Learning Lyapunov-Stable Polynomial Dynamical Systems Through Imitation

Anonymous Author(s)
Affiliation
Address
email

Abstract: Imitation learning is a paradigm to address complex motion planning problems by learning a policy to imitate an expert's behavior. However, relying solely on the expert's data might lead to unsafe actions when the robot deviates from the demonstrated trajectories. Stability guarantees have previously been provided utilizing nonlinear dynamical systems, acting as high-level motion planners, in conjunction with the Lyapunov stability theorem. Yet, these methods are prone to convergence issues or inaccurate policies upon replicating complex and highly nonlinear trajectories. To mitigate this problem, we present an approach for learning a globally stable nonlinear dynamical system as a motion planning policy. We model the nonlinear dynamical system as a parametric polynomial and learn the polynomial's coefficients jointly with a Lyapunov candidate. To showcase its success, we compare our method against the state of the art in simulation and conduct real-world experiments with the Kinova Gen3 Lite manipulator arm. Our experiments demonstrate that our method is data efficient and is capable of more accurate reproduction of a vast variety of expert demonstrations while remaining stable in the face of perturbations.

Keywords: Imitation learning, Safe learning, Motion planning, Dynamical system, Semi-definite programming, Robotic manipulation

1 Introduction

Motion planning for robotic systems is generally regarded as a decomposition of a desired motion into a series of configurations that potentially satisfy a set of constraints [1]. Imitation learning tackles motion planning by imitating an expert's behavior to learn a planning policy consistent with demonstrated trajectories [2]. To this day, only a handful of imitation learning methods provide mathematical stability guarantees for their resultant policy. Stability is a critical factor when deploying imitation policies in environments exposed to external perturbations. Therefore, unpredictable environments require a policy that reasonably responds in unexplored regions of state-space, away from original demonstrations.

Researchers have turned to autonomous dynamical systems (DS) as a means to learn stable motion planning policies [3, 4, 5, 6]. Essentially, a parametric time-invariant DS is optimized to provide an action (velocity) given the current state (position), while adhering to constraints that attain global Lyapunov stability. This approach leads to safety and predictability of planned trajectories, even in areas of state-space without expert demonstrations. Previous work is mostly confined to simple Lyapunov functions. The choice of Lyapunov candidate affects the performance when these methods are applied to highly nonlinear trajectories, for instance, demonstrations where the distance to target is not decreased monotonically.

We propose a method to simultaneously learn a polynomial dynamical system (PLYDS) and a polynomial Lyapunov candidate to generate globally stable imitation policies. Polynomials, depending on

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

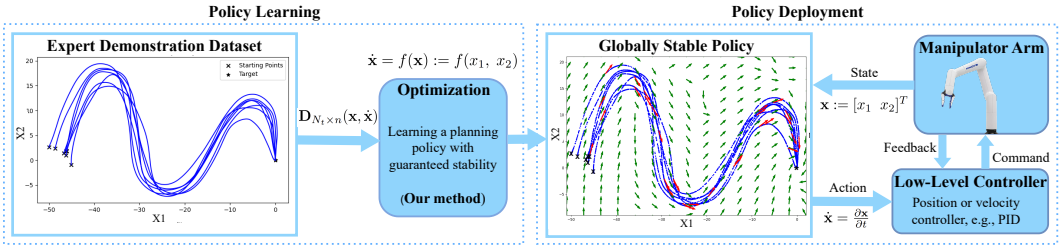


Figure 1: Overview of the stable policy learning framework. Policy learning (left) optimizes a stable polynomial DS from expert's demonstration data. The policy (green and red arrows) is then deployed (right) to plan globally stable and safe trajectories.

the degree, possess an expressive power to approximate highly nonlinear systems, and polynomial regression can empirically compete with neural networks on challenging tasks [7, 8]. Unlike most neural policies, global stability can be naturally expressed with polynomials. Polynomials also enable us to utilize efficient semi-definite programming [9, 10] and sum-of-squares (SOS) optimization techniques [11, 12], and offer adaptability to expert's demonstrations.

Our main contribution is twofold. We propose a polynomial representation of the planning policy and Lyapunov candidate function, coupled with concrete mathematical stability certification, for precise and safe replication of the expert's demonstrations. Then, we define a regularized semi-definite optimization problem to jointly learn the DS and the Lyapunov candidate with higher flexibility and precision. We compare the reproduction accuracy of PLYDS with alternatives in the literature and evaluate the performance in both simulation and real robotic systems. The process of the execution is illustrated in Figure 1.

2 Background and Notation

Consider a system operating in a state-space $\mathcal{X} \subset \mathbb{R}^n$, e.g., a robot in its task- or configuration-space. The system can execute actions in $\mathcal{A} \subset \mathbb{R}^n$, for instance, velocity or torque commands, leading to state evolution. We denote the state variable with $\mathbf{x} \triangleq [x_1\ x_2\ \dots\ x_n]^T \in \mathcal{X}$, and consider the action variable to be the state's derivative $\dot{\mathbf{x}} \in \mathcal{A}$. Within this space, our goal is to learn an imitation policy through a dataset of experts' state-action pairs, referred to as trajectories.

Let $N_d \in \mathbb{N}$ be the number of trajectories demonstrated by the expert. Each trajectory contains $N_s \in \mathbb{N}$ state-action pairs. The dataset of expert trajectories stacks all state-action pairs, defined as:

$$\mathbf{D} \triangleq \Big\{ \big(\mathbf{x}^d(s),\ \dot{\mathbf{x}}^d(s)\big) \ \big|\ d \in \{1, \dots, N_d\},\ s \in \{1, \dots, N_s\} \Big\}, \tag{1}$$

where $(\mathbf{x}^d(s),\ \dot{\mathbf{x}}^d(s))$ is the dataset entry corresponding to the s-th sample of the d-th demonstrated trajectory. The dataset $\mathbf{D}$ holds $N_t = N_d N_s$ samples. We assume that the trajectories contain the same sample size (N_s), share a common target ($\mathbf{x}^* \in \mathcal{X}$), and have zero velocity at the target, i.e., $\mathbf{x}^d(N_s) = \mathbf{x}^*$ and $\dot{\mathbf{x}}^d(N_s) = \mathbf{0}$ for all trajectories $d \in \{1, \dots, N_d\}$.

Definition 2.1. *(Dynamical Systems). The mapping between the state and the action in each sample can be modelled with a time-invariant autonomous dynamical system (DS), denoted by:*

$$\dot{\mathbf{x}} = f(\mathbf{x}) + \epsilon = \hat{f}(\mathbf{x}), \quad f, \hat{f} : \mathcal{X} \to \mathcal{A}. \tag{2}$$

In Equation (2), f is an ordinary differential equation for the true underlying DS. The term $\epsilon \in \mathbb{R}^n$ captures measurement and recording noise of expert's demonstrations. Following [3], we aim at learning a noise-free estimation of $f(\mathbf{x})$, denoted by $\hat{f}(\mathbf{x})$. One can view $\hat{f}(\mathbf{x})$ in Equation (2) as a *policy* that maps states to actions for reproducing the demonstrated trajectories in the state-space. For instance, when the robot is located in $\mathbf{x}_0 \in \mathcal{X}$, the policy yields an action $\dot{\mathbf{x}}_0 = \hat{f}(\mathbf{x}_0)$, which can be passed to the robot's velocity controller.

The estimated DS in Equation (2), $\hat{f}(\mathbf{x})$, is globally asymptotically stable (GAS) around an equilibrium point $\mathbf{x}^e$, if and only if for every initial state, $\mathbf{x} \to \mathbf{x}^e$ as the system evolves and time goes to infinity [13]. A popular tool to study the GAS property of a DS is the Lyapunov stability theorem. According to this theorem, a DS exhibits GAS if there exists a positive-definite function $v : \mathcal{X} \to \mathbb{R}$, known as Lyapunov potential function (LPF), such that $\dot{v}(\mathbf{x}) < 0$ for all $\mathbf{x} \neq \mathbf{x}^e$ and $\dot{v}(\mathbf{x}^e) = 0$. To ensure GAS for $\hat{f}(\mathbf{x})$, we simultaneously learn the policy, $\hat{f}$, and the LPF, v.

3 Related work

Extensive research is conducted on imitation learning and its applications in robotic motion planning for a variety of tasks. Existing efforts can be divided into the following predominant research tracks.

Dynamical systems for motion planning. Dynamical systems have proved to effectively counter autonomous motion planning problems by proposing a time-invariant policy [14]. Traditional methods of encoding trajectories are based on spline decomposition [15], Gaussian process regression [16], or unstable dynamical systems [17, 18]. They either lack robustness because of time-variance or fail to provide GAS. SEDS [3] is the first attempt to learn stable planning policies. However, its performance declines when applied to highly nonlinear expert trajectories. Most notably, it suffers from trajectories where the distance to the target is not monotonically decreasing. The intrinsic limitation of SEDS comes from the choice of a simple Lyapunov function. Follow-up research introduces more complex Lyapunov candidates to stably mimic nonlinear trajectories [4, 19, 20], but are still restricted in representing the Lyapunov candidate. Others have tried to tackle SEDS limitations through diffeomorphic transformations [5] or contraction theory [21].

Imitation learning. Recent imitation learning developments can be applied to motion planning tasks with minimal modifications, since motion planning can be achieved by finding a (not necessarily stable) policy in the robot's task-space from the expert's behavior. For instance, GAIL [22] introduces an adversarial imitation learning approach that directly optimizes the expert's policy, but requires a large set of expert's data (low sample efficiency) and extensive training iterations. The growing interest in neural policies has also led to the development of end-to-end autonomous driving [23] and behavioral cloning [24, 25, 26] methods. Nevertheless, they generally lack GAS, and it is unclear whether the robot can recover from perturbations. The same drawbacks exist with apprenticeship learning approaches, such as [27] and inverse reinforcement learning, such as [28], and the computational demand is even higher for the latter.

Stability in neural dynamical systems. Methods such as [29] and [30] represent the dynamics with a Neural Network, and propose the joint training of dynamics and a Lyapunov function to guarantee the stability. Though theoretically sound, these methods have only been applied to rather simple settings and require large demonstration sets. Neural Lyapunov methods [31, 32, 33] promise data driven and potentially stable approach to control and model nonlinear dynamics, but typically lack GAS since they softly enforce the Lyapunov stability conditions.

4 Methodology

We instantiate the policy and the corresponding LPF candidate, $\hat{f}$ and v, with two polynomials in Section 4.1 and Section 4.2, respectively. This allows us to accurately imitate an expert's behavior, while providing formal GAS guarantees. Subsequently, we formulate a tractable optimization problem for jointly learning the policy and the LPF in Section 4.3.

4.1 Dynamical system policy formulation

We need to approximate the unknown underlying DS in Equation (2) to discover the mapping between states and actions from expert's behavior. To this end, we opt to model the policy with a parametric polynomial. The representative power of polynomials was originally established through the Weierstrass approximation theorem, stating that every continuous function defined on a closed

interval can be approximated with desired precision by a polynomial. This idea is fortified by recent studies, such as [7, 8], that compare polynomials to neural networks on a variety of tasks.

Definition 4.1. *(Polynomial Dynamical Systems). A Polynomial Dynamical System (PLYDS) is a polynomial approximation of the policy in Equation (2), and is expressed as,*

$$\dot{\mathbf{x}} = \hat{f}(\mathbf{x};\ \mathbf{P}) \triangleq \left[\mathbf{b}_{\mathbf{x},\alpha}^T \mathbf{P}_1 \mathbf{b}_{\mathbf{x},\alpha}\ \ \mathbf{b}_{\mathbf{x},\alpha}^T \mathbf{P}_2 \mathbf{b}_{\mathbf{x},\alpha}\ \ \dots\ \ \mathbf{b}_{\mathbf{x},\alpha}^T \mathbf{P}_n \mathbf{b}_{\mathbf{x},\alpha}\right]^T, \tag{3}$$

where $\mathbf{b}_{\mathbf{x},\alpha} \triangleq [1 \quad (\mathbf{x}^T)^{\circ 1} \quad (\mathbf{x}^T)^{\circ 2} \dots (\mathbf{x}^T)^{\circ \alpha}]^T$ *is the polynomial basis of degree* $\alpha \in \mathbb{N}$*, and* $(\mathbf{x}^T)^{\circ k}$ *is the element-wise* k*-th power of* $\mathbf{x}^T$*. Every row* i *of* $\hat{f}$ *is a polynomial of degree* 2α*,* $\hat{f}_i(\mathbf{x};\ \mathbf{P}_i) = \mathbf{b}_{\mathbf{x},\alpha}^T \mathbf{P}_i \mathbf{b}_{\mathbf{x},\alpha}$*, where* $\mathbf{P}_i \in \mathbb{S}^{\alpha n+1}$ *and* $\mathbb{S}^k \triangleq \{S \in \mathbb{R}^{k\times k} | S^T = S\}$*. The matrix* $\mathbf{P} \in \mathbb{S}^{\alpha n^2+n}$ *encapsulates the block-diagonal form of all* $\mathbf{P}_i$ *matrices.*

Below, we present an example to show how PLYDS, as defined in Definition 4.1, captures nonlinear time-invariant policies. One can further complicate the policy by increasing α, which in turn produces a larger basis vector and a more flexible polynomial.

Example 4.1.1. *A second-order polynomial representation of a one-dimensional DS is:*

$$\dot{\mathbf{x}} = \hat{f}(\mathbf{x};\ \mathbf{P}) = \left[\mathbf{b}_{\mathbf{x},\alpha}^T\ \mathbf{P}_1\ \mathbf{b}_{\mathbf{x},\alpha}\right] = [1 \quad x] \begin{bmatrix} p_{00} & p_{01} \\ p_{01} & p_{11} \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} = p_{00}x^2 + (p_{01} + p_{10})x + p_{00},$$

where $\alpha = 1, \mathbf{b}_{\mathbf{x},\alpha} = [1\ x]^T$*. Note how* $\mathbf{P}$ *can be symmetric without loss of generality.*

4.2 Global stability guarantees for polynomial dynamical systems

As explained in Section 4.1, a polynomial policy allows for accurately imitating the expert's demonstrations. Yet, there is no formal GAS guarantee that the robot will ultimately converge to the target in the face of perturbations, deflecting it from the expert's trajectories. Owing to the Lyapunov stability theorem, finding an LPF that meets the criteria in Section 2 ensures the desired stability [34].

The major challenge lies in learning an LPF, v, which is a positive definite function with negative gradient. We tackle this by confining to the class of polynomial LPF candidates.

Definition 4.2. *(Polynomial Lyapunov Candidate). A multidimensional polynomial LPF is given by,*

$$v(\mathbf{x};\ \mathbf{Q}) \triangleq \left[\mathbf{b}_{\mathbf{x},\beta}^T \mathbf{Q}_1 \mathbf{b}_{\mathbf{x},\beta}\ \ \mathbf{b}_{\mathbf{x},\beta}^T \mathbf{Q}_2 \mathbf{b}_{\mathbf{x},\beta}\ \ \dots\ \ \mathbf{b}_{\mathbf{x},\beta}^T \mathbf{Q}_n \mathbf{b}_{\mathbf{x},\beta}\right]^T,\ v : \mathcal{X} \to \mathbb{R}^n, \tag{4}$$

where $\beta \in \mathbb{N}$ *is the polynomial basis degree. Each row is defined by* $v_i(\mathbf{x};\ \mathbf{Q}_i) = \mathbf{b}_{\mathbf{x},\beta}^T \mathbf{Q}_i \mathbf{b}_{\mathbf{x},\beta}, v_i : \mathcal{X} \to \mathbb{R}$*, and can be viewed as a scalar Lyapunov function. The parameters matrix,* $\mathbf{Q} \in \mathbb{S}^{\beta n^2+n}$*, is a block-diagonal of all* $\mathbf{Q}_i \in \mathbb{S}^{\beta n+1}$ *matrices.*

Definition 4.2 introduces a non-conventional LPF candidate. Rather than considering a single LPF, we designate a distinct polynomial LPF for each dimension of the state-space and stack them into $v(\mathbf{x};\ \mathbf{Q})$. This characterization is less restrictive than a single polynomial LPF and enables the policy and LPF to be learned independently for each dimension of the state-space.

We highlight that the GAS of the policy in each dimension, $\hat{f}_i(\mathbf{x};\ \mathbf{P}_i)$, implies the GAS of the entire policy, $\hat{f}(\mathbf{x};\ \mathbf{P})$. Proposition 4.3 establishes a link between the policy stability in each row and the global stability of the multidimensional policy.

Proposition 4.3. *Assuming each pair* $(\hat{f}_i(\mathbf{x};\ \mathbf{P}_i),\ v_i(\mathbf{x};\ \mathbf{Q}_i))$ *individually satisfies the GAS conditions. Then, the sum* $\hat{v} = \sum_{i=1}^n v_i(\mathbf{x};\ \mathbf{Q}_i)$ *yields a valid standard Lyapunov function for* $\hat{f}(\mathbf{x};\ \mathbf{P})$*, proving that the policy satisfies GAS conditions. The proof is given in Appendix A.1.*

The formulation of the policy and the LPF as multidimensional polynomials empowers us to leverage tools from sum-of-squares (SOS) [12, 35]. The SOS approach boils the Lyapunov GAS conditions down to verifying positive-definiteness of a set of specified matrices. The next two lemmas illustrate the SOS formulation of Lyapunov stability conditions.

Lemma 4.4. *The first Lyapunov stability criterion,* $v_i(\mathbf{x};\ \mathbf{Q}_i) \succeq 0$*, is satisfied for each* $i \in \{1, \dots, n\}$ *if* $\mathbf{Q}_i \succeq 0$ *and* $\mathbf{Q}_i \in \mathbb{S}^{\beta n+1}$*. The proof is outlined in Appendix A.2.*

Lemma 4.5. *The second Lyapunov criterion, $\frac{\partial}{\partial t} v_i(\mathbf{x};\ \mathbf{Q}_i) \prec 0$, is fulfilled for each $i \in \{1, \dots, n\}$ if there exists a symmetric matrix $\mathbf{G}_i \prec 0$ and $\mathbf{G}_i \in \mathbb{S}^{(\alpha+\beta)n+1}$ such that:*

$$\frac{\partial}{\partial t} v_i(\mathbf{x};\ \mathbf{Q}_i) = \frac{\partial v_i(\mathbf{x};\ \mathbf{Q}_i)}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial t} = \frac{\partial v_i(\mathbf{x};\ \mathbf{Q}_i)}{\partial \mathbf{x}} \hat{f}(\mathbf{x};\ \mathbf{P}) = \mathbf{b}_{\mathbf{x},\alpha+\beta}^T \mathbf{G}_i \mathbf{b}_{\mathbf{x},\alpha+\beta}, \tag{5}$$

where $\alpha + \beta$ is the basis degree. The matrix $\mathbf{G}_i$ is acquired by polynomial coefficient matching, and depends on $\mathbf{P}$ and $\mathbf{Q}_i$. We summarize this dependence with the function $\mathcal{G}(\mathbf{P}, \mathbf{Q}) = \mathbf{G}$, where $\mathbf{G}$ symbolizes the block-diagonal form of all $\mathbf{G}_i$ matrices. The proof is outlined in Appendix A.3.

Finally, with the necessary tools at our disposal, we can establish the connection between the global stability of the policy and finding SOS polynomials in Theorem 4.6. This theorem serves as the fundamental basis for the subsequent policy optimization process.

Theorem 4.6. *A polynomial DS policy, $\hat{f}(\mathbf{x};\ \mathbf{P})$, is GAS if the following conditions are satisfied:*

$$(a)\ \mathbf{Q} \succeq 0, \qquad (b)\ \mathbf{G} \prec 0, \qquad (c)\ \mathcal{G}(\mathbf{P}, \mathbf{Q}) = \mathbf{G}. \tag{6}$$

The proof is straightforward and is sketched in Appendix A.4.

4.3 Joint optimization problem

At this stage, we have established polynomial representations for both the policy and the LPF, along with a firm connection that confirms global stability. Now, we develop an objective function using the Mean-Squared Error (MSE) cost with the Elastic Net Regularization [36]. The MSE is calculated between the policy output and the expert's actions across demonstrated trajectories, and it solely depends on the policy parameters. Essentially, this problem entails regularized polynomial regression to minimize the imitation MSE to expert's demonstrations, subject to the existence of an LPF that satisfies the Lyapunov conditions. The optimization problem is framed as:

$$\begin{aligned} &\min_{\mathbf{Q},\mathbf{G},\mathbf{P}} J(\mathbf{P}) = \frac{1}{2N_t} \sum_{d=1}^{N_d} \sum_{s=1}^{N_s} (\hat{f}(\mathbf{x}^d(s);\ \mathbf{P}) - \dot{\mathbf{x}}^d(s))^2 + \lambda_1 \|\mathbf{P}\|_1 + \lambda_2 \|\mathbf{P}\|_F^2, \\ &s.t. \quad (a)\ \mathbf{Q} \succeq 0 \quad (b)\ \mathbf{G} \prec 0 \quad (c)\ \mathcal{G}(\mathbf{P}, \mathbf{Q}) = \mathbf{G} \quad (d)\ \mathbf{Q} = \mathbf{Q}^T,\ \mathbf{G} = \mathbf{G}^T,\ \mathbf{P} = \mathbf{P}^T, \end{aligned} \tag{7}$$

where $\|.\|_1$ and $\|.\|_F^2$ denote the first and Frobenius norms, and $\lambda_1,\ \lambda_2 \in \mathbb{R}^+$ represent the regularization coefficients.

Equation 7 is a semi-definite programming with nonlinear cost function [37, 35], and can be solved using standard semi-definite solvers [38, 39]. Semi-definite programming facilitates optimization over the convex cone of symmetric, positive semi-definite matrices or its affine subsets.

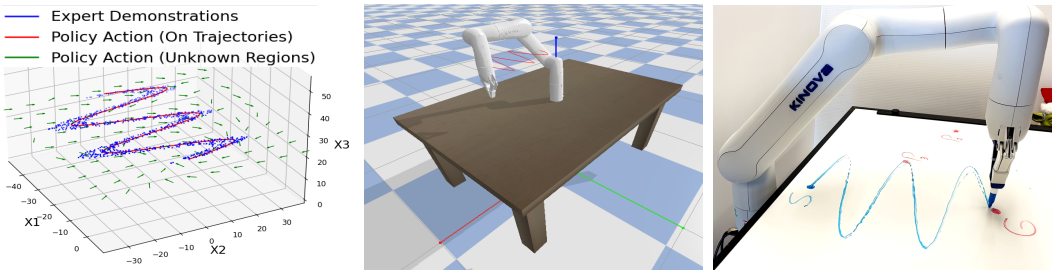
Notice that the negative-definite constraints can be transformed to a semi-definite constraints by a constant shift. Furthermore, to ensure that the resulting LPF is convex, one can either choose a basis with even powers, or add a constraint, such as $g_{ij} = 0\ \forall i + j \neq 2k,\ k \in \mathbb{N}$, where g_{ij} are elements of $\mathbf{Q}$. We opt for the first option to avoid further constraints during optimization.

5 Experiments

We employ two motion planning datasets for experiments. Our primary data comes from the widely recognized LASA Handwriting Motion Dataset [40], which comprises data recorded from handwritten trajectories. The second dataset contains expert demonstrations collected through teleoperating a robotic arm on realistic manipulation tasks. Details about both datasets can be found in Appendix B.1.

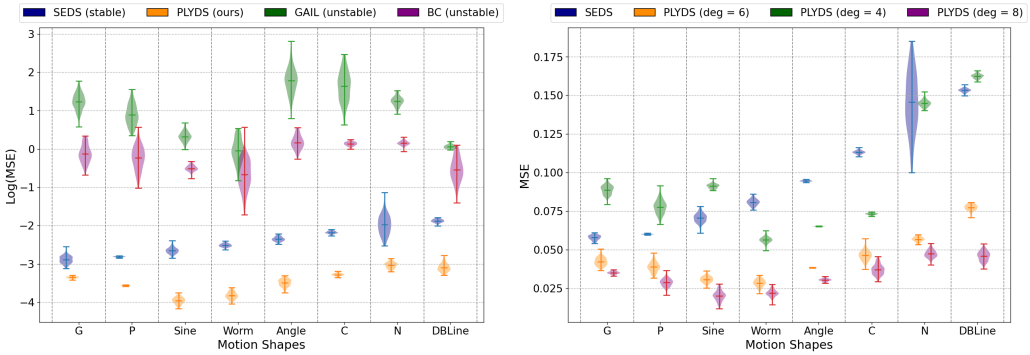
5.1 Evaluation

For evaluation purposes, we apply PLYDS and the baselines to the dataset (Figure 2a) and evaluate the performance of policy rollouts in PyBullet simulation (Figure 2b) before deploying safe policies onto a manipulator (Figure 2c). In all experiments, we randomly split the demonstrated trajectories in



(a) (b) (c)

Figure 2: Overview of the evaluation sequence: (a) learning from demonstrated data, (b) numerical evaluation in simulation and (c) deployed in real-world Gen3 Lite manipulator.



(a) (b)

Figure 3: Comparison of mean and standard deviation of reproduction MSE to other imitation learning methods (a) and the effect of having more complex polynomial policy (b). We use log(MSE) to plot (a) as GAIL and BC yield higher MSE due to their intrinsic need for large sample sizes.

the dataset into train and test sets. The policy learning stage, introduced in Equation (7), is carried out on the training data. The learned policy is subsequently evaluated by calculating the MSE between the policy predictions and the ground truth in the test data, $\frac{1}{2N_d^{test}N_s}\sum_{d=1}^{N_d^{test}}\sum_{s=1}^{N_s}(\hat{f}(\mathbf{x}^d(s);\ \mathbf{P}) - \dot{\mathbf{x}}^d(s))^2$. Recall that the policy output, $\dot{\mathbf{x}}$, is the velocity passed to the robot's low-level controller. We repeat this procedure over 20 different random seeds, and report average and standard deviation.

We compare the imitation accuracy of our approach to three existing baselines: Stable Estimator of Dynamical Systems (SEDS) [3], Behavioral Cloning (BC) [25], and Generative Adversarial Imitation Learning (GAIL) [22]. Note that among these, BC and GAIL do not provide mathematical stability guarantees, but the results could provide further comparison ground for the accuracy and computation time. The hyperparameters and architecture are discussed in Appendix B.3.

5.2 LASA Handwriting Dataset

This dataset contains handwriting demonstrations for several shapes. We compare the learned policies of PLYDS, with the policy and the LPF polynomial degrees of $\alpha = 6$ and $\beta = 2$, against the baselines on eight selected motions. The MSE for different experiments are reported in Figure 3a and confirm that our method consistently outperforms other baselines across all motion shapes. To understand how the degrees of polynomials affect the performance, we repeated the same experiments with degrees of $\alpha = 4$, 6, and 8. The results for other policy degrees are depicted in Figure 3b. We observe that a higher complexity leads to improved precision, if not halted by overfitting. Additionally, we study different LPF complexities in Appendix C.4.

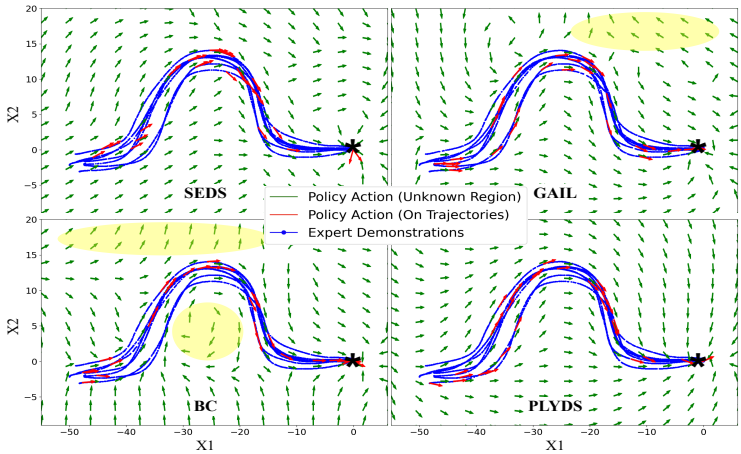


Figure 4: Policy rollout for Worm-shaped motion (blue) of the handwriting dataset. Each figure represents a trained policy (red and green arrows). Note the stability issues with GAIL and BC (highlighted regions) where the vector field diverges from the target.

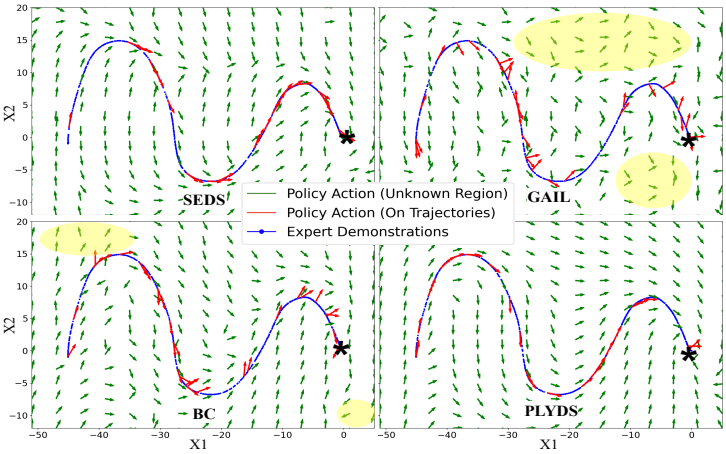


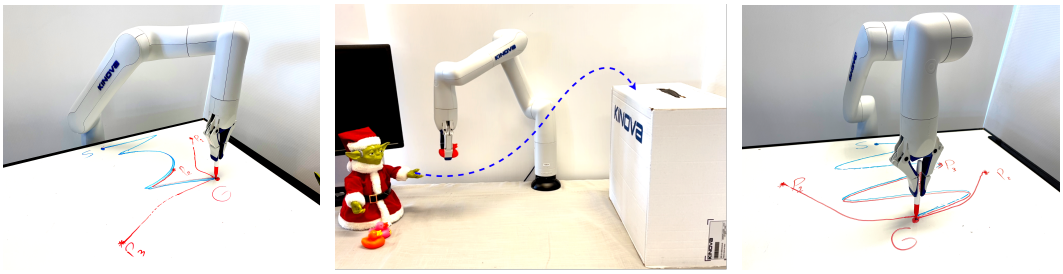
Figure 5: Policy rollout for Sine-shaped motion (blue) of the handwriting dataset, with access to only one expert demonstration. Each figure represents a trained policy (red and green arrows). Methods requiring large datasets, such as GAIL and BC, exhibit inaccurate and unsteady performance.

To analyze GAS, we visualize the learned policies of all methods as vector fields. Figure 4 illustrates the policy rollouts for Worm-shaped motion (blue) of the handwriting dataset. Each sub-figure represents a trained policy (red and green arrows). It is evident that SEDS and PLYDS maintain GAS, while GAIL and BC fail to demonstrate converging trajectories for the entire state-space (highlighted regions). This pattern persists for other motions (see Appendix C.1 for full experimental results).

Finally, we examine the sample efficiency of our method by reducing the input data to **one** demonstrated trajectory. From Figure 5, we can see that PLYDS learns a stable policy with such limited training samples, while the baseline methods may generate motion that diverges from the target.

5.3 Manipulation tasks

To conduct real-world trials, we collect a second set of expert demonstrations through teleoperating Kinova Gen3 Lite, a manipulator arm with six degrees of freedom. This new dataset holds three distinct motions: (a) prolonged-sine, (b) root-parabola, and (c) pick-and-place that represent typical nonlinear trajectories (see Figure 6) and more details are available in Appendix B.



(a) (b) (c)

Figure 6: Manipulation tasks: (a) prolonged-sine, (b) root-parabola, and (c) standard pick and place.

Expert Motion	Prolonged Sine	Root Parabola	Pick-and-Place
SEDS	0.234 ± 0.015	0.152 ± 0.023	0.094 ± 0.012
BC	1.650 ± 0.133	0.931 ± 0.078	0.725 ± 0.133
GAIL	2.322 ± 0.098	1.322 ± 0.094	0.663 ± 0.098
PLYDS (ours)	0.111 ± 0.007	0.176 ± 0.015	0.021 ± 0.003

Table 1: Policy rollout reproduction MSE in PyBullet simulations (for numerical comparisons and visual affirmation only)

The performance of all methods is summarized in Table 1, where PLYDS often outperforms other baselines. Next, PLYDS is deployed the learnt policy on the physical arm (Figure 2c), and successfully imitates the introduced manipulation tasks. We also start the robot at regions that are further away from the demonstrations and introduce perturbations by randomly pushing the robot to reveal the inherent GAS of PLYDS. As expected, PLYDS manage to successfully recover to the goal. Video recording of these experiments is provided in the supplementary video and Section 7.

6 Conclusion and Limitations

We introduced an approach that aims to learn globally stable nonlinear policies represented by polynomial dynamical systems. We employ the learned policies for motion planning based on imitating expert demonstrations. Our approach jointly learns a polynomial policy along with a parametric Lyapunov candidate that verifies global asymptotic stability by design. The resulting DS is utilized as a motion planning policy, guiding robots to stably imitate the expert's behavior. A comprehensive experimental evaluation is presented in real-world and simulation, where the method is compared against prominent imitation learning baselines. Future work includes incorporating other safety criteria, such as control barrier functions [41] or obstacle avoidance, into our learning objectives.

Limitations. A limitation of SOS is that the set of non-negative polynomials is larger than the ones expressed as SOS [42]. Though rare in motion planning tasks, this implies that finding a Lyapunov candidate could be difficult. Lasserre hierarchy and SOS extensions [43] can search in a broader class of LPF candidates and tackle this issue. Another limitation occurs when finding highly complex policies that lead to a violation of stability guarantees. This often happens when the regularization coefficients or the optimization tolerance are not set properly. We discuss this trade-off between stability and accuracy in Appendix C.3. Further, the computation complexity of PLYDS is feasible with a reasonable choice of polynomial degrees. Extremely higher degrees (than the ones in our experiments) are computationally demanding, but are often unnecessary in normal tasks.

7 Video, Codebase, and Reproducibility

The codebase, video supplements, etc. related to this project are available on sites.google.com/view/stable-planning-policy. Reproducing the experiments is as straightforward as installing dependent software packages, and running a Unix commands in `README` files.

References

[1] J.-C. Latombe. Robot motion planning, volume 124. Springer Science & Business Media, 2012.

[2] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. ACM Computing Surveys (CSUR), 50(2):1–35, 2017.

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

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

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

[6] N. Figueroa and A. Billard. A physically-consistent bayesian non-parametric mixture model for dynamical system learning. In A. Billard, A. Dragan, J. Peters, and J. Morimoto, editors, Proceedings of The 2nd Conference on Robot Learning, volume 87 of Proceedings of Machine Learning Research, pages 927–946. PMLR, 29–31 Oct 2018. URL https://proceedings.mlr.press/v87/figueroa18a.html.

[7] X. Cheng, B. Khomtchouk, N. Matloff, and P. Mohanty. Polynomial regression as an alternative to neural nets. arXiv preprint arXiv:1806.06850, 2018.

[8] P. Morala, J. A. Cifuentes, R. E. Lillo, and I. Ucar. Towards a mathematical framework to inform neural network modelling via polynomial regression. Neural Networks, 142:57–72, 2021.

[9] L. Vandenberghe and S. Boyd. Semidefinite programming. SIAM review, 38(1):49–95, 1996.

[10] H. Wolkowicz, R. Saigal, and L. Vandenberghe. Handbook of semidefinite programming: theory, algorithms, and applications, volume 27. Springer Science & Business Media, 2012.

[11] M. Laurent. Sums of squares, moment matrices and optimization over polynomials. In Emerging applications of algebraic geometry, pages 157–270. Springer, 2009.

[12] P. A. Parrilo. Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. California Institute of Technology, 2000.

[13] R. L. Devaney. An introduction to chaotic dynamical systems. CRC press, 2021.

[14] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Survey: Robot programming by demonstration. Technical report, Springrer, 2008.

[15] J. Aleotti and S. Caselli. Robust trajectory learning and approximation for robot programming by demonstration. Robotics and Autonomous Systems, 54(5):409–413, 2006.

[16] M. Muhlig, M. Gienger, S. Hellbach, J. J. Steil, and C. Goerick. Task-level imitation learning using variance-based movement optimization. In 2009 IEEE International Conference on Robotics and Automation, pages 1177–1184. IEEE, 2009.

[17] M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. IEEE Transactions on Robotics, 24(6):1463–1467, 2008.

[18] S. Schaal. Scalable locally weighted statistical techniques for real time robot learning. Applied Intelligence-Special issue on Scalable Robotic Applications of Neural Networks, 17:49–60, 2002.

[19] N. B. Figueroa Fernandez and A. Billard. A physically-consistent bayesian non-parametric mixture model for dynamical system learning. Technical report, EPFL, 2018.

[20] A. Coulombe and H.-C. Lin. Generating stable and collision-free policies through Lyapunov function learning. International Conference on Robotics and Automation, 2023.

[21] H. Ravichandar, I. Salehi, and A. Dani. Learning partially contracting dynamical systems from demonstrations. In Conference on Robot Learning, pages 369–378. PMLR, 2017.

[22] J. Ho and S. Ermon. Generative adversarial imitation learning. Advances in neural information processing systems, 29, 2016.

[23] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316, 2016.

[24] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, pages 4950–4957, 2018.

[25] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. Advances in neural information processing systems, 1, 1988.

[26] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In Conference on Robot Learning, pages 158–168. PMLR, 2022.

[27] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the twenty-first international conference on Machine learning, page 1, 2004.

[28] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al. Maximum entropy inverse reinforcement learning. In Aaai, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[29] J. Z. Kolter and G. Manek. Learning stable deep dynamics models. Advances in neural information processing systems, 32, 2019.

[30] N. Lawrence, P. Loewen, M. Forbes, J. Backstrom, and B. Gopaluni. Almost surely stable deep dynamics. Advances in Neural Information Processing Systems, 33:18942–18953, 2020.

[31] Y.-C. Chang, N. Roohi, and S. Gao. Neural Lyapunov control. Advances in neural information processing systems, 32, 2019.

[32] S. M. Richards, F. Berkenkamp, and A. Krause. The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In Conference on Robot Learning, pages 466–476. PMLR, 2018.

[33] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake. Lyapunov-stable neural-network control. In Proceedings of Robotics: Science and Systems, Virtual, July 2021. doi:10.15607/RSS.2021.XVII.063.

[34] A. Papachristodoulou and S. Prajna. On the construction of Lyapunov functions using the sum of squares decomposition. In Proceedings of the 41st IEEE Conference on Decision and Control, 2002., volume 3, pages 3482–3487. IEEE, 2002.

[35] G. Blekherman, P. A. Parrilo, and R. R. Thomas. Semidefinite optimization and convex algebraic geometry. SIAM, 2012.

[36] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. Journal of the royal statistical society: series B (statistical methodology), 67(2):301–320, 2005.

[37] S. Burer and R. D. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Mathematical Programming, 95(2):329–357, 2003.

[38] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. Nonlinear programming: theory and algorithms. John Wiley & Sons, 2013.

[39] M. ApS. MOSEK Optimizer API for Python 10.0.46, 2021. URL http://docs.mosek.com/9.0/toolbox/index.html.

[40] S. M. Khansari-Zadeh and A. Billard. Lasa Handwriting Dataset. https://cs.stanford.edu/people/khansari/download.html#SEDS_reference, 2011.

[41] 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), pages 3420–3431. IEEE, 2019.

[42] G. Blekherman. There are significantly more nonegative polynomials than sums of squares. Israel Journal of Mathematics, 153:355–380, 2006.

[43] A. Papachristodoulou and S. Prajna. On the construction of Lyapunov functions using the sum of squares decomposition. In Proceedings of the 41st IEEE Conference on Decision and Control, 2002., volume 3, pages 3482–3487 vol.3, 2002. doi:10.1109/CDC.2002.1184414.

[44] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17:261–272, 2020. doi:10.1038/s41592-019-0686-2.