

## 359 A Mathematical Proofs

360 Due to space limitations and to maintain coherency, proofs for propositions, lemmas, and theorems  
361 are presented in this section in the order in which they appeared in the main text.

### 362 A.1 Proof of Proposition 4.3

363 *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*  
364  *$\hat{v} = \sum_{i=1}^n v_i(\mathbf{x}; \mathbf{Q}_i)$  yields a valid standard Lyapunov function for  $\hat{f}(\mathbf{x}; \mathbf{P})$ , proving that the policy*  
365 *satisfies GAS conditions.*

366 **Proof.** As  $v_i(\mathbf{x}; \mathbf{Q}_i)$  is an LPF candidate for  $\hat{f}_i(\mathbf{x}; \mathbf{P}_i)$ , both the first and second Lyapunov  
367 conditions must be satisfied, i.e.,  $\forall i \in \{1, \dots, n\}$ :

$$(a) v_i(\mathbf{x}; \mathbf{Q}_i) \succeq 0, \forall \mathbf{x} \in \mathcal{X}, \quad (b) \frac{\partial v_i(\mathbf{x}; \mathbf{Q}_i)}{\partial t} \prec 0, \forall \mathbf{x} \in \mathcal{X}.$$

368 Define the sum of elements  $\hat{v}(\mathbf{x}; \mathbf{Q}) = \sum_{i=1}^n v_i(\mathbf{x}; \mathbf{Q}_i)$ . We show that  $\hat{v}(\mathbf{x}; \mathbf{Q})$  satisfies both  
369 Lyapunov global stability conditions:

$$(i) v_i(\mathbf{x}; \mathbf{Q}_i) \succeq 0 \Rightarrow v_1(\mathbf{x}; \mathbf{Q}_1) + \dots + v_n(\mathbf{x}; \mathbf{Q}_n) = \hat{v}(\mathbf{x}; \mathbf{Q}) \succeq 0,$$

$$(ii) \frac{\partial \hat{v}(\mathbf{x}; \mathbf{Q})}{\partial t} = \frac{\partial \sum_{i=1}^n v_i(\mathbf{x}; \mathbf{Q}_i)}{\partial t} = \sum_{i=1}^n \frac{\partial v_i(\mathbf{x}; \mathbf{Q}_i)}{\partial t}, \quad \frac{\partial v_i(\mathbf{x}; \mathbf{Q}_i)}{\partial t} \prec 0 \quad (b). \quad \square$$

### 370 A.2 Proof of Lemma 4.4

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

**Proof.** Considering that  $v_i(\mathbf{x}; \mathbf{Q}_i) = \mathbf{b}_{\mathbf{x},\beta}^T \mathbf{Q}_i \mathbf{b}_{\mathbf{x},\beta}$  and  $\mathbf{Q}_i$  is not singular, we can perform a Cholesky  
factorization on the parameters' matrix  $\mathbf{Q}_i$ . The result is  $\mathbf{Q}_i = L_i^T L_i$ , and the positivity of  $v_i(\mathbf{x}; \mathbf{Q}_i)$   
comes from,

$$v_i(\mathbf{x}; \mathbf{Q}_i) = \mathbf{b}_{\mathbf{x},\beta}^T \mathbf{Q}_i \mathbf{b}_{\mathbf{x},\beta} = \mathbf{b}_{\mathbf{x},\beta}^T L_i^T L_i \mathbf{b}_{\mathbf{x},\beta} = (L_i \mathbf{b}_{\mathbf{x},\beta})^T (L_i \mathbf{b}_{\mathbf{x},\beta}) = \|L_i \mathbf{b}_{\mathbf{x},\beta}\|^2 \succeq 0,$$

373 that represents  $v_i(\mathbf{x}; \mathbf{Q}_i)$  as an SOS and therefore achieves the first Lyapunov condition.  $\square$

### 374 A.3 Proof of Lemma 4.5

375 *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*  
376 *symmetric matrix  $\mathbf{G}_i \prec 0$  and  $\mathbf{G}_i \in \mathbb{S}^{(\alpha+\beta)n+1}$  such that:*

$$\frac{\partial}{\partial t} v_i(\mathbf{x}; \mathbf{Q}_i) = \frac{\partial v_i(\mathbf{x}; \mathbf{Q}_i)}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial t} = \frac{\partial v_i(\mathbf{x}; \mathbf{Q}_i)}{\partial \mathbf{x}} \hat{f}(\mathbf{x}; \mathbf{P}) = \mathbf{b}_{\mathbf{x},\alpha+\beta}^T \mathbf{G}_i \mathbf{b}_{\mathbf{x},\alpha+\beta}, \quad (8)$$

377 *where  $\alpha + \beta$  is the basis degree. The matrix  $\mathbf{G}_i$  is acquired by polynomial coefficient matching,*  
378 *and depends on  $\mathbf{P}$  and  $\mathbf{Q}_i$ . We summarize this dependence for all  $i \in \{1, \dots, n\}$  with the function*  
379  *$\mathcal{G}(\mathbf{P}, \mathbf{Q}) = \mathbf{G}$ , where  $\mathbf{G}$  symbolizes the block-diagonal form of all  $\mathbf{G}_i$  matrices.*

380 **Proof.** We know that LPF rows are denoted by  $v_i(\mathbf{x}; \mathbf{Q}_i) = \mathbf{b}_{\mathbf{x},\beta}^T \mathbf{Q}_i \mathbf{b}_{\mathbf{x},\beta}$ . Hence, we write the  
381 second Lyapunov condition by taking the derivative of each row:

$$\begin{aligned} \frac{\partial v_i(\mathbf{x}; \mathbf{Q}_i)}{\partial t} &= \frac{\partial v_i(\mathbf{x}; \mathbf{Q}_i)}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial v_i(\mathbf{x}; \mathbf{Q}_i)}{\partial x_2} \frac{\partial x_2}{\partial t} + \dots + \frac{\partial v_i(\mathbf{x}; \mathbf{Q}_i)}{\partial x_n} \frac{\partial x_n}{\partial t} \\ \frac{\partial x_j}{\partial t} &= \hat{f}_j(\mathbf{x}; \mathbf{P}_j) \Rightarrow \frac{\partial v_i(\mathbf{x}; \mathbf{Q}_i)}{\partial t} = \sum_{j=1}^n \frac{\partial v_i(\mathbf{x}; \mathbf{Q}_i)}{\partial x_j} \hat{f}_j(\mathbf{x}; \mathbf{P}_j) \\ &= \sum_{j=1}^n \frac{\partial [\mathbf{b}_{\mathbf{x},\beta}^T \mathbf{Q}_i \mathbf{b}_{\mathbf{x},\beta}]}{\partial x_j} [\mathbf{b}_{\mathbf{x},\alpha}^T \mathbf{P}_j \mathbf{b}_{\mathbf{x},\alpha}] \end{aligned}$$

382 Within the last summation, both the derivative of Lyapunov function and the policy are polynomials.  
 383 The idea is that their multiplication could also be written as an SOS polynomial if the parameters  $\mathbf{P}_i$   
 384 and  $\mathbf{Q}_i$  are chosen carefully. For this polynomial, we define a new basis  $\mathbf{b}_{\mathbf{x},\alpha+\beta}$  and  $\mathbf{G}_i \in \mathbb{S}^{(\alpha+\beta)n+1}$ .  
 385 Note that the degree of this basis is calculated by  $\frac{1}{2}[(2\beta - 1) + (2\alpha) + 1]$ , which is the rounded-up  
 386 degree of the above multiplication term.

387 Next, we match polynomial coefficients on both sides, yielding  $\mathbf{G}_i$  parameters as a function of both  
 388  $\mathbf{P}$  and  $\mathbf{G}_i$ , i.e.,

$$\begin{aligned} \mathbf{b}_{\mathbf{x},\alpha+\beta}^T \mathbf{G}_i \mathbf{b}_{\mathbf{x},\alpha+\beta} &\stackrel[\text{Coefficients}]{\text{Matching}} \sum_{j=1}^n \frac{\partial [\mathbf{b}_{\mathbf{x},\beta}^T \mathbf{Q}_i \mathbf{b}_{\mathbf{x},\beta}]}{\partial x_j} [\mathbf{b}_{\mathbf{x},\alpha}^T \mathbf{P}_j \mathbf{b}_{\mathbf{x},\alpha}] \\ &\Rightarrow \mathbf{G}_i = \mathcal{G}_i(\mathbf{P}, \mathbf{Q}_i) \end{aligned}$$

389 We summarize the same relationship for all  $\mathbf{G}_i$  matrices, and call the resulting function  $\mathcal{G}$ . Hence,  
 390 the second condition can be represented by  $\mathbf{G} = \mathcal{G}(\mathbf{P}, \mathbf{Q})$  and  $\mathbf{G} \prec 0$ , and be viewed as SOS.  $\square$

#### 391 A.4 Proof of Theorem 4.6

392 Assuming the polynomial representation of a nonlinear autonomous dynamical system (Defini-  
 393 tion 4.1),

$$\hat{f}(\mathbf{x}; \mathbf{P}) = [\mathbf{b}_{\mathbf{x},\alpha}^T \mathbf{P}_1 \mathbf{b}_{\mathbf{x},\alpha} \quad \mathbf{b}_{\mathbf{x},\alpha}^T \mathbf{P}_2 \mathbf{b}_{\mathbf{x},\alpha} \quad \dots \quad \mathbf{b}_{\mathbf{x},\alpha}^T \mathbf{P}_n \mathbf{b}_{\mathbf{x},\alpha}]^T,$$

394 the existence of a corresponding polynomial Lyapunov function (Definition 4.2),

$$v(\mathbf{x}; \mathbf{Q}) = [\mathbf{b}_{\mathbf{x},\beta}^T \mathbf{Q}_1 \mathbf{b}_{\mathbf{x},\beta} \quad \mathbf{b}_{\mathbf{x},\beta}^T \mathbf{Q}_2 \mathbf{b}_{\mathbf{x},\beta} \quad \dots \quad \mathbf{b}_{\mathbf{x},\beta}^T \mathbf{Q}_n \mathbf{b}_{\mathbf{x},\beta}]^T$$

395 guarantees the asymptotic global stability of the policy, if the following conditions are satisfied:

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

396 **Proof.** The proof is straightforward and follows both Lemma 4.4 and Lemma 4.5. We know that  
 397 each partial DS  $\hat{f}_i(\mathbf{x}; \mathbf{P}_i)$  is stable if the corresponding parameterized LPF satisfies (a), (b), and  
 398 (c), where  $\mathcal{G}$  is an affine function found in Lemma 4.5 by polynomial coefficient matching. Since  
 399 each  $\hat{f}_i(\mathbf{x}; \mathbf{P}_i)$  explains the derivative along one of the orthogonal basis of  $\hat{f}(\mathbf{x}; \mathbf{P})$ , their individual  
 400 global stability is equivalent to the stability of the entire system. In other words,

$$\begin{aligned} &\forall x_i \in \mathcal{D}\{\hat{f}_i(\mathbf{x}; \mathbf{P}_i)\}, \quad \lim_{t \rightarrow \infty} x_i = x_i^* \\ &\Rightarrow \lim_{t \rightarrow \infty} \mathbf{x} = [\lim_{t \rightarrow \infty} x_1 \quad \lim_{t \rightarrow \infty} x_2 \quad \dots \quad \lim_{t \rightarrow \infty} x_n]^T = \mathbf{x}^* \end{aligned}$$

401 Another proof can be provided using the LPF introduced in Proposition 4.3 as a Lyapunov candidate  
 402 for the whole system. Both proofs equally validate the stability of the polynomial DS.  $\square$

## 403 B Experiment Setup and Details

404 Enclosed in this section are detailed descriptions of our experiment setup, main software packages,  
 405 and datasets. Due to space limitations, crucial details from the experiments are explained here. Even  
 406 though reading the section is not necessary to understand the paper, it provides useful insight into our  
 407 setup and can aid reproducibility and future research.

### 408 B.1 Datasets

409 **Handwriting dataset.** The LASA Handwriting Dataset (Figure 7) is a collection of 2D hand-  
 410 writing motions recorded from a Tablet-PC and by user's input. The dataset includes 30 human  
 411 handwriting motions, where each motion represents a desired pattern. For each pattern, there are  
 412 seven demonstrations recorded, with each demonstration starting from a slightly different (but fairly  
 413 close) initial position but ending at the same final point. These demonstrations may intersect with  
 414 each other. Out of the 30 motions, 26 correspond to a single pattern, while the remaining four motions  
 415 include multiple patterns, referred to as Multi Models. In all the handwriting motions, the target  
 416 position is defined as (0, 0), without loss of generality. The dataset provides the following features:

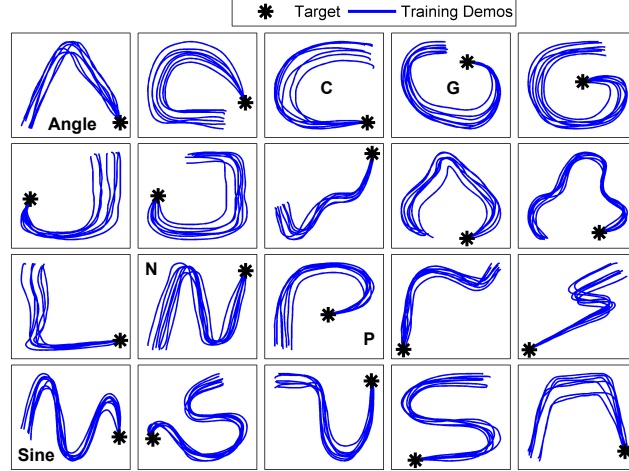


Figure 7: Plots of handwriting dataset motions used in our experiments. We select a representative subset of motions for baselining to keep the experiments computationally feasible. Each plot shows 7 demonstrations with 1000 recorded samples per each. Notice that the time indexing is included in the dataset, but it is irrelevant to our work as we learn time-invariant policies.

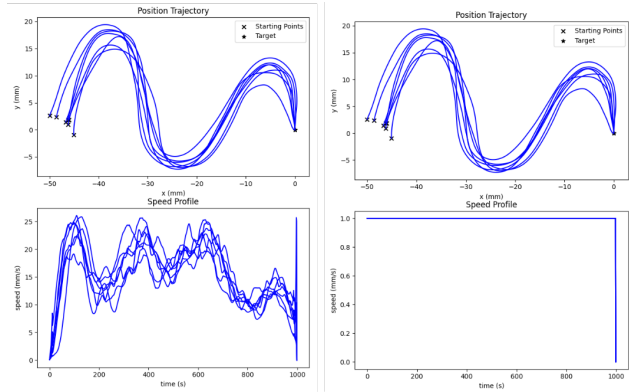


Figure 8: Speed profile for the sine motion, normalized vs. natural. When we normalize the speed, policies fail to capture the difference in speed along the trajectories. PLYDS works with both normalized and regular velocities, but we mostly opt for normalized velocities for baselining and comparisons, especially when plotting the policy vector fields.

- 417 • Position ( $2 \times 1000$ ) representing the motion in 2D space. The first row corresponds to the
- 418 x-axis, and the second row corresponds to the y-axis in Cartesian coordinates.
- 419 • Time ( $1 \times 1000$ ) being the time-stamp for each data point in the motion. We do not use this
- 420 property, as our proposed method generates time-invariant policies.
- 421 • Velocity ( $2 \times 1000$ ) representing the velocity corresponding to each position. We use this
- 422 feature as a label and form our MSE cost function to calculate the difference between the
- 423 predicted velocity and this data.
- 424 • Acceleration ( $2 \times 1000$ ) matrix representing the acceleration. Not applicable to our research,
- 425 but could potentially be utilized for future research.

426 We will not experiment on the entire dataset of 30 motions due to computational unfeasibility. Instead,

427 we select a representative set of motions with ( $8 \times 5 \times 2 \times 1000$ ) samples in total. The experiments

428 are mainly conducted with this designated set, but since the set is chosen to be representative, we

429 expect the results to generalize to other motions as well.

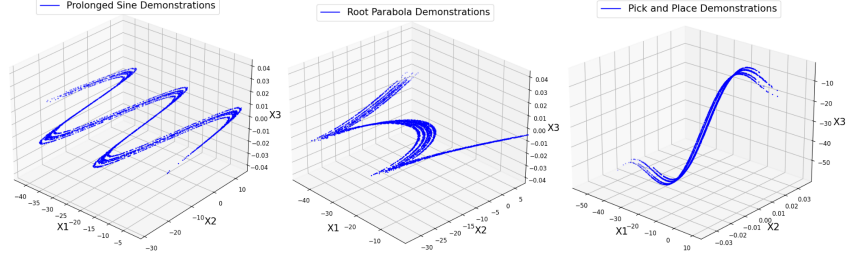


Figure 9: Plots of the dataset collected using Kinova Gen3 Lite and teleoperation. The robot is operated to complete the following trajectories multiple times, while the position and velocity data are recorded in real-time. Expert’s demonstrations can also come from robot’s low-level controllers, which leads to faster data gathering process. We tried to keep the scale of these trajectories aligned with the handwriting dataset for further consistency.

**Velocity normalization.** Moreover, for some experiments, we opt to normalize the velocity values, such as in Figure 8, to avoid large cost values. This can cause a loss of generality, since the policy actions are now restricted to the direction of the action vector, and will not try to replicate its size. The size of this arrow might be important in scenarios where parts of the motion need to be carried out at a different pace. PLYDS can handle the dataset with or without velocity normalization. The dataset is also referenced and provided as a part of our reproducibility efforts.

**Real-world collected dataset.** We collect data by teleoperating the Kinova Gen3 Lite Arm. Teleoperation involves employing human agents to operate a robotic device or system, recording their actions as expert demonstrations. The teleoperated actions are then recorded and utilized as training data for algorithmic learning. We have two options for teleoperation. First, a human expert can manually control the robot arm using a joystick or keyboard. This process results in natural but non-smooth trajectories. The second approach employs the robot’s internal control systems and trajectory planning systems to perform as an expert and execute some patterns. This leads to a smoother data collection process with higher reliability. Please keep in mind that planners only connect a series of few points, with no guarantee of stability, and are time-dependent. Consequently, the role of policies generated by PLYDS will not be obsolete.

We gather an open-source dataset holding three distinct motions: the prolonged sine, root parabola, and pick-and-place (Figure 9). Each motion is represented by 50 demonstrations in a 3-dimensional world. Each demonstration contains a state (position) vector ( $3 \times 1000$ ) and a corresponding action (velocity) vector ( $3 \times 1000$ ). Note that orientation is not recorded in the dataset, as we assume the robot’s gripper will always face downwards. There will not be any loss of generality because controlling the orientation with PLYDS can be done in parallel, and in the exact same way as the end-effector’s position. The dataset is provided as a part of our reproducibility efforts in Section 7.

## B.2 SDP Optimization

We primarily use the commercially available MOSEK [39] optimization software that provides solutions for numerous types of optimization issues, including nonlinear semidefinite programming. The flexibility and high-performance capabilities of MOSEK make it ideal for challenging optimization tasks in both commercial and academic settings. We currently use the MOSEK under an academic license. SCP is another solver specifically designed for solving semidefinite complementarity problems, which include nonlinear SDP as a special case. SCP employs an augmented Lagrangian method combined with the Fischer-Burmeister function to handle the complementarity conditions in the SDP. At this time, we do not have any solid comparison between the efficiency of these solvers for our setup, but commercial software products often perform more efficient than open-source products.

We also use SciPy [44], an open-source scientific computing library for Python that has many modules for numerical optimization. SciPy can handle a wide range of optimization problems, including

nonlinear programming with semidefinite constraints, even though it may not provide specialized solvers for nonlinear SDP. Our software still supports SciPy; however, it is not as efficient in solving nonlinear SDP problems as MOSEK and SCP.

### B.3 Hyperparameters and architecture.

We provide a summary of parameters related to each baseline we used in the paper. Note that we accelerate the computation of GAIL and BC with an NVIDIA GeForce RTX 3060 GPU, but SEDS and PLYDS use only a Core-i7 Gen8 CPU for optimization.

**PLYDS** For our experiments, we primarily utilize parameters  $\alpha = 3$  and  $\beta = 1$ , which have proven effective in most cases. However, we also explore higher degrees to cover a broader range of settings. Additionally, to strike a balance between stability and accuracy, we occasionally adjust the tolerance level from  $10^{-4}$  to  $10^{-9}$ . This allows us to trade off precision for stability when necessary. For the Lyapunov candidate to remain convex beyond the quadratic form, we opt for only square elements in the LPF basis vector. This ensures stability and reliability in our system. Although it is possible to enforce a positive Hessian for the Lyapunov function, it incurs additional computational time while providing greater flexibility in stability conditions.

**GAIL.** The discriminator network takes as input the state-action pairs or observations generated by the policy network and expert demonstrations. *Hidden Layers:* The network may consist of two or three hidden layers, each with 256 or 512 units. *Activation Function:* Rectified Linear Unit (ReLU) activation function is commonly used between the layers. *Output:* The discriminator produces a single output value, representing the probability of the input being from the expert or the generated policy. *Hyperparameters:* Learning Rate: 0.0001, Number of Discriminator Updates per Generator Update: 1 or 2, Discount Factor (for reinforcement learning algorithm): 0.99, Batch Size: 64 or 128, Number of Training Iterations: 1000.

**BC.** The behavioral cloning network takes the state-action pairs as input. *Hidden Layers:* The network may have two or three hidden layers, each consisting of 128 or 256 units. *Activation Function:* Rectified Linear Unit (ReLU), *Output:* The output layer of the network corresponds to the action space dimensionality, producing the predicted action. *Hyperparameters:* Learning Rate: 0.0001, Number of Training Iterations: 5000, Batch Size: 64, Regularization Strength (L2 regularization): 0.001, Optimizer: Adam, Loss Function: Mean Squared Error (MSE).

**SEDS.** Takes position-velocity pairs as input (or state-action pairs in general). *Number of Gaussian Components:* Typically ranges from 3 to 10, depending on the complexity of the motion being learned. *Gaussian Mixture Model (GMM) Parameters:* Covariance Type: Diagonal covariance is commonly used for efficiency and simplicity, Regularization Weight: Often set to a small value, such as  $1e-6$ , to avoid singularities and overfitting, Maximum Number of Iterations: 100 iterations. *Convergence Tolerance:*  $1e-6$  or  $1e-7$ . SEDS is implemented in Python Scipy, and the original MATLAB code is not used in our comparisons to remain consistent with other baselines.

## C Supplemental Results

We present a comprehensive set of additional experiments aimed at putting the proposed framework to test from a variety of angles including access to fewer demonstrations, demonstrating the variety of LPFs, more baseline policy rollouts, additive noise, and lastly, we conduct an ablation study by removing the stability guarantee, and present computation times in comparison with the baselines.

### C.1 Baseline policy rollouts

We plot more policy rollouts for PLYDS in comparison to the baselines. The key takeaway is the pattern of instability among neural based imitation learning methods for unknown areas of state-space.

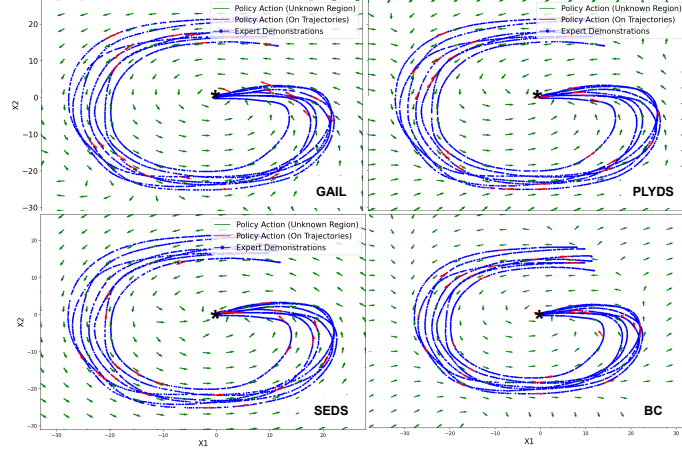


Figure 10: Policy rollout for G-shaped motion in handwriting dataset.

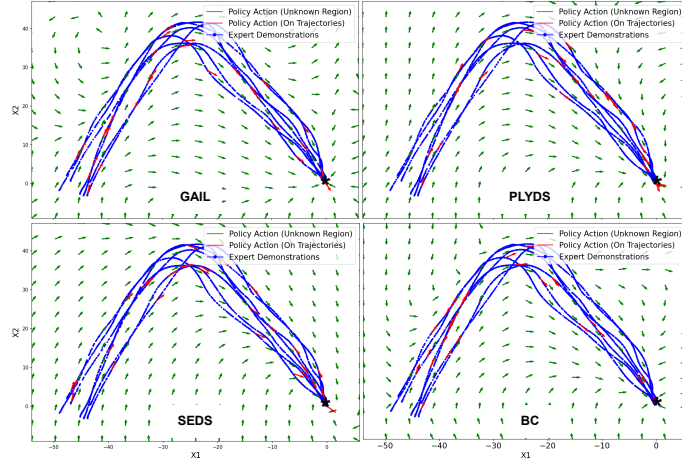


Figure 11: Policy rollout for Angle-shaped motion in handwriting dataset.

509 Note that even though some arrows might appear diverging from the target, they can be part of a  
 510 converging trajectory with decreasing energy, that circles back towards the goal in the end. We make  
 511 sure of this by plotting a larger portion of state-space and energy functions after each experiment.

512 We present more optimized policies for the following handwriting dataset motions: G (Figure 10),  
 513 Angle (Figure 11), C (Figure 12), and P (Figure 13). Each figure represents a trained policy (red  
 514 and green arrows) along with the original trajectories in the dataset (blue dots). The same instability  
 515 patterns continue to persist with GAIL and BC as the vector field diverges from the target. The same  
 516 patterns as the plots in the main text continue to emerge.

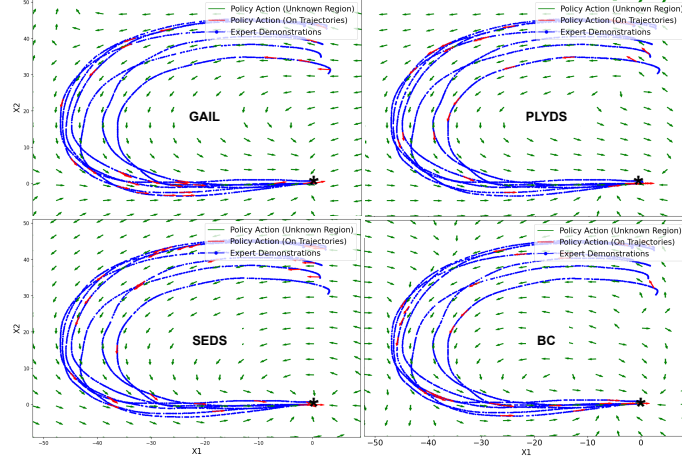


Figure 12: Policy rollout for C-shaped motion in handwriting dataset.

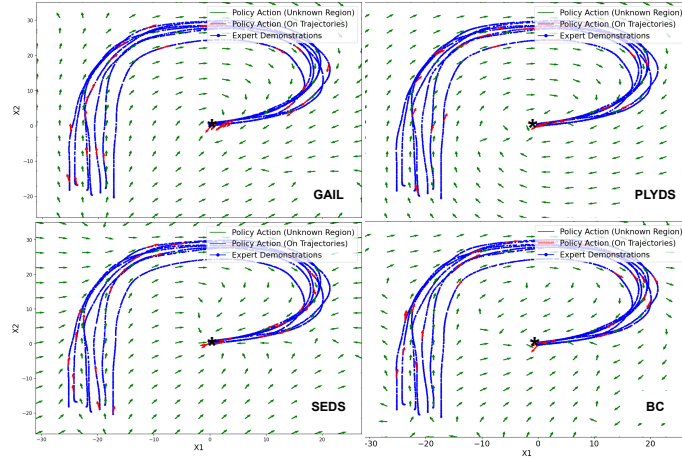


Figure 13: Policy rollout for P-shaped motion in handwriting dataset.

## 517 C.2 Performance with additive noise

518 The presence of noise in imitation learning has substantial impacts on the learning process and the  
 519 resultant policies. Excessive levels of noise can lead to instability, inhibiting the algorithm from  
 520 converging to an optimal policy. To address this, we adjust the noise levels to gauge the performance  
 521 of PLYDS. We impose a uniform additive noise, where samples are uniformly distributed over the  
 522 specified interval. We change the size of this interval and expand it bilaterally around zero.

523 The results in Figure 14 demonstrate a moderate level of noise-robustness that can be further improved  
 524 in future studies. Noisy data also increases the error bands, leading to increased uncertainty in the  
 525 outcome of policy optimization.

## 526 C.3 Ablation study: stability vs. accuracy

527 When working with DS policies, there is a dilemma known as the stability-accuracy trade-off [19].  
 528 This means that a balance must be struck between the reliability and robustness of generated policies  
 529 to guarantee global convergence to the target (referred to as stability), and minimizing errors to obtain  
 530 precise solutions (referred to as accuracy). It is important to find a compromise between these two  
 531 factors, as more stable algorithms may not be as accurate, while more accurate algorithms may be  
 532 sensitive to instabilities.



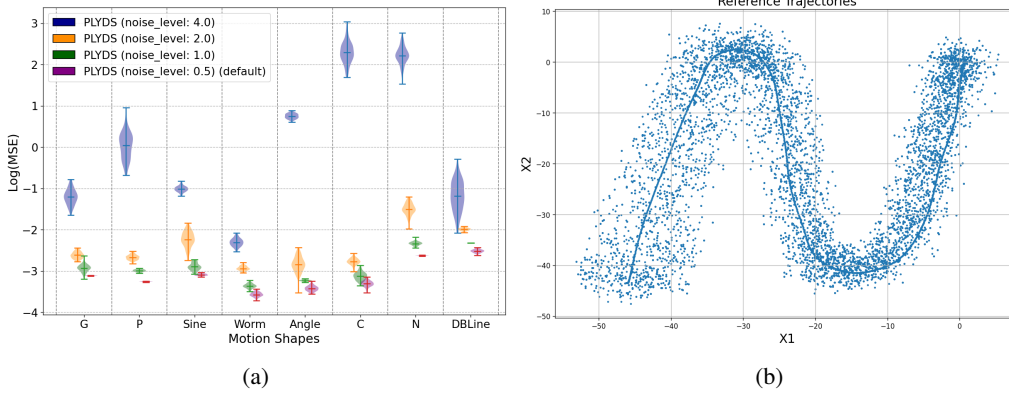


Figure 14: Performance of PLYDS in the face of uniform additive noise (a) and a sample of a noisy trajectory with noise-level set to 2 (b). Noise levels are in centimeters, therefore, a noise-level of 4 means each reading could be deviated from its true value by  $\pm 4cm$ .

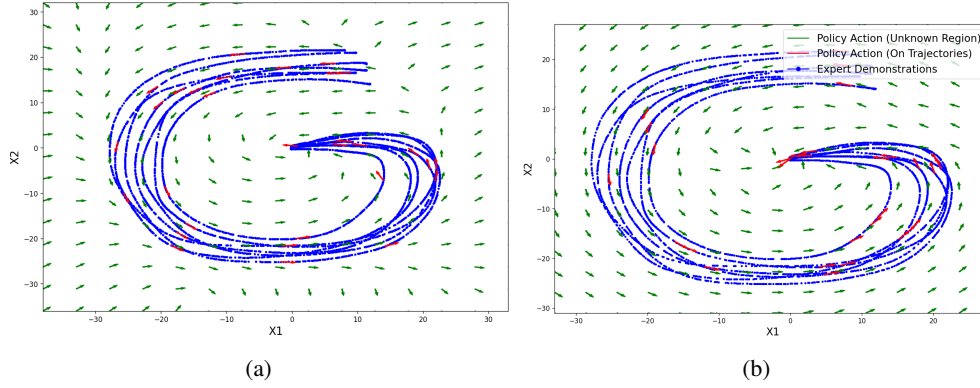


Figure 15: PLYDS policy rollout: (a) without enforced stability, and (b) with the stability constraints in effect. The instability is caused by having a complex polynomial without regularization, while setting the tolerance parameter in favor of accuracy.

533 The higher the degree of the polynomial, the more accurate imitation of expert behavior. Hence,  
 534 in theory, any nonlinearity may be approximated by our DS formulation. However, in practice, we  
 535 introduce regularization and tolerance parameters in the code. The tolerance can be used to choose in  
 536 the favor of accuracy or stability (see Figure 15). Another way to balance this equation is to start with  
 537 lower-degree polynomials, and increase the policy’s complexity when the accuracy is not sufficient.

#### 538 C.4 Complexity of Lyapunov functions

539 Figure 16 demonstrates the complexity of the Lyapunov function affects trajectory planning in  
 540 the state-space in various ways, such as optimization efficiency, trajectory smoothness, obstacle  
 541 avoidance, robustness to perturbations, and planning accuracy. If the Lyapunov function is more  
 542 complex, it may increase computational costs but in turn result in more complex and nonlinear  
 543 trajectories. On the other hand, simpler Lyapunov functions may offer faster computations, smoother  
 544 trajectories, and satisfactory planning accuracy, but they may not be adaptable to intricate expert  
 545 demonstrations.

546 When deciding on the complexity of the Lyapunov function, it is necessary to consider the feasibility  
 547 of computations, and the desired smoothness and accuracy of planned trajectories and always start  
 548 with the most simple: quadratic distance function. Figure 16 illustrates this by showing both stable  
 549 and unstable Lyapunov possibilities gauged across various LPF complexities. Additionally, Figure 17  
 550 depicts two same executions of PLYDS with high and low complexity of the Lyapunov function.



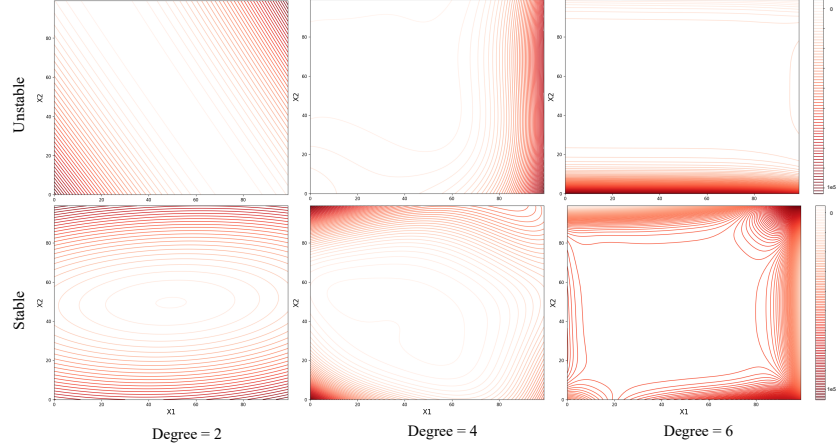


Figure 16: During our policy optimization, we obtain LPF samples such as the ones depicted above. Even though variation in complexity notably affects the computation time, employing more complex Lyapunov functions (polynomials of higher degrees) appears necessary to achieve stable and precise policies in some cases. Currently, we manually determine the complexity of the Lyapunov function and shift to higher complexities only if the optimization fails to deliver satisfactory results.

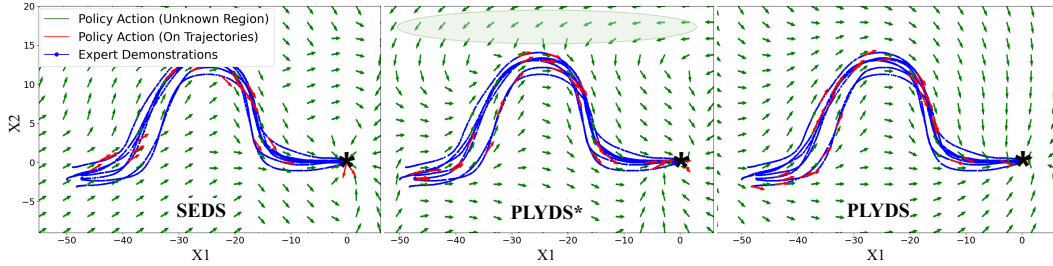


Figure 17: Comparison of policy trajectories between PLYDS and SEDS. PLYDS\* uses an LPF representation with higher nonlinearity, and, therefore, results in trajectories where distance to the target is not reduced monotonically (highlighted).

## 551 C.5 Computation times

552 We conduct all the experiments on a machine with a Core-i7 Gen8 CPU, NVIDIA GeForce RTX  
 553 3060 GPU, and 32 GB RAM DDR2. Among the methods we experimented on, only GAIL and BC  
 554 utilize a GPU to accelerate the computation of neural networks. PLYDS and SEDS only use CPU for  
 555 optimization in our implementations. Even so, the computation times are compared in Figure 18.

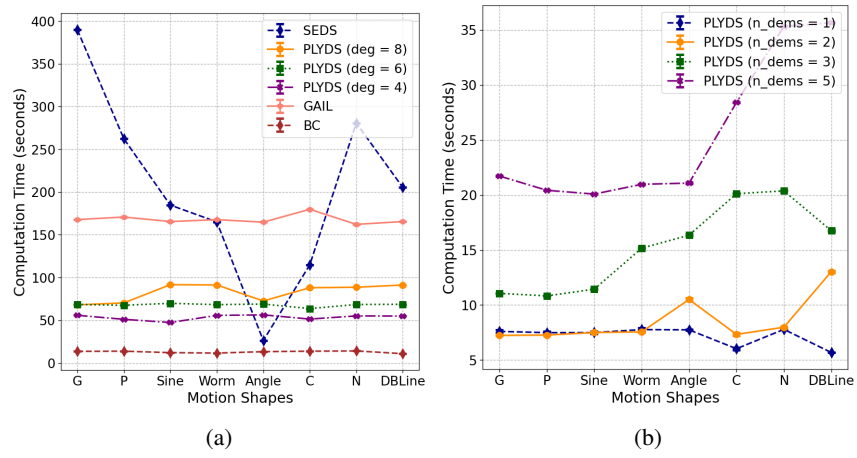


Figure 18: Computation time (averaged over 20 trials) of PLYDS compared to other baselines (a), and for different dataset sizes (b). It is noteworthy that GAIL and BC are utilizing a GPU to expedite their computations.