Direct Policy Optimization using Deterministic Sampling and Collocation

Taylor A. Howell[1], Chunjiang Fu[2], and Zachary Manchester[3]

***Abstract*—We present an approach for approximately solving discrete-time stochastic optimal-control problems by combining direct trajectory optimization, deterministic sampling, and policy optimization. Our feedback motion-planning algorithm uses a quasi-Newton method to simultaneously optimize a reference trajectory, a set of deterministically chosen sample trajectories, and a parameterized policy. We demonstrate that this approach exactly recovers LQR policies in the case of linear dynamics, quadratic objective, and Gaussian disturbances. We also demonstrate the algorithm on several nonlinear, underactuated robotic systems to highlight its performance and ability to handle control limits, safely avoid obstacles, and generate robust plans in the presence of unmodeled dynamics.**

***Index Terms*—Motion and Path Planning, Robust/Adaptive Control, Optimization and Optimal Control**

I. Introduction

TRAJECTORY optimization (TO) is a powerful tool for solving deterministic optimal-control problems in which accurate models of the system and its environment are available. However, when disturbances or unmodeled dynamics are significant, a stochastic optimal-control approach, in which a feedback policy is optimized directly, can often produce more robust performance [3].

Unfortunately, general solution methods for solving stochastic optimal-control problems suffer from the curse of dimensionality and are only applicable to low-dimensional systems in practice. To scale to many interesting robotic systems, approximations must be made. Typically, these include simplifying or linearizing dynamics, approximating value functions or policies with polynomials or neural networks, or approximating distributions with Gaussians or Monte Carlo sampling.

We present Direct Policy Optimization (DPO), a computationally tractable algorithm for finding approximate solutions to stochastic optimal-control problems that jointly optimizes a small number of trajectories and a policy. This algorithm combines several key ideas:

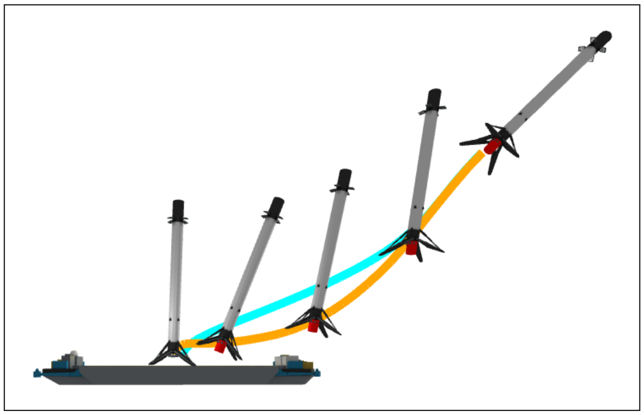

Fig. 1: Rocket soft-landing position trajectories generated by TO (blue) and DPO (orange). Unlike the TO solution tracked with LQR, the DPO policy successfully lands the rocket despite fuel slosh.

- Joint optimization of parameterized policies along with trajectories.
- Deterministic sampling of trajectories and approximation of expectations using the unscented transform.
- Direct collocation for enforcing dynamics along trajectories without performing rollouts.
- Use of large-scale constrained nonlinear programming solvers based on quasi-Newton methods for fast and robust convergence.

In contrast to many other approaches, DPO is able to easily enforce constraints like torque limits and obstacle avoidance, makes extensive use of analytical models and their derivatives, and is extremely sample efficient.

We first provide background for the discrete-time stochastic optimal-control problem, present related work on feedback motion planning, and give an overview of the unscented transform in Section II. In Section III, we present the DPO algorithm. We then demonstrate that DPO exactly recovers LQR policies when the dynamics are linear, the objective is quadratic, and disturbance inputs are Gaussian, and provide examples using DPO for several nonlinear, underactuated-control problems in Section IV. Section V offers discussion of the experimental results and, finally, we summarize our work and propose directions for future research in Section VI.

Manuscript received: October 15, 2020; Revised January 14, 2021; Accepted February 13, 2021.

This paper was recommended for publication by Editor Nancy Amato upon evaluation of the Associate Editor and Reviewers' comments.

[1]Taylor A. Howell is with the Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA thowell@stanford.edu

[2]Chunjiang Fu is with Frontier Robotics, Innovative Research Excellence, Honda R&D Co., Ltd, Wako-shi, Saitama 3510188, Japan chunjiang_fu@jp.honda

[3]Zachary Manchester is with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA zacm@cmu.edu

Digital Object Identifier (DOI): see top of this page.

II. Background

This section provides brief reviews of the discrete-time stochastic optimal-control problem and the unscented transform, as well as a survey of related solution approaches.

A. Discrete-Time Stochastic Optimal Control

We formulate the discrete-time stochastic optimal-control problem as,

$$\begin{array}{lll} \underset{\Theta}{\text{minimize}} & \mathbf{E}[J(\tau)] & \\ \text{subject to} & x_{t+1} = f_t(x_t, u_t, w_t), & t = 1, \ldots, T-1, \\ & u_t = \pi_t(x_t, \theta_t), & t = 1, \ldots, T-1, \\ & \mathbf{Prob}(c_j(\tau) > 0) \leq \epsilon_j, & j = 1, \ldots, k. \end{array} \tag{1}$$

The system's state, $x_t \in \mathbf{R}^n$, and control inputs, $u_t \in \mathbf{R}^m$, define a trajectory, $\tau = (x_1, \ldots, x_T, u_1, \ldots, u_{T-1}) \in \mathbf{R}^z$, with a subscript denoting the time index t, over a planning horizon T. The initial state, x_1, is a random variable. The discrete-time stochastic dynamics, $f_t : \mathbf{R}^n \times \mathbf{R}^m \times \mathcal{W} \to \mathbf{R}^n$, are subject to random disturbance inputs, $w_t \in \mathcal{W}$. We seek a policy, $\pi_t : \mathbf{R}^n \times \mathbf{R}^p \to \mathbf{R}^m$, parameterized by $\Theta = (\theta_1, \ldots, \theta_{T-1}) \in \mathbf{R}^{p(T-1)}$, to minimize the objective, $J : \mathbf{R}^z \to \mathbf{R}$, with the expectation taken over the initial state and disturbance inputs. Chance constraints, with $c_j : \mathbf{R}^z \to \mathbf{R}$ and probability of violation less than tolerance $\epsilon_j \in \mathbf{R}_+$, can further constrain trajectories.

B. Related Work

Model-based approaches often tackle problem (1) in a decoupled fashion: First, generate a reference trajectory assuming no disturbances; then, design a tracking feedback policy to reject disturbances. Using collocation methods [23] or differential dynamic programming (DDP) [6] to optimize a trajectory, then tracking it with a time-varying LQR controller has achieved impressive results for complex, real-world systems [7, 16].

There are two primary drawbacks to these classic synthesis techniques: First, there is no explicit consideration of uncertainty or disturbances. Robustness is often achieved heuristically via *post hoc* Monte Carlo testing and tuning. Second, by decoupling the synthesis of the reference trajectory and policy, performance is often sacrificed.

DIRTREL [10] optimizes a reference trajectory with an additional cost term that penalizes a linearized approximation of the tracking error from an LQR policy. Chance constraints are approximated by enforcing constraints at a finite number of samples. There is also a variation of DDP that can account for multiplicative noise applied to the controls [21]. Unlike these methods, DPO directly optimizes policies and can propagate uncertainty through nonlinear dynamics.

Several learning-based approaches exist that directly optimize feedback policies. Policy gradient methods [18, 26] use first-order or derivative-free methods to optimize parameterized policies, typically without direct access to the underlying dynamics model. Domain randomization [20] can be employed to vary model and environment parameters to encourage policy robustness. Random search [12], stochastic gradient descent [4], and Newton methods [27] have also been used to optimize linear feedback and MPC policies [1, 2]. A major benefit of stochastic methods is their inherent exploration of the policy space.

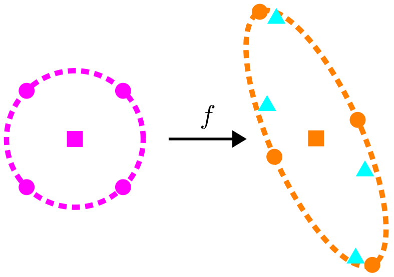

Fig. 2: Unscented transform visualized in 2D for initial (left) and transformed (right) distributions. Sigma points (circles) with sample mean (square) and sample covariance (dashed) are propagated through a nonlinear function (triangles) and then resampled to compute new sigma points.

Guided Policy Search (GPS) [9] is a hybrid approach that alternates between optimizing sample trajectories and fitting a policy. DDP is used to generate high-reward trajectories around the current policy that are subsequently employed to improve the policy. This procedure is then iterated with a regularizer to keep new trajectories close to those generated by the previous policy.

While GPS does not make strong assumptions about state and disturbance distributions or explicitly require dynamics models, it relies heavily on Monte Carlo techniques that require a large number of samples. As a result, training can require significant time and computational resources. In contrast, DPO leverages analytical models and uses only a small number of deterministically chosen samples, making it much more efficient.

Finally, we note that many of the approaches outlined in this section—including DDP and many of the policy gradient methods—rely on explicit forward simulation or "rollouts" along with some form of backpropagation. These techniques are vulnerable to numerical ill-conditioning issues colloquially known as the "tail-wagging-the-dog problem" in control and the "vanishing" or "exploding" gradient problem in reinforcement learning. In contrast, DPO employs collocation methods that simultaneously optimize state and control trajectories with dynamics enforced as constraints. Such "direct" methods enjoy far better numerical conditioning and robustness, especially with long-horizon plans.

C. Unscented Transform

The unscented transform is a procedure for propagating a unimodal probability distribution through a nonlinear function using deterministic samples, often referred to as sigma points. This tool is commonly used for state estimation, where it is generally considered to be superior to the extended Kalman

Algorithm 1 Unscented dynamics

```
function $g_t(\tau_t^{(1:N)}, D_t)$
    for $i = 1 : N$ do
        $x_t^{(i)},\ w_t^{(i)} \leftarrow$ compute sigma points (2)
        $u_t^{(i)},\ x_{t+1}^{(i)} \leftarrow$ propagate sigma points (3, 4)
    end for
    $\mu_{t+1},\ \nu_t \leftarrow$ compute sample means (5, 7)
    $P_{t+1},\ L_t \leftarrow$ compute sample covariances (6, 8)
    return $x_{t+1}^{(1:N)}$
end function
```

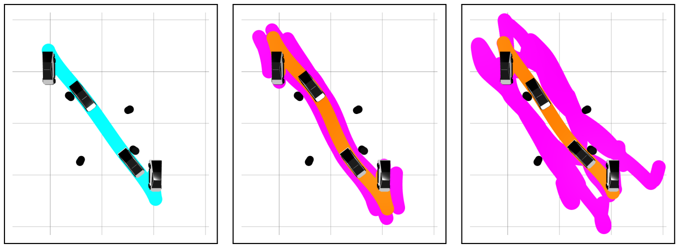

Fig. 3: Position trajectories for autonomous car avoiding obstacles. TO (left) finds a path that is near the obstacles, whereas DPO with $\delta = 0.001$ (center) finds a path that avoids obstacles with a margin for safety. With larger disturbances, $\delta = 0.01$, samples (magenta) diverge and the DPO policy fails to safely avoid obstacles (right).

filter's linear propagation of covariance matrices [22]. There are many variations of the unscented transform [15]; the version we utilize is visualized in Fig. 2 and outlined below.

Assuming a unimodal state distribution with mean $\mu_t \in \mathbf{R}^n$, covariance $P_t \in \mathbf{S}_{++}^n$, and an uncorrelated zero-mean disturbance distribution with covariance $D_t \in \mathbf{S}_{++}^d$, we generate $N = 2(n+d)$ sigma points,

$$\begin{bmatrix} x_t^{(i)} \\ w_t^{(i)} \end{bmatrix} \leftarrow \begin{bmatrix} \mu_t \\ 0 \end{bmatrix} \pm \beta_t \,\mathbf{col}\left(\sqrt{\begin{bmatrix} P_t & 0 \\ 0 & D_t \end{bmatrix}}\right). \tag{2}$$

These samples, denoted with superscripts, are constructed using the column vectors of a square root of the joint covariance matrix. In this work we use the principle (symmetric) matrix square root, although other decompositions, such as the Cholesky factorization, could be employed. The parameter $\beta_t \in \mathbf{R}_+$ controls the spread of the samples around the mean. For a Gaussian distribution propagated through linear dynamics, the unscented transform will exactly recover the updated distribution. For nonlinear systems, the selection of β can affect performance, and has been explored extensively in the literature on unscented Kalman filters [15].

Sample points are first propagated through the policy,

$$u_t^{(i)} = \pi_t(x_t^{(i)}), \tag{3}$$

and then sample dynamics,

$$x_{t+1}^{(i)} = f_t^{(i)}(x_t^{(i)}, u_t^{(i)}, w_t^{(i)}), \tag{4}$$

in order to compute a sample mean,

$$\mu_{t+1} = \frac{1}{N}\sum_{i=1}^{N} x_{t+1}^{(i)}, \tag{5}$$

and sample covariance,

$$P_{t+1} = \frac{1}{2\beta_t^2}\sum_{i=1}^{N}(x_{t+1}^{(i)} - \mu_{t+1})(x_{t+1}^{(i)} - \mu_{t+1})^T, \tag{6}$$

for the state distribution at the next time step. Similarly, we compute a sample mean,

$$\nu_t = \frac{1}{N}\sum_{i=1}^{N} u_t^{(i)}, \tag{7}$$

and sample covariance,

$$L_t = \frac{1}{2\beta_t^2}\sum_{i=1}^{N}(u_t^{(i)} - \nu_t)(u_t^{(i)} - \nu_t)^T, \tag{8}$$

for the control inputs at the current time step.

III. Direct Policy Optimization

We now present the Direct Policy Optimization algorithm. DPO makes several strategic approximations to the discrete-time stochastic optimal-control problem: First, the expectation of the objective in (1) is approximated using the unscented transform. Second, DPO explicitly optimizes a reference trajectory, $\bar{\tau}$, and N sample trajectories, $\tau^{(i)}$, in order to approximate the stochastic dynamics. We use an overbar to denote reference or nominal quantities and superscripts to denote sample indices throughout the paper. Next, we seek a local feedback policy that is valid in the neighborhood of the reference trajectory. Finally, chance constraints are approximately enforced by applying inequality constraints to a set of sample points chosen from level sets of the state and input distributions. Using these approximations, we formulate a nonlinear program (NLP) that is amenable to optimization with large-scale quasi-Newton solvers.

A. *Objective*

DPO minimizes the following objective,

$$J(\bar{\tau}) + \mathbf{E}[S(\bar{\tau}, \tau)], \tag{9}$$

with cost function J applied to the reference trajectory and a quadratic tracking-cost function,

$$\begin{aligned} S(\bar{\tau}, \tau) = {} & (x_T - \bar{x}_T)^T Q_T (x_T - \bar{x}_T) \\ & + \sum_{t=1}^{T-1}\{(x_t - \bar{x}_t)^T Q_t (x_t - \bar{x}_t) \\ & + (u_t - \bar{u}_t)^T R_t (u_t - \bar{u}_t)\}, \end{aligned} \tag{10}$$

$S : \mathbf{R}^z \times \mathbf{R}^z \to \mathbf{R}$, with $Q_t \in \mathbf{S}_+^n$ and $R_t \in \mathbf{S}_+^m$, that penalizes deviations from the reference trajectory under disturbances.

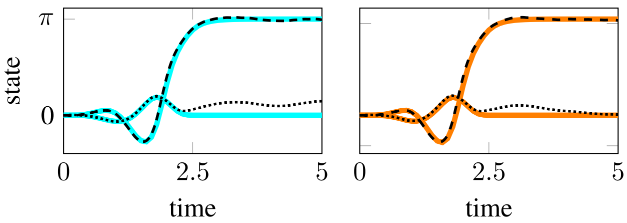

Fig. 4: Simulated tracking (black) for position (dotted) and orientation (dashed) of cart-pole experiencing Coulomb friction. Comparison between LQR (blue) and DPO (orange) policies.

After simple manipulations, we can write (9) in terms of the quantities introduced in Sec. II-C,

$$\begin{aligned}\mathbf{E}[(x_t - \bar{x}_t)^T Q_t (x_t - \bar{x}_t)] \\ = \mathbf{Tr}(P_t Q_t) + (\mu_t - \bar{x}_t)^T Q_t (\mu_t - \bar{x}_t),\end{aligned} \quad (11)$$

$$\begin{aligned}\mathbf{E}[(u_t - \bar{u}_t)^T R_t (u_t - \bar{u}_t)] \\ = \mathbf{Tr}(L_t R_t) + (\nu_t - \bar{u}_t)^T R_t (\nu_t - \bar{u}_t).\end{aligned} \quad (12)$$

Since the expectation in (9) depends only on the first and second moments of the state and control distributions, we can efficiently compute it using the unscented transform. As an aside, the second terms in (11) and (12) are zero in the linear-quadratic Gaussian (LQG) case, and their presence reflects the invalidity of the separation principle in more general settings.

B. Dynamics

DPO utilizes direct collocation and enforces dynamics for each trajectory via equality constraints at each time step. A unimodal distribution over the state and policy is maintained along the planning horizon by resampling the sample trajectories at each time step using the unscented transform (2-8); this procedure is summarized in Algorithm 1. For motion-planning applications, maintaining a unimodal distribution over a planning horizon is a reasonable modeling choice because the policy explicitly works to keep sample trajectories near the reference. In addition to computing terms required by the expectation over the objective, resampling at each time step prevents samples from collapsing to the reference at later time steps, encouraging the policy to be robust throughout the entire trajectory.

Initial sample states and disturbance inputs are deterministically drawn from normal distributions, $x_1^{(i)} \sim \mathcal{N}(\mu_1, P_1)$ and $w_t \sim \mathcal{N}(0, D_t)$, using (2). The initial reference state is $\bar{x}_1 = \mu_1$. By utilizing the discrete-time dynamics, it is possible to generate non-Gaussian noise for the system and capture model uncertainty.

C. Feedback Policy

Each sample trajectory is subject to a policy constraint,

$$u_t^{(i)} = \pi_t(x_t^{(i)}, \bar{x}_t, \bar{u}_t, \theta_t), \quad (13)$$

at each time step. Instead of optimizing global policies, we search for local feedback policies that can depend on the reference trajectory. While these policies can be from any differentiable function class, we focus on linear policies for simplicity and leave extensions to more complex functions to future work.

TABLE I: Tracking error, computed with (10), for cart-pole experiencing Coulomb friction. Comparison between LQR and DPO policies.

cost	**LQR**	**DPO**
state	3.18	**2.26**
control	0.84	**0.39**
total	4.02	**2.64**

D. Chance Constraints

Chance constraints are approximated by constraining the reference and sample trajectories. The probability of violation, ϵ, corresponds to particular level sets of the state and control distributions. The sampling parameter β can be selected in order to sample sigma points that approximate these level sets. Constraints are only enforced at these points. While more accurate methods for evaluating chance constraints exist, they are much more computationally demanding. Instead, DPO trades computational tractability for exact guarantees.

E. Nonlinear Programming Formulation

DPO can be formulated as the following NLP:

$$\begin{array}{lll} \underset{\Theta, \bar{\tau}, \tau^{(1:N)}}{\text{minimize}} & J(\bar{\tau}) + \sum_{i=1}^{N} S(\bar{\tau}, \tau^{(i)}) & (14) \\ \text{subject to} & \bar{x}_{t+1} = \bar{f}_t(\bar{x}_t, \bar{u}_t, 0), & t = 1, \ldots, T-1, \\ & x_{t+1}^{(1:N)} = g_t(\tau_t^{(1:N)}, D_t), & t = 1, \ldots, T-1, \\ & u_t^{(1:N)} = & \\ & \quad \pi_t(x_t^{(1:N)}, \bar{x}_t, \bar{u}_t, \theta_t), & t = 1, \ldots, T-1, \\ & c_j(\bar{\tau}) \leq 0, & j = 1, \ldots, k, \\ & c_j(\tau^{(1:N)}) \leq 0, & j = 1, \ldots, k. \end{array}$$

Additional constraints, for example, conditions on the policy parameters or trajectories, can be directly added to the formulation. Off-the-shelf large-scale NLP solvers like Ipopt [24] and SNOPT [5] can be employed to efficiently optimize (14) by taking advantage of sparsity in its associated Jacobian and Hessian matrices.

IV. Examples

The following examples were implemented in Julia, used SNOPT to optimize (14), and were performed on a laptop computer with an Intel Core i7-7600U 2.80 GHz CPU and 16 GB of memory.

To verify the performance of DPO, optimized policies are simulated at ten times the sample rate used during optimization, systems are simulated with explicit third-order Runge-Kutta integration, zero-order-hold control interpolation, cubic

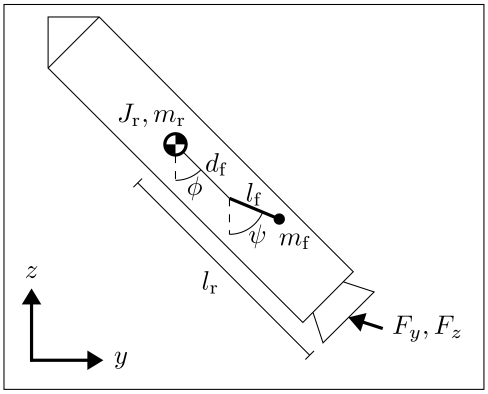

Fig. 5: Rocket with planar dynamics. The model has lateral and vertical positions y and z, orientation ϕ, and thrust inputs F_y and F_z. Parameters are inertia J_r, mass m_r, and length l_r from center of mass to thruster. A pendulum positioned d_f from rocket center of mass, with orientation ψ, length l_f, and mass m_f, models fuel slosh.

spline state interpolation, and are subject to additive zero-mean Gaussian noise. TO and DPO utilize the same objective for the reference trajectory. Throughout, the tracking cost (10) and LQR policies have the same weights and, unless specified, $\beta = 1$. The reported tracking error is computed using the tracking cost (10) for a single trajectory. During optimization we employ implicit midpoint integration and use discrete dynamics with additive noise,

$$x_{t+1}^{(i)} = f_t^{(i)}(x_t^{(i)}, u_t^{(i)}) + w_t^{(i)}. \tag{15}$$

We find that additive noise with state augmentation and different sample models is quite general and capable of capturing interesting dynamical effects. Policies use linear feedback to track the reference trajectory,

$$u_t^{(i)} = \bar{u}_t - \theta_t(x_t^{(i)} - \bar{x}_t). \tag{16}$$

Our implementation of DPO and additional details for each example are available at, http://roboticexplorationlab.org/projects/dpo.html.

A. Double Integrator

We first demonstrate that DPO recovers the LQR solution for double integrator dynamics,

$$x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_t + w_t \tag{17}$$

with $n = 2$ states, $m = 1$ controls, and $d = 2$ disturbances, regulated to the origin over a horizon $T = 51$. The policy has $p = 2$ parameters at each time step. Initial states are sampled from a distribution with $\mu_1 = 0$ and $P_1 = I$, disturbances have $D_{1:T-1} = I$, the weights are $Q_{1:T} = I$ and $R_{1:T-1} = I$.

The decision variables are initialized by uniformly sampling between -1 and 1 for 1000 trials. The maximum, mean, and standard deviations of the normalized error between the DPO policies and the exact LQR solution computed with the Riccati equation are near the tolerance of the solver: 2.4e-5, 4.0e-7, 8.5e-7, respectively—reinforcing that the approximations made in the derivation of DPO are exact in the LQG case. Additionally, the second term in (11) is zero, demonstrating that the separation principle holds for this problem.

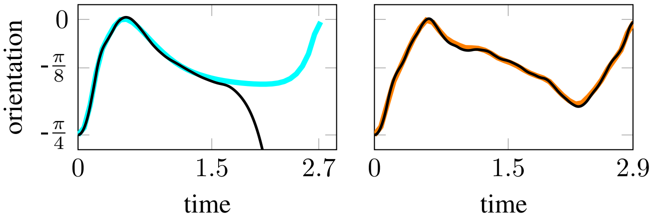

Fig. 6: Simulated tracking (black) for the orientation of a rocket experiencing fuel slosh during landing. Comparison between LQR (blue) and DPO (orange) polices.

B. Car

We plan a collision-free path through four obstacles for an autonomous car, modeled as a unicycle [8],

$$\begin{bmatrix} \dot{y} \\ \dot{z} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} v\cos(\phi) \\ v\sin(\phi) \\ \omega \end{bmatrix}. \tag{18}$$

The $n = 3$ states are positions y and z, and orientation ϕ. The $m = 2$ controls are the forward and angular velocities, v and ω. There are $d = 3$ disturbance inputs and the policy has $p = 6$ parameters at each time step. Initial states are sampled from a distribution with $\mu_1 = 0$, $P_1 = \textbf{diag}(1, 1, 0.1)$, weights are $Q_{1:T-1} = \textbf{diag}(10, 10, 1)$, $Q_T = 100I$, and $R_{1:T-1} = 0.1I$, and disturbances have $D_{1:T-1} = \textbf{diag}(\delta, \delta, 0.1\delta)$. The planning horizon is $T = 51$ with fixed time step $h = 0.02$. The choice of $\beta = 1$ over the planning horizon results in constraints being enforced on the 1-sigma level set of the state distribution, corresponding to $\epsilon = 0.32$.

For this scenario, TO finds a path that is in close proximity to the obstacles. In contrast, by considering noise, DPO with $\delta = 0.001$ finds a path that remains a safe distance from the obstacles. Decreasing the noise results in convergence to the TO solution. However, larger disturbance inputs (e.g., $\delta = 0.01$), expose a limitation of DPO; samples can diverge, resulting in a multimodal state distribution with trajectories taking different paths around the obstacles (Fig. 3). Ultimately, unlike *ad hoc* techniques like obstacle inflation, DPO considers the coupling of dynamics, constraints, and disturbances to generate safer paths for the system.

C. Cart-Pole

A swing-up trajectory for a cart-pole [19] with a slider that experiences Coulomb friction is synthesized for horizon $T = 51$ and fixed time step $h = 0.1$. The state, $x = (y, \phi, \dot{y}, \dot{\phi})$, has cart position y, pendulum orientation ϕ, and their respective time derivatives. The optimality conditions for the maximum-dissipation principle [17] are used as constraints to explicitly

TABLE II: Tracking error, computed with (10), for rocket landing with fuel slosh. Comparison between LQR and DPO policies.

cost	LQR	DPO
state	4725.39	**5.02**
control	75.21	**0.39**
total	4800.60	**5.41**

model friction [11] with coefficient $\mu_{\text{f}} = 0.1$ for each trajectory. The cart position is controlled and $m = 1$, there are $d = 4$ disturbance inputs, and the policy has $p = 4$ parameters at each time step. Initial states are sampled from $\mu_1 = 0$ and $P_1 = I$, the weights are $Q_{1:T-1} = \textbf{diag}(10, 10, 1, 1)$, $Q_T = 100I$, and $R_{1:T-1} = I$, and disturbances have $D_{1:T-1} = 0.001I$.

The performance of LQR tracking reference trajectories optimized subject to friction, as well as the DPO policy, are verified in simulation. Results are provided in Table I and tracking for the position and orientation is shown in Fig. 4. By first designing a trajectory that explicitly models friction, it is possible to subsequently synthesize an LQR policy for tracking. In contrast, DPO, which can handle nonsmooth dynamics, produces superior tracking by simultaneously optimizing the reference trajectory and policy.

D. Rocket Landing

We plan a soft landing for a rocket [13]. The nominal system with planar dynamics has state $x = (y, z, \phi, \dot{y}, \dot{z}, \dot{\phi})$, with lateral and vertical positions y and z, orientation ϕ, and their respective time derivatives. The rocket is controlled with gimballed thrust, $u = (F_y, F_z)$. It is initialized with non-zero displacements and velocities relative to its target state: a zero-velocity, vertical-orientation touchdown inside a landing zone.

The rocket experiences fuel slosh during landing that is not accounted for in the nominal model. This is a critical dynamical effect, but it is difficult to model and observe. In practice, these unmodeled effects are typically handled in *ad hoc* ways, often with extensive Monte Carlo simulation and controller tuning. To approximately model fuel slosh, we augment the nominal model with two additional states associated with an unobservable and unactuated pendulum (Fig. 5). A common frequency-domain control approach is to include a notch filter at the pendulum's natural frequency in order to prevent excitation of these fuel-slosh dynamics. We instead use DPO to synthesize a policy that is robust to fuel slosh.

The reference trajectory is optimized with the nominal model, while the sample trajectories are subject to the augmented model in order to capture fuel-slosh dynamics, and a linear output feedback policy is optimized that does not have access to the fuel-slosh states. There are $d = 8$ disturbance inputs to the augmented model and the policy has $p = 12$ parameters at each time step. Initial states for the augmented model are sampled with uncertainty $P_1 = I$, weights are $Q_{1:T-1} = 100I$, $Q_T = 1000I$, and $R_{1:T-1} = \textbf{diag}(1, 1, 100)$, and disturbances have $D_{1:T-1} = 0.001I$. Over the planning horizon $T = 41$ with free final time, thrust limits are enforced.

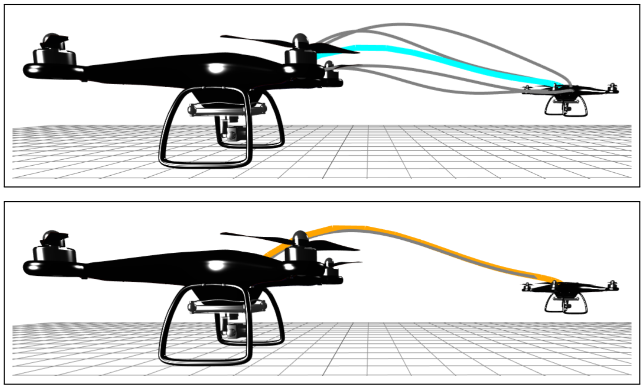

Fig. 7: Simulated position trajectories (gray) for quadrotor controlled with LQR (top) and DPO (bottom) policies while experiencing different propellers breaking. LQR (blue) has degrading tracking while DPO (orange) has superior, and nearly identical tracking regardless of which propeller breaks.

TO finds a 2.72 second solution, whereas DPO finds a longer 2.91 second path to the landing zone. The position trajectories are shown in Fig. 1, simulation results with fuel slosh are provided in Table II, and tracking results for the rocket's orientation are shown in Fig. 6. Due to fuel slosh, LQR fails to track the path generated by TO and the rocket tips over, whereas the DPO policy successfully lands the rocket. DPO is able to optimize a policy for a system with unobservable dynamical effects by sampling augmented models.

E. Quadrotor

A point-to-point maneuver is planned for a quadrotor [14] that experiences a random blade breaking off from a propeller during flight. A broken propeller is modeled by constraining the corresponding control input to be half its nominal maximum value. We use DPO to optimize a policy that is robust to this event occurring for any of the propellers. The reference model has no broken propellers, but the sample trajectories are optimized with different models from four groups, each experiencing a different broken propeller. There are $n = 12$ states, $m = 4$ controls, $d = 12$ input disturbances, and the policy has $p = 48$ parameters at each time step. Initial states are sampled around the initial nominal state with uncertainty $P_1 = I$, weights are $Q_{1:T-1} = 10I$, $Q_T = 100I$, and $R_{1:T-1} = I$, and disturbances have $D_{1:T-1} = 0.001I$. Thrust limits are enforced over the planning horizon $T = 31$ with a free final time.

TO finds a 2.71 second trajectory, while DPO finds a longer 3.38 second trajectory with lower maximum controls (Fig. 8). The policies are compared in simulation over 100 initial conditions with noise sampled from $\mathcal{N}(0, 0.001I)$. One set of initial conditions is visualized in Fig. 7. The average tracking performance over each propeller breaking is provided in Table III. DPO finds a policy with consistent and superior tracking compared with LQR for all cases by optimizing over different models.

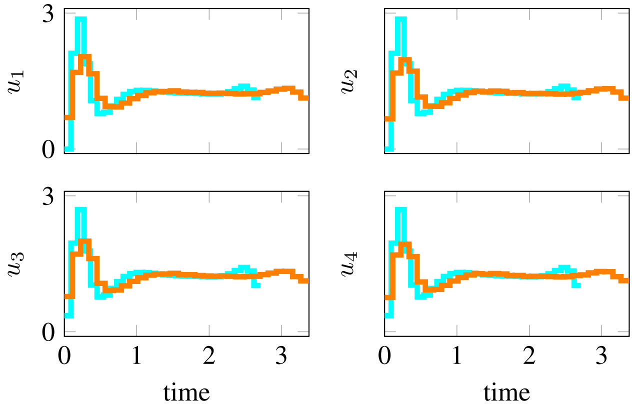

Fig. 8: Nominal controls for quadrotor performing point-to-point maneuver generated with TO (blue) and DPO (orange). The controls found with DPO are applied over a longer horizon and have lower maximum values compared to TO.

TABLE III: Tracking errors with mean (μ) and standard deviation (σ), computed with (10), comparing LQR and DPO policies for quadrotor over 100 initial conditions for each propeller being broken.

cost (μ, σ)	**LQR**	**DPO**
state	(13.43, 12.64)	**(4.74, 5.87)**
control	(0.074, 0.067)	**(0.020, 0.022)**
total	(13.50, 12.71)	**(4.76, 5.89)**

(a) propeller 1 broken

cost (μ, σ)	**LQR**	**DPO**
state	(13.13, 11.98)	**(4.74, 5.87)**
control	(0.073, 0.065)	**(0.020, 0.023)**
total	(13.21, 12.05)	**(4.76, 5.89)**

(b) propeller 2 broken

cost (μ, σ)	**LQR**	**DPO**
state	(3431.91, 12895.93)	**(4.74, 5.88)**
control	(2.21, 7.86)	**(0.020, 0.023)**
total	(3434.12, 12903.75)	**(4.76, 5.90)**

(c) propeller 3 broken

cost (μ, σ)	**LQR**	**DPO**
state	(4843.89, 14892.19)	**(4.74, 5.87)**
control	(3.21, 9.41)	**(0.020, 0.023)**
total	(4847.10, 14901.59)	**(4.76, 5.89)**

(d) propeller 4 broken

V. Discussion

We empirically demonstrate through Monte Carlo simulations that DPO recovers LQR policies for a double integrator system. It seems possible that for LQG problems, despite the non-convexity of the policy constraints, there exists a unique global solution for DPO. However, we leave a formal convergence analysis to future work.

The motion-planning examples highlight a number of DPO's capabilities compared to DIRTREL and GPS. In the cart-pole example, DPO is able to subject each trajectory to discontinuous Coulomb friction calculated with the maximum-dissipation principle, whereas a similar DIRTREL example only applied white-noise disturbances to a single model [10]. This example also highlights how jointly optimizing trajectories and a policy leads to superior performance, a distinction compared to the decoupled approach employed by GPS. Next, in the rocket landing example, DPO optimizes an output feedback policy, whereas DIRTREL is limited to state feedback. Additionally, directly modeling fuel-slosh dynamics with a pendulum is likely simpler than designing input disturbances for DIRTREL to capture the same effects. Finally, with DPO, additional constraints can be applied to any of the decision variables, whereas the ability of GPS to handle constraints using DDP and stochastic gradient descent is limited.

Because we use local optimization methods to solve DPO, providing good initial guesses to the solver is crucial for performance of the algorithm. In practice, we use a standard TO solution to warm start the reference and sample trajectories for DPO. For linear state-feedback policies, we use the corresponding LQR solution as an initial guess. While not necessary in any of our examples, initial guesses for more complex policies could be found by running DPO without the policy constraint, then performing an offline regression to fit approximate policy parameters for warm starting.

For a TO problem solved with a sparsity-exploiting second-order method, the computational complexity is approximately $O\Big(T(n+m)^3\Big)$ [25]. For DPO, we can consider an augmented state and control with dimensions $n(2n+2)$ and $m(2n+2)$, respectively. The complexity of DPO is, therefore, $O\Big(T(n^6+m^3n^3)\Big)$. In the examples, which do not employ a specialized solver, solution times range from seconds to 1.5 hours for a state with $n = 12$ on a laptop computer. It is likely possible to exploit the DPO problem's structure to improve the complexity by using custom linear solvers. First-order methods, which can scale much better, become attractive for DPO as problems become large.

Lastly, LQR is a powerful tool and can likely be tuned to qualitatively match the performance of the linear policies found with DPO in many cases. However, the strength of DPO lies in its ability to explicitly reason about robustness and the complex coupling between dynamics, constraints, and disturbances during synthesis, instead of relying on hand tuning and heuristics.

VI. Conclusion

We have presented a new algorithm, Direct Policy Optimization, for approximately solving stochastic optimal-control problems. We demonstrate that the algorithm is exact in the LQG case and is capable of optimizing policies for nonlinear systems that respect control limits and obstacles, and are robust to unmodeled dynamics.

Many interesting avenues for future work remain: Extensions to nonlinear policies could be made by introducing high-dimensional features and regularization or constraints on policy parameters, or neural networks could be used to parameterize policies since the policy parameters scale much better compared to state and control dimensions. Another potential direction is to optimize a local value function approximation in place of an explicit policy. A much richer class of disturbances and model parameter uncertainty could also be modeled by augmenting the state vector and dynamics. Lastly, more complex systems with larger state and input dimensions may be more amenable to optimization with first-order or matrix-free methods.

References

[1] Akshay Agrawal et al. "Learning convex optimization control policies". In: *Learning for Dynamics and Control*. 2020, pp. 361–373.

[2] Brandon Amos et al. "Differentiable MPC for end-to-end planning and control". In: *Advances in Neural Information Processing Systems*. 2018, pp. 8289–8300.

[3] Dimitri P. Bertsekas and Steven Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Athena Scientific, 2004.

[4] Jingjing Bu et al. "LQR through the lens of first order methods: Discrete-time case". In: *arXiv preprint arXiv:1907.08921* (2019).

[5] Philip E. Gill, Walter Murray, and Michael A. Saunders. "SNOPT: An SQP algorithm for large-scale constrained optimization". In: *SIAM Review* 47.1 (2005), pp. 99–131.

[6] David H. Jacobson and David Q. Mayne. *Differential Dynamic Programming*. North-Holland, 1970.

[7] Scott Kuindersma et al. "Optimization-based locomotion planning, estimation, and control design for the Atlas humanoid robot". In: *Autonomous Robots* 40.3 (2016), pp. 429–455.

[8] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[9] Sergey Levine and Vladlen Koltun. "Guided policy search". In: *International Conference on Machine Learning*. 2013, pp. 1–9.

[10] Zachary Manchester and Scott Kuindersma. "Robust direct trajectory optimization using approximate invariant funnels". In: *Autonomous Robots* 43.2 (2019), pp. 375–387.

[11] Zachary Manchester and Scott Kuindersma. "Variational contact-implicit trajectory optimization". In: *Robotics Research*. Springer, 2020, pp. 985–1000.

[12] Horia Mania, Aurelia Guy, and Benjamin Recht. "Simple random search of static linear policies is competitive for reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2018, pp. 1800–1809.

[13] James S. Meditch. "On the problem of optimal thrust programming for a lunar soft landing". In: *IEEE Transactions on Automatic Control* 9.4 (1964), pp. 477–484.

[14] Daniel Mellinger, Nathan Michael, and Vijay Kumar. "Trajectory generation and control for precise aggressive maneuvers with quadrotors". In: *The International Journal of Robotics Research* 31.5 (2012), pp. 664–674.

[15] Henrique M. T. Menegaz et al. "A systematization of the unscented Kalman filter theory". In: *IEEE Transactions on Automatic Control* 60.10 (2015), pp. 2583–2598.

[16] Joseph Lawrence Moore. "Robust post-stall perching with a fixed-wing UAV". PhD thesis. Massachusetts Institute of Technology, 2014.

[17] Jean Jacques Moreau. "On unilateral constraints, friction and plasticity". In: *New Variational Techniques in Mathematical Physics*. Springer, 2011, pp. 171–322.

[18] David Silver et al. "Deterministic policy gradient algorithms". In: *International Conference on Machine Learning*. 2014.

[19] Russ Tedrake. "Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for MIT 6.832)". In: *Downloaded in Fall* (2020).

[20] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2017, pp. 23–30.

[21] Emanuel Todorov and Weiwei Li. "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems". In: *Proceedings of the 2005, American Control Conference, 2005.* IEEE. 2005, pp. 300–306.

[22] Jeffrey K. Uhlmann. "Dynamic map building and localization: New theoretical foundations". PhD thesis. University of Oxford, 1995.

[23] Oskar Von Stryk. "Numerical solution of optimal control problems by direct collocation". In: *Optimal Control*. Springer, 1993, pp. 129–143.

[24] Andreas Wächter and Lorenz T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106.1 (2006), pp. 25–57.

[25] Yang Wang and Stephen Boyd. "Fast model predictive control using online optimization". In: *IFAC Proceedings Volumes* 41.2 (2008), pp. 6974–6979.

[26] Ronald J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine Learning* 8.3-4 (1992), pp. 229–256.

[27] Matt Wytock and J. Zico Kolter. "A fast algorithm for sparse controller design". In: *arXiv preprint arXiv:1312.4892* (2013).