U} \subset \mathbb{R}^m$ are the set of allowable states and control inputs, respectively. The DS-based motion plan (2) is a nonlinear control affine DS with $m = d$, $g(x) = \hat{f}(x)$ and $h(x) = I$.

We first define safety with respect to a safe set $\mathcal{C} \subseteq \mathcal{X}$ for the system (5). The safe set $\mathcal{C}$ is defined as the super-level set of a function $B(\cdot) : \mathbb{R}^d \to \mathbb{R}$:

$$\mathcal{C} = \{x \in \mathcal{X} : B(x) \geq 0\}. \tag{6}$$

Our objective is to find a control input u such that the states x that evolve according to the dynamics (5) always stay inside the safe set $\mathcal{C}$. Such an objective is formalized using *forward invariance* of the safe set $\mathcal{C}$. Forward invariance and CBFs are defined as follows [18].

Definition 1: The safe set $\mathcal{C}$ is forward invariant if for every initial point $x(0) = x_0 \in \mathcal{C}$, the future states $x(t) \in \mathcal{C}$ for all $t \geq 0$.

Definition 2: Let $\mathcal{C}$ be the super-level set of a continuously differentiable function $B(\cdot) : \mathbb{R}^d \to \mathbb{R}$ as given in (6). Then, B is a Control Barrier Function (CBF) for the dynamical system (5) and safe set $\mathcal{C}$ if there exists an extended class $\mathcal{K}_\infty$ function $\gamma(\cdot)$ that satisfies

$$\sup_{u \in \mathcal{U}} \nabla_x B(x)^\top (g(x) + h(x)u) \geq -\gamma(B(x)), \ \forall \ x \in \mathcal{X}. \tag{7}$$

The set of all control inputs that satisfy the condition in (7) for each $x \in \mathcal{X}$ is

$$K(x) := \{u \in \mathcal{U} : \nabla_x B(x)^\top (g(x) + h(x)u) \geq -\gamma(B(x))\}. \tag{8}$$

The formal result on safety follows from [18].

Theorem 1: Let B be a Control Barrier Function (CBF) for a safe set $\mathcal{C}$ and a nonlinear control affine system (5). Let $u(x) \in K(x)$ be a locally Lipschitz feedback control law. Then, the following holds: $x(0) \in \mathcal{C} \implies x(t) \in \mathcal{C}$ for all $t \in [0, t_{max})$. If the set $\mathcal{C}$ is compact, then, $\mathcal{C}$ is forward invariant, i.e., $t_{max} = \infty$, and $\mathcal{C}$ is asymptotically stable, i.e., $\lim_{t \to \infty} x(t) \in \mathcal{C}$ for all $x(0) \in \mathcal{X}$.

A *Control Lyapunov Function (CLF)* $V(\cdot)$ is a special case of a CBF. In particular, for $V(\cdot) : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ a positive definite function, if we set $B(\cdot) = -V(\cdot)$ and define the singleton safe set $\mathcal{C} = \{0\}$, then Theorem 1 states that any control law $u(x) \in K(x)$ with $B(\cdot) = -V(\cdot)$ and $\mathcal{C} = \{0\}$ will asymptotically stabilize the system (5) to the origin: see [18] for more details.

Since the inequality in (8) is affine in u, they can be included in efficient optimization-based controllers for control affine systems. We present such an optimization-based planner in Section V-B that has strong stability and safety

guarantees as claimed in Theorem 1, but, with respect to a target trajectory as opposed to a single attractor.

B. Computing the Virtual Control Input

We now show how to incorporate CLFs and CBFs into the DS-based motion plan (2) to generate safe and stable motions that converge to a target trajectory, not just a single attractor that is presented in prior work [3], [13]. In particular, we use the learned NODE f_θ to generate a nominal motion plan, and compute $u(x)$ using CLFs and CBFs to enforce stability and safety, resulting in a motion plan of the form:

$$\dot{x} = f_\theta(x) + u(x), \tag{9}$$

where x is the state of the robot, and $u(x)$ is the virtual control input.

1) Stability using Control Lyapunov Functions: We utilize CLFs described in Section V-A to generate a motion plan that always converge to the target trajectory $x^*(t)$ even in the presence of disturbances. We note that $x^*(t)$ is different from the unknown target trajectory $z^*(t)$ introduced in Section I. Previous work [20] have utilized CLFs only for convergence to a single target point. In contrast, we present a framework that integrates Neural ODEs for rich behaviors, CLFs to ensure convergence to a target trajectory $x^*(t)$, and CBFs for safety. To that end, we first define the error $e(t)$ between the robot state and the target trajectory: $e(t) = x(t) - x^*(t)$. For ease of notation, we drop the explicit dependence on time t, and write e, x, and x^* for the current error, state, and target point at time t, respectively. From (9), the error dynamics are given by

$$\dot{e} = \dot{x} - \dot{x}^* \Rightarrow \dot{e} = f_\theta(x) - \dot{x}^* + u(x). \tag{10}$$

The error dynamics (10) define a nonlinear control affine system (5), where the state of the system is e, and $u(x)$ is the control input. Hence, by Theorem 1, if there exists a CLF $V(\cdot) = -B(\cdot)$ for the error dynamics (10), then, any feedback virtual control law $u(\cdot)$ that satisfies

$$\nabla_{e(t)} V(e)^\top (f_\theta(x) - \dot{x}^* + u(x)) \leq -\alpha(V(e)), \ \forall \ e \in \mathbb{R}^d \tag{11}$$

will drive the error asymptotically to zero, where $\alpha(\cdot)$ is a class $\mathcal{K}_\infty$ function for CLF, which is introduced to differentiate from $\gamma(\cdot)$ in Definition 2 for the CBF. During online motion planning, given the current state of the robot x and information about the target trajectory x^*, we compute the minimal control effort $u(x)$ that satisfies (11) by setting

$$u(x) = \arg\min_v \ \|v\|_2^2 \tag{12}$$
$$\text{s.t.} \quad \nabla_e V(e)^\top (f_\theta(x) - \dot{x}^* + v) \leq -\alpha(V(e)),$$

where $\alpha(\cdot)$ defines how aggressively the robot tracks the target trajectory. We describe how we choose x^* and $\dot{x}^*$ in detail in Section V-B.3. The optimization problem (12) is a quadratic program with a single affine inequality and has a closed form solution [20]. The Lyapunov function we use is $V(e) = \|e\|_2^2$, but, note that any positive definite function is a valid CLF. The presence of the virtual actuation term v makes the optimization problem (12) always feasible. We refer the

reader to Fig. 1 to differentiate between the paths generated by only $f_\theta(\cdot)$, and by (9) using the correction term $u(\cdot)$. We refer to this approach as the CLF-NODE.

2) Safety using Control Barrier Functions: We build on the framework in Section V-A by integrating CBFs into the virtual control input computation to guarantee safety for the generated motion plan. We define safety with respect to a safe set $\mathcal{C} \subseteq \mathcal{X}$ as described in Section V-A for the system (9). From Theorem 1, if there exists a CBF $B(\cdot)$ for the dynamics (9), then, any feedback control law $u(\cdot)$ that satisfies

$$\nabla_x B(x)^\top (f_\theta(x) + u(x)) \geq -\gamma(B(x)), \ \forall \ x \in \mathcal{X} \quad (13)$$

will render the system (9) safe, where, $\gamma(\cdot)$ is an extended class $\mathcal{K}_\infty$ function. At inference, the DS-based motion plan is still given by (9), but the virtual control input $u(x)$ is computed such that it satisfies the CBF condition in (13) for the dynamics $\dot{x}$ and a given CBF $B(\cdot)$ for the safe set $\mathcal{C}$.

In cases where an obstacle obstructs the robot moving along the nominal trajectory, the robot should automatically avoid the obstacle, but converge back to complete the desired task when possible. However, this may lead to a conflict between preserving safety and stability: during the obstacle avoidance phase, the CLF constraint in (12) may be violated as the robot takes safety preserving actions that increase tracking error. We prioritize safety and adapt the approach proposed in [18] for balancing these competing objectives to our setting, and solve an optimization problem with the CBF condition (13) as a hard constraint and the CLF condition (11) as a soft constraint. Given the current state of the robot x and the target point x^*, the optimization problem that guarantees a safe motion plan is

$$(u(x), \_) = \underset{\{v, \epsilon\}}{\arg\min} \ \|v\|_2^2 + \lambda \epsilon^2$$

$$\text{s.t.} \quad \nabla_x B(x)^\top (f_\theta(x) + v) \geq -\gamma(B(x)) \quad (14)$$

$$\nabla_e V(e)^\top (f_\theta(x) - \dot{x}^* + v) \leq -\alpha(V(e)) + \epsilon$$

where ϵ is a relaxation variable to ensure feasibility of (14) and is penalized by $\lambda > 0$. The problem in (14) is a parametric program, where the parameters of interest are $\{x, x^*, \dot{x}^*\}$. We abuse notation and denote the optimal virtual control input $u(x)$ for (14) to be $u^*(x, x^*, \dot{x}^*)$ which will be used in the next section. Problem (14) is a QP that we solve efficiently in real-time using OSQP solver [34]. Multiple CBFs and CLFs can be composed in a way analogous to problem (14) to represent multiple obstacles of complex non-convex shapes as given in Section VI. We refer to this approach as the CLF-CBF-NODE.

3) Choosing a Target Point: As shown in Fig. 2, we first integrate the learnt model $f_\theta(\cdot)$ offline to generate the *target array* $\mathcal{T} := \{x^*(t_k)\}_{k=1}^T$ from a given initial condition $x^*(t_1)$. The target array $\mathcal{T}$ is exactly the target trajectory $x^*(t)$, but only at time steps $\{t_k\}_{k=1}^T$. Given an observation of the current state of the robot x at time t, and the target array $\mathcal{T}$, we select the next target point x^* for the robot to follow using the map $\pi(x)$ defined in Algorithm 1. We remove the direct dependence of the target point x^* on time t, which leads to a more reactive motion plan that adapts to

---

Algorithm 1: Choose target point

Input: $\mathcal{T} := \{x^*(t_k)\}_{k=1}^T, f_\theta(\cdot), x, N$
Output: $\pi(x)$
$m \leftarrow \arg\min_k \|x - x^*(t_k)\|_2$;
$\mathcal{T}_N := \{x^*(t_m), x^*(t_{m+1}), \ldots, x^*(t_{m+N-1})\}$;
Solve (14) with parameters $\{x, y, f_\theta(y)\}$ for each $y \in \mathcal{T}_N$;
$\pi(x) \leftarrow \arg\min_{y \in \mathcal{T}_N} \|u^*(x, y, f_\theta(y))\|_2^2$;

---

both the time delays that are often present during online deployment, and to unforeseen perturbations of the robot away from the nominal plan, e.g., due to human interaction or obstacle avoidance. The look-ahead horizon length N is used to construct the array $\mathcal{T}_N$, consisting of N future points starting at the current target state $x^*(t_m)$. We choose the target point $\pi(x)$ from $\mathcal{T}_N$ that results in the smallest norm of virtual control input when solving (14) for all $y \in \mathcal{T}_N$. We use a forward looking horizon N to ensure the robot moves forward along the target trajectory, which is illustrated in the video submission. To the best of our knowledge, this is the first time that the norm of the correction input $u(\cdot)$ is used as a metric for choosing an appropriate nearest neighbor point in motion planning. We use $\dot{x}^* := f_\theta(\pi(x))$, since $\pi(x) \in \mathcal{T}$ and we obtained the target array $\mathcal{T}$ by integrating $f_\theta(\cdot)$.

VI. Experimental Validation and Results

A. LASA handwriting dataset

We validate our approach on the LASA handwriting data set [3] that contains 30 2D nonlinear motions. Each motion set has 7 demonstrations: we use 4 as the training set, and the remaining 3 as the test set. In Fig. 3, we compare the performance of our NODE model with two existing DS-based learning approaches: SEDS [3] and LPV-DS [13] using the mean and variance of Dynamic Time Warping (DTW) distance [35]. DTW distance measures the dissimilarity between the demonstrations and the corresponding reproductions starting from the same initial condition. We note that although SEDS and LPV-DS use velocity data for regression, which our approach does not have access to, the mean of the DTW distance for our NODE approach is approximately half of the existing methods [3], [13]. We illustrate disturbance rejection in Fig. 4a using CLF-NODE and obstacle avoidance in Fig. 4b using CLF-CBF-NODE with a circular obstacle and a non-convex obstacle. The barrier function $B(\cdot)$ is defined to be less than zero inside the obstacles, and we choose linear $\alpha(\cdot)$ and $\gamma(\cdot)$ functions.

B. Periodic trajectories

We validate our approach on handwritten 2D periodic motions of the letters **I, R, O** and **S** given in [17] and 3D periodic trajectories that encode three wiping tasks as given in Figs. 5e, 6a, and 6b. We compare our method with Imitation Flow (IFlow) [17] and a Gaussian Process (GP) [11] based approach, since SEDS and LPV-DS compared in Section VI-A model trajectories that converge only to a single attractor. IFlow is based on normalizing flows that learns stable motions using prior knowledge on whether the underlying dynamics of the demonstrations has a single attractor or a limit cycle, but our approach requires no prior knowledge. We present the DTW distance in Fig. 5a, training
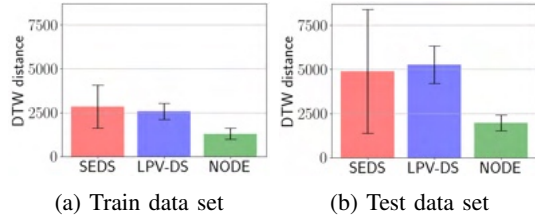
(a) Train data set (b) Test data set

Fig. 3: Comparison of DTW distance on the LASA data set.

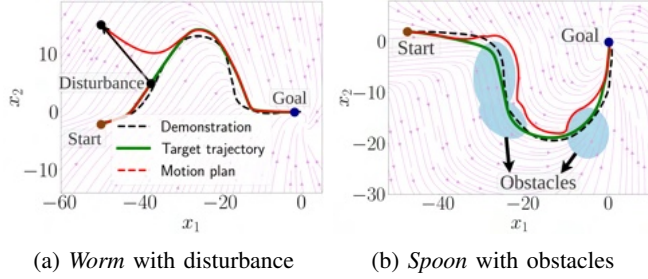(a) *Worm* with disturbance (b) *Spoon* with obstacles

Fig. 4: Illustration of (a) disturbance rejection using CLF-NODE and (b) obstacle avoidance using CLF-CBF-NODE.

time in Fig. 5b and execution time in Fig. 5c. The execution time is the computation time for a single forward pass of the model to generate the entire trajectory from a given initial point and the time span of the demonstration. We also compare the trajectory reproductions for the **R** shape in Fig. 5d, and for the spiral wiping task by the Franka robot arm in Fig. 5e. The IFlow approach is not able to learn the complex motion for spiral wiping, but our NODE approach learns with high accuracy and lesser computation time. The execution time comparison in Fig. 5c is plotted in log scale and we note that our approach (NODE) has much lesser execution time, which is important for real-time robot experiments. Although the GP based method [11] learns complex trajectories with comparable accuracy and training time to NODE, the execution time is much smaller and they rely on time inputs for desired roll outs with no capability to generate safe and stable motion plans. All computations are performed on Google Colab.

C. Robotic experiments

We validate our approach on the Franka robot arm performing complex periodic motions: wiping a mannequin with a towel (Fig. 6a) and wiping a white board with an eraser (Fig. 6b). We use the passive DS impedance controller [36]. We used 2 demonstrations for the mannequin task, and 3 demonstrations for the board wiping task. Each demonstration had between 300 and 600 data samples. The average training time (offline) is $3 - 6$ minutes for each task on Google Colab. The obstacle shown in Fig. 6b at $t = 2$ has markers on it that are tracked in real-time by OptiTrack motion capture system. We observe that the robot tracks the desired nominal trajectories while remaining compliant to human interaction, robust to perturbations, and safe with respect to unforeseen dynamic obstacles. The online motion planning module runs at 1KHz that solves (14) and implements Algorithm 1. We include the stirring task, the effect of Algorithm 1 and some preliminary results for full pose (position in $\mathbb{R}^3$ and orientation in $\mathcal{SO}(3)$) mo-
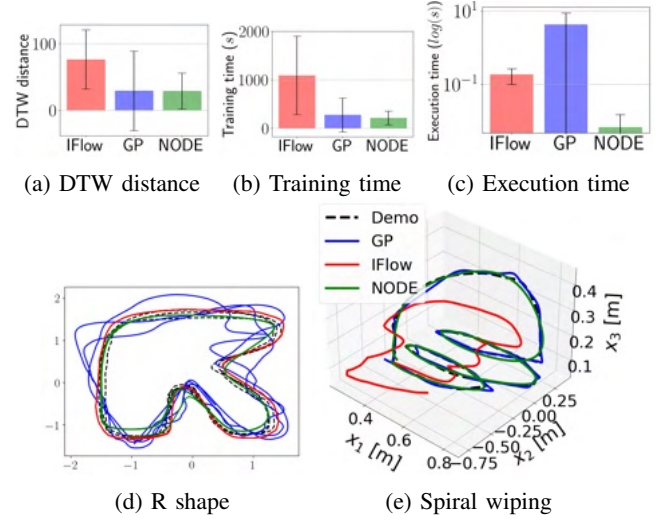

(a) DTW distance (b) Training time (c) Execution time

(d) R shape (e) Spiral wiping

Fig. 5: Comparison of performance metrics and trajectory reproductions between IFlow, GP and our approach (NODE) on periodic trajectories.
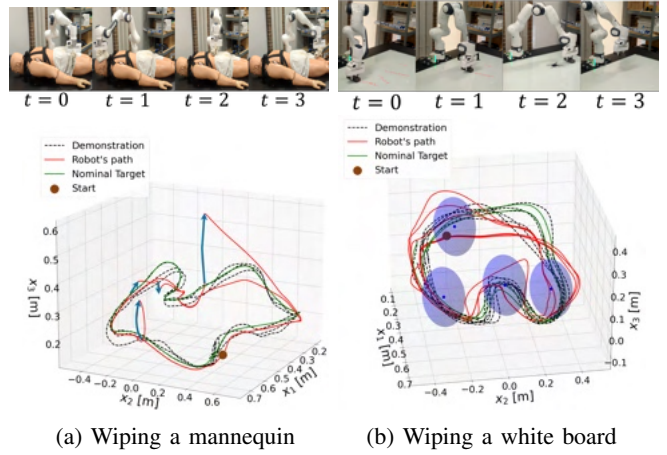

(a) Wiping a mannequin (b) Wiping a white board

Fig. 6: The robot performing different wiping tasks with complex periodic motions. The blue arrows denote the perturbations, and the purple spheres is the dynamic obstacle at different time.

tion planning in `https://sites.google.com/view/lfd-neural-ode/home` and the video submission.

VII. CONCLUSION & FUTURE WORK

We propose a modular DS-based motion planner using NODE with an additive CLF and CBF-based correction term that guarantees stability and safety on the error dynamics for convergence to a target trajectory, rather than a single attractor. We validate our approach on complex non-linear and periodic handwritten motions and on the Franka robot arm for wiping and stirring tasks.

Our robot experiments are limited to the Cartesian end-effector position: future work will address this by extending our approach to higher dimensions such as (a) orientation in $\mathcal{SO}(3)$ and position in $\mathbb{R}^3$ of the end-effector [15], [37], [38], (b) full pose of end-effector in $\mathcal{SE}(3)$ [39], and (c) joint space [40] that can include joint limits and avoid collision with arm links. Future work will also incorporate the CLF-CBF NODE approach with end-to-end learning frameworks using image observations [41], [42].

REFERENCES

[1] R. Ma, J. Chen, and J. Oyekan, "A learning from demonstration framework for adaptive task and motion planning in varying package-to-order scenarios," *Robotics and Computer-Integrated Manufacturing*, vol. 82, p. 102539, 2023.

[2] J. Yang, J. Zhang, C. Settle, A. Rai, R. Antonova, and J. Bohg, "Learning periodic tasks from human demonstrations," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8658–8665.

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

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

[5] B. Akgun and K. Subramanian, "Robot learning from demonstration : Kinesthetic teaching vs . teleoperation," 2011.

[6] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

[7] M. C. Priess, J. Choi, and C. Radcliffe, "The inverse problem of continuous-time linear quadratic gaussian control with application to biological systems analysis," in *Dynamic Systems and Control Conference*, vol. 46209. American Society of Mechanical Engineers, 2014, p. V003T42A004.

[8] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[9] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[10] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5628–5635.

[11] N. Jaquier, D. Ginsbourger, and S. Calinon, "Learning from demonstration with model-based gaussian process. arxiv preprint arxiv: 191005005," 2019.

[12] A. Billard, S. Mirrazavi, and N. Figueroa, *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach*. MIT Press, 2022.

[13] N. Figueroa and A. Billard, "A physically-consistent bayesian non-parametric mixture model for dynamical system learning," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 87. PMLR, 29–31 Oct 2018, pp. 927–946.

[14] S. S. Mirrazavi Salehian, N. Figueroa, and A. Billard, "A unified framework for coordinated multi-arm motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1205–1232, 2018.

[15] N. Figueroa, S. Faraji, M. Koptev, and A. Billard, "A dynamical system approach for adaptive grasping, navigation and co-manipulation with humanoid robots," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 7676–7682.

[16] N. Figueroa and A. Billard, "Locally active globally stable dynamical systems: Theory, learning, and experiments," *The International Journal of Robotics Research*, vol. 41, no. 3, pp. 312–347, 2022.

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

[18] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[19] Y. Wang, N. Figueroa, S. Li, A. Shah, and J. Shah, "Temporal logic imitation: Learning plan-satisficing motion policies from demonstrations," in *6th Annual Conference on Robot Learning*, 2022. [Online]. Available: https://openreview.net/forum?id=ndYsaoyzCWv

[20] S. M. Khansari-Zadeh and A. Billard, "Learning control lyapunov function to ensure stability of dynamical system-based robot reaching

[21] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning control barrier functions from expert demonstrations," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 3717–3724.

[22] K. Kronander and A. Billard, "Passive interaction control with dynamical systems," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 106–113, 2016.

[23] R. T. Chen, B. Amos, and M. Nickel, "Learning neural event functions for ordinary differential equations," *arXiv preprint arXiv:2011.03902*, 2020.

[24] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," 2019.

[25] E. Haber and L. Ruthotto, "Stable architectures for deep neural networks," *Inverse problems*, vol. 34, no. 1, p. 014004, 2017.

[26] J. C. Butcher, "A history of runge-kutta methods," *Applied numerical mathematics*, vol. 20, no. 3, pp. 247–260, 1996.

[27] P. Kidger, "On neural differential equations," *arXiv preprint arXiv:2202.02435*, 2022.

[28] S. Purwar, I. N. Kar, and A. N. Jha, "Adaptive output feedback tracking control of robot manipulators using position measurements only," *Expert systems with applications*, vol. 34, no. 4, pp. 2789–2798, 2008.

[29] B. Xiao, L. Cao, S. Xu, and L. Liu, "Robust tracking control of robot manipulators with actuator faults and joint velocity measurement uncertainty," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 3, pp. 1354–1365, 2020.

[30] A. Singletary, K. Klingebiel, J. Bourne, A. Browning, P. Tokumaru, and A. Ames, "Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8129–8136.

[31] Y. Chen, L. Chen, J. Ding, and Y. Liu, "Research on real-time obstacle avoidance motion planning of industrial robotic arm based on artificial potential field method in joint space," *Applied Sciences*, vol. 13, no. 12, p. 6973, 2023.

[32] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, pp. 433–454, 2012.

[33] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 2587–2592.

[34] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[35] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[36] K. Kronander and A. Billard, "Passive interaction control with dynamical systems," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 106–113, 2016.

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

[38] J. Zhang, H. B. Mohammadi, and L. Rozo, "Learning riemannian stable dynamical systems via diffeomorphisms," in *6th Annual Conference on Robot Learning*, 2022.

[39] J. Urain, D. Tateo, and J. Peters, "Learning stable vector fields on lie groups," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 569–12 576, 2022.

[40] Y. Shavit, N. Figueroa, S. S. M. Salehian, and A. Billard, "Learning augmented joint-space task-oriented dynamical systems: A linear parameter varying and synergetic control approach," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2718–2725, 2018.

[41] D. Totsila, K. Chatzilygeroudis, D. Hadjivelichkov, V. Modugno, I. Hatzilygeroudis, and D. Kanoulas, "End-to-end stable imitation learning via autonomous neural dynamic policies," *arXiv preprint arXiv:2305.12886*, 2023.

[42] S. Bahl, M. Mukadam, A. Gupta, and D. Pathak, "Neural dynamic policies for end-to-end sensorimotor learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5058–5069, 2020.

motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014.