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 nominal 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 cost, and Gaussian noise. 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.**

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 noise, 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 sample trajectories and a stabilizing feedback policy. This new algorithm combines several key ideas:

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

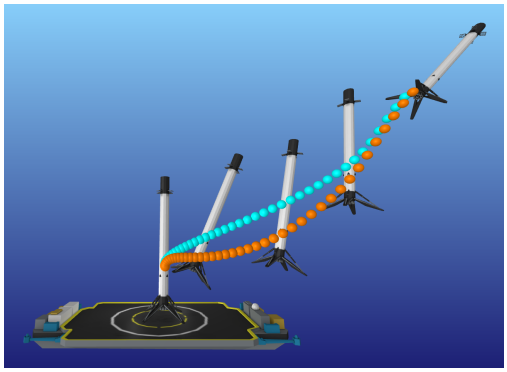

Fig. 1: Rocket landing trajectories generated by TO (blue) and DPO (orange). Unlike the TO solution tracked with TVLQR, the DPO policy lands the rocket despite unmodeled fuel slosh.

In contrast to many reinforcement learning techniques, 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.

The paper proceeds as follows: We 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, costs are quadratic, and noise is Gaussian, and provide examples

[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, Pittsburg, PA 15213, USA zacm@cmu.edu

using DPO for several nonlinear, underactuated control problems in Section IV. Section V offers some discussion of the experimental results and, finally, we summarize our work and propose directions for future work in Section VI.

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,

$$\begin{array}{ll} \underset{\theta}{\text{minimize}} & \mathbf{E}_{w_t}\left[\ell_T(x_T) + \sum_{t=1}^{T-1} \ell_t(x_t, u_t)\right] \\ \text{subject to} & x_{t+1} = f_t(x_t, u_t, w_t), \\ & (x_1 \text{ given}), \\ & u_t = \pi_t(x_t, \theta_t), \\ & \text{prob}(c(\tau) > 0) \leq \epsilon, \end{array} \tag{1}$$

where we seek a policy, π_t, parameterized by $\theta_t \in \mathbf{R}^p$, t is the time index, and T is the planning horizon. The system's state, $x_t \in \mathbf{R}^n$, and control inputs, $u_t \in \mathbf{R}^m$, define a trajectory, $\tau = (u_1, \ldots, u_{T-1}, x_1, \ldots, x_T) \in \mathbf{R}^z$. The objective includes terminal and stage costs ℓ_T and ℓ_t, respectively. The discrete-time stochastic dynamics, f_t, are subject to random disturbance inputs, $w_t \in \mathbf{R}^d$, drawn from some distribution. Inequality constraints, $c : \mathbf{R}^z \to \mathbf{R}^r$, are encoded as element-wise chance constraints on the trajectory, each with probability of violation less than tolerance ϵ.

B. Related Work

Model-based approaches often tackle problem (1) in a decoupled fashion: First, generate a nominal 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 linear-quadratic regulator (TVLQR) has achieved impressive results for complex, real-world systems [7, 16].

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

DIRTREL [10] optimizes a nominal trajectory with an additional cost term that penalizes a linearized approximation of the tracking error of a TVLQR policy. Chance constraints are approximated by enforcing them 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 propagates uncertainty through nonlinear dynamics.

Several learning-based approaches exist that directly optimize feedback policies. Policy gradient methods [17] use first-order or derivative-free methods to optimize parameterized policies, typically without direct access to the underlying dynamics model [18]. Domain randomization [20] can be employed to vary model and environment parameters to encourage robustness. Random search [12], stochastic gradient descent [4], and Newton methods [26] have also been used to optimize linear feedback policies and MPC controllers [1, 2]. A major benefit of stochastic methods is the inherent exploration of the policy space.

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 close to the current policy that are subsequently employed to train a neural-network 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 require explicit dynamics models or make strong assumptions about disturbance and state distributions, it relies heavily on Monte Carlo sampling 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 their derivatives 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, especially with long horizon plans.

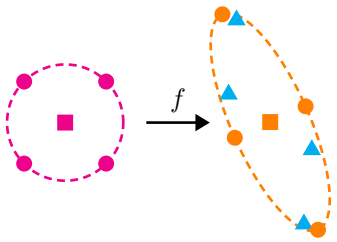

Fig. 2: The unscented transform visualized in 2D for initial (left) and transformed (right) distributions. Sigma points (circles) with sample mean (square) and sample covariance (dashed) are propagated through a nonlinear function (triangle) and then resampled to compute new sigma points.

C. Unscented Transform

The unscented tranform is a procedure for propagating a unimodal probability distribution through a nonlinear function using deterministic samples, often referred to as sigma points [22]. This tool is commonly used for state estimation, where it is generally considered to be superior to the extended Kalman filter's linear propagation of covariance matrices [22]. There are many variations of the unscented transform [15]; the version we utilize is outlined below and visualized in Fig. 2.

Assuming a unimodal state distribution with mean $\mu \in \mathbf{R}^n$ and covariance $P \in \mathbf{S}^n_{++}$, and a zero-mean disturbance distribution with covariance $W \in \mathbf{S}^d_{++}$, we generate $M = 2(n+d)$ sigma points:

$$\begin{bmatrix} x^{(j)} \\ w^{(j)} \end{bmatrix} \leftarrow \begin{bmatrix} \mu \\ 0 \end{bmatrix} \pm \beta \left(\sqrt{\begin{bmatrix} P & 0 \\ 0 & W \end{bmatrix}} \right)^{(j)}. \tag{2}$$

The samples are constructed using the column vectors of a square root of the joint covariance matrix. In this work we use the principle matrix square root, although other decompositions, such as the Cholesky factorization, could be employed. The parameter β can be used to control the spread of the samples around the mean.

The sigma points are first propagated through the policy,

$$u^{(j)} = \pi(x^{(j)}), \tag{3}$$

and then dynamics,

$$x_+^{(j)} = f(x^{(j)}, u^{(j)}, w^{(j)}), \tag{4}$$

in order to compute a sample mean,

$$\mu_+ = \frac{1}{M} \sum_{j=1}^{M} x_+^{(j)}, \tag{5}$$

and sample covariance,

$$P_+ = \frac{1}{\beta^2 M} \sum_{j=1}^{M} (x_+^{(j)} - \mu_+)(x_+^{(j)} - \mu_+)^T, \tag{6}$$

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

$$\nu = \frac{1}{M} \sum_{j=1}^{M} u^{(j)}, \tag{7}$$

and sample covariance,

$$L = \frac{1}{\beta^2 M} \sum_{j=1}^{M} (u^{(j)} - \nu)(u^{(j)} - \nu)^T, \tag{8}$$

for the control policy 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, we seek a local feedback policy that is only valid in the neighborhood of a nominal trajectory. Second, the expectation of the objective in (1) is approximated using the unscented transform. Finally, chance constraints are approximately enforced by applying constraints to a set of sample points chosen from desired level sets of the state and input distributions.

A. Cost Function

DPO minimizes the following cost function,

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

where J is a general cost applied to the nominal trajectory and S is a quadratic tracking cost,

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

that penalizes deviations from the nominal trajectory under disturbances. After some simple manipulations, we can write (9) in terms of the quantities introduced in Sec. II-C:

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

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

Algorithm 1: Sample dynamics

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

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. Feedback Policy

Each sample trajectory is subject to a policy constraint,

$$u_t = \pi_t(\theta_t, x_t, \bar{x}_t, \bar{u}_t). \tag{13}$$

Instead of optimizing global policies, we restrict the class to local feedback controllers. These policies could be linear state feedback, nonlinear feedback on features constructed from the state, or neural networks.

C. Constraints

Since DPO relies on direct collocation, dynamics are enforced through equality constraints. The nominal trajectory is not subject to disturbances, and the dynamics associated with the sigma-point trajectories are summarized in Algorithm 1. We maintain unimodal distributions over the state and policy by resampling at each time step using the unscented transform (2-8). Disturbances are assumed to be drawn from a zero-mean normal distribution, parameterized by a covariance matrix, $W_t \in \mathbf{S}_{++}^d$. By utilizing the discrete-time dynamics, it is possible to generate non-Gaussian noise for the system and to capture model uncertainty.

Chance constraints are approximated by constraining the nominal trajectory and each sample trajectory. With appropriate selection of the scaling parameter β, we can approximately enforce constraints on any desired level set of the state and control distributions to achieve a desired confidence bound, defined by ϵ. While it is possible to check constraints on ellipsoidal level sets by solving semidefinite programs, enforcing constraints on sigma points is computationally much cheaper. Additionally, deterministic constraints, $b : \mathbf{R}^z \to \mathbf{R}^q$, can easily be enforced on the nominal trajectory.

D. Nonlinear Programming Formulation

In summary, DPO can be formulated as the following nonlinear program (NLP):

$$\begin{aligned}
\underset{\theta, \bar{\tau}, \tau^{(1:N)}}{\text{minimize}} \quad & J(\bar{\tau}) + \mathbf{E}_{w_t}\left[S(\bar{\tau}, \tau)\right] \\
\text{subject to} \quad & \bar{x}_{t+1} = f_t(\bar{x}_t, \bar{u}_t, 0), \\
& x_{t+1}^{(i)} = g_t(\tau_t^{(1:N)}, W_t), \\
& (\bar{x}_1, x_1^{(i)} \text{ given}), \\
& u_t^{(i)} = \pi_t(\theta_t, x_t^{(i)}, \bar{x}_t, \bar{u}_t), \\
& b(\bar{\tau}) \leq 0, \\
& c(\bar{\tau}) \leq 0, \\
& c(\tau^{(i)}) \leq 0.
\end{aligned} \tag{14}$$

Off-the-shelf large-scale NLP solvers like Ipopt [24] and SNOPT [5] can be employed to efficiently solve (14) by taking advantage of sparsity in its associated Hessian and Jacobian matrices.

IV. Examples

The following examples were implemented in Julia, optimized with the SNOPT solver [5], and run on a laptop computer with an Intel Core i7-7600U 2.80 GHz CPU and 16 GB of memory. Our code is available at `https://github.com/thowell/direct_policy_optimization`.

In all examples, implicit midpoint integration is used and the same weights are used for the sample costs (10) and LQR controllers. To verify online tracking performance, the closed-loop policies are simulated at ten times the sample rate used in the optimization with additive zero-mean white Gaussian noise, third-order Runge-Kutta integration, zero-order-hold control interpolation, and cubic spline state interpolation. The tracking error is computed using the sample costs (10). All examples use quadratic objectives for the nominal trajectories, have discrete dynamics with additive noise,

$$f_t(x_t, u_t) + w_t, \tag{15}$$

and optimize linear feedback policies,

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

A. Double Integrator

We first demonstrate that DPO recovers the optimal discrete-time LQR solution for a double integrator,

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

that is regulated to the origin over a horizon $T = 51$. Because this is a regulator problem, the DPO nominal

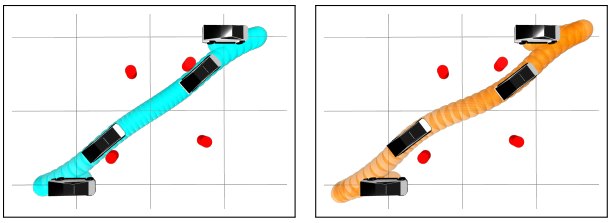

Fig. 3: Position trajectories for autonomous car avoiding obstacles. TO (left) finds a path (blue) that is near the obstacles, whereas the DPO (right) path (orange) avoids obstacles with a margin for safety.

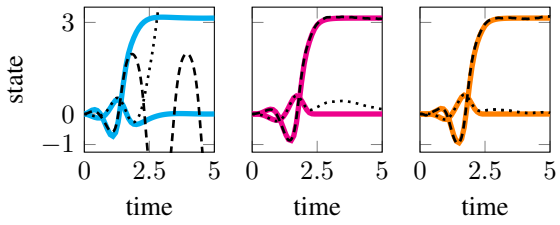

Fig. 4: Simulated tracking (black) for position (dotted) and orientation (dashed) of cart-pole experiencing Coulomb friction. Comparison between TVLQR tracking a nominal trajectory without (blue) and with (magenta) modeled friction, and the DPO policy (orange).

trajectory is fixed to zeros. There are $N = 4$ sample trajectories with initial conditions, $x_1^{(i)} = (\pm 1, \pm 1)$. Resampling is performed with $\beta = 1$ and $W = 0.1I$, the sample-cost weights are $Q = I$ and $R = I$, and the solver is initialized with ones.

The maximum normalized error between the LQR and DPO policies along the trajectory is 7.68e-7, 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.

B. Car

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

$$\begin{bmatrix} \dot{y} \\ \dot{z} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} u_1 \cos(\phi) \\ u_1 \sin(\phi) \\ u_2 \end{bmatrix}, \tag{18}$$

with $T = 51$ and $\Delta t = 0.02$. The state has positions y and z, and orientation ϕ. The controls are the forward and angular velocities, u_1 and u_2. For DPO, we set the sample-cost weights to be $Q_t = \text{diag}(10, 10, 1)$ for $t = 1 : T-1$, $Q_T = 100I$, and $R = 0.1I$, sample initial conditions for the $N = 6$ sample trajectories around the origin, and apply the unscented transform with $\beta = 1$ and $W = \text{diag}(0.001, 0.001, 0.0001)$. The choice of $\beta = 1$ results in constraints being enforced on the 1-sigma level set of the state distribution, corresponding to $\epsilon = 0.32$.

TO naturally finds a solution that is in close proximity to the obstacles. In contrast, DPO finds a path that reaches the goal state while remaining a safe distance from the obstacles (Fig. 3). Unlike *ad-hoc* techniques for obstacle avoidance, like obstacle inflation, the path generated by DPO results from consideration of the coupled dynamics, constraints, and disturbances of the system.

C. Cart-Pole

A swing-up trajectory for a cart-pole [19] with a slider that experiences Coulomb friction is synthesized for $T = 51$ and $\Delta t = 0.1$. The state, $x = (y, \phi, \dot{y}, \dot{\phi})$, has cart position y, pendulum orientation ϕ, and their respective time derivatives. The cart position is controlled. The optimality conditions for the maximum dissipation principle [16] are used as deterministic constraints to explicitly model this behavior [11] for a coefficient of friction $\mu_f = 0.1$. The DPO sample-cost weights are $Q_t = \text{diag}(10, 10, 1, 1)$ for $t = 1 : T-1$, $Q_T = 100I$, and $R = I$, initial conditions for the $N = 8$ sample trajectories are sampled around the origin, and resampling is performed with $\beta = 1$ and $W = 0.001I$. The performance of TVLQR tracking nominal trajectories optimized with and without friction, and the DPO policy is verified in simulation. Results are provided in Table I and tracking for the position and orientation is shown in Fig. 4. Without accounting for friction, the TVLQR policy fails to track. By first designing a trajectory that explicitly models friction, it is possible to subsequently synthesize a TVLQR policy for tracking. In contrast, DPO produces the best tracking results by simultaneously optimizing the nominal trajectory and policy.

D. Rocket Landing

We plan a soft landing trajectory for a rocket [13]. The nominal system with planar dynamics has state

TABLE I: Tracking error, computed with (10), for cart-pole experiencing Coulomb friction. Comparison between TVLQR tracking nominal trajectories with and without modeled friction, and the DPO policy.

cost	TVLQR (no friction)	TVLQR	DPO
state	1.03e4	2.32	**1.54**
control	62.87	0.38	**0.28**
total	1.04e4	2.70	**1.82**

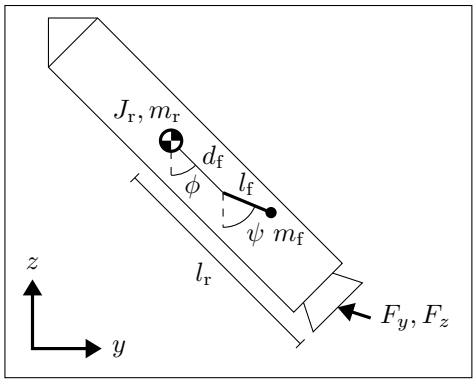

Fig. 5: Rocket with planar dynamics having lateral and vertical positions y and z, orientation ϕ, and controlled with thrust inputs F_y and F_z. The model parameters are the rocket inertia J_r, mass m_r, and length from center of mass to thruster l_r. An additional pendulum, with orientation ψ, and parameterized by distance from center of mass to rotation point d_f, length l_f, and mass m_f, is included to model fuel slosh.

$x = (y, z, \phi, \dot{y}, \dot{z}, \dot{\phi})$, with lateral and vertical positions, y and z, orientation ϕ, and their respective time derivatives (Fig. 5). 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 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 for the nominal model that is

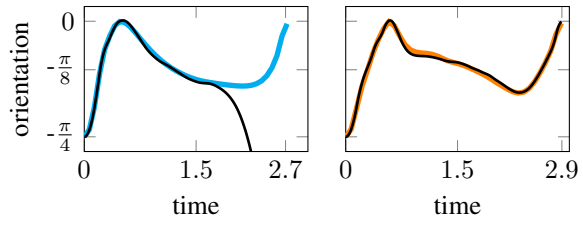

Fig. 6: Simulated tracking (black) for the orientation of a rocket experiencing fuel slosh during landing. Comparison between tracking of TVLQR (blue) and DPO (orange) polices.

TABLE II: Tracking error, computed with (10), for rocket that experiences fuel slosh while landing. The simulation uses a rocket with modeled fuel slosh to compare TVLQR tracking a trajectory generated with the nominal rocket model and the DPO policy.

cost	TVLQR	DPO
state	7294.55	**3.54**
control	35.51	**0.28**
total	7330.06	**3.82**

robust to fuel slosh by utilizing sample trajectories with the augmented model that capture fuel-slosh effects.

For DPO, the nominal trajectory is optimized with the nominal model because its states can be observed. The augmented model is utilized for each of $N = 16$ sample trajectories, and an output feedback policy is optimized that does not have access to the fuel-slosh states. The sample-cost weights are $Q_t = 100I$ for $t = 1 : T-1$, $Q_T = 1000I$, and $R = \mathrm{diag}(1, 1, 100)$. The sample initial conditions, including the fuel-slosh states, are sampled around the initial augmented state. The initial fuel-slosh state is a zero-velocity, downward orientation corresponding to no excitation. Resampling is performed with $\beta = 1$ and $W = 0.001I$. Thrust limits are enforced, the problem has a free final time, and the planning horizon is $T = 41$.

TO finds a 2.72 second solution, whereas DPO finds a slightly longer, 2.89 second, path to the landing zone. The position trajectories for the nominal model 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 the fuel slosh, TVLQR 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 model that is observable while capturing additional, unobservable dynamical effects through sample models.

E. Quadrotor

A point-to-point maneuver is planned for a quadrotor, with dynamics from [14], that experiences a random

TABLE III: Average tracking error, computed with (10), for quadrotor with different broken propellers. In simulation, each propeller is broken in order to compare the overall robustness of the TVLQR and DPO policies.

cost (avg.)	TVLQR	DPO
state	4.06	**0.50**
control	0.03	**0.01**
total	4.09	**0.51**

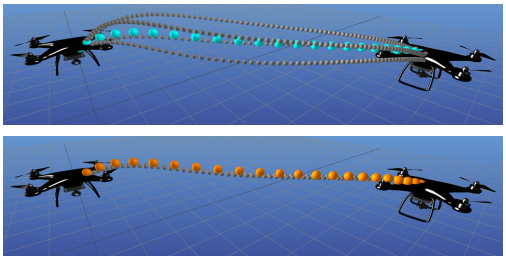

Fig. 7: Simulated position trajectories (gray) for quadrotor controlled with TVLQR (top) and DPO (bottom) policies while experiencing different propellers breaking. The TVLQR policy (blue) experiences degrading tracking and generates different paths depending on which propeller breaks. The DPO policy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

blade breaking off from a propeller during flight. A broken propeller is modeled by constraining the corresponding control input to only half its nominal maximum value. We use DPO to optimize a policy that is robust to this event occurring for any propeller. The nominal model has no broken propellers, but the $N = 24$ sample trajectories are divided into four groups, each with a different broken propeller. The sample-cost weights are $Q_t = 10I$ for $t = 1 : T - 1$, $Q_T = 100I$, and $R = I$. Initial conditions are sampled around the initial nominal state, and resampling is performed with $\beta = 1$ and $W_t = 0.001I$. The planning horizon is $T = 31$ with free final time.

Both TO and DPO find nominal trajectories that take about 2.6 seconds, but DPO finds lower maximum controls (Fig. 8). The policies are compared in simulation and are 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 TVLQR for all cases.

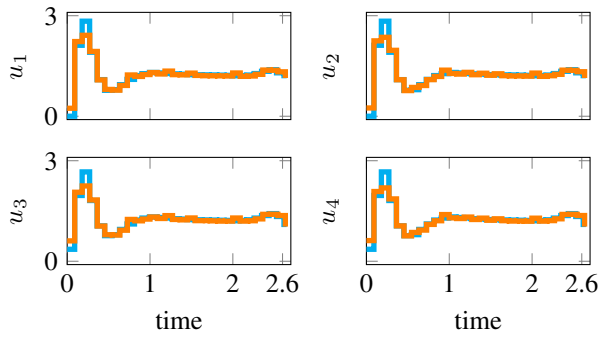

Fig. 8: Nominal controls for quadrotor performing point-to-point maneuver generated with TO (blue) and DPO (orange). The controls found with DPO have lower maximum values compared to TO.

V. Discussion

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 nominal and sample trajectories for DPO. For linear state-feedback policies, we use the corresponding TVLQR 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 solver, 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 $n(2n+1)$ and $m(2n+1)$, respectively. The complexity of DPO is then $O\Big(T(n^6 + m^3 n^3)\Big)$. In the examples, the largest state dimension is $n = 12$ and the solve times range from seconds to tens of minutes on a laptop computer. However, it is likely possible to exploit more of the DPO problem's structure to improve the complexity. Parallelization across sample points is also possible, which would enable faster solve times and scaling to much larger systems.

Lastly, LQR is a powerful tool and can likely be tuned to qualitatively match the performance of the linear policies found with DPO. 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. 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: It should be possible to optimize sparse nonlinear policies by introducing high-dimensional features and regularizing or constraining policy parameters. Neural networks could also be used to parameterize policies. Another potential direction is to optimize a local value function approximation in place of an explicit policy. Finally, a much richer class of disturbances and model parameter uncertainty could be modeled by augmenting the state vector and dynamics.

References

[1] Akshay Agrawal et al. "Differentiable convex optimization layers". In: *Advances in Neural Information Processing Systems*. 2019, pp. 9562–9574.

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

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

[18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

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