Enforcing the consensus between Trajectory Optimization and Policy Learning for precise robot control

Quentin Le Lidec[*,1]   Wilson Jallet[1,2]   Ivan Laptev[1]   Cordelia Schmid[1]   Justin Carpentier[1]

Abstract— **Reinforcement learning (RL) and trajectory optimization (TO) present strong complementary advantages. On one hand, RL approaches are able to learn global control policies directly from data, but generally require large sample sizes to properly converge towards feasible policies. On the other hand, TO methods are able to exploit gradient-based information extracted from simulators to quickly converge towards a locally optimal control trajectory which is only valid within the vicinity of the solution. Over the past decade, several approaches have aimed to adequately combine the two classes of methods in order to obtain the best of both worlds. Following on from this line of research, we propose several improvements on top of these approaches to learn global control policies quicker, notably by leveraging sensitivity information stemming from TO methods via Sobolev learning, and Augmented Lagrangian (AL) techniques to enforce the consensus between TO and policy learning. We evaluate the benefits of these improvements on various classical tasks in robotics through comparison with existing approaches in the literature.**

I. INTRODUCTION

BY leveraging derivative information from the dynamics and costs, optimal control (OC) algorithms [1], [2], [3], [4] efficiently compute local controllers. Model Predictive Control (MPC) [5] aims at retrieving a local controller with state feedback through online re-planning. However, current algorithms remain computationally too expensive to be run at high frequencies, are sensible to local minima and integration of information from sensors (e.g. force, vision) is technically difficult. This limits both their real-world capabilities and the practicality of deployment on complex settings requiring various sensor modalities. For these reasons, control policies learned directly from data are appealing as they offer the possibility to perform sensor-fusion and executed at very high frequencies.

Reinforcement learning (RL) [6] is the prevailing paradigm when it comes to learning policies directly from observations. In RL, the physical dynamics are usually considered as unknown and classical algorithms [7] rely on stochastic policies to build 0th-order gradient estimators. Although they are unbiased, the high variance of such estimators make them very sample-inefficient. Actor critic approaches [8] combine ideas from policy gradient and approximate dynamic programming by using an estimate of the Q function to reduce variance of policy gradient algorithms. This improves the sample efficiency but often at the cost of less stable training, and more advanced strategies, *e.g.* Proximal Policy Optimization (PPO) [9] or Soft Actor-Critic (SAC) [10], are often focused on regaining more stability. Eventually, even on problems easily solved via standard *model-based* approaches such as LQR, model-free algorithms exhibit several limitations and simpler rather than more complex methods may perform better [11].

In robotics, RL is never truly model-free, as samples are often generated by simulators which are actually based on models of physics. Therefore, we propose to embrace models when they are available, and, by using gradient information, to move from 0th-order to 1st-order methods with superlinear convergence properties. Such an approach is made possible by the recent emergence of differentiable physics engines, relying on derivatives of rigid body algorithms [12] and frictional contacts [13], [14]. A second key ingredient for model-based approaches is access to efficient numerical solvers for trajectory optimization [4], [15]. Indeed, specialized OC algorithms exploit physical models, their derivatives, and the inherent temporal structure of the problem via the Bellman equation (or Pontryagin's principle). In particular, quasi-Newton (*e.g.,* iterative Linear Quadratic Regulator (iLQR) [1], [3]) or second-order methods (*e.g.,* Differential Dynamic Programming (DDP) [2]), lead to improved convergence rates.

For complex scenarios, OC is often too slow to be run online, especially when it includes constraints on the system. Yet, it remains attractive to draw benefits from its capabilities offline; typically, for training policies. Following this path, guided policy search (GPS) [16], [17] aims at combining the strengths of OC and policy learning. Closer to our work, [18] considers known deterministic dynamics and uses an ADMM framework to embed the policy learning problem within the OC formulation. The dual update from ADMM allows to automatically and gradually reach a consensus between the learned policy and the local controllers that OC yields. However, they relax the agreement constraint between RL and OC, which results in shifted, thus degraded, solutions. Additionally, they further exploit the feedback gains from trajectory optimization in order to supervise the first order derivatives of the policy. This technique, also known as Sobolev training [19], [20], regularizes the training and improves generalization and thus the stability of the control policy. In this work, we extend it by using a stochastic variant [20] to boost computational efficiency. Finally, a more recent work [21] improves rollouts of the policy by performing approximate Newton steps in the trajectory space. Here we establish some links between such an approach and OC, which in turn allows us to set the stage for a general formulation incorporating constraints while resulting in more

*Corresponding author
[1]Inria - Département d'Informatique de l'École normale supérieure, PSL Research University. Email: `firstname.lastname@inria.fr`
[2]LAAS-CNRS, 7 av. du Colonel Roche, 31400 Toulouse

	Precise	Efficient	Constraints	Sobolev	Model-free
RL					✓
DPL [18]		✓		✓	
PODS [21]	✓	✓			
AL (Ours)	✓	✓	✓	✓	
PDDP (Ours)	✓	✓	✓	✓	

efficient algorithms. Table I summarizes some characteristics of our approach and the related works, *i.e.* the precision of the learned policy (Precision), the number of samples required to converge (Efficient), the ability to handle physical constraints (Constraints) or to leverage higher order information (Sobolev) and the need for a physical model (Model-free).

In this paper, we first present a general formulation, including physical and geometrical constraints, for policy learning which lays the ground for a better interplay between OC and policy learning (Sec. II). In Sec. III, we propose a first algorithm based on a projected descent and detail how stochastic Sobolev learning can be leveraged to provide a computationally efficient higher-order supervision of policy training. A second variation based on Augmented Lagrangian (AL) techniques is introduced in Sec. IV to automatically and precisely reach a consensus between the policy and the trajectory optimization oracle. To properly account for the lack of consensus at the beginning of the learning process, we propose to add multiple shooting component which allows to sensibly reduce the computational timings of learning a control policy. Finally, in Sec. V, the influence of each component of the approach is evaluated separately and we validate the final algorithms on a set of standard robotics tasks.

II. INTERPLAY BETWEEN POLICY LEARNING AND TRAJECTORY OPTIMIZATION

We consider deterministic and known system dynamics

$$x_{t+1} = f(x_t, u_t) \tag{1a}$$
$$x_t \in \mathcal{X} \text{ and } u_t \in \mathcal{U} \tag{1b}$$

while other parameters of the optimal control problem (OCP) are randomly distributed. These parameters, denoted by β, include the initial condition of the problem $x^0 \in \mathcal{X}$, as well as other parameters (desired terminal state or end-effector position); they follow a random distribution $\mathcal{P}$. The set $\mathcal{U} = [u_{\min}, u_{\max}]$ contains the admissible control inputs, of dimension n_u, while $\mathcal{X}$ refers to the set of feasible states also accounting for obstacles. The deterministic assumption on the dynamics is reasonable, as a wide range of systems (robotic arms, manipulators, quadrupeds, drones) can accurately be described without stochasticity. We denote $\boldsymbol{u} = (u_0, \ldots, u_{T-1})$ the sequence of controls, and $\boldsymbol{x} = (x_0, \ldots, x_T)$ the state sequence. The total cost associated with a given control sequence $\boldsymbol{u}$ is:

$$R(\beta; \boldsymbol{u}) = \sum_{t=0}^{T-1} \ell_t(x_t, u_t; \beta) + \ell_T(x_T; \beta) \tag{2}$$

Unlike previous works in the literature [18], [21], we also account in this work for state and control constraints in Eq. (1b) directly in the TO problems. Experiments from Sec. V notably demonstrate how this consideration of the constraints substantially improves the precision and the stability of the resulting policy.

The goal of this paper is to learn closed-loop control policies $\pi \colon \mathcal{X} \to \mathcal{U}$ which approximately minimizes the trajectory cost. The policy π is constrained to lie in a set of parameterized policies Π; typically, we will consider neural network policies with two hidden layers of 256 units each. The policy parameters are denoted by θ and lie in a set Θ. The policy space is thus

$$\Pi = \{\pi_\theta : \theta \in \Theta\}.$$

The activation functions are ReLU except for the final layer which is activated by a hyperbolic tangent so that the output is restricted to $[u_{\min}, u_{\max}]$. We denote by R, the cost function and $J(\beta; \pi_\theta)$ the *policy cost*, which is the trajectory cost with controls given by $u_t = \pi_\theta(x_t)$:

$$J(\beta; \pi_\theta) = R(\beta; \pi_\theta(x_0), \ldots, \pi_\theta(x_{T-1}))$$
$$\text{with } x_0 = x^0, \ x_{t+1} = f(x_t, \pi_\theta(x_t)) \tag{3}$$

Our goal is to solve the *policy optimization problem*:

$$\min_{\theta \in \Theta} \mathbb{E}_\beta [J(\beta; \pi_\theta))] \tag{4}$$

where $\mathbb{E}_\beta[\cdot]$ is the expectation.

To do so, we will follow the path initiated in [18], [22] and adopt a constrained formulation linking together trajectory optimization and policy optimization:

$$\min_{\theta, \boldsymbol{u}} \mathbb{E}_\beta [R(\beta; \boldsymbol{u})] \tag{5a}$$
$$\text{s.t. } \forall t \ u_t = \pi_\theta(x_t), \text{ a.s.} \tag{5b}$$

where we now optimize over a *distribution* of control sequences $\boldsymbol{u}$, and $\boldsymbol{x} = (x_t)_t$ is defined by (1a). The constraint enforces that the controls and the policy are consistent on almost all states x_t reached by randomization of β.

III. ALTERNATING BETWEEN OC AND SUPERVISED LEARNING: A PROJECTED DDP APPROACH

Projected DDP algorithm. A classical way of solving a constrained optimization problem consists in alternating between making a step on the unconstrained problem and projecting the new iterate on the constraints (e.g., projected gradient algorithms). In this spirit, (5) can be solved by optimizing the control $\boldsymbol{u}$ via trajectory optimization before projecting the new trajectory on the set of trajectories obtainable through Π. Discarding the constraints, the problem (5) can be solved by approximating the expectation with a finite sum $\hat{R}(\boldsymbol{u})$ obtained by sampling N instances of the problem $\beta^{(i)} \sim \mathcal{P}$:

$$\hat{R}(\boldsymbol{U}) = \frac{1}{N} \sum_{i=1}^{N} R(\beta^{(i)}; \boldsymbol{u}^{(i)}). \tag{6}$$

Algorithm 1: Projected DDP descent (PDDP)

Input: Distribution on the parameters $\beta \sim \mathcal{P}$, model
for the policy: $\pi_\theta \in \Pi$
Output: Optimal policy $\pi_\theta^\star$

1 for $k = 1$ **to** M **do**
2 $\boldsymbol{U}^{k+1,0} \leftarrow \text{Rollout}(\pi_{\theta^k});$ // Initial guess
3 $\boldsymbol{U}^{k+1} \leftarrow \arg\min_{\boldsymbol{U}} \hat{R}(\boldsymbol{U});$ // OC (with guess)
     // Supervised learning (can also use $\hat{d}_2$)
4 $\theta^{k+1} \leftarrow \arg\min_\theta \hat{d}(\theta, \boldsymbol{U}^{k+1});$
5 end

---

where $\boldsymbol{U} = (\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(N)})$ is the entire set of control sequences. The objective above is separable, hence each $\boldsymbol{u}^{(i)}$ can be optimized separately in parallel.

The sampled problems are then optimized via discrete-time trajectory optimization methods such as Differential Dynamic Programming (DDP) [2] or its variant iLQR [3], using a rollout originating from the learned policy π_θ as a warm-start. Then, the local control trajectories $(\boldsymbol{u}^{(i)})_i$ are projected onto an element of Π in the least-squares sense. The projection problem is equivalent to the following supervised learning problem:

$$\min_{\theta \in \Theta} \hat{d}(\theta, \boldsymbol{U}) = \sum_{i=1}^{N} \sum_{t=0}^{T-1} \frac{1}{2} \left\| u_t^{(i)} - \pi_\theta\left(x_t^{(i)}\right) \right\|_2^2. \quad (7)$$

which is solved using classical stochastic mini-batch gradient algorithms such as Adam [23]. Alternating between these two operations results in Alg. 1 which we call Projected DDP (PDDP).

Such an approach encompasses the one proposed in [21] where Gauss-Newton (GN) steps are successively projected. By establishing an interplay with OC, here we rather perform a DDP step which is just a more efficient way to compute a GN step that also handles the constraints from (1). Previous works [18], [21] often ignore these constraints during the trajectory optimization phase. Typically, the box constraint on the control is rather lifted to the policy space via a tanh activation function on the last layer. This discrepancy can be fatal as trajectories obtained from OC might not be reproducible by the neural network and, thus, could cause divergence of the combined approach. Experiments in Sec. V demonstrate how crucial this specificity of our methods is to ensure convergence. Another obvious advantage is the possibility to consider more complex tasks involving geometric constraints or dynamic constraints on the systems to control (*e.g.*, obstacles or joint limits).

Stochastic Sobolev learning. If the constraint from (5) enforces the output of the learned policy to match the local controllers, this should also be true for higher order derivatives. This idea, exploited in [18], refers to the concept of Sobolev training [20]. Second-order methods such as DDP additionally yield local *feedback* gains K_t around the optimized trajectory. These gains can be used for stabilization of the controlled system around the local optimum, as [24] has shown on

real-world systems. For policy learning, we can make use of these gains to regularize the policy π_θ during the supervised learning phase (Alg. 1, line 4), leading to the following supervised learning problem:

$$\min_{\theta \in \Theta} \hat{d}_2(\theta, \boldsymbol{U}) = \hat{d}(\theta, \boldsymbol{U}) + \frac{1}{2} \sum_{i,t} \left\| K_t^{(i)} - \frac{\partial \pi_\theta}{\partial x}(x_t^{(i)}) \right\|_2^2 \quad (8)$$

However, getting the second order derivative $\partial_{x\theta} \pi_\theta$, as in [18], is computationally expensive as it requires to perform multiple backpropagation operations. For this reason, we propose to use a stochastic version of Sobolev learning [20] which is done by matching projections of K and $\partial_x \pi_\theta$ on control directions $v^{(i)}$ randomly sampled on the unit sphere. The regularization term from (8) becomes $\frac{1}{2} \sum_{i,t} \|(v^{(i)})^\top K_t^{(i)} - \partial_x (v^{(i)\top} \pi_\theta(x_t^{(i)}))\|_2^2$ and this greatly improves computational efficiency by avoiding backpropagating several times across the neural network.

IV. GRADUALLY ENFORCING A CONSENSUS: AN AUGMENTED LAGRANGIAN APPROACH

The Lagrangian function associated to the constrained formulation (5) is given by:

$$\mathcal{L}(\theta, \boldsymbol{u}, \boldsymbol{\lambda}) = \mathbb{E}_\beta \left[R(\beta; \boldsymbol{u}) + \sum_{t=0}^{T-1} \lambda_t^\top (u_t - \pi_\theta(x_t)) \right] \quad (9)$$

where $\boldsymbol{\lambda} = (\lambda_t)_{0 \leq t \leq T}$ now designate random variables which are the Lagrange multipliers. We can define the corresponding augmented Lagrangian by:

$$\mathcal{L}_A^\mu(\theta, \boldsymbol{u}, \boldsymbol{\lambda}) = \mathcal{L}(\theta, \boldsymbol{u}, \boldsymbol{\lambda}) + \mathbb{E}_\beta \left[\sum_{t=0}^{T-1} \frac{\mu_t}{2} \|u_t - \pi_\theta(x_t)\|^2 \right] \quad (10)$$

ADMM algorithm. We first consider that a set of sample parameters $(\beta^{(i)})$ is fixed during training. Indeed, the Lagrange multipliers are intrinsically linked to the sampled values of β and thus, re-sampling them would require to update $\boldsymbol{\lambda}$ without losing progress on the optimization, which is a difficult task left as future work. This differs from Alg. 1 which can be naturally run online. Thus, we consider the following sampled version of the constrained problem (5):

$$\min_{\theta, \boldsymbol{U}} \frac{1}{N} \sum_{i=1}^{N} R(\beta^{(i)}; \boldsymbol{u}^{(i)}) \quad (11a)$$

$$\text{s.t. } u_t^{(i)} = \pi_\theta(x_t^{(i)}), \ \forall t, \ \forall i \in [\![1, N]\!] \quad (11b)$$

where $(\beta^{(i)})_{1 \leq i \leq N}$. The classical and augmented Lagrangians for this sampled variant are respectively given by:

$$\hat{\mathcal{L}}(\theta, \boldsymbol{U}, \boldsymbol{\Lambda}) = \hat{R}(\boldsymbol{U}) + \sum_{i,t} (\lambda_t^{(i)})^\top (u_t^{(i)} - \pi_\theta(x_t^{(i)})), \quad (12)$$

$$\hat{\mathcal{L}}_A^\mu(\theta, \boldsymbol{U}, \boldsymbol{\Lambda}) = \hat{\mathcal{L}}(\theta, \boldsymbol{U}, \boldsymbol{\Lambda}) + \sum_{i,t} \frac{\mu_t}{2} \|u_t^{(i)} - \pi_\theta(x_t^{(i)})\|^2, \quad (13)$$

where $\boldsymbol{\Lambda} = (\boldsymbol{\lambda}^{(1)}, \ldots, \boldsymbol{\lambda}^{(N)})$ are the Lagrange multipliers for each constraint block $\{u_t^{(i)} = \pi_\theta(x_t^{(i)})\}_t$. Both functions are separable with respect to the entry $\boldsymbol{u}^{(i)}$.

Algorithm 2: Policy learning via ADMM (PLAL)

Input: Distribution on the parameters $\beta \sim \mathcal{P}$, model for the policy: $\pi_\theta \in \Pi$
Output: Optimal policy $\pi_{\theta^\star}$

1 for $k = 1$ **to** M **do**
2 $\quad \boldsymbol{U}^{k+1} \leftarrow \arg\min_u \hat{\mathcal{L}}_A^\mu(\theta^k, \boldsymbol{U}, \boldsymbol{\Lambda}^k)$;
$\quad$ // Supervised learning
3 $\quad \theta^{k+1} \leftarrow \arg\min_\theta \hat{\mathcal{L}}_A^\mu(\theta, \boldsymbol{U}^{k+1}, \boldsymbol{\Lambda}^k)$;
$\quad$ // Dual update
4 $\quad \boldsymbol{\Lambda}^{k+1} \leftarrow (\lambda_t^{(i)} + \mu(u_t^{(i)} - \pi_\theta(x_t^{(i)})))_{t,i}$;
5 end

The problem is solved using the alternating direction method of multipliers (ADMM) [25], as described in Alg. 2 which we call PLAL. The algorithm alternatively minimizes $\mathcal{L}_\mathcal{A}^\mu$ w.r.t. the primal variables $\boldsymbol{U}$ and θ, before updating the dual variable $\boldsymbol{\Lambda}$ through a simple dual ascent step. As done in Sec. III, minimization w.r.t. each $\boldsymbol{u}^{(i)}$ is done using a DDP-type algorithm such as FDDP [4], while the minimization w.r.t. θ is performed using a classical stochastic optimization algorithm [23]. If [18] also uses ADMM, the consensus is approximated via a quadratic penalty term, while in this work we enforce it with a hard constraint (11b). This results in more stable and precise solutions as shown in Sec. V.

Contrary to Alg. 1, the OC (Alg. 2, line 2) and supervised learning (Alg. 2, line 3) phases are now linked via the Lagrange multipliers $\boldsymbol{\Lambda}$. Indeed, the additional terms appearing in (13) enforces the demonstrations from OC to be adapted to the current capabilities of the policy, while the dual update (Alg. 2, line 4) gradually leads to an agreement between the local controllers from OC and the policy. The previously introduced stochastic Sobolev term can naturally be integrated in the approach as a regularization of the learning phase (Alg. 2, line 3), and the algorithm is then interpretable as an occurrence of *Global Variable Consensus with Regularization* [25].

Multiple shooting formulation. Due to the augmented Lagrangian terms, the trajectory optimization phase (Alg. 2, line 2) requires to optimize the state $\boldsymbol{x}$ through the neural network policy and thus involves the costly computation of the Jacobian $\partial_x \pi_\theta$. Inspired by [18], we propose to decouple the state variables of the supervised learning and OC problems, in a way similar to what is done in multiple shooting [26]. To do so, two state variables are introduced ($^1\boldsymbol{X} = (^1\boldsymbol{x}^{(i)})_i$ and $^2\boldsymbol{X} = (^2\boldsymbol{x}^{(i)})_i$):

$$\min_{\theta, \boldsymbol{U}, ^1\boldsymbol{X}, ^2\boldsymbol{X}} \frac{1}{N} \sum_{i=1}^N R(\beta^{(i)}; ^1\boldsymbol{x}^{(i)}, \boldsymbol{u}^{(i)}) \quad (14a)$$

$$\text{s.t. } \forall t, \forall i, \; u_t^{(i)} = \pi_\theta(^2x_t^{(i)}) \quad (14b)$$

$$^2\boldsymbol{X} = ^1\boldsymbol{X} \quad (14c)$$

The corresponding Augmented Lagrangian is:

$$\mathcal{L}_A^\mu(\theta, ^1\boldsymbol{X}, ^2\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{\Lambda}, \boldsymbol{\Gamma}) = \frac{1}{N} \sum_{i=1}^N \left[R(\beta^{(i)}; ^1\boldsymbol{x}^{(i)}, \boldsymbol{u}^{(i)}) \right.$$

$$+ \sum_{t=0}^{T-1} \frac{\mu_t^{(i)}}{2} \left\| u_t^{(i)} + \frac{\lambda_t^{(i)}}{\mu_t^{(i)}} - \pi_\theta(^2x_t^{(i)}) \right\|^2$$

$$+ \frac{\mu_t^{(i)}}{2} \left\| ^1x_t^{(i)} + \frac{\gamma_t^{(i)}}{\mu_t^{(i)}} - ^2x_t^{(i)} \right\|^2$$

$$\left. - \frac{1}{2\mu_t^{(i)}} \left\| \lambda_t^{(i)} \right\|^2 - \frac{1}{2\mu_t^{(i)}} \left\| \gamma_t^{(i)} \right\|^2 \right] \quad (15)$$

and the update rule for γ is:

$$\gamma_t^{(i)} \leftarrow \gamma_t^{(i)} + \mu_t^{(i)}(^1x_t^{(i)} - ^2x_t^{(i)}) \quad (16)$$

By decoupling the state variables between the control and learning problems, we avoid the need of computing the Jacobian of π_θ when doing optimal control, lifting the burden to a simple dual update (16).

V. EXPERIMENTS

Our implementation uses Pinocchio [27] and Crocoddyl [4] for defining and solving OCPs, and relies on PyTorch [28] for learning the neural network. We also propose our own implementations of PODS [21] and DPL [18] as they were not provided along with the original papers. For RL algorithms, *i.e.* PPO [9] and SAC [10], we use implementations provided by [29]. Our code is open-source and will be publicly released upon publication acceptance. Every experiment was run on a single laptop without GPUs and took about 5 minutes to train policies, even on complex systems such as UR5.

The first problem we consider is a constrained LQR where the control inputs are forced to stay into a box. This is a typical convex optimization problem already very well-studied in the control community. This control problem already highlights different characteristics and limitations of approaches under consideration. The second set of experiments are related to robotics systems of increasing complexity: an inverted simple pendulum, a double pendulum and an UR5 robotic arm. These problems are challenging as their dynamics are highly non-linear and the considered targets are unstable. In addition, we consider non-linear cost functions of the form:

$$\ell(x_t, u_t) = \frac{1}{2} \| p(x_t) - \overline{p} \|_{W_p}^2 + \frac{1}{2} \| u_t \|_{W_u}^2 + \frac{1}{2} \| x_t \|_{W_x}^2$$

where p corresponds to the position of the end-effector which should reach a desired position $\overline{p}$ and W_x, W_u, W_p are given weight matrices. On robot systems, we only penalize the joint velocities in the state penalty term $\|x_t\|_{W_x}^2$.

A. Ablation study: constrained LQR and inverted pendulum

Model-based vs model-free. Fig. 3 compares model-based algorithms, *i.e.* Algs. 1,2, DPL and PODS, to the well-established deep RL algorithms PPO and SAC, on a constrained LQR problem. It appears that model-based ap-
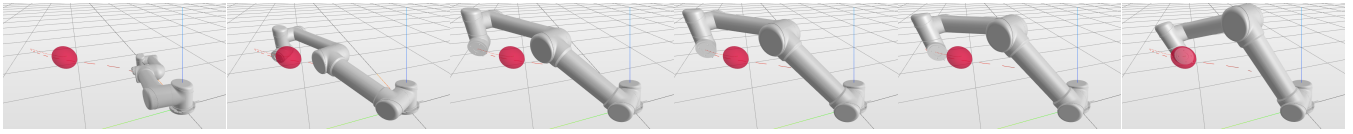
Fig. 1. **Learned policy on UR5.** After training, a rollout of the policy leads to precise control on a test problem.
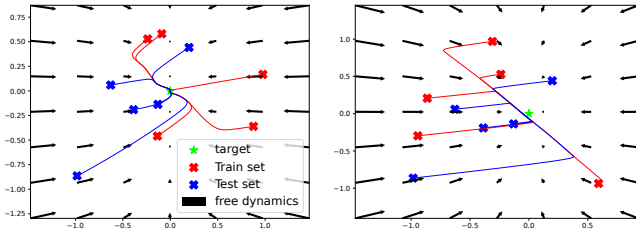

Fig. 2. **Precision of the policy.** Model-based Alg. 1 (**left**) leads to policies more precise than RL algorithms such as PPO (**right**).
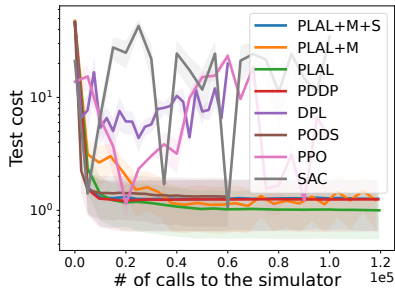

Fig. 3. **Sample efficiency: constrained LQR.** Model-based methods outperform model-free RL on tasks with linear dynamics and a stable target. We also evaluate the impact Sobolev training (+S) and multiple shooting (+M) or removing the constraint handling (-C).
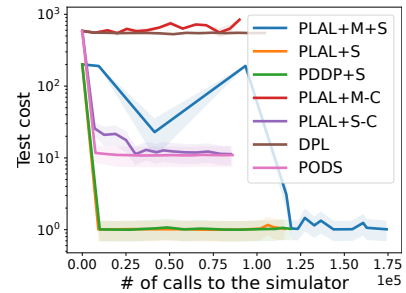

Fig. 4. **Sample efficiency: inverted pendulum.** Using a constrained formulation (5) along with supervision of the derivatives appears essential to reach a precise solution on a non-linear dynamics with unstable target.
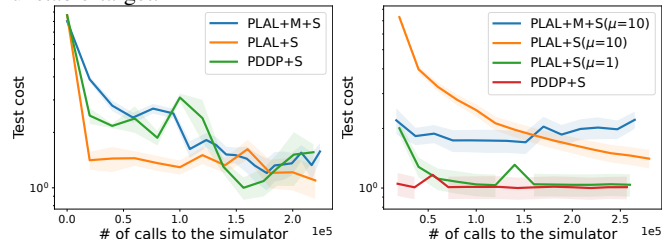

Fig. 5. **Samples efficiency: robotics tasks.** Due to chaotic dynamics, the double pendulum (**left**) and UR5 (**right**) require both handling constraints and the Sobolev training to converge. When it is so, it converges quickly towards a precise solution. Even though policy parameters are identically initialized across the different algorithms, on UR5 the first test evaluations are only performed after a few training steps to avoid numerical divergence during rollouts.

proaches are an order of magnitude more efficient than model-free algorithms. Fig. 2 also demonstrates how our approach is able to reach precise solutions while RL algorithms are limited by the inherent noise from 0th-order gradients estimates. Among already existing model-based algorithms, PODS seems to dominate DPL both in terms of precision and efficiency. Mainly, DPL suffers from the approximated formulation which shifts the initial problem and hinders progress towards the solution even on the constrained LQR.

Constrained vs unconstrained OC. If including the trajectory constraints (1b) in the trajectory optimization phase allows to slightly improve results on instances of the constrained LQR (Fig. 3), the results appear even more pronounced on the inverted pendulum problem (Fig. 4). In particular, we observe a significant performance drop when removing this component (see PDDP v. PODS or PLAL+S v. PLAL+S-C on Fig. 4). Intuitively, as the policy tries to mimic an unreachable control from the OC phase, it deteriorates the final solution, resulting in a significant gap in performance.

Multiple shooting. As mentioned in Sec. IV, multiple shooting allows to reduce the computational burden of the TO phase without modifying the final solution (Fig. 3). In practice, we observe that the OC phase of PLAL+M is 5 times faster than the one of PLAL. However, the results from Fig. 3,4 also exhibit how multiple shooting can require more simulator calls to learn a good policy (see PLAL+M v. PLAL on Fig. 3, and PLAL+M+S v. PLAL+S on Fig. 4).

Sobolev training allows to obtain gains in terms of sample efficiency by exploiting higher-order information made available by the trajectory optimization solver (see PLAL+M v. PLAL+M+S on Fig. 3). In addition, it also yields better generalization properties which significantly improve the stability of the policy rollouts and facilitate the training (see PLAL+M+S v. PLAL+M on Fig. 4). The latter characteristic becomes even more important for tasks on more complex systems with chaotic dynamics (Sec. V-B).

B. Robotics systems

Here, we consider tasks on the double pendulum and UR5 robotic arm. These tasks are made very challenging by the highly non-linear and chaotic dynamics and the instability of the desired target configuration.

As already noticed during experiments on the inverted pendulum, taking into account the physical constraints and exploiting the higher-order information contained in the feedback gains is crucial. In the case of the double pendulum and UR5, the combination of these components even appears to be necessary to get converging algorithms (Fig. 5). Due to the chaotic dynamics, for the PODS algorithm, even slight drifts from the policy w.r.t the TO controller induce very large deviations which, in turn, drive the policy towards saturation
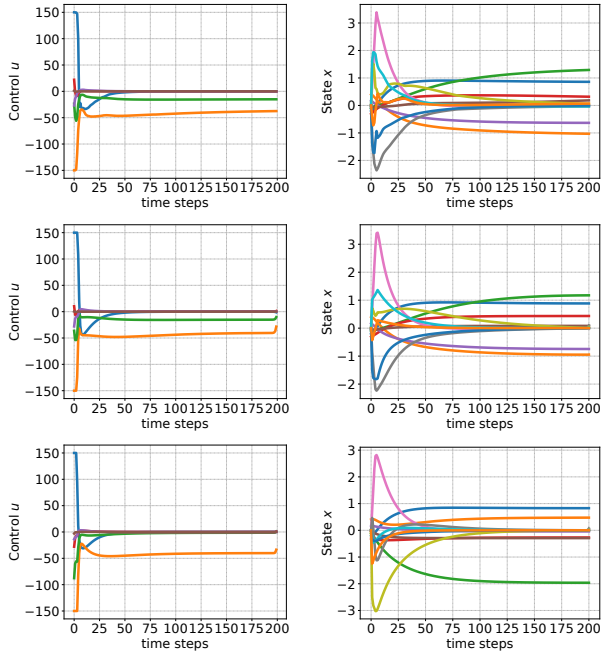
Fig. 6. **Comparisons of controllers from TO and policy learning.** We represent the trajectories (**right**) and the corresponding control (**left**) on a testing sample of UR5. The 1st row corresponds to the learned policy while the 2nd and 3rd represent the trajectories obtained via TO when the solver is initialized respectively with and without the learned controller.

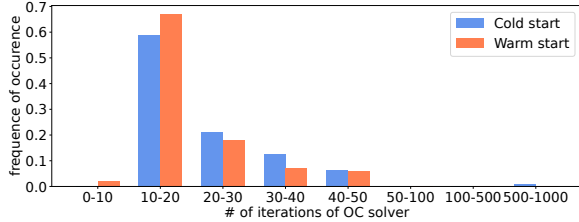Fig. 7. **Benefits of policy learning on TO.** We record the number of iterations required by the TO solver to reach a given precision on OCPs from the testing set. We compare the cases where the solver is initialized with and without a rollout of the learned policy.

and cause the rollouts to diverge. Including the physical constraints in the OC phase and regularizing the learning phase with the feedback gains allow to reduce the gap, and thus stabilize the training process.

Fig. 5 (right) highlights the influence of the augmented Lagrangian parameter μ on the convergence of ADMM algorithms. As described in V-A, multiple shooting already hinders the stability of the learning process, and when using it, we suspect the setting of μ to be even more crucial for the quality of the final solution (Fig. 5, right).

If we already demonstrated how policy learning can take advantage of TO, reciprocally, we observe that TO can also benefit from being warm-started by the learned policy (Fig. 7). In our case, doing so allows to systematically avoid worst case scenarii where the solver needs more than 100 steps.

Eventually, Fig. 6 demonstrates that warm-starting the solver with the policy also impacts the final control as it changes the local minima found by TO algorithms.

VI. DISCUSSION, CONCLUSION AND FUTURE WORK

In this paper, we have introduced a general framework to leverage the interplay between policy learning and trajectory optimization. Our formulation results in two algorithmic variants which both exploit two constrained optimization techniques: a projected DDP method, and the alternating direction method of multipliers for establishing a strong consensus between the learned control policy and TO. Both algorithms proceed in two alternating steps: solving an OCP and solving a supervised learning problem. We have introduced enhancements for either part: constraints handling, stochastic Sobolev regularization and multiple shooting. Although these algorithms achieve faster convergence towards more precise solutions when compared to classical RL algorithms, the experiments also highlight the need for advanced numerical trajectory optimization solvers. Indeed, for challenging robotic tasks, constraint handling and higher order information appear to be necessary to stabilize the training of the policy. The experiments and the ablation study also reveal the interest in our proposed enhancements: stochastic Sobolev learning enhanced sample efficiency and the stability of learned policy rollouts, and multiple shooting eliminates drift in the optimized trajectories.

A possible enhancement is the convergence speed of the method towards a consensus between the OCP and the supervised learning loop. It is known that Augmented Lagrangian methods (which includes ADMM) can reach much higher performance by having a strategy for updating the penalty parameter (μ_t). One such strategy in the literature is that of the bound-constrained Lagrangian (BCL) method of [30], which could be a direction for further enhancements.

In this work, we assumed the samples $(\beta^{(i)})_i$ used for policy optimization are fixed; they are never re-sampled (which is how stochastic algorithms such as SGD work) nor are new samples ever added. This is the domain of constrained stochastic optimization; one of the difficulties here is updating the Lagrange multipliers in a way where optimization progress is not lost at every sampling. Such an extension could allow active sampling of the initial condition x^0 or other parameters so to explore the state space and learn policies more effectively.

We could then explore extensions towards setups closer to GPS [17] by considering partially unknown system dynamics, with limited knowledge of some physical parameters (*e.g.* friction coefficients) – in order to take advantage of physical models once again. This would require embedding a system identification step in the loop which would also benefit from differentiable simulation techniques.

REFERENCES

[1] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems.," in *ICINCO (1)*, pp. 222–229, 2004.

[2] D. Mayne, "A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems," *International Journal of Control*, vol. 3, pp. 85–95, Jan. 1966.

[3] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems.*, pp. 4906–4913, 2012.

[4] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2536–2542, IEEE, 2020.

[5] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[7] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[8] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[10] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

[11] H. Mania, A. Guy, and B. Recht, "Simple random search of static linear policies is competitive for reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[12] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and systems (RSS 2018)*, 2018.

[13] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," *Advances in neural information processing systems*, vol. 31, 2018.

[14] Q. Le Lidec, I. Kalevatykh, I. Laptev, C. Schmid, and J. Carpentier, "Differentiable simulation for physical system identification," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3413–3420, 2021.

[15] W. Jallet, N. Mansard, and J. Carpentier, "Implicit Differential Dynamic Programming," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE Robotics and Automation Society, 2022.

[16] S. Levine and V. Koltun, "Guided policy search," in *International conference on machine learning*, pp. 1–9, PMLR, 2013.

[17] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[18] I. Mordatch and E. Todorov, "Combining the benefits of function approximation and trajectory optimization.," in *Robotics: Science and Systems*, vol. 4, 2014.

[19] P. Y. Simard, Y. A. LeCun, J. S. Denker, and B. Victorri, "Transformation invariance in pattern recognition—tangent distance and tangent propagation," in *Neural networks: tricks of the trade*, pp. 239–274, Springer, 1998.

[20] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu, "Sobolev training for neural networks," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[21] M. A. Z. Mora, M. P. Peychev, S. Ha, M. Vechev, and S. Coros, "Pods: Policy optimization via differentiable simulation," in *International Conference on Machine Learning*, pp. 7805–7817, PMLR, 2021.

[22] S. Levine and V. Koltun, "Learning complex neural network policies with trajectory optimization," in *International Conference on Machine Learning*, pp. 829–837, PMLR, 2014.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] E. Dantec, M. Taïx, and N. Mansard, "First Order Approximation of Model Predictive Control Solutions for High Frequency Feedback," *IEEE Robotics and Automation Letters*, vol. 7, pp. 4448–4455, Apr. 2022.

[25] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[26] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[27] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *2019 IEEE/SICE International Symposium on System Integration (SII)*, pp. 614–619, IEEE, 2019.

[28] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," *NIPS 2017 Autodiff Workshop*, 2017.

[29] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, 2021.

[30] A. Conn, N. Gould, and P. Toint, "A Globally Convergent Augmented Lagrangian Algorithm for Optimization with General Constraints and Simple Bounds," *SIAM Journal on Numerical Analysis*, vol. 28, Apr. 1991.