2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Learning Stable Dynamics via Iterative Quadratic Programming

Paul Gesel[1] and Momotaz Begum[1]

Abstract— **This paper proposes a novel autonomous dynamic system (ADS) based controller for trajectory learning from demonstration (LfD). We call our method Learning Stable Dynamics via Iterative Quadratic Programming (LSD-IQP). LSD-IQP learns an energy function and an ADS from demonstrations via semi-infinite quadratic programming. Energy function constraints are imposed on the learned ADS to ensure convergence to a single goal position. Unlike other energy-based methods, LSD-IQP allows the energy function to have both local maximums and saddle points. This flexibility enables LSD-IQP to learn a broader class of motions compared to other ADS-based controllers. We demonstrate the capabilities of LSD-IQP via several experiments, including: 1) learning handwritten symbols and comparing the swept error area to several other ADS methods 2) learning a pick-and-place task with novel goal positions for a robot, and 3) learning a point to point motion in the presence of a non-convex obstacle for a robot.**

I. Introduction

We are interested in designing a controller for trajectory learning from demonstration (LfD) that is robust against both spatial and temporal perturbations, offers high accuracy, and is computationally efficient. These properties are often required for robots operating in the real world. In unstructured environments, the robot's trajectory can be blocked at any moment by an unexpected obstacle. Robustness to temporal perturbation enables a robot to successfully resume its previous trajectory once the path is cleared. The robot must exhibit robustness to spatial perturbations, e.g. the goal position of a manipulation task may vary, even during execution. Finally, high reproduction accuracy is required for tasks where constraints are implicitly captured, such as pick and place.

Two primary trajectory LfD paradigms have emerged in the literature: 1) autonomous control and 2) non-autonomous control. Non-autonomous methods, such as Dynamic Motion Primitive (DMP) [1] and Probabilistic Movement Primitives (ProMPs) [2], are generally more accurate, but lack temporal robustness. Since they are (at least implicitly) time-dependent, a separate mechanism is needed to re-index the motion if temporal perturbations occur [3]. Autonomous methods, such as Stable Estimator of Dynamical Systems (SEDS) [3] and the Control Lyapunov Function-based Dynamic Movements (CLF-DM) [4], benefit from temporal and spatial robustness, but are generally less accurate, especially when sharp movements are required. The trade-off between accuracy and stability has been previously identified as the accuracy vs. stability dilemma [5].

[1] Department of Computer Science, University of New Hampshire, USA
paul.gesel@unh.edu, mbegum@cs.unh.edu

In this paper, we propose an autonomous dynamic system (ADS) for trajectory LfD, named Learning Stable Dynamics via Iterative Quadratic Programming (LSD-IQP), that exhibits accuracy and robustness to spatial and temporal perturbations. Our method performs trajectory learning in two stages: 1) learning an energy function from demonstrations and 2) learning an ADS that is subjected to a set of stability criteria derived from the learned energy function. The accuracy and robustness of LDS-IQP is achieved through the learned energy function that can have both local maximums and saddle points. Additionally, LSD-IQP has the inherent ability to avoid non-convex obstacles since we employ semi-infinite programming. Finally, we explicitly consider demonstration tracking in our formulation to avoid visiting a state not seen in the demonstrations, thus minimizing the effect of covariant shift.

II. Related work

DMP [1], [6], [7], ProMPs [2], [8], [9], and SEDS variations [3], [10], [11] are widely used in trajectory LfD. Since the DMPs and ProMPs methods are (at least implicitly) time-dependent, we consider them to be a different class of methods. Consequently, they are not the focus of this paper. More recent ADS approaches, namely CLF-DM and FSM-DS, have been developed in [4] and [12], which benefit from time-invariant control and increased accuracy.

SEDS is a dynamic system (DS) based controller that learns a direct mapping from state (position) to action (velocity) [3]. The controller is parameterized with a linear combination of basis functions. Conditions on the controller's parameters are developed via a Lyapunov stability analysis, ensuring that the motion always converges to a single goal position. Learning is then formulated as a non-linear constrained optimization problem. The method is autonomous and exhibits both spatial and temporal robustness to perturbations. However, the Lyapunov function is quadratic, which limits the reproduction accuracy for more complex motions. Additionally, non-linear constrained optimization can be computationally expensive if many parameters are used.

CLF-DM exploits multi-stage learning to achieve improved accuracy [4]. In the first step, a control Lyapunov function is learned from demonstrations via a constrained non-linear optimization. The second step learns a potentially unstable ADS with a standard regression technique. Finally, the last step uses the learned Lyapunov function to stabilize the learned ADS via a constrained convex optimization. The Lyapunov function learning is formulated in a way, such that the gradient of the Lyapunov function cannot be zero, other than at the goal point. Consequently, the Lyapunov function

2023 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-2365-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRA48891.2023.10161237

cannot have local maximums, hence the class of possible motions is restricted. Compared to SEDS, the method achieves improved accuracy, but still suffers from the computational complexity of general non-linear programming.

In [12], a more recent ADS method named FSM-DS applies Extreme Learning Machines (ELMs) to learn demonstrated motions. Similar to SEDS, a constrained non-linear optimization is developed to learn the controller. Constraints on the optimization are determined that guarantee stability through a Lyapunov analysis. FSM-DS obtained a reduced swept error area compared to other methods, including SEDS and CLF-DM. However, as with other methods, the Lyapunov function cannot have local maximums or saddle points.

In [13], learning is achieved by applying a diffeomorphic transformation to a base ADS. The transformation distorts the velocity field to match that of the demonstrated motions. The base ADS has a quadratic Lyapunov function and, thus, is stable. The diffeomorphic transformation cannot change the underlying topology, hence the deformed Lyapunov function must still have only one critical point. As stated earlier, this ultimately limits the types of motions that can be expressed. Another limitation of this approach is that it only handles a single demonstration.

Recent advances in hierarchical policy learning have successfully integrated dynamic system-based controllers as layers in the deep learning pipeline [14], [15], [16], [17]. For example, Neural Dynamic Policies (NDP) [14] is a deep policy learning method that integrates DMP into its network. The method showed a substantially higher success rate for simulated tasks in Mujoco compared to a neural network policy. The drawback of the aforementioned method is that it requires interaction with the environment and is computationally expensive to learn.

In this paper, we develop LSD-IQP to overcome the problems of other ADS methods. Our method exhibits both high reproduction accuracy and increased flexibility, including concave obstacle avoidance. We are able to achieve this result because our energy function is allowed to have local maximums and saddle points. To the best of our knowledge, there is no other ADS-based method that exhibits this capability.

III. Proposed Approach

We formulate a closed-loop control policy as an ADS, namely

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

where $f : R^n \rightarrow R^n$ is a function mapping from state $\boldsymbol{x}$ to action $\dot{\boldsymbol{x}}$ of an n dimensional motion. This formulation yields two important implications: 1) the rate of change of the state is directly controllable and 2) the control policy is closed-loop and time-invariant. The goal in trajectory LfD is to learn a function $f(\boldsymbol{x})$, such that some characteristics of the demonstration are learned. We will parameterize the control policy $f(\boldsymbol{x})$ with a weighted linear combination of basis functions and develop objective functions for learning the demonstration's key characteristics. A convex optimization will then be formulated with quadratic penalties on the basis function weights. In general, optimizing the weights with respect to an arbitrary objective function directly does not guarantee convergence to a single stable goal point, e.g. spurious attractors and limit cycles will exist. Hence, we impose constraints on the optimization to ensure stability.

A. Stability analysis

The use of an energy function to prove a dynamic system's asymptotic stability is ubiquitous. This type of analysis is often performed when proving stability via a Lyapunov function. We aim to develop a general energy function $V(\boldsymbol{x})$, which can have local maximums and saddle points. Our objective is to use the energy function to stabilize $f(\boldsymbol{x})$.

We require the following three conditions on the energy function to be met,

$$V(\mathbf{0}) = 0 \tag{2}$$

$$\forall_{\boldsymbol{x} \in \mathcal{X} \setminus \{\mathbf{0}\}} V(\boldsymbol{x}) > 0 \tag{3}$$

$$\forall_{\boldsymbol{x} \in \mathcal{X} \setminus \{\mathbf{0}\}} \exists_{\{\boldsymbol{x}_0 \in \mathcal{X} \,:\, 0 < ||\boldsymbol{x} - \boldsymbol{x}_0|| < \epsilon\}} V(\boldsymbol{x}_0) - V(\boldsymbol{x}) < 0 \tag{4}$$

where $\mathcal{X} \subset \mathbb{R}^n$ is the set of all feasible states and ϵ is a small value. The stated constraints imply that 1) $V(\boldsymbol{x})$ has only one minimum, which is global at $\boldsymbol{x} = \mathbf{0}$ and 2) $V(\boldsymbol{x})$ is locally negative definite or indefinite everywhere except $\mathbf{0}$, e.g. there is always a direction that decreases $V(\boldsymbol{x})$. Additionally, since $V(\mathbf{0})$ is a global minimum, $V(\boldsymbol{x})$ is locally positive definite at the origin. In summary, all stationary points of $V(\boldsymbol{x})$ must be either maximums or saddles, other than the minimum at the origin. Given the energy function, we then require the following condition on the ADS ($f(\boldsymbol{x})$) to be satisfied.

$$\forall_{\boldsymbol{x} \in \mathcal{X} \setminus \{\mathbf{0}\}} \|\nabla V(\boldsymbol{x})\| > 0 \implies \nabla V(\boldsymbol{x}) \cdot f(\boldsymbol{x}) < 0 \tag{5}$$

Effectively, the ADS is required to move in the negative gradient of $V(\boldsymbol{x})$ when it is non-zero.

Two additional conditions are required to handle infeasible regions. We assume each infeasible region is entirely contained inside or outside an n-dimensional closed polytope. We require that the normal vector at every point on the polytope is in the direction of the feasible states. The following conditions then must be satisfied,

$$\forall_{\boldsymbol{x} \in \mathcal{O}} \nabla V(\boldsymbol{x}) \cdot N(\boldsymbol{x}) > 0 \tag{6}$$

$$\forall_{\boldsymbol{x} \in \mathcal{O}} f(\boldsymbol{x}) \cdot N(\boldsymbol{x}) > 0 \tag{7}$$

where $\mathcal{O}$ is the union of all points on each infeasible region's polytope and $N : \mathbb{R}^n \rightarrow \mathbb{R}^n$ maps points to their corresponding normals. The conditions require that the direction of motion points into the feasible states if $\boldsymbol{x}$ is on the polytope. It also prevents $\boldsymbol{x}$ from entering into the infeasible states.

Given the functions $V(\boldsymbol{x})$ and $f(\boldsymbol{x})$ which satisfy the required conditions, we can show asymptotic stability for all non-stationary starting points in the feasible state space.

Lemma 3.1: The value of $V(\boldsymbol{x})$ always decreases when $\boldsymbol{x}$ is not in the set of a stationary points $\mathcal{S}$ or at the origin, specifically $\forall_{\boldsymbol{x} \in \mathcal{X} \setminus (\{\mathbf{0}\} \cup \mathcal{S})} \dot{V}(\boldsymbol{x}) < 0$

Proof: We can use a first order approximation of $V(\boldsymbol{x})$ around a fixed point $\boldsymbol{x}_0$:

$$V(\boldsymbol{x}) = V(\boldsymbol{x}_0) + \nabla V(\boldsymbol{x}_0)^T(\boldsymbol{x} - \boldsymbol{x}_0)$$

It then follows that:

$$\boldsymbol{x}(t) = \boldsymbol{x}_0 \implies \boldsymbol{x}(t+dt) = \boldsymbol{x}_0 + f(\boldsymbol{x}_0)dt$$
$$V(\boldsymbol{x}(t+dt)) = V(\boldsymbol{x}_0) + \nabla V(\boldsymbol{x}_0)^T(f(\boldsymbol{x}_0)dt)$$

Eq. (5) requires that $\nabla V(\boldsymbol{x})$ dotted with $f(\boldsymbol{x})$ is negative, which implies:

$$V(\boldsymbol{x}(t+dt)) - V(\boldsymbol{x}_0) < 0$$
$$\dot{V}(\boldsymbol{x}) < 0$$

■

Lemma 3.1 states that $V(\boldsymbol{x})$ always decreases until $\boldsymbol{x}$ becomes a stationary point or the origin. Since $V(\boldsymbol{x})$ has only one bounded minimum at $\boldsymbol{x} = \mathbf{0}$, $\boldsymbol{x}$ will asymptotically converge to the origin.

$$\forall_{\boldsymbol{x} \in \mathcal{X} \setminus (\{\mathbf{0}\} \cup \mathcal{S})} \lim_{t \to \infty} \boldsymbol{x}(t) = \mathbf{0} \tag{8}$$

We can also show that for all stationary starting points in the feasible state space, there exists a piece-wise function $f(\boldsymbol{x})$ that will decrease $V(\boldsymbol{x})$.

Lemma 3.2: There always exists an $f(\boldsymbol{x}_0)$, such that $\dot{V}(\boldsymbol{x}_0) < 0$ for $\boldsymbol{x}_0 \in \mathcal{S}$, specifically $\forall_{\boldsymbol{x} \in \mathcal{S}} \dot{V}(\boldsymbol{x}) < 0$

Proof: We can use a second order approximation of $V(\boldsymbol{x})$ around a fixed point $\boldsymbol{x}_0$

$$V(\boldsymbol{x}) = V(\boldsymbol{x}_0) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_0)^T \nabla^2 V(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0)$$

It then follows that:

$$\boldsymbol{x}(t) = \boldsymbol{x}_0 \implies \boldsymbol{x}(t+dt) = \boldsymbol{x}_0 + f(\boldsymbol{x}_0)dt$$
$$V(\boldsymbol{x}(t+dt)) = V(\boldsymbol{x}_0) + \frac{1}{2}(f(\boldsymbol{x}_0)dt)^T \nabla^2 V(\boldsymbol{x}_0)(f(\boldsymbol{x}_0)dt)$$

Eq. (4) requires that there exists a vector $(\boldsymbol{x} - \boldsymbol{x}_0)$ with length less than ϵ, such that $(\boldsymbol{x} - \boldsymbol{x}_0)^T \nabla^2 V(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0)$ is negative, which implies:

$$\exists_{f: R^n \to R^n} \frac{1}{2}(f(\boldsymbol{x}_0)dt)^T \nabla^2 V(\boldsymbol{x}_0)(f(\boldsymbol{x}_0)dt) < 0$$
$$\exists_{f: R^n \to R^n} V(\boldsymbol{x}(t+dt)) - V(\boldsymbol{x}_0) < 0$$
$$\exists_{f: R^n \to R^n} \dot{V}(\boldsymbol{x}_0) < 0$$

■

Lemma 3.2 states that there is at least one direction around maximums and saddle points of $V(\boldsymbol{x})$, where $V(\boldsymbol{x})$ decreases. The direction is not known in general. However, the eigenvectors of the Hessian at the stationary points can be calculated. Then the direction of a non-zero eigenvector can be used. Practically speaking, a random $\dot{\boldsymbol{x}}$ can be used, since the stationary points are unstable.

The constraints imposed on the energy functions for methods like SEDS, CLF-DM, and FSM-DS are stricter than the conditions that we require. The common constraint for each method is that the energy function must have only one critical point, which is a minimum. In other words, the energy function and a bowl-shaped manifold are diffeomorphic. The constraint is expressed through eq. (9).

$$\forall_{\boldsymbol{x} \in \mathcal{X} \setminus \{\mathbf{0}\}} \exists_{\boldsymbol{d} \in \mathbb{R}^n} \nabla V(\boldsymbol{x}) \cdot d < 0 \tag{9}$$

Energy functions with peaks, saddles, and ridges violate the constraint in eq. (9), but the constraint in eq. (9) does not violate our method's constraints. Lemma 3.3 illustrates the latter.

Lemma 3.3: If the $V(\boldsymbol{x})$ satisfies eq. (9), then $V(\boldsymbol{x})$ also satisfies eq. (4).

Proof: Eq. (9) implies the gradient is non-zero everywhere other than the origin.

$$\forall_{\boldsymbol{x} \in \mathcal{X} \setminus \{\mathbf{0}\}} \exists_{\boldsymbol{d} \in \mathbb{R}^n} \nabla V(\boldsymbol{x}) \cdot d < 0 \implies \forall_{\boldsymbol{x} \in \mathcal{X} \setminus \{\mathbf{0}\}} ||\nabla V(\boldsymbol{x})|| > 0$$

If the gradient is non-zero, then there must be a nearby point less than ϵ distance away, such that the value of the energy function decreases.

$$\forall_{\boldsymbol{x} \in \mathcal{X} \setminus \{\mathbf{0}\}} ||\nabla V(\boldsymbol{x})|| > 0 \implies$$
$$\forall_{\boldsymbol{x} \in \mathcal{X} \setminus \{\mathbf{0}\}} \exists_{\{\boldsymbol{x}_0 \in \mathcal{X} \,:\, 0 < ||\boldsymbol{x} - \boldsymbol{x}_0|| < \epsilon\}} V(\boldsymbol{x}_0) - V(\boldsymbol{x}) < 0$$

■

Since any valid energy function according to the constraints of eq. (9) is also a valid energy function according to our constraints, our set of possible energy functions is a superset of those from methods like CLF-DM and SEDS. Hence, our method can express a broader class of ADS.

B. Energy function learning

We propose the following parameterized energy function shown in eq. (10).

$$V(\boldsymbol{w^l}, \boldsymbol{x}) = \sum_{i}^{b^l} \boldsymbol{w^l_i} \phi^l_i(\boldsymbol{x}) \tag{10}$$

The basis functions ϕ^l can be arbitrary functions, however, at least one must be bowl-shaped to guarantee at least one feasible solution exists. For our implementation, we use one quadratic basis function and $b^l - 1$ radial basis functions.

To learn the weights $\boldsymbol{w^l}$ from the demonstrations, we need an optimization metric to shape the energy function. We propose two objectives: 1) position tracking and 2) direction following. For position tracking, we want the negative gradient of $V(\boldsymbol{x})$ to point towards the nearest demonstration. This will allow the ADS to approach and follow the demonstrations. We express the objective in eq. (12).

$$near(\boldsymbol{x}) = \arg\min_{\boldsymbol{\xi} \in \mathcal{D}} \|\boldsymbol{\xi} - \boldsymbol{x}\| \tag{11}$$

$$J^{lp}(\boldsymbol{w^l}) = \sum_{\boldsymbol{x} \in \mathcal{X}} \left(\nabla V(\boldsymbol{w^l}, \boldsymbol{x}) \cdot \frac{near(\boldsymbol{x}) - \boldsymbol{x}}{\|near(\boldsymbol{x}) - \boldsymbol{x}\|} - \beta \right)^2 \tag{12}$$

Here, the $near(\boldsymbol{x})$ function returns the point closest to $\boldsymbol{x}$ from the position demonstration set $\mathcal{D} = \left\{ \boldsymbol{\xi^{t,n}} \right\}_{t=0,n=1}^{T^n,N}$, β is a hyperparameter, and $\boldsymbol{\xi^{t,n}}$ is a point from the demonstration

set, where t is the time index and n is the demonstration number. The position tracking objective J^{lp} incentivizes the energy function gradient to point towards the nearest demonstration point.

The second objective is direction following. The negative gradient of the $V(\boldsymbol{x})$ should point in the direction of motion along each demonstration. This allows the controller to precisely follow the demonstrated trajectories. We express the second objective in eq. (13).

$$J^{ld}(\boldsymbol{w^l}) = \sum_{n=1}^{N} \sum_{t=0}^{T^n} \left(\nabla V(\boldsymbol{w^l}, \boldsymbol{\xi^{t,n}}) \cdot \frac{\boldsymbol{\dot{\xi}^{t,n}}}{\|\boldsymbol{\dot{\xi}^{t,n}}\|} - 1 \right)^2 \quad (13)$$

Here, $\dot{\xi}^{t,n}$ are velocities from the demonstrations, N is the number of demonstrations, and T^N is the length of the Nth demonstration. The objective incentivizes the gradient of $V(\boldsymbol{x})$ to point in the direction of the demonstrated motion at each demonstration point.

Given the two objectives, we formulate the optimization in eq. (14) in order to learn the weights $\boldsymbol{w^l}$.

$$\arg\min_{\boldsymbol{w^l}} \left\{ \alpha^l J^{lp}(\boldsymbol{w^l}) + (1-\alpha^l) J^{ld}(\boldsymbol{w^l}) \;\middle|\; \mathcal{C}^l \right\} \quad (14)$$

Here, the hyperparameter α^l varies between 0 and 1 and scales the influence of each objective. Notably, the set $\mathcal{C}^l$ is the set of constraints described by eqs. (2-4) and eq. (6).

Although the objective function is quadratic and the constraints are linear, the optimization cannot be solved with a quadratic program because the constraints are continuous. We, therefore, implemented algorithm 1. Line 1 initializes

Algorithm 1 Optimization via Iterative Quadratic Programming

Input: J, GET_VIOLATION
Output: $\boldsymbol{w}$

1: $\mathcal{C} \leftarrow \{V(\mathbf{0}) = 0\}$
2: **do**
3: $\quad \boldsymbol{w} =$ quadprog(J, $\mathcal{C}$)
4: $\quad \mathcal{C}^i \leftarrow$ GET_VIOLATION($\boldsymbol{w}$)
5: $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}^i$
6: **while** $|\mathcal{C}^i| > 0$

the constraints set $\mathcal{C}$ with the condition from eq. (2). On the first iteration, the optimization is solved via a quadratic program with just one constraint. Line 4 then checks for any constraint violations. If there is one, it is returned, otherwise, an empty set is returned. If no violation occurred, the while loop terminates, otherwise, the new constraint is added to the constraint set and another iteration starts.

Finding the constraint violations for the condition in eq. (4) is not trivial, hence we propose algorithm 2. Furthermore, we present fig. 1 to visually demonstrate how we identify constraint violations. Lines 2-3 initialize an empty open set and an empty closed set for the energy first expansion. Lines 4-6 check the condition for eq. (6). Lines 7-16 correspond to steps 1-3 in fig 1. The closed list starts empty and expands the minimum energy regions first (shown in green). Step 4 in fig.

Algorithm 2 Energy Function Constraint Violation

Input: ϵ

1: **function** GET_VIOLATION(w^l)
2: $\quad open \leftarrow \{\mathbf{0}\}$
3: $\quad closed \leftarrow \{\}$
4: $\quad$ **for** $\boldsymbol{x^i} \in \mathcal{O}$ **do**
5: $\quad\quad$ **if** $\nabla V(\boldsymbol{w^l}, \boldsymbol{x^i}) \cdot N(\boldsymbol{x^i}) \leq 0$ **then**
6: $\quad\quad\quad$ **return** $\left\{-\nabla V(\boldsymbol{w^l}, \boldsymbol{x^i}) \cdot N(\boldsymbol{x^i}) < 0\right\}$
7: $\quad$ **while** $|open| > 0$ **do**
8: $\quad\quad \boldsymbol{x^i} \leftarrow \arg\min_{\boldsymbol{x} \in open} V(\boldsymbol{w^l}, \boldsymbol{x})$
9: $\quad\quad open \leftarrow open \setminus \boldsymbol{x^i}$
10: $\quad\quad closed \leftarrow closed \cup \boldsymbol{x^i}$
11: $\quad\quad$ **for** $j \in \{1..n\}$ **do**
12: $\quad\quad\quad \boldsymbol{x^c} \leftarrow \boldsymbol{x^i}$
13: $\quad\quad\quad$ **for** $k \in \{-1, 1\}$ **do**
14: $\quad\quad\quad\quad \boldsymbol{x^c_j} \leftarrow \boldsymbol{x^i_j} + k\epsilon$
15: $\quad\quad\quad\quad$ **if** $\boldsymbol{x^c} \notin closed$ **then**
16: $\quad\quad\quad\quad\quad open \leftarrow open \cup \boldsymbol{x^c}$
17: $\quad\quad\quad\quad\quad$ **if** $V(\boldsymbol{w^l}, \boldsymbol{x^i}) - V(\boldsymbol{w^l}, \boldsymbol{x^c}) \geq 0$ **then**
18: $\quad\quad\quad\quad\quad\quad$ **return** $\left\{V(\boldsymbol{w^l}, \boldsymbol{x^i}) - V(\boldsymbol{w^l}, \boldsymbol{x^c}) < 0\right\}$
19: $\quad$ **return** $\{\}$

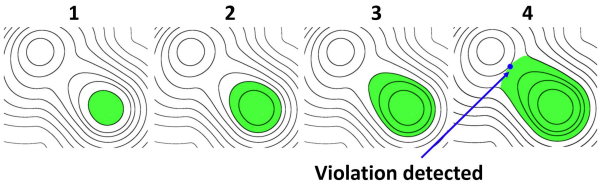

Fig. 1: Four steps of the constraint violation detection are shown. Steps 1-3 show an energy-first expansion from the origin. Step 4 is the first time a constraint violation is detected.

1 corresponds to the condition in line 17 being true, when the energy of the expanded state is lower than the current state. The algorithm will continue to run until all $\mathcal{X}$ are expanded or a constraint violation is found.

Notably, algorithm 2 finds constraint violations on a regular grid determined with the input variable ϵ. There is a possibility that constraint violations exist between the discrete points, e.g. there could be a local minimum. This is, however, unlikely with wide smooth basis functions, such as radial basis functions. Nevertheless, we handle this possibility by using linear interpolation for points between grid points. Minimums of a linearly interpolated function can only occur on the grid. Since the constraints are enforced on the grid, we can say there will be no constraint violating minimums.

C. Controller learning

Given the energy function, we can learn a stable ADS $f(\boldsymbol{x})$ from the demonstrations. We parameterize the $f(\boldsymbol{x})$ the following way.

$$f(\boldsymbol{w^c}, w^v, \boldsymbol{x}) = \sum_{i}^{b^c} \boldsymbol{w^c_i} \nabla \phi^c_i(\boldsymbol{x}) + w^v \nabla V(\boldsymbol{x}) \quad (15)$$

Here, b^c is the number of basis functions used for $f(\boldsymbol{x})$. Notably, the controller is composed of a linear combination of arbitrary basis functions $\nabla\phi_i^c(\boldsymbol{x})$ and the gradient of the energy function. By including ∇V as one of the basis functions, we guarantee that the condition in eq. (4) can be satisfied. The solution is trivially obtained when $w^v = -1$ and $\boldsymbol{w^c} = \mathbf{0}$, as the controller will follow the energy function's negative gradient.

We want the controller to move toward the nearest demonstration and then begin to follow it. We again propose two objectives to achieve this; one for position tracking and the other for velocity tracking. The first objective function is shown in eq. (17).

$$g(\boldsymbol{x}) = \|\dot{near}(\boldsymbol{x})\| \frac{near(\boldsymbol{x}) - \boldsymbol{x}}{\|near(\boldsymbol{x}) - \boldsymbol{x}\|} \tag{16}$$

$$J^{cp}(\boldsymbol{w^c}, w^v) = \sum_{\boldsymbol{x}\in\mathcal{X}} \left\| f(\boldsymbol{w^c}, w^v, \boldsymbol{x}) - g(\boldsymbol{x}) \right\|^2 \tag{17}$$

The $\dot{near}(\boldsymbol{x})$ function refers to the velocity corresponding to the nearest demonstration point. The second objective aims to follow the demonstrated velocity once the demonstration is reached. This objective is shown in eq. (18).

$$J^{cd}(\boldsymbol{x}) = \sum_{n=1}^{N} \sum_{t=0}^{T^n} \left\| f(\boldsymbol{\xi^{t,n}}) - \boldsymbol{\dot{\xi}^{t,n}} \right\|^2 \tag{18}$$

Finally, the optimization in eq. (19) is formulated.

$$\underset{\boldsymbol{w^c}, w^v}{\arg\min} \left\{ \alpha^c J^{cp}(\boldsymbol{w^c}, w^v) + (1-\alpha^c) J^{cd}(\boldsymbol{w^c}, w^v) \;\middle|\; \mathcal{C}^c \right\} \tag{19}$$

As before, the constraints required in eqs. (5) and (7) are continuous and cannot be solved with a regular quadratic program. Instead, we apply algorithm 1. To find the constraint violation points, we implemented algorithm 3. The algorithm

Algorithm 3 Controller Constraint Violation Function

Input: ϵ

function GET_VIOLATION($\boldsymbol{w^c}, w^v$)
  for $\boldsymbol{x^i} \in \mathcal{O}$ **do**
    if $f(\boldsymbol{w^c}, w^v, \boldsymbol{x^i}) \cdot N(x^i) \leq 0$ **then**
      return $\{-f(\boldsymbol{w^c}, w^v, \boldsymbol{x^i}) \cdot N(\boldsymbol{x^i})\}$
  for $\boldsymbol{x^i} \in \mathcal{X}$ **do**
    if $f(\boldsymbol{w^c}, w^v, \boldsymbol{x^i}) \cdot \nabla V(\boldsymbol{w^l}, \boldsymbol{x^i}) \geq 0$ **then**
      return $\{f((\boldsymbol{w^c}, w^v, \boldsymbol{x^i}) \cdot \nabla V(\boldsymbol{w^l}, \boldsymbol{x^i})\}$
  return $\{\}$

checks the conditions in eqs. (5) and (7) and returns a violation if found. Similar to algorithm 2, a parameter ϵ is required to specify the discrete interval to check for violations.

D. Dependent dimensions

For many pick-and-place tasks, the orientation of the end effector can be estimated as a function of the Cartesian position. Therefore, we directly model dependent dimensions as a function of the independent dimensions (state) and the goal.

$$\boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{g}) \tag{20}$$

Here, $\boldsymbol{y} \in \mathbb{R}^m$ represents the dependent dimensions, $h : R^n \rightarrow R^m$ is a function that estimates the dependent dimensions given the state $\boldsymbol{x}$, and $\boldsymbol{g} \in \mathbb{R}^n$ is the goal parameter. We parameterize $h(\boldsymbol{x}, \boldsymbol{g})$ with radial basis functions and estimate $\boldsymbol{y}$ via least square. Finally, we apply a proportional control to track estimated dependent dimensions. Notably, it is not necessary to model any dependent dimensions, but doing so is more computationally and data-efficient.

IV. Results

We evaluate LSD-IQP with three experiments: 1) learning motions from the handwritten LASA data set [18], 2) learning a pick and place task for a robot, and 3) learning a point-to-point motion with a non-convex obstacle for a robot. We directly compare LSD-IQP to SEDS, CLF-DM, and FSM-DS in experiment 1 since they are all ADS methods. Results are quantified with the swept error area (SEA) metric and learning time for 30 motions. For both robot experiments, two Cartesian dimensions were used for the state and all other dimensions were made dependent. The source code for each method's implementation is located at [19], [20], and [21], respectively.

A. Experiment 1: learning handwritten symbols

This experiment demonstrates LSD-IQP's increased accuracy in learning complex motion compared to other ADS methods. We present quantitative results in terms of SEA and total learning time in table I. SEA is a commonly used benchmark for the LASA data set. The metric effectively measures spatial and temporal accuracy. A low SEA value corresponds to a motion that has the same shape as the demonstrations and is aligned in time. For a more detailed explanation, see [4]. Compared to the other approaches, LSD-IQP showed the best performance in terms of mean SEA and total learning time.

TABLE I Comparison of the mean swept error area (mm^2) and total learning time (sec) for the SEDS, CLF-DM, FSM-DS, and LSD-IQP methods.

	SEDS	CLF-DM	FSM-DS	LSD-IQP
SEA	351.3	181.3	164.3	**93.5**
time	376.7	412.2	351.3	**182.7**

While SEA is the standard metric used to measure performance on the LASA data set, it does not capture all important characteristics of ADS learning. Namely, the SEA is evaluated by reproducing motions from the same points as the original demonstrations. This is shown for a sample written letter in fig. 2 with green dots. In this case, LSD-IQP reproduces the demonstrations correctly, while SEDS fails to learn the motion. SEDS's quadratic Lyapunov function is too restricted to generate motion that moves too directly away from the goal. The increased performance of LSQ-IQP over other methods is apparent when reproducing motions from novel starting positions. Three arbitrary novel starting points are shown in fig. 2. LSQ-IQP quickly and smoothly converges

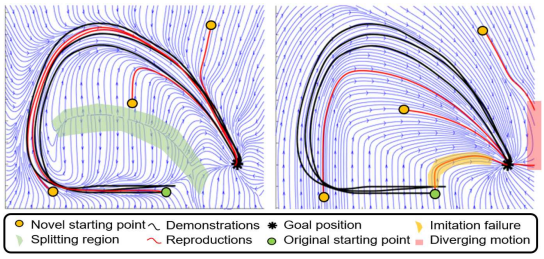

Fig. 2: Comparison of streamline plots for LSD-IQP (left) and SEDS (right). LSD-IQP shows stable and accurate reproduction under spatial perturbation.

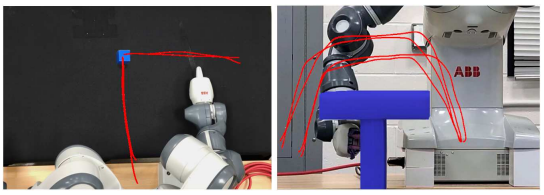

Fig. 3: Demonstrations for pick and place task (left) and point to point task (right).

to the original demonstrations from each point, while SEDS generally maintains the shape of the demonstrations. The effects of the distributional shift are highlighted in the red region. The motion divergences from the demonstrations, which could lead to the robot moving outside of its workspace.

B. Experiment 2: pick and place task with novel goals

Our second experiment validated LSD-IQP ability to learn multi-modal demonstrations via a pick-and-place task on a robotic platform. Two demonstrations were provided for both picking and placing at two different goal locations, totaling eight demonstrations. The demonstrations were provided by a human teacher via kinesthetic teaching. Fig 3 illustrates the demonstrations for both robot experiments. Note, the demonstrations are shown relative to a single block location. The task requires precision and a proper approach angle to avoid the robot's fingers from colliding with the block. Also, since the block is picked from the tabletop, precise clearance is required to avoid a collision. For this experiment, motions were executed by mapping learned Cartesian velocities to joint velocities with the pseudo-inverse of the Jacobian. The robot was successfully able to pick and place the block at novel goal positions along a 3x3 grid with 5 cm spacing.

This experiment demonstrates LSD-IQP's ability to model

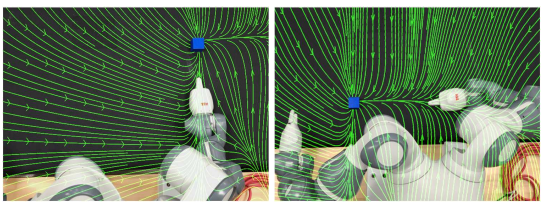

Fig. 4: Streamline plots of LSD-IQP for the pick and place task for two different goal positions.

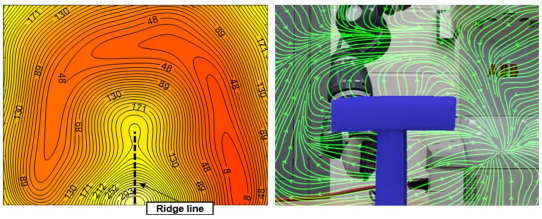

Fig. 5: Energy function learned via LSD-IQP (left) and streamline plot for the learned controller.

multi-modal trajectories. As seen in fig. 4, the flow lines drive the robot's end effector to the goal location via two primary modes. If the block is on the robot's left, it will grab the block from the right side, while if the block is in front of the robot, it will grab the block by moving straight forward. These two different modes were present in the demonstrations.

C. Experiment 3: point-to-point motion with an obstacle

We further validated LSD-IQP with a point-to-point obstacle avoidance task on a robotic platform. In this task, the robot must maneuver its hand above the obstacle and then slide along the surface while maintaining a suitable clearance. The key challenge is that the obstacle is non-convex and must be avoided from any starting position. For this experiment, four demonstrations were provided.

This experiment demonstrates LSD-IQP's ability to avoid infeasible regions of the state space. The learned controller converges to the goal from all feasible states. As seen in fig. 5, the flow on the boundary of the obstacle always has some component in the normal direction. This result is achievable only because the learned energy function has two maximums, forming a ridge line inside of the obstacle. This effect is illustrated by the dashed black line in fig. 5. The energy functions of other ADS methods, such as CLF-DM, must have only one critical point, which is a minimum. The topology of those types of energy functions is fundamentally diffident from that of the LSD-IQP, consequently, they are not able to generate motion that circumvents the obstacle. The flexible energy function topology allowed in our formulation enables learning of complex ADS, even in a non-convex feasible state space.

V. Conclusion

In this paper, we developed LSD-IQP to learn an ADS from multiple demonstrations. Learning was broken into two stages: 1) learning an energy function and 2) learning an ADS. In both cases, iterative quadratic programming with constraint generation was employed to solve the optimization problem. We validated LSD-IQP in three experiments and showed increased accuracy and capability compared to other ADS methods.

Acknowledgment

This work was supported in part by the National Science Foundation (IIS 1830597).

REFERENCES

[1] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 763–768.

[2] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," 01 2013.

[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, Oct 2011.

[4] S. Mohammad Khansari-Zadeh and A. Billard, "Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889014000372

[5] K. Neumann and J. J. Steil, "Learning robot motions with stable dynamical systems under diffeomorphic transformations," *Robotics and Autonomous Systems*, vol. 70, pp. 1–15, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889015000883

[6] Z. Li, T. Zhao, F. Chen, Y. Hu, C.-Y. Su, and T. Fukuda, "Reinforcement learning of manipulation and grasping using dynamical movement primitives for a humanoidlike mobile manipulator," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 121–131, 2018.

[7] C. Yang, C. Chen, W. He, R. Cui, and Z. Li, "Robot learning system based on adaptive neural control and dynamic movement primitives," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 3, pp. 777–787, 2019.

[8] F. Frank, A. Paraschos, P. van der Smagt, and B. Cseke, "Constrained probabilistic movement primitives for robot trajectory adaptation," *IEEE Transactions on Robotics*, vol. 38, pp. 2276–2294, 2022.

[9] R. A. Shyam, P. Lightbody, G. Das, P. Liu, S. Gomez-Gonzalez, and G. Neumann, "Improving local trajectory optimisation using probabilistic movement primitives," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2666–2671.

[10] 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.

[11] 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.

[12] J. Duan, Y. Ou, J. Hu, Z. Wang, S. Jin, and C. Xu, "Fast and stable learning of dynamical systems based on extreme learning machine," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 6, pp. 1175–1185, 2019.

[13] N. Perrin and P. schlehuber caissier, "Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems," *Systems & Control Letters*, vol. 96, pp. 51–59, 10 2016.

[14] S. Bahl, M. Mukadam, A. Gupta, and D. Pathak, "Neural dynamic policies for end-to-end sensorimotor learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 5058–5069. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/354ac345fd8c6d7ef634d9a8e3d47b83-Paper.pdf

[15] S. Bahl, A. K. Gupta, and D. Pathak, "Hierarchical neural dynamic policies," in *RSS*, 2023.

[16] S. Pirk, K. Hausman, A. Toshev, and M. Khansari, "Modeling long-horizon tasks as sequential interaction landscapes," in *CoRL*, 2020.

[17] R. Pahič, A. Gams, A. Ude, and J. Morimoto, "Deep encoder-decoder networks for mapping raw images to dynamic movement primitives," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5863–5868.

[18] S. M. Khansari-Zadeh and A. Billard. Lasa handwriting dataset. [Online]. Available: https://cs.stanford.edu/people/khansari/download.html

[19] S. M. Khansari-Zadeh. Stable estimator of dynamical systems (seds) - source code. [Online]. Available: https://bitbucket.org/khansari/seds/src/master/

[20] ——. Control lyapunov function-based dynamics movement (clf-dm) - source code. [Online]. Available: https://bitbucket.org/khansari/clfdm/src/master/

[21] J. Duan, Y. Ou, J. Hu, Z. Wang, S. Jin, and C. Xu. Fast and stable learning of dynamical systems based on extreme learning machine - source code. [Online]. Available: https://github.com/SIAT-CIBS/FSM-DS/