Robotics: Science and Systems 2022
New York City, NY, USA, June 27-July 1, 2022

Parameterized Differential Dynamic Programming

Alex Oshin*†‡, Matthew D. Houghton†, Michael J. Acheson†, Irene M. Gregory† and Evangelos A. Theodorou*
*School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA
†NASA Langley Research Center, Hampton, VA
‡Correspondence to: alexoshin@gatech.edu

***Abstract*—Differential Dynamic Programming (DDP) is an efficient trajectory optimization algorithm relying on second-order approximations of a system's dynamics and cost function, and has recently been applied to optimize systems with time-invariant parameters. Prior works include system parameter estimation and identifying the optimal switching time between modes of hybrid dynamical systems. This paper generalizes previous work by proposing a general parameterized optimal control objective and deriving a parametric version of DDP, titled Parameterized Differential Dynamic Programming (PDDP). A rigorous convergence analysis of the algorithm is provided, and PDDP is shown to converge to a minimum of the cost regardless of initialization. The effects of varying the optimization to more effectively escape local minima are analyzed. Experiments are presented applying PDDP on multiple robotics systems to solve model predictive control (MPC) and moving horizon estimation (MHE) tasks simultaneously. Finally, PDDP is used to determine the optimal transition point between flight regimes of a complex urban air mobility (UAM) class vehicle exhibiting multiple phases of flight.**

I. Introduction

Classically, optimal control research has focused on the study of methods for trajectory optimization of underactuated nonlinear systems. This attention is due to the fact that underactuated systems are more difficult to control due to having more degrees of freedom relative to the number of controls available [25]. However, the emerging urban air mobility (UAM) sector allows the opportunity to study the application of optimal control methods to overactuated aircraft whose complexity appears in the nonlinear transition dynamics between flight phases [12, 11]. These UAM class vehicles commonly exhibit three phases of flight — including vertical takeoff and landing (VTOL), fixed-wing cruise, and a complex transition phase between the two — and can thus be classified as hybrid systems [6, 9, 13], transitioning between multiple modes of the dynamics during a flight. While many algorithms have been developed for hybrid systems trajectory optimization in the past, most adopt a linear or reduced-order model, with the focus on bipedal or quadrupedal robotics systems, e.g. [22, 3, 7]. The lack of applications to UAM class vehicles reveals an opportunity to develop hybrid systems optimization techniques for unique aircraft dynamics which have not been previously studied.

Past research has shown Differential Dynamic Programming (DDP) is an effective algorithm for planning in high-dimensional state spaces [28], and DDP has shown recent success planning trajectories for UAM vehicles [14]. DDP is a shooting method that achieves computational efficiency using second-order approximations along a nominal trajectory and admits quadratic convergence properties under mild assumptions [21, 15]. DDP is beneficial in that it requires no reduction in the complexity of the dynamics model, and its convergence properties make it a solid candidate for solving realtime model predictive control (MPC) tasks [27].

Fig. 1: Rendering of the NASA Lift+Cruise aircraft.

Recent work has shown the value function derivatives produced by DDP can be used to update the initial conditions of the nominal trajectory, which allows optimization over time-invariant parameters [16, 17]. While this update corresponds to a Newton step on the parameters, modifying the initial condition of DDP significantly affects the convergence of the algorithm since the change induces a large shift of the nominal trajectory in the state space. To remedy this issue, this work introduces a general parameterized optimal control objective. Explicitly introducing the parameters into the system dynamics and cost function allows a second-order algorithm to be derived for iteratively updating both the controls and parameters simultaneously, independent of the form of the parameterization. This parameterized version of DDP is referred to as Parameterized Differential Dynamic Programming (PDDP), and a rigorous convergence analysis is provided showing the derived algorithm converges to a minimum of the cost regardless of state or parameter initialization. Further, the effects of the simultaneous optimization of parameters and controls is studied. An optimization scheme is proposed that more effectively escapes local minima, which is a common issue of local methods such as DDP. The PDDP algorithm is applied to two important robotics tasks, model parameter estimation and hybrid systems optimization, and is used to

identify the optimal transition points between flight regimes for the NASA Lift+Cruise UAM class vehicle developed under NASA's Revolutionary Vertical Lift Technology (RVLT) project [24]. A rendering of the Lift+Cruise aircraft is provided in Fig. 1.

The contributions of this work are as follows:

1) Generalize previous work by deriving a parameterized form of DDP, referred to as PDDP.
2) Provide theoretical analysis of the convergence behavior of the proposed PDDP algorithm and show it is globally convergent to a minimum of the cost for optimization problems with arbitrary dynamics and cost function parameters.
3) Discuss and analyze three optimization choices for iteratively solving for the optimal controls and parameters.
4) Apply the proposed method on multiple robotics systems, including a cartpole, a quadrotor, and an ant quadruped system using the open-source physics engine Brax [10], and show PDDP is able to solve adaptive MPC tasks using moving horizon estimation (MHE).
5) Apply PDDP to find the optimal transition point between multi-modal dynamical systems, including a UAM class vehicle exhibiting multiple phases of flight.

The paper is organized as follows. Section II derives the PDDP algorithm and proves the convergence of the method. Section III discusses applications of PDDP to system parameter estimation and switching time optimization for hybrid systems. Section IV analyzes experimental results applying PDDP to the previously discussed tasks. The paper concludes with Section V by discussing future work.

II. Parameterized Differential Dynamic Programming

In this section, the parameterized version of DDP is derived. While the similarities are highlighted here, a full derivation of standard DDP can be found in recent works such as [28].

A. Problem Formulation

This work considers parameterized discrete-time dynamics that evolve according to

$$\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t, \mathbf{u}_t; \boldsymbol{\theta}), \tag{1}$$

where $\mathbf{F} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_x}$ with $\mathbf{x}_t \in \mathbb{R}^{n_x}$, $\mathbf{u}_t \in \mathbb{R}^{n_u}$, and $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$, and $\mathbf{x}_1 \in \mathbb{R}^{n_x}$ denoting the initial condition. The semicolon emphasizes that the parameters are time-invariant and separate from states or controls that are time-varying. The same assumptions made in standard DDP [15] are adopted here. Namely, the dynamics are assumed to be twice differentiable with respect to state and control, with the additional assumption that the dynamics are twice differentiable with respect to the parameters.

Defining the control sequence $\mathbf{U} := \{\mathbf{u}_1, \ldots, \mathbf{u}_T\}$, the parameterized discrete-time optimal control problem is given by

$$\min_{\mathbf{U}, \boldsymbol{\theta}} \mathcal{J}(\mathbf{U}; \boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \min_{\mathbf{U}} \sum_{t=1}^{T} \mathcal{L}(\mathbf{x}_t, \mathbf{u}_t; \boldsymbol{\theta}) + \phi(\mathbf{x}_{T+1}; \boldsymbol{\theta}), \tag{2}$$

where $\mathcal{L}$ and ϕ are the twice differentiable running cost and terminal cost function, respectively, and T is the time horizon. In the most general case, the parameters $\boldsymbol{\theta}$ affect both the dynamics and the cost.

The inner objective of Eq. (2) with $\boldsymbol{\theta}$ fixed is equivalent to the standard DDP optimal control problem. The value function, now parameterized by $\boldsymbol{\theta}$, is given recursively by

$$V(\mathbf{x}_t; \boldsymbol{\theta}) = \min_{\mathbf{u}_t} \Big[\underbrace{\mathcal{L}(\mathbf{x}_t, \mathbf{u}_t; \boldsymbol{\theta}) + V(\mathbf{x}_{t+1}; \boldsymbol{\theta})}_{:= Q(\mathbf{x}_t, \mathbf{u}_t; \boldsymbol{\theta})} \Big], \tag{3}$$

with $V(\mathbf{x}_{T+1}; \boldsymbol{\theta}) = \phi(\mathbf{x}_{T+1}; \boldsymbol{\theta})$. Thus, the optimal $\boldsymbol{\theta}$ is given at the initial time $t = 1$ by

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} V(\mathbf{x}_1; \boldsymbol{\theta}). \tag{4}$$

B. Algorithm Derivation

Given the parameterized dynamics defined by Eq. (1), this subsection will derive an iterative algorithm for finding the control trajectory $\mathbf{U}$ and parameters $\boldsymbol{\theta}$ that minimize the cost function defined in Eq. (2).

As in standard DDP, the parameterized optimal control problem is solved by considering quadratic approximations of the value function along a nominal trajectory $\bar{\mathbf{x}}_t$, $\bar{\mathbf{u}}_t$. However, this derivation also includes terms expanding the value function about a set of nominal parameters $\bar{\boldsymbol{\theta}}$. Let $\delta\mathbf{x}_t, \delta\mathbf{u}_t, \delta\boldsymbol{\theta}$ be the variation in state, control, and parameters, respectively, so that

$$\mathbf{x}_t := \bar{\mathbf{x}}_t + \delta\mathbf{x}_t, \quad \mathbf{u}_t := \bar{\mathbf{u}}_t + \delta\mathbf{u}_t, \quad \boldsymbol{\theta} := \bar{\boldsymbol{\theta}} + \delta\boldsymbol{\theta}. \tag{5}$$

The quadratic expansion of the value function about $\bar{\mathbf{x}}_t, \bar{\boldsymbol{\theta}}$ has the form

$$\begin{aligned} V(\mathbf{x}_t; \boldsymbol{\theta}) \approx V_t^0 + (V_t^x)^\top \delta\mathbf{x}_t + (V_t^\theta)^\top \delta\boldsymbol{\theta} \\ + \frac{1}{2} \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\boldsymbol{\theta} \end{bmatrix}^\top \begin{bmatrix} V_t^{xx} & V_t^{x\theta} \\ V_t^{\theta x} & V_t^{\theta\theta} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\boldsymbol{\theta} \end{bmatrix}, \end{aligned} \tag{6}$$

where the partial derivatives of V are denoted using superscripts and are evaluated at $\bar{\mathbf{x}}_t, \bar{\boldsymbol{\theta}}$, e.g., $V_t^0 = V(\bar{\mathbf{x}}_t; \bar{\boldsymbol{\theta}})$, $V_t^x = \nabla_x V(\bar{\mathbf{x}}_t; \bar{\boldsymbol{\theta}})$, etc. This superscript notation will be adopted throughout the paper. In order to solve for the partial derivatives of V, the quadratic expansion of the Q function defined in Eq. (3) is taken about $\bar{\mathbf{x}}_t$, $\bar{\mathbf{u}}_t, \bar{\boldsymbol{\theta}}$:

$$\begin{aligned} Q(\mathbf{x}_t, \mathbf{u}_t; \boldsymbol{\theta}) \approx Q_t^0 + (Q_t^x)^\top \delta\mathbf{x}_t + (Q_t^u)^\top \delta\mathbf{u}_t + (Q_t^\theta)^\top \delta\boldsymbol{\theta} \\ + \frac{1}{2} \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \\ \delta\boldsymbol{\theta} \end{bmatrix}^\top \begin{bmatrix} Q_t^{xx} & Q_t^{xu} & Q_t^{x\theta} \\ Q_t^{ux} & Q_t^{uu} & Q_t^{u\theta} \\ Q_t^{\theta x} & Q_t^{\theta u} & Q_t^{\theta\theta} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \\ \delta\boldsymbol{\theta} \end{bmatrix}, \end{aligned} \tag{7}$$

where

$$
\begin{aligned}
Q_t^0 &= \mathcal{L}_t^0 + V_{t+1}^0, \\
Q_t^x &= \mathcal{L}_t^x + (\mathbf{F}_t^x)^\top V_{t+1}^x, \\
Q_t^u &= \mathcal{L}_t^u + (\mathbf{F}_t^u)^\top V_{t+1}^x, \\
Q_t^\theta &= \mathcal{L}_t^\theta + V_{t+1}^\theta + (\mathbf{F}_t^\theta)^\top V_{t+1}^x, \\
Q_t^{xx} &= \mathcal{L}_t^{xx} + (\mathbf{F}_t^x)^\top V_{t+1}^{xx} \mathbf{F}_t^x, \\
Q_t^{xu} &= \mathcal{L}_t^{xu} + (\mathbf{F}_t^x)^\top V_{t+1}^{xx} \mathbf{F}_t^u = (Q_t^{ux})^\top, \\
Q_t^{x\theta} &= \mathcal{L}_t^{x\theta} + (\mathbf{F}_t^x)^\top V_{t+1}^{x\theta} + (\mathbf{F}_t^x)^\top V_{t+1}^{xx} \mathbf{F}_t^\theta = (Q_t^{\theta x})^\top, \\
Q_t^{uu} &= \mathcal{L}_t^{uu} + (\mathbf{F}_t^u)^\top V_{t+1}^{xx} \mathbf{F}_t^u, \\
Q_t^{u\theta} &= \mathcal{L}_t^{u\theta} + (\mathbf{F}_t^u)^\top V_{t+1}^{x\theta} + (\mathbf{F}_t^u)^\top V_{t+1}^{xx} \mathbf{F}_t^\theta = (Q_t^{\theta u})^\top, \\
Q_t^{\theta\theta} &= \mathcal{L}_t^{\theta\theta} + V_{t+1}^{\theta\theta} + 2(\mathbf{F}_t^\theta)^\top V_{t+1}^{x\theta} + (\mathbf{F}_t^\theta)^\top V_{t+1}^{xx} \mathbf{F}_t^\theta.
\end{aligned} \tag{8}
$$

Note the partial derivatives with respect to $\boldsymbol{\theta}$ include extra terms because both the value function and the dynamics depend on $\boldsymbol{\theta}$. In Eq. (8), the second-order dynamics terms have been dropped following the iterative Linear-Quadratic Regulator (iLQR) algorithm [18]. A full derivation with all second-order terms is given in Appendix A.

Substituting this quadratic approximation into Eq. (3) and dropping the terms not dependent on $\delta\mathbf{u}_t$ gives

$$
\begin{aligned}
V(\mathbf{x}_t;\boldsymbol{\theta}) = \min_{\delta\mathbf{u}_t} \Big[&(Q_t^u)^\top \delta\mathbf{u}_t + \delta\mathbf{x}_t^\top Q_t^{xu} \delta\mathbf{u}_t \\
&+ \delta\boldsymbol{\theta}^\top Q_t^{\theta u} \delta\mathbf{u}_t + \frac{1}{2} \delta\mathbf{u}_t^\top Q_t^{uu} \delta\mathbf{u}_t \Big].
\end{aligned} \tag{9}
$$

Taking the gradient of the argument in the minimization and setting it equal to zero yields the optimal control update

$$
\delta\mathbf{u}_t^* = \mathbf{k}_t + \mathbf{K}_t \delta\mathbf{x}_t + \mathbf{M}_t \delta\boldsymbol{\theta} \tag{10}
$$

with

$$
\begin{aligned}
\mathbf{k}_t &:= -(Q_t^{uu})^{-1} Q_t^u, \\
\mathbf{K}_t &:= -(Q_t^{uu})^{-1} Q_t^{ux}, \\
\mathbf{M}_t &:= -(Q_t^{uu})^{-1} Q_t^{u\theta}.
\end{aligned} \tag{11}
$$

Note the gains $\mathbf{k}_t$ and $\mathbf{K}_t$ match the standard DDP gains, with an additional feedback gain $\mathbf{M}_t$ on $\delta\boldsymbol{\theta}$ providing a correction because the algorithm is optimizing over $\delta\mathbf{u}_t$ and $\delta\boldsymbol{\theta}$ simultaneously. Taking $\delta\boldsymbol{\theta} = \mathbf{0}$ in Eq. (10) means the parameters are held constant which yields the usual DDP control update.

The optimal $\boldsymbol{\theta}$ minimizes Eq. (3) at time $t = 1$. Since the initial condition is fixed, $\mathbf{x}_1 = \bar{\mathbf{x}}_1$, which implies $\delta\mathbf{x}_1 = \mathbf{0}$. Substituting in the quadratic approximation of Q into Eq. (4) and dropping the terms not dependent on $\delta\boldsymbol{\theta}$, yields

$$
\begin{aligned}
\delta\boldsymbol{\theta}^* &= \underset{\delta\boldsymbol{\theta}}{\operatorname{argmin}} \left[(Q_1^\theta)^\top \delta\boldsymbol{\theta} + \delta\mathbf{u}_1^\top Q_1^{u\theta} \delta\boldsymbol{\theta} + \frac{1}{2} \delta\boldsymbol{\theta}^\top Q_1^{\theta\theta} \delta\boldsymbol{\theta} \right] \\
&= -(Q_1^{\theta\theta})^{-1} Q_1^\theta - (Q_1^{\theta\theta})^{-1} Q_1^{\theta u} \delta\mathbf{u}_1.
\end{aligned} \tag{12}
$$

At the initial time $t = 1$, there are two equations and two unknowns $\delta\boldsymbol{\theta}^*$ and $\delta\mathbf{u}_1^*$. Substituting in $\delta\mathbf{u}_1^*$ from Eq. (10) results in the parameter update

$$
\begin{aligned}
\delta\boldsymbol{\theta}^* = \mathbf{m} := -(&Q_1^{\theta\theta} - Q_1^{\theta u} (Q_1^{uu})^{-1} Q_1^{u\theta})^{-1} \\
&(Q_1^\theta - Q_1^{\theta u} (Q_1^{uu})^{-1} Q_1^u).
\end{aligned} \tag{13}
$$

When the feedforward gains $\mathbf{k}_t$ of Eq. (10) and $\mathbf{m}$ of Eq. (13) are too aggressive, the state trajectory may stray from the region where the quadratic approximation is accurate, and the cost may not decrease. To ensure convergence, the feedforward gains are scaled by the parameter $0 < \epsilon \leq 1$, such that

$$
\delta\mathbf{u}_t^* = \epsilon\mathbf{k}_t + \mathbf{K}_t \delta\mathbf{x}_t + \mathbf{M}_t \delta\boldsymbol{\theta}^*, \tag{14}
$$

$$
\delta\boldsymbol{\theta}^* = \epsilon\mathbf{m}. \tag{15}
$$

The parameter ϵ is determined using line search and ensures the optimizer makes sufficient progress towards the minimum at each iteration, i.e. ensuring the update step satisfies Armijo's condition or the Wolfe conditions [5]. Further discussion is given in Section II-C.

What remains is to solve the value function derivatives by substituting in the expression for $\delta\mathbf{u}_t^*$. The expression for $\delta\boldsymbol{\theta}^*$ is not substituted back in, otherwise the terms in $\delta\boldsymbol{\theta}^*$ are incorporated into the zero-order value function term and the derivatives with respect to $\boldsymbol{\theta}$ go to zero. This means that the information for $\boldsymbol{\theta}$ cannot be propagated backwards in time. In Section II-C, to prove the cost reduction achievable, the full expression for $\delta\boldsymbol{\theta}^*$ is substituted into Eq. (7).

Substituting Eq. (14) into Eq. (7) and equating like powers of the value function expansion in Eq. (6) gives the following expressions for the value function derivatives:

$$
\begin{aligned}
V_t^0 &= Q_t^0 + (\frac{1}{2}\epsilon^2 - \epsilon)(Q_t^u)^\top (Q_t^{uu})^{-1} Q_t^u, \\
V_t^x &= Q_t^x - Q_t^{xu} (Q_t^{uu})^{-1} Q_t^u, \\
V_t^\theta &= Q_t^\theta - Q_t^{\theta u} (Q_t^{uu})^{-1} Q_t^u, \\
V_t^{xx} &= Q_t^{xx} - Q_t^{xu} (Q_t^{uu})^{-1} Q_t^{ux}, \\
V_t^{x\theta} &= Q_t^{x\theta} - Q_t^{xu} (Q_t^{uu})^{-1} Q_t^{u\theta} = (V_t^{\theta x})^\top, \\
V_t^{\theta\theta} &= Q_t^{\theta\theta} - Q_t^{\theta u} (Q_t^{uu})^{-1} Q_t^{u\theta}.
\end{aligned} \tag{16}
$$

Eq. (16) and Eq. (8) provide the equations for the backward pass of PDDP, with boundary conditions $V_{T+1}^0 = \phi(\bar{\mathbf{x}}_{T+1}; \bar{\boldsymbol{\theta}})$, $V_{T+1}^x = \nabla_x \phi(\bar{\mathbf{x}}_{T+1}; \bar{\boldsymbol{\theta}})$, etc. The derivatives of V and Q are solved for backwards in time starting from $t = T+1$ down to the initial time $t = 1$ along the nominal trajectory. The forward pass of the algorithm consists of using the control updates from Eq. (14) and parameter updates from Eq. (15) to compute the new state trajectory starting from the initial condition $\mathbf{x}_1$. This yields an updated nominal trajectory, and this process can be repeated until some convergence criteria is met. Discussion on regularization of the value function derivatives as well as methods for updating the controls and parameters is given in Section II-D once the convergence of the method is proven.

C. Convergence Analysis

This section provides a mathematically rigorous convergence analysis of the proposed PDDP algorithm. The main result is summarized in Theorem 1, which shows PDDP is globally convergent to a minimum of the cost function. The same assumptions as in [15] for classic DDP are made in this work, including the differentiability of the dynamics and cost function up to second-order.

The following proposition establishes the optimality of the parameter update step by drawing comparisons to Newton's method.

Proposition 1. *The parameter update $\delta\boldsymbol{\theta}^*$ is a (damped) Newton step towards the minimum of the value function.*

Proof: See Appendix B. ∎

Proposition 1 implies PDDP converges quadratically fast to the optimal parameters $\boldsymbol{\theta}^*$, adopting the convergence rate of Newton's method. It is important to note that PDDP differs from a pure stagewise Newton's method in that it uses the full nonlinear dynamics during the forward pass, thus achieving better numerical convergence properties [19].

However, for general nonlinear, nonconvex problems, it cannot be guaranteed that $V_1^{\theta\theta}$ or Q_1^{uu} remain positive definite, which motivates the addition of regularization to ensure convergence. Regularization can be accomplished through addition of a Levenberg-Marquardt parameter ensuring the Hessian matrices are always positive definite [29]. Throughout this work, the regularization scheme proposed in [27] is adopted. Further discussion is given in Section II-D.

Next, the control updates are proven to reduce the cost after each iteration of PDDP. To this end, an expression for the gradient of the cost function with respect to an individual control is derived, and it is shown that the variations in control, state, and parameters are $O(\epsilon)$.

Lemma 1. *The gradient of the cost function with respect to the control at time t is given by*

$$\nabla_{\mathbf{u}_t}\mathcal{J} = \mathcal{L}_t^u + (\mathbf{F}_t^u)^\top \eta_{t+1}, \tag{17}$$

with $\eta_t = \mathcal{L}_t^x + (\mathbf{F}_t^x)^\top \eta_{t+1}$ for $t = 1, \ldots, T$ and $\eta_{T+1} = \phi_{T+1}^x$.

Proof: See Appendix C. ∎

Lemma 2. *For the form of $\delta\boldsymbol{\theta}$ given in Eq. (15), it is true that*

$$\delta\boldsymbol{\theta} = O(\epsilon), \tag{18}$$

where $O(\cdot)$ corresponds to big-O notation in $\|\cdot\|_2$.

Further, for all $t = 1, \ldots, T$,

$$\delta\mathbf{u}_t = O(\epsilon), \tag{19a}$$

$$\delta\mathbf{x}_{t+1} = O(\epsilon). \tag{19b}$$

Proof: See Appendix D. ∎

Next, the control updates are shown to be a descent direction of the cost.

Proposition 2. *The optimal control updates satisfy*

$$(\nabla_{\mathbf{U}}\mathcal{J})^\top \Delta\mathbf{U} = \sum_{t=1}^{T} (\nabla_{\mathbf{u}_t}\mathcal{J})^\top \delta\mathbf{u}_t < 0. \tag{20}$$

Proof: It can be shown that

$$\sum_{t=1}^{T} (\nabla_{\mathbf{u}_t}\mathcal{J})^\top \delta\mathbf{u}_t = -\epsilon \sum_{t=1}^{T} \lambda_t + \epsilon \left(\sum_{t=1}^{T} \gamma_t\right) (V_1^{\theta\theta})^{-1} V_1^\theta + O(\epsilon^2),$$

where $\lambda_t = (Q_t^u)^\top (Q_t^{uu})^{-1} Q_t^u$ and $\gamma_t = (Q_t^u)^\top (Q_t^{uu})^{-1} Q_t^{u\theta}$. This implies there exists some ϵ sufficiently small such that Eq. (20) holds. The full proof is given in Appendix E. ∎

Remark 1. The row vector $\sum_{t=1}^{T} \gamma_t$ can be thought of as correcting for the change in $\boldsymbol{\theta}$, as both the parameters and the controls are optimized simultaneously. The inner product between $\sum_{t=1}^{T} \gamma_t$ and the parameter ascent direction $(V_1^{\theta\theta})^{-1} V_1^\theta$ ensures that the optimizer does not take too large of a step in the case when the controls and the parameters both contribute to a reduction in cost.

To show the cost reduction achievable after a single iteration of PDDP, the expression for $\delta\boldsymbol{\theta}^*$ must be substituted into the value function approximation given by Eq. (7). This substitution means the terms in $\delta\boldsymbol{\theta}$ are now incorporated into the zero-order term V_t^0. The previous expression for V_t^0 given in Eq. (16) only accounts for the change in controls $\delta\mathbf{u}_t^*$, which was necessary to derive a backwards rule for the value function derivatives with respect to $\boldsymbol{\theta}$. The following lemma establishes the value function change by applying the updates $\delta\mathbf{u}_t^*$ and $\delta\boldsymbol{\theta}^*$.

Lemma 3. *The zero-order value function approximation term satisfies*

$$V_t^0 = Q_t^0 - \epsilon(1 - \frac{1}{2}\epsilon)\lambda_t + \epsilon(V_t^\theta)^\top \mathbf{m} + \frac{1}{2}\epsilon^2 \mathbf{m}^\top V_t^{\theta\theta} \mathbf{m}. \tag{21}$$

Further, at time $t = 1$, the following is true:

$$V_1^0 = Q_1^0 - \epsilon(1 - \frac{1}{2}\epsilon)(\lambda_1 + \psi), \tag{22}$$

where $\psi = (V_1^\theta)^\top (V_1^{\theta\theta})^{-1} V_1^\theta$.

Proof: See Appendix F. ∎

Remark 2. ψ can be interpreted as the Newton decrement on the parameters. Likewise, λ_t can be interpreted as the Newton decrement on the controls at each timestep. These values can be used as an effective stopping criteria for the algorithm as well as verification for the line search parameter ϵ [5]. Further discussion is provided in the following proposition.

Now, it is possible to show the cost reduction achievable after an iteration of PDDP.

Proposition 3. *The total cost reduction after each iteration of PDDP is*

$$\Delta\mathcal{J} = -\epsilon(1 - \frac{1}{2}\epsilon)\left(\sum_{t=1}^{T} \lambda_t + \psi\right) + O(\epsilon^3). \tag{23}$$

Proof: See Appendix G. ∎

Remark 3. Proposition 3 implies that a valid line search candidate at each iteration should satisfy the following cost reduction:

$$\Delta\mathcal{J} \leq -\kappa\epsilon\left(\sum_{t=1}^{T} \lambda_t + \psi\right), \tag{24}$$

for a small constant $\kappa > 0$. This condition is similar to the line search criteria for Newton's method, and ensures that Armijo's condition or the Wolfe conditions hold [5].

Finally, the convergence of PDDP to a minimum of the cost is proven.

Theorem 1. *As the number of iterations of PDDP approaches infinity, the cost $\mathcal{J}$, the control trajectory $\mathbf{U}$, and the parameters $\boldsymbol{\theta}$ converge to a stationary point regardless of initialization.*

Proof: See Appendix H. ∎

D. Algorithm Design

The convergence result presented in Section II-C assumes that the Hessian matrices Q_t^{uu} and $V_1^{\theta\theta}$ remain positive definite throughout the optimization process. One method of ensuring this condition always holds is to use Levenberg-Marquardt regularization corresponding to adding a quadratic cost around the nominal trajectory [29]. This regularization can be employed to ensure $V_1^{\theta\theta}$ is positive definite since this Hessian matrix only needs to be invertible at the first timestep. However, the authors of [27] have shown a more robust approach for regularizing Q_t^{uu} places the regularization on the states rather than the controls, ensuring the control perturbation does not have different effects at different timesteps. Therefore, the regularized derivatives can be computed using

$$\begin{aligned} \tilde{V}_{t+1}^{xx} &= V_{t+1}^{xx} + \mu\mathbf{I} \\ \tilde{V}_1^{\theta\theta} &= V_1^{\theta\theta} + \nu\mathbf{I}, \end{aligned} \tag{25}$$

with hyperparameters $\mu, \nu \geq 0$. The regularized Hessian $\tilde{V}_{t+1}^{xx}$ is used in place of the original during the backward pass calculation of Eq. (8). The regularization of V_{t+1}^{xx} and $V_1^{\theta\theta}$ correspond to adding a quadratic cost to the deviation of the state and parameters from the nominal values, thus ensuring the update step does not stray too far from the region where the quadratic approximation is accurate. However, the addition of the regularization on V_{t+1}^{xx} when calculating the derivatives in Eq. (16) means the same cancellations of Q_t^{uu} and its inverse are not possible. The updated derivatives are given as

$$\begin{aligned} V_t^x &= Q_t^x + \mathbf{K}_t^\top Q_t^u + Q_t^{xu}\mathbf{k}_t + \mathbf{K}_t^\top Q_t^{uu}\mathbf{k}_t, \\ V_t^\theta &= Q_t^\theta + \mathbf{M}_t^\top Q_t^u + Q_t^{\theta u}\mathbf{k}_t + \mathbf{M}_t^\top Q_t^{uu}\mathbf{k}_t, \\ V_t^{xx} &= Q_t^{xx} + Q_t^{xu}\mathbf{K}_t + \mathbf{K}_t^\top Q_t^{ux} + \mathbf{K}_t^\top Q_t^{uu}\mathbf{K}_t, \\ V_t^{x\theta} &= Q_t^{x\theta} + Q_t^{xu}\mathbf{M}_t + \mathbf{K}_t^\top Q_t^{u\theta} + \mathbf{K}_t^\top Q_t^{uu}\mathbf{M}_t \\ V_t^{\theta\theta} &= Q_t^{\theta\theta} + Q_t^{\theta u}\mathbf{M}_t + \mathbf{M}_t^\top Q_t^{u\theta} + \mathbf{M}_t^\top Q_t^{uu}\mathbf{M}_t. \end{aligned} \tag{26}$$

An additional benefit of the proposed method is that the optimization of the controls and parameters can be coupled or decoupled without affecting the convergence. The result in Theorem 1 assumes both the control and parameter feedforward gains are applied simultaneously, but the same result holds if the gains are applied in an alternating fashion, e.g. at every odd iteration the controls are updated by applying the control feedforward gain $\mathbf{k}_t$ and at every even iteration the parameters are updated by applying the parameter feedforward gain $\mathbf{m}$. The addition of the feedback term $\mathbf{M}_t\delta\boldsymbol{\theta}$ on the parameter update when calculating the updated controls in Eq. (14) ensures stability in the controls even when only the parameters are being updated. Likewise, [17] adopt an approach where the parameters are only updated once the controls have fully converged. This approach is referred to as Switching Time Optimization Differential Dynamic Programming (STO-DDP). The alternating and simultaneous schemes are benchmarked against STO-DDP through a numerical comparison on three systems, which is presented in Section IV-B.

In summary, Algorithm 1 describes the proposed PDDP algorithm.

Algorithm 1: Parameterized Differential Dynamic Programming

Input: Nominal trajectory $\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t$ and parameters $\bar{\boldsymbol{\theta}}$
while *not converged* **do**
  Calculate derivatives of $\mathcal{L}$, ϕ along $\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t, \bar{\boldsymbol{\theta}}$
  `// Backward pass`
  $V_{T+1}^0 \leftarrow \phi_{T+1}^0$, $V_{T+1}^x \leftarrow \phi_{T+1}^x$, $V_{T+1}^\theta \leftarrow \phi_{T+1}^\theta$,
  $V_{T+1}^{xx} \leftarrow \phi_{T+1}^{xx}$, $V_{T+1}^{x\theta} \leftarrow \phi_{T+1}^{x\theta}$, $V_{T+1}^{\theta\theta} \leftarrow \phi_{T+1}^{\theta\theta}$
  for $t = T, \ldots, 1$ **do**
    Calculate derivatives of Q according to Eq. (8) and Eq. (25)
    Calculate gains $\mathbf{k}_t, \mathbf{K}_t, \mathbf{M}_t$ according to Eq. (11)
    Calculate derivatives of V according to Eq. (26)
  `// End backward pass`
  Calculate gain $\mathbf{m}$ according to Eq. (13)
  `// Line search`
  $\epsilon \leftarrow 1$
  while *cost decrease not sufficient* **do**
    `// Forward pass`
    if *update parameters* **then**
      $\delta\boldsymbol{\theta} \leftarrow \epsilon\mathbf{m}$
    else
      $\delta\boldsymbol{\theta} \leftarrow \mathbf{0}$
    $\boldsymbol{\theta} \leftarrow \bar{\boldsymbol{\theta}} + \delta\boldsymbol{\theta}$
    $\mathbf{x}_1 \leftarrow \bar{\mathbf{x}}_1$
    for $t = 1, \ldots, T$ **do**
      $\delta\mathbf{x}_t \leftarrow \mathbf{x}_t - \bar{\mathbf{x}}_t$
      if *update controls* **then**
        $\delta\mathbf{u}_t \leftarrow \epsilon\mathbf{k}_t + \mathbf{K}_t\delta\mathbf{x}_t + \mathbf{M}_t\delta\boldsymbol{\theta}$
      else
        $\delta\mathbf{u}_t \leftarrow \mathbf{K}_t\delta\mathbf{x}_t + \mathbf{M}_t\delta\boldsymbol{\theta}$
      $\mathbf{u}_t \leftarrow \bar{\mathbf{u}}_t + \delta\mathbf{u}_t$
      $\mathbf{x}_{t+1} \leftarrow \mathbf{F}(\mathbf{x}_t, \mathbf{u}_t; \boldsymbol{\theta})$
    `// End forward pass`
    $\epsilon \leftarrow \rho\epsilon$ `// Reduce` ϵ
  `// End line search`
  $\bar{\mathbf{x}}_t \leftarrow \mathbf{x}_t$, $\bar{\mathbf{u}}_t \leftarrow \mathbf{u}_t$, $\bar{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$
return *optimal controls* $\mathbf{u}_t$, *optimal parameters* $\boldsymbol{\theta}$

III. Applications

In this section the proposed PDDP algorithm is applied to two important robotics control tasks: parameter estimation and switching time optimization.

A. Parameter Estimation

Estimating the unknown parameters and states of a dynamical system can be achieved through moving horizon estimation (MHE). This framework reformulates the state estimation problem as an optimization problem dual to MPC [8]. This work assumes full state observability, with the realized discrete-time dynamics given by

$$\begin{aligned}\mathbf{x}_{t+1} &= \mathbf{F}(\mathbf{x}_t, \mathbf{u}_t; \boldsymbol{\theta}^*) + \mathbf{w}_t, \\ \mathbf{x}_1 &\sim \mathcal{N}(\hat{\mathbf{x}}_1, \boldsymbol{\Sigma}_{\mathbf{x}_1}),\end{aligned}$$

where $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_w)$ is some unknown additive Gaussian process noise. If the controls are given, this corresponds to a trajectory smoothing problem where the goal is to identify the parameters $\boldsymbol{\theta}$ and the realized disturbances $\mathbf{w}_t$ that maximize the likelihood of the observed states. The objective can be formulated through maximum likelihood estimation (MLE) by minimizing the negative log-likelihood

$$\begin{aligned}\mathcal{J}_{\text{est}}(\boldsymbol{\theta}, \mathbf{x}_1) &= -\log p(\mathbf{x}_1, \boldsymbol{\theta}) \prod_{t=1}^{T} p(\mathbf{x}_{t+1} | \mathbf{x}_t, \boldsymbol{\theta}) \\ &= -\log p(\mathbf{x}_1, \boldsymbol{\theta}) - \sum_{t=1}^{T} \log p(\mathbf{x}_{t+1} | \mathbf{x}_t, \boldsymbol{\theta}),\end{aligned}$$

with $p(\mathbf{x}_{t+1} | \mathbf{x}_t, \boldsymbol{\theta})$ describing the likelihood of observing the next state $\mathbf{x}_{t+1}$ given the current state $\mathbf{x}_t$ and parameters $\boldsymbol{\theta}$, and $p(\mathbf{x}_1, \boldsymbol{\theta})$ describing the prior distribution over the initial state and parameters.

Assuming a Gaussian prior over the parameters $\boldsymbol{\theta} \sim \mathcal{N}(\hat{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_\theta)$, the estimation cost takes the form

$$\begin{aligned}\mathcal{J}_{\text{est}}(\boldsymbol{\theta}, \mathbf{x}_1) = \frac{1}{2} \sum_{t=1}^{T_{\text{est}}} &\|\mathbf{x}_{t+1} - \mathbf{F}(\mathbf{x}_t, \mathbf{u}_t; \boldsymbol{\theta})\|^2_{\boldsymbol{\Sigma}_w^{-1}} \\ &+ \frac{1}{2}\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|^2_{\boldsymbol{\Sigma}_\theta^{-1}} + \frac{1}{2}\|\mathbf{x}_1 - \hat{\mathbf{x}}_1\|^2_{\boldsymbol{\Sigma}_{\mathbf{x}_1}^{-1}}.\end{aligned} \tag{27}$$

The terms $\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|^2_{\boldsymbol{\Sigma}_\theta^{-1}}$ and $\|\mathbf{x}_1 - \hat{\mathbf{x}}_1\|^2_{\boldsymbol{\Sigma}_{\mathbf{x}_1}^{-1}}$ ensure the new estimates of the parameters and initial state do not deviate too far from the prior, while $\|\mathbf{x}_{t+1} - \mathbf{F}(\mathbf{x}_t, \mathbf{u}_t; \boldsymbol{\theta})\|^2_{\boldsymbol{\Sigma}_w^{-1}}$ ensures the predicted state trajectory matches the actual observed states. This cost function setup allows PDDP to be used to identify the true system parameters as well as an updated estimate of the true state. The observed state trajectory is assumed to be given, and the parameters are updated to match the model predictions to the observed states through minimization of Eq. (27).

The time horizon T_{est} corresponds to the estimation horizon, i.e. the number of steps in the history to estimate. In practice, the horizon is shifted forward every step when the number of observed states exceeds the time horizon T_{est} to maintain computational tractability, particularly in realtime applications. This approach is why MHE is often referred to as the dual problem to MPC [8], as the same receding horizon approach is adopted to solve realtime tasks using solvers such as DDP [27]. In the MPC problem, the goal is to find the sequence of controls that minimizes a finite-horizon cost function of the form

$$\mathcal{J}_{\text{mpc}}(\mathbf{U}) = \sum_{t=1}^{T_{\text{mpc}}} \mathcal{L}(\mathbf{x}_t, \mathbf{u}_t) + \phi(\mathbf{x}_{T_{\text{mpc}}+1}). \tag{28}$$

While MHE optimizes over the past horizon T_{est} up to the current state at time t, MPC plans over the future time horizon from the current time t up to T_{mpc}. In both methods, as a new state observation is made, the time horizon is shifted forward one step and the optimization proceeds again. A combined cost can be derived to optimize over both tasks simultaneously using PDDP:

$$\mathcal{J}_{\text{combined}}(\mathbf{U}; \boldsymbol{\theta}, \mathbf{x}_1) = \mathcal{J}_{\text{est}}(\boldsymbol{\theta}, \mathbf{x}_1) + \mathcal{J}_{\text{mpc}}(\mathbf{U}). \tag{29}$$

This combined cost function is used in Section IV-A to find the optimal parameters of a system while solving an MPC task. At each timestep t, PDDP minimizes the MHE cost over the past time horizon $[t - T_{\text{est}}, t]$, updating the parameter estimate to best match the observed states. Simultaneously, PDDP minimizes the MPC cost function, planning over the future time horizon $[t, t+T_{\text{mpc}}]$ and generating a control update corresponding to standard DDP plus feedback based on the simultaneous update of the parameters. This approach can be interpreted as a variation of adaptive MPC [1] since the system parameters are adapted to fit the true model at every step of the MPC algorithm. The simultaneous optimization is possible due to the explicit feedback gains provided during the derivation of PDDP. This generalizes previous work by [16], which uses DDP to only solve the MHE task.

B. Switching Time Optimization

Switching time optimization (STO) is proposed by the authors of [17] and reformulates the trajectory optimization of hybrid systems into an equivalent objective that is easier to solve using iterative numerical solvers, including DDP.

A hybrid system is defined by a sequence of N modes such that for each mode $i = 1, \ldots, N$, the dynamics obey

$$\dot{\mathbf{x}} = \mathbf{f}_i(\mathbf{x}(t), \mathbf{u}(t)). \tag{30}$$

Example hybrid systems include bipedal robotic animals, which transition between modes of flight and contact with the ground while running, and UAM class vehicles, which exhibit modes of vertical takeoff and landing, fixed-wing cruise, and a transition phase between the two.

Given this fixed sequence of modes, the goal in hybrid systems trajectory optimization is to find the optimal control trajectory and the optimal set of times $\mathbf{t} = [t_1, \ldots, t_N]^\top$ that minimize the cost, with each t_i describing the terminal time of mode i. This problem can be formulated through the

continuous-time objective

$$\min_{\mathbf{u},\mathbf{t}} \sum_{i=1}^{N} \left[\int_{t_{i-1}}^{t_i} \ell_i(\mathbf{x}(t), \mathbf{u}(t))\, \mathrm{d}t + \varphi_i(\mathbf{x}(t_i)) \right], \tag{31}$$

with the dynamics transitioning accordingly through the modes defined in Eq. (30).

The authors of [17] show that this objective can be equivalently expressed as trying to find the optimal amount of time to spend in each mode through switching time optimization (STO). The dynamics in Eq. (30) and objective in Eq. (31) are reformulated over fixed time intervals of unit length, with the dynamics and running cost scaled by the amount of time that is spent in each mode. Letting $\theta_i = t_i - t_{i-1}$ and adopting the change of variable $\tau = (t - t_{i-1})/\theta_i + i - 1$ results in the time scaled dynamics

$$\dot{\mathbf{x}} = \theta_i \mathbf{f}_i(\mathbf{x}(\tau), \mathbf{u}(\tau)), \tag{32}$$

and the objective can be rewritten as

$$\min_{\mathbf{u},\boldsymbol{\theta}} \sum_{i=1}^{N} \left[\int_{i-1}^{i} \theta_i \ell_i(\mathbf{x}(\tau), \mathbf{u}(\tau))\, \mathrm{d}\tau + \varphi_i(\mathbf{x}(i)) \right]. \tag{33}$$

The equivalent discrete-time optimal control problem is given as

$$\min_{\mathbf{U},\mathbf{t}} \sum_{i=1}^{N} \left[\phi_i(\mathbf{x}_{T_i+1}) + \sum_{t=T_{i-1}+1}^{T_i} \theta_i \mathcal{L}_i(\mathbf{x}_t, \mathbf{u}_t) \right] \tag{34a}$$

$$\text{subject to } \mathbf{x}_{t+1} = \mathbf{F}_i(\mathbf{x}_t, \mathbf{u}_t; \theta_i), \tag{34b}$$

with a chosen integration step size h and $T_i = T_{i-1} + \theta_i/h$, $T_0 = 0$ denoting the number of timesteps in the horizon up to the end of mode i. This objective is a parameterized discrete-time optimal control problem that can be solved using PDDP. In this problem, the optimized parameters are the intervals of time spent in each mode θ_i. For a single mode, this objective is an approximation to a free-horizon optimal control problem, where the terminal time of the trajectory parameterizes the dynamics and can be optimized using PDDP. Similar parameterizations have been discussed in previous works such as [20, 26].

Three key improvements are introduced to stabilize the optimization using a numerical solver such as PDDP. First, the number of timesteps per mode is set to be constant, meaning timesteps do not have to be interpolated when the amount of time in a mode changes during the optimization. This improvement ensures that the convergence properties described in Section II-C hold, as changing the number of timesteps in the optimization modifies the objective, which can cause unexpected problems and results in poor convergence.

Second, an improved integration scheme is adopted for calculating the discrete-time dynamics. Using the typical forward Euler integration with fixed step size h, the discrete-time dynamics from Eq. (34b) are given as

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \theta_i \mathbf{f}_i(\mathbf{x}_t, \mathbf{u}_t) h. \tag{35}$$

Note the time scaling in the dynamics by the amount of time spent in each mode θ_i results in an equivalent scaling of the integration step size h used in the forward Euler integration, which causes inaccuracies in the dynamics, particularly as the time scale θ_i becomes large. To remedy this issue, the dynamics are integrated at a smaller underlying step size Δt for multiple substeps, which prevents numerical errors in the dynamics causing the optimization to fail.

Finally, constraints are necessary in order to ensure the time scales are always nonnegative. A popular choice to handle box control limits is the projected Newton quadratic program solver proposed in [28]. The same projected Newton solver is used when calculating the parameter feedforward gain $\mathbf{m}$ to ensure that the time scales never become negative.

IV. Experiments

In this section, PDDP is applied to two separate tasks: optimal parameter estimation and switching time optimization. The results show PDDP can solve MHE and MPC problems simultaneously, as well as optimize for time-optimal trajectories through STO.

A. Parameter Estimation

The dual optimization over the combined MHE and MPC tasks proposed in Section III-A is solved by PDDP on multiple nonlinear, underactuated systems including a cartpole, quadrotor, and ant quadruped.

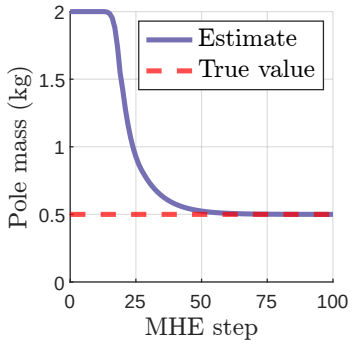
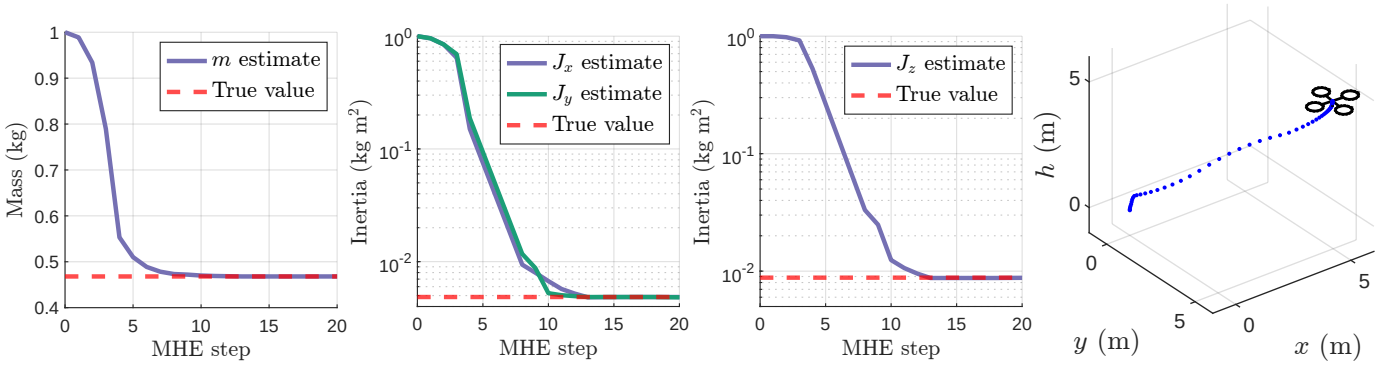


Fig. 2: Pole mass estimate for the cartpole system.

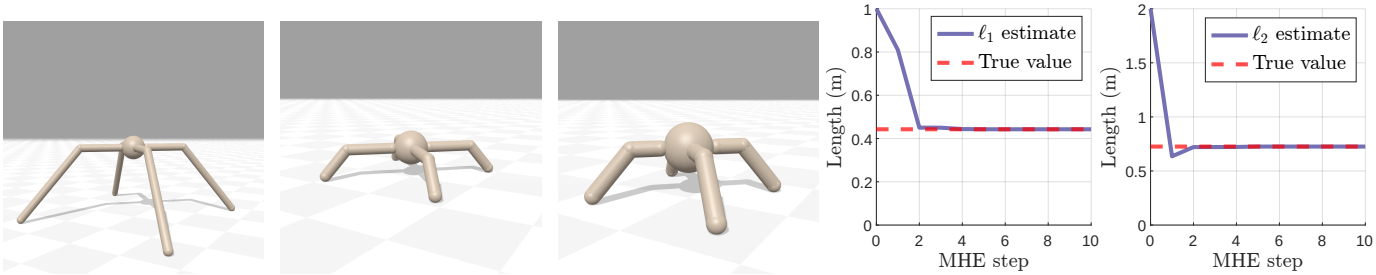
1) Cartpole: The cartpole is an underactuated, nonlinear dynamical system with four states and one control. PDDP is given a bad initial estimate of the pole mass of 2 kg in its dynamics model and is tasked with finding the correct pole mass of 0.5 kg while simultaneously bringing the pole to the upright position starting from the downward configuration. The MPC and MHE horizons are chosen to be $T_{\text{mpc}} = T_{\text{est}} = 100$ steps long, while the underlying discretization step of the environment is given as $\Delta t = 0.02$ s. The optimization is run for 200 MPC steps.

The convergence of PDDP to the true mass is given in Fig. 2. PDDP makes little progress for the first 15 steps since the dynamics of the pole evolve slowly and the true mass is hard to estimate. Once the pole velocity increases, the mass can be estimated successfully within 50 steps.



(a) Mass m estimate (b) Inertia J_x and J_y estimates (c) Inertia J_z estimate (d) Quadrotor true trajectory

Fig. 3: (a-c): Parameter estimates for the quadrotor system. PDDP converges to the true values within 15 MHE steps. (d): The spatial trajectory of the quadrotor as it arrives at the target state. Note the slow progress at the start while the dynamics model is inaccurate.



(a) Initial estimate (b) Estimate, MHE step 1 (c) Estimate, MHE step 2 (d) Upper limb length ℓ_1 (e) Lower limb length ℓ_2

Fig. 4: (a-c): Leg length estimates for the ant system, rendered using Brax [10]. The estimate after MHE step 2 very closely matches the true model. (d, e): Estimate history. PDDP converges to the true values within 4 MHE steps.

2) Quadrotor: The quadrotor has 6 degrees of freedom, resulting in twelve states $\mathbf{x} = [x, y, h, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{h}, p, q, r]^\top$ and four controls $\mathbf{u} = [F, \tau_\phi, \tau_\theta, \tau_\psi]^\top$ that correspond to the total thrust produced by the rotors, and the rolling, pitching, and yawing torques, respectively. The full description of the dynamics is given by [4].

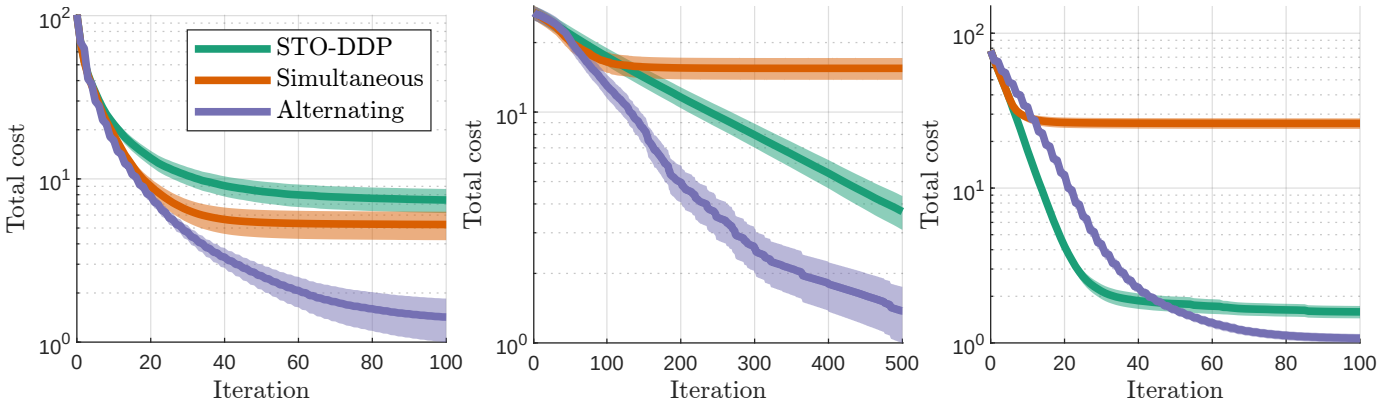
PDDP is tasked with estimating the mass and diagonal inertia terms of the quadrotor while driving the quadrotor to a target position of (5, 5, 5) m starting from the origin. It is assumed that the quadrotor is symmetric about all three axes, so the off-diagonal terms of the inertia matrix are zero. The initial guesses of the parameters are given as $m = 1\,\text{kg}$, $J_x = 1\,\text{kg}\,\text{m}^2$, $J_y = 1\,\text{kg}\,\text{m}^2$, and $J_z = 1\,\text{kg}\,\text{m}^2$, while the true parameters values are $m = 0.468\,\text{kg}$, $J_x = 4.856 \times 10^{-3}\,\text{kg}\,\text{m}^2$, $J_y = 4.856 \times 10^{-3}\,\text{kg}\,\text{m}^2$, and $J_z = 8.801 \times 10^{-3}\,\text{kg}\,\text{m}^2$, which are adapted from [2]. The initial estimate is very poor, and assumes the rotors are very heavy and/or far from the center of mass of the quadrotor. For this task, the MPC and MHE horizons are chosen as $T_\text{mpc} = T_\text{est} = 100$ steps, and the dynamics is integrated at a step size of $\Delta t = 0.01\,\text{s}$.

The results are plotted in Fig. 3. PDDP converges to the true values within 15 MHE steps, meaning PDDP is able to find the true parameters in under 0.2 s of total execution time. The MPC task is then solved using the corrected dynamics model, taking a total of 3 s to arrive at the target state.

3) Ant: The ant is a popular robotics system for continuous control and reinforcement learning. The physics used for this system is adapted from Brax [10]. This quadruped system is described by nine rigid bodies and has a state dimension of 117 in Brax, 13 states for each rigid body describing their position, quaternion rotation, linear velocity, and angular velocity. The four legs of the ant have two torque-controlled joints each, resulting in a control dimension of 8.

The MPC task is to maximize the forward velocity of the robot, requiring periodicity in the motion of the legs to generate the necessary forward force. The MPC horizon was chosen as $T_\text{mpc} = 100$ steps with an underlying environment discretization step size of $\Delta t = 0.05\,\text{s}$, resulting in a total planning horizon of 5 s. The MPC cost function rewards positive x velocity and penalizes deviation of the y position and total control effort.

The simultaneous MHE task is to estimate the length of the eight rigid bodies of the legs of the ant. Symmetry is enforced to prevent instabilities, so the legs lengths are parameterized by the upper limb length ℓ_1 and the lower limb length ℓ_2. The true values are $\ell_1 = 0.443\,\text{m}$ and $\ell_2 = 0.726\,\text{m}$, respectively, with the initial estimate given by $\ell_1 = 1.0\,\text{m}$ and $\ell_2 = 2.0\,\text{m}$,



(a) Cartpole (b) Quadrotor (c) Lift+Cruise

Fig. 5: Convergence of PDDP on three switching time tasks for different choices of optimization scheme. A normal distribution is fit to the cost at each iteration, with the dark lines corresponding to the mean and the shaded region showing the 95% confidence interval. To make the comparison clearer, the total costs for each task have been normalized so the minimum cost trajectory corresponds to a cost of 1.

resulting in vastly different dynamics.

The optimization is run for 250 total timesteps, with the parameter estimate history shown in Figs. 4d and 4e. PDDP converges to the true leg lengths in 4 steps. While the dynamics model is inaccurate, the ant makes little progress moving forward. However, once the true parameters are found, PDDP is able to successfully drive the ant forward. The fast convergence to the optimal parameters is in large part due to the choice of parameterization. Since the limb lengths ℓ_1 and ℓ_2 parameterize all four legs of the ant simultaneously, only a small subset of lengths can accurately describe the observed change in the state of all four legs.

B. Switching Time Optimization

In this section, PDDP is used to solve an STO task for three systems, finding the minimum time to bring a cartpole and quadrotor to two sequential target orientations, and solving for the optimal transition point between flight regimes for an overactuated UAM class vehicle. A numerical comparison is performed to benchmark STO-DDP from [17] with the simultaneous and alternating schemes proposed in Section II-D. The results are summarized in Fig. 5.

For each of the following experiments, the underlying environment integration step size is chosen as $\Delta t = 0.01\,\mathrm{s}$ (see discussion in Section III-B).

1) Multi-target cartpole: Starting from the origin with the pole in the downward position, the goal of the multi-target cartpole problem is to first bring the pole to the upright position at an x position of $-5\,\mathrm{m}$ with zero velocity, and then bring the pole to the upright position at an x position of $5\,\mathrm{m}$ with zero velocity.

In practice, the simultaneous optimization is highly dependent on the initial guess of the nominal control trajectory. Therefore, the method is warm started by only updating the controls for the first 5 iterations, corresponding to standard DDP.

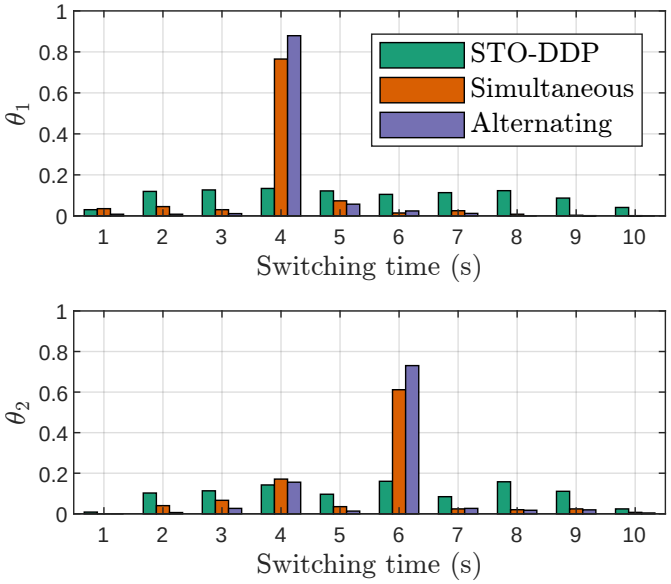


Fig. 6: Histogram of optimal time scales θ_1 and θ_2 for the multi-target cartpole task. Bins are centered at each second from 1 up to 10. From Fig. 5a, the alternating method consistently finds low cost solutions despite different initializations of the switching times, while STO-DDP and the simultaneous method converge to suboptimal solutions.

For each optimization scheme, PDDP was run for 1000 initial guesses of the time scales θ_1 and θ_2, each sampled uniformly from the interval $[1, 10]$. The convergence behavior is plotted in Fig. 5a. The alternating scheme converges to a solution half an order of magnitude lower in cost on average than both the simultaneous scheme and the STO-DDP approach that waits for the controls to converge. Note that the simultaneous scheme reduces the cost quicker than waiting for the controls to converge, but the final solutions are comparable.

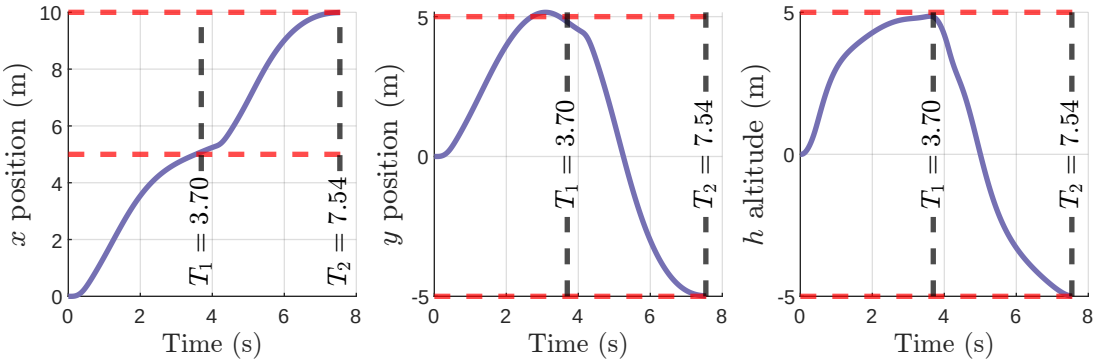


Fig. 7: Time-optimal trajectory for quadrotor STO task found using PDDP. The terminal times of each mode T_1 and T_2 are plotted as dashed black lines. The target states are plotted as dashed red lines.

Overall, the alternating scheme vastly outperforms the others on average. This success is attributed to the fact that alternating between updating the controls and updating the parameters allows the solution to more effectively escape local minima throughout the optimization process. Fig. 6 shows a histogram of the optimal time scales for each of the different optimization schemes. The simultaneous and alternating scheme converge to a solution of $\theta_1 = 4\,\mathrm{s}$ and $\theta_2 = 6\,\mathrm{s}$ in the majority of cases. Waiting for the controls to converge in STO-DDP results in a fairly flat distribution, suggesting the method quickly falls into a suboptimal solution when analyzed together with Fig. 5a. Likewise, the simultaneous method converges to similar switching times as the alternating method, but does not find an optimal control solution, describing the poor convergence in Fig. 5a. The alternating scheme is able to consistently converge in both switching times and controls, resulting in lower cost solutions.

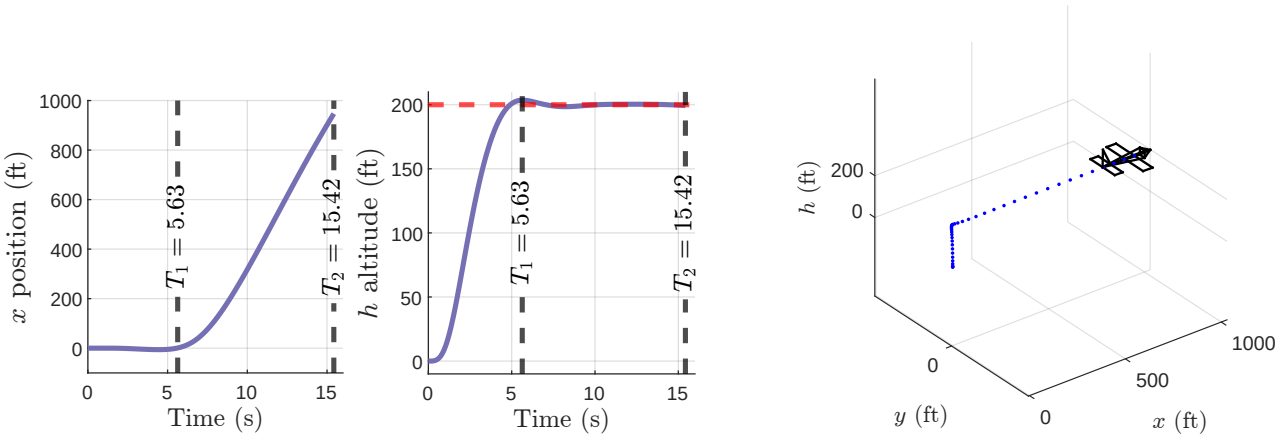
2) Quadrotor: For the quadrotor STO task, starting from the origin, the goal is to reach a first target position of $(5, 5, 5)$ m with zero velocity and then to reach a second target of $(10, -5, -5)$ m with zero velocity while optimizing for the amount of time to reach both targets. The PDDP algorithm was run until convergence for 1000 initial guesses of the switching times, sampled uniformly from the range [1, 10]. The convergence behavior is plotted in Fig. 5b. The simultaneous scheme quickly falls into local minima, while waiting for the controls to converge in STO-DDP results in linear convergence. The alternating scheme outperforms STO-DDP. The minimum cost trajectory is shown in Fig. 7.

3) NASA Lift+Cruise vehicle: The NASA Lift+Cruise vehicle is a VTOL aircraft that operates as a fixed-wing aircraft in forward flight, but has access to eight lifting rotors allowing for VTOL capabilities. While optimal control algorithms, including DDP, have been applied to these vehicles in the past [14], to the authors' best knowledge this work is the first application of switching time optimization applied to changing flight regimes within trajectory planning. The STO task performed here is a vertical takeoff followed by a partial transition from hover into cruise. PDDP was run for 500 initial guesses of the switching times sampled uniformly from [5, 15] seconds, with the convergence results of each optimization scheme plotted in Fig. 5c. On this complex task, the simultaneous scheme converges to a poor solution. Meanwhile, the alternating scheme and STO-DDP result in comparable performance, with STO-DDP beating the alternating scheme for the first 50 iterations. The alternating scheme makes little change to the switching times while the control solution is still suboptimal during the first 50 iterations, but as the controls converge to a solution that is better able to solve the problem, the alternating scheme is able to successfully find a low cost solution for the switching times. A representative time-optimal trajectory is shown in Fig. 8. The aircraft requires $5.63\,\mathrm{s}$ to perform the vertical takeoff maneuver and reach the desired altitude, then takes $9.79\,\mathrm{s}$ to increase its forward velocity to the desired target speed.

To further demonstrate the planning capabilities of PDDP, the reverse partial transition maneuver was tested. This task requires the aircraft to transition from cruise to a hover configuration by bringing its forward velocity to zero within $500\,\mathrm{ft}$, and then perform a vertical landing. The solved trajectory is shown in Fig. 9. This particular aircraft model does not have access to flaps or a speed brake to decelerate, and thus reduces its forward velocity by pitching back, pointing the body z-axis forwards to generate the backwards thrust necessary to reduce the aircraft's speed. This cruise-to-hover transition is accomplished within $6.87\,\mathrm{s}$. The aircraft then proceeds to lower its altitude down to the target, requiring $4.87\,\mathrm{s}$ to complete the landing.

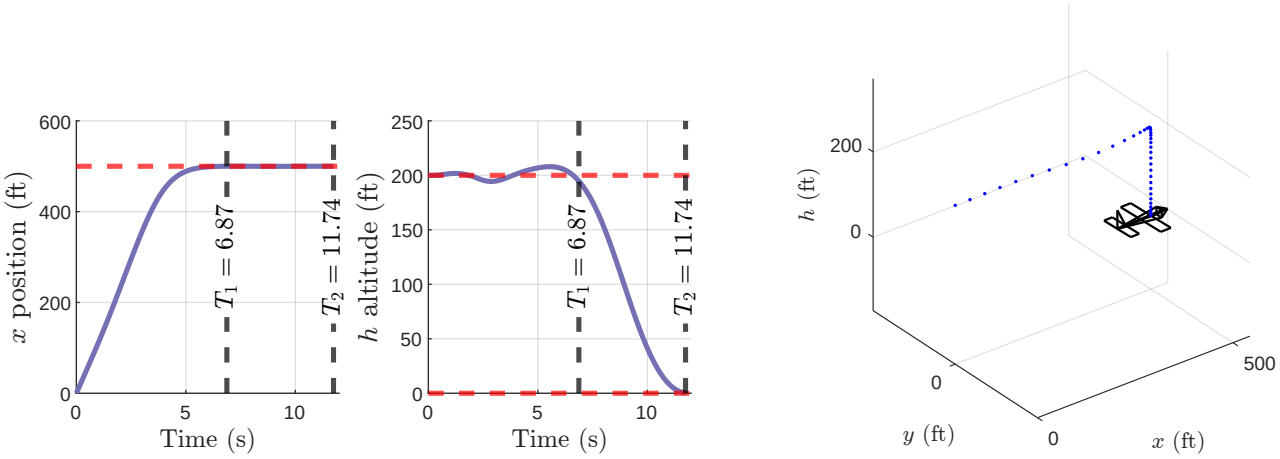
V. Conclusions and Future Work

This paper has derived a method generalizing previous work for solving problems with time-invariant parameters using DDP. The proposed PDDP algorithm is based on a general parameterized optimal control objective and allows direct optimization over time-invariant parameters with theoretical convergence guarantees. PDDP was applied to multiple robotics systems through adaptive MPC using MHE and hybrid systems optimization via STO. In particular, PDDP is able to identify the optimal transition point between flight regimes for a UAM class vehicle exhibiting complex transition



(a) x position trajectory. (b) h altitude trajectory. (c) Spatial trajectory.

Fig. 8: Time-optimal vertical takeoff to cruise maneuver for the Lift+Cruise vehicle found using PDDP. The terminal times of each mode T_1 and T_2 are plotted as dashed black lines. The target altitude is plotted as a dashed red line. PDDP increases the altitude up to the target state during the first mode, then increases the forward velocity while maintaining altitude during the second mode.



(a) x position trajectory. (b) h altitude trajectory. (c) Spatial trajectory.

Fig. 9: Time-optimal cruise to hover to vertical landing maneuver for the Lift+Cruise vehicle found using PDDP. The terminal times of each mode T_1 and T_2 are plotted as dashed black lines. The target states are plotted as dashed red lines. PDDP decreases the forward velocity of the aircraft by pitching back, then initiates a vertical landing.

dynamics. Various optimization schemes were analyzed and an alternating approach between updating the controls and parameters was shown to converge to a better solution by effectively escaping local minima throughout the optimization process.

In terms of runtime, a purely CPU-based MATLAB implementation of Algorithm 1 with limited parallelization runs the NASA Lift+Cruise STO task presented in Section IV-B3 in roughly 1.5 s per iteration (including the line search) on an Intel i7-9850H 2.60 GHz processor. The runtime is dominated by the complex dynamics calculations of the NASA Lift+Cruise vehicle, especially the dynamics Jacobians calculations. The other tasks run in under a second per iteration based on the complexity of the dynamics. This is fairly slow for an MPC algorithm, and thus improvements can be made, such as improved parallelization [23], in order to make the algorithm more suitable for realtime MPC tasks.

Additionally, as described in [20], DDP can be used as a second-order optimizer for speeding up the training process of neural networks. The addition of parameter-based optimization into this approach can help improve the training performance of modern machine learning methodologies.

References

[1] Veronica Adetola, Darryl DeHaan, and Martin Guay. Adaptive model predictive control for constrained nonlinear systems. *Systems & Control Letters*, 58(5):320–326, 2009.

[2] Nigar Ahmed and Mou Chen. Sliding mode control for quadrotor with disturbance observer. *Advances in Mechanical Engineering*, 10(7):1687814018782330, 2018.

[3] Taylor Apgar, Patrick Clary, Kevin Green, Alan Fern, and Jonathan W Hurst. Fast online trajectory optimization for the bipedal robot Cassie. In *Robotics: Science and Systems*, volume 101, page 14, 2018.

[4] Randal W Beard. Quadrotor dynamics and control. *Brigham Young University*, 19(3):46–56, 2008.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[6] Michael S Branicky. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):475–482, 1998.

[7] Jared Di Carlo, Patrick M Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[8] Moritz Diehl, Hans Joachim Ferreau, and Niels Haverbeke. *Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation*, pages 391–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[9] Jelel Ezzine and AH Haddad. Controllability and observability of hybrid systems. *International Journal of Control*, 49(6):2045–2055, 1989.

[10] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax — a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

[11] Irene M. Gregory. Urban air mobility: A control-centric approach to addressing technical challenge. IEEE FoRCE Webinar, June 14, 2021. URL http://ieeecss.org/index.php/presentation/force-webinars/urban-air-mobility-control-centric-approach-addressing-technical.

[12] Irene M. Gregory, Natasha A. Neogi, Newton H. Campbell, Jon Holbrook, Barton J. Bacon, Patrick C. Murphy, Daniel D. Moerder, Benjamin M. Simmons, Michael J. Acheson, Thomas C. Britton, and Jacob Cook. Intelligent contingency management for urban air mobility. In *AIAA Scitech Forum*, 2021. doi: 10.2514/6.2021-1000.

[13] Sven Hedlund and Anders Rantzer. Optimal control of hybrid systems. In *Proceedings of the 38th IEEE Conference on Decision and Control*, volume 4, pages 3972–3977. IEEE, 1999.

[14] Matthew D Houghton, Alexander B Oshin, Michael J Acheson, Evangelos A Theodorou, and Irene M Gregory. Path planning: Differential dynamic programming and model predictive path integral control on VTOL aircraft. In *AIAA Scitech Forum*, page 0624, 2022.

[15] David H Jacobson and David Q Mayne. *Differential dynamic programming*. Number 24. Elsevier Publishing Company, 1970.

[16] Marin Kobilarov, Duy-Nguyen Ta, and Frank Dellaert. Differential dynamic programming for optimal estimation. In *International Conference on Robotics and Automation (ICRA)*, pages 863–869. IEEE, 2015.

[17] He Li and Patrick M Wensing. Hybrid systems differential dynamic programming for whole-body motion planning of legged robots. *IEEE Robotics and Automation Letters*, 5(4):5448–5455, 2020.

[18] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229. Citeseer, 2004.

[19] Li-zhi Liao and Christine A Shoemaker. Advantages of differential dynamic programming over Newton's method for discrete-time optimal control problems. Technical report, Cornell University, 1992.

[20] Guan-Horng Liu, Tianrong Chen, and Evangelos Theodorou. Second-order neural ODE optimizer. *Advances in Neural Information Processing Systems*, 34, 2021.

[21] David Mayne. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3(1): 85–95, 1966.

[22] David E Orin, Ambarish Goswami, and Sung-Hee Lee. Centroidal dynamics of a humanoid robot. *Autonomous Robots*, 35(2):161–176, 2013.

[23] Brian Plancher and Scott Kuindersma. A performance analysis of parallel differential dynamic programming on a GPU. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 656–672. Springer, 2018.

[24] Christopher Silva, Wayne R Johnson, Eduardo Solis, Michael D Patterson, and Kevin R Antcliff. VTOL urban air mobility concept vehicles for technology development. In *Aviation Technology, Integration, and Operations Conference*, page 3847, 2018.

[25] Mark W Spong. Underactuated mechanical systems. In *Control Problems in Robotics and Automation*, pages 135–150. Springer, 1998.

[26] Kyle Stachowicz and Evangelos A Theodorou. Optimal-horizon model-predictive control with differential dynamic programming. *arXiv preprint arXiv:2111.09207*, 2021.

[27] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4906–4913. IEEE, 2012.

[28] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE, 2014.

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

APPENDIX

A. Full Q derivatives

The full Q function derivatives are given by

$$
\begin{aligned}
Q_t^0 &= \mathcal{L}_t^0 + V_{t+1}^0, \\
Q_t^x &= \mathcal{L}_t^x + (\mathbf{F}_t^x)^\top V_{t+1}^x, \\
Q_t^u &= \mathcal{L}_t^u + (\mathbf{F}_t^u)^\top V_{t+1}^x, \\
Q_t^\theta &= \mathcal{L}_t^\theta + V_{t+1}^\theta + (\mathbf{F}_t^\theta)^\top V_{t+1}^x, \\
Q_t^{xx} &= \mathcal{L}_t^{xx} + (\mathbf{F}_t^x)^\top V_{t+1}^{xx} \mathbf{F}_t^x + V_{t+1}^x \cdot \mathbf{F}_t^{xx}, \\
Q_t^{xu} &= \mathcal{L}_t^{xu} + (\mathbf{F}_t^x)^\top V_{t+1}^{xx} \mathbf{F}_t^u + V_{t+1}^x \cdot \mathbf{F}_t^{xu} = (Q_t^{ux})^\top, \\
Q_t^{uu} &= \mathcal{L}_t^{uu} + (\mathbf{F}_t^u)^\top V_{t+1}^{xx} \mathbf{F}_t^u + V_{t+1}^x \cdot \mathbf{F}_t^{uu}, \\
Q_t^{x\theta} &= \mathcal{L}_t^{x\theta} + (\mathbf{F}_t^x)^\top V_{t+1}^{x\theta} + (\mathbf{F}_t^x)^\top V_{t+1}^{xx} \mathbf{F}_t^\theta \\
&\quad + V_{t+1}^x \cdot \mathbf{F}_t^{x\theta} = (Q_t^{\theta x})^\top, \\
Q_t^{u\theta} &= \mathcal{L}_t^{u\theta} + (\mathbf{F}_t^u)^\top V_{t+1}^{x\theta} + (\mathbf{F}_t^u)^\top V_{t+1}^{xx} \mathbf{F}_t^\theta \\
&\quad + V_{t+1}^x \cdot \mathbf{F}_t^{u\theta} = (Q_t^{\theta u})^\top, \\
Q_t^{\theta\theta} &= \mathcal{L}_t^{\theta\theta} + V_{t+1}^{\theta\theta} + V_{t+1}^{\theta x} \mathbf{F}_t^\theta + (\mathbf{F}_t^\theta)^\top V_{t+1}^{x\theta} \\
&\quad + (\mathbf{F}_t^\theta)^\top V_{t+1}^{xx} \mathbf{F}_t^\theta + V_{t+1}^x \cdot \mathbf{F}_t^{\theta\theta}.
\end{aligned}
\tag{36}
$$

The terms highlighted in red denote contractions of the second-order derivative dynamics tensors with the vector V_{t+1}^x. These terms are dropped in the implementation of PDDP following iLQR [18].

B. Proof of Proposition 1

Proof: From Eqs. (13) and (15), $\delta\boldsymbol{\theta}^*$ has the following form:

$$
\begin{aligned}
\delta\boldsymbol{\theta}^* &= \epsilon \mathbf{m} \\
&= -\epsilon (\underbrace{Q_1^{\theta\theta} - Q_1^{\theta u}(Q_1^{uu})^{-1} Q_1^{u\theta}}_{V_1^{\theta\theta}})^{-1} \\
&\quad \times (\underbrace{Q_1^{\theta} - Q_1^{\theta u}(Q_1^{uu})^{-1} Q_1^{u}}_{V_1^{\theta}}) \\
&= -\epsilon (V_1^{\theta\theta})^{-1} V_1^\theta.
\end{aligned}
\tag{37}
$$

V_1^θ and $V_1^{\theta\theta}$ are the gradient and Hessian of the value function with respect to the parameters at the first timestep. The step in Eq. (37) corresponds exactly to the direction given by an iteration of Newton's method for minimizing the value function at the initial time with respect to the parameters. ■

C. Proof of Lemma 1

Proof: Using the expression for the cost function $\mathcal{J}$ given in Eq. (2),

$$
\begin{aligned}
\nabla_{\mathbf{u}_t} \mathcal{J} &= \nabla_{\mathbf{u}_t} \left[\sum_{i=1}^{T} \mathcal{L}(\mathbf{x}_i, \mathbf{u}_i; \boldsymbol{\theta}) + \phi(\mathbf{x}_{T+1}; \boldsymbol{\theta}) \right] \\
&= \mathcal{L}_t^u + \left((\mathcal{L}_{t+1}^x)^\top \mathbf{F}_t^u \right)^\top + \dots \\
&\quad + \left((\mathcal{L}_T^x)^\top \mathbf{F}_{T-1}^x \dots \mathbf{F}_{t+1}^x \mathbf{F}_t^u \right)^\top \\
&\quad + \left((\phi_{T+1}^x)^\top \mathbf{F}_T^x \mathbf{F}_{T-1}^x \dots \mathbf{F}_t^u \right)^\top \\
&= \mathcal{L}_t^u + (\mathbf{F}_t^u)^\top \Big(\mathcal{L}_{t+1}^x + (\mathbf{F}_{t+1}^x)^\top \mathcal{L}_{t+2}^x + \dots \\
&\qquad + (\mathbf{F}_{t+1}^x)^\top \dots (\mathbf{F}_{T-1}^x)^\top \mathcal{L}_T^x \\
&\qquad + (\mathbf{F}_{t+1}^x)^\top \dots (\mathbf{F}_T^x)^\top \phi_{T+1}^x \Big) \\
&= \mathcal{L}_t^u + (\mathbf{F}_t^u)^\top \\
&\quad \times \Bigg(\underbrace{\mathcal{L}_{t+1}^x + (\mathbf{F}_{t+1}^x)^\top \Bigg(\underbrace{\mathcal{L}_{t+2}^x + \dots + (\mathbf{F}_{t+2}^x)^\top \dots (\mathbf{F}_T^x)^\top \phi_{T+1}^x}_{\eta_{t+2}} \Bigg)}_{\eta_{t+1}} \Bigg) \\
&= \mathcal{L}_t^u + (\mathbf{F}_t^u)^\top \eta_{t+1}.
\end{aligned}
$$

This shows Eq. (17) of Lemma 1 is true. ■

D. Proof of Lemma 2

Proof: From the definition of $\delta\boldsymbol{\theta}$ in Eq. (15),

$$
\delta\boldsymbol{\theta} = \epsilon \mathbf{m} = O(\epsilon).
$$

Now, Eq. (19) is proven true by induction. Starting with $t = 1$,

$$
\begin{aligned}
\delta\mathbf{u}_1 &= \epsilon \mathbf{k}_1 + \mathbf{K}_1 \underbrace{\delta\mathbf{x}_1}_{=0} + \underbrace{\mathbf{M}_1 \delta\boldsymbol{\theta}}_{=O(\epsilon)} = O(\epsilon), \\
\delta\mathbf{x}_2 &= \mathbf{x}_2 - \bar{\mathbf{x}}_2 \\
&= \mathbf{F}(\bar{\mathbf{x}}_1, \bar{\mathbf{u}}_1 + \delta\mathbf{u}_1; \bar{\boldsymbol{\theta}} + \delta\boldsymbol{\theta}) - \mathbf{F}(\bar{\mathbf{x}}_1, \bar{\mathbf{u}}_1; \bar{\boldsymbol{\theta}}) \\
&= \mathbf{F}_1^u \delta\mathbf{u}_1 + \mathbf{F}_1^\theta \delta\boldsymbol{\theta} + \underbrace{O(\|\delta\mathbf{u}_1\|_2^2 + \|\delta\boldsymbol{\theta}\|_2^2)}_{=O(\epsilon^2)} = O(\epsilon).
\end{aligned}
$$

This shows Eq. (19) is true for $t = 1$. Now, assume Eq. (19) holds for $t = i$, namely $\delta\mathbf{u}_i = O(\epsilon)$ and $\delta\mathbf{x}_{i+1} = O(\epsilon)$. Then, for $t = i + 1$,

$$
\begin{aligned}
\delta\mathbf{u}_{i+1} &= \epsilon \mathbf{k}_{i+1} + \mathbf{K}_{i+1} \delta\mathbf{x}_{i+1} + \mathbf{M}_{i+1} \delta\boldsymbol{\theta} = O(\epsilon), \\
\delta\mathbf{x}_{i+2} &= \mathbf{F}_{i+1}^x \delta\mathbf{x}_{i+1} + \mathbf{F}_{i+1}^u \delta\mathbf{u}_{i+1} + \mathbf{F}_{i+1}^\theta \delta\boldsymbol{\theta} \\
&\quad + O(\|\delta\mathbf{x}_{i+1}\|_2^2 + \|\delta\mathbf{u}_{i+1}\|_2^2 + \|\delta\boldsymbol{\theta}\|_2^2) = O(\epsilon).
\end{aligned}
$$

Therefore, by induction, Eq. (19) is true for all $t = 1, \dots, T$. ■

E. Proof of Proposition 2

Proof: To show Eq. (20) is true, it is sufficient to show

$$\sum_{t=1}^{T}(\nabla_{\mathbf{u}_t}\mathcal{J})^\top\delta\mathbf{u}_t = -\epsilon\sum_{t=1}^{T}(\lambda_t+\gamma_t\mathbf{m})+O(\epsilon^2), \tag{38}$$

with $\lambda_t = (Q_t^u)^\top(Q_t^{uu})^{-1}Q_t^u$ and $\gamma_t = (Q_t^u)^\top(Q_t^{uu})^{-1}Q_t^{u\theta}$. To show Eq. (38), it is sufficient to prove using induction that

$$\begin{aligned}\sum_{t=i}^{T}(\nabla_{\mathbf{u}_t}\mathcal{J})^\top\delta\mathbf{u}_t = &-\epsilon\sum_{t=i}^{T}(\lambda_t+\gamma_t\mathbf{m})\\ &+(V_i^x-\eta_i)^\top\delta\mathbf{x}_i+O(\epsilon^2)\end{aligned}. \tag{39}$$

Starting at $i=T$,

$$\begin{aligned}&(\nabla_{\mathbf{u}_T}\mathcal{J})^\top\delta\mathbf{u}_T\\ &=(\underbrace{\mathcal{L}_T^u+(\mathbf{F}_T^u)^\top\phi_{T+1}^x}_{Q_T^u})^\top(\epsilon\mathbf{k}_T+\mathbf{K}_T\delta\mathbf{x}_T+\mathbf{M}_T\delta\boldsymbol{\theta})\\ &=\epsilon(Q_T^u)^\top\mathbf{k}_T+(Q_T^u)^\top\mathbf{K}_T\delta\mathbf{x}_T+(Q_T^u)^\top\mathbf{M}_T\delta\boldsymbol{\theta}\\ &=-\epsilon\underbrace{(Q_T^u)^\top(Q_T^{uu})^{-1}Q_T^u}_{\lambda_T}-\epsilon\underbrace{(Q_T^u)^\top(Q_T^{uu})^{-1}Q_T^{u\theta}}_{\gamma_T}\mathbf{m}\\ &\quad+\left[\underbrace{Q_T^x+(\mathbf{K}_T)^\top Q_T^u}_{V_T^x}-Q_T^x\right]\delta\mathbf{x}_T\\ &=-\epsilon(\lambda_T+\gamma_T\mathbf{m})\\ &\quad+\left[V_T^x-(\underbrace{\mathcal{L}_T^x+(\mathbf{F}_T^x)^\top\phi_{T+1}^x}_{\eta_T})\right]^\top\delta\mathbf{x}_T\\ &=-\epsilon(\lambda_T+\gamma_T\mathbf{m})+(V_T^x-\eta_T)^\top\delta\mathbf{x}_T.\end{aligned}$$

Next, assume Eq. (39) holds for $i=k+1$, namely

$$\begin{aligned}\sum_{t=k+1}^{T}(\nabla_{\mathbf{u}_t}\mathcal{J})^\top\delta\mathbf{u}_t = &-\epsilon\sum_{t=k+1}^{T}(\lambda_t+\gamma_t\mathbf{m})\\ &+(V_{k+1}^x-\eta_{k+1})^\top\delta\mathbf{x}_{k+1}+O(\epsilon^2).\end{aligned}$$

Then, for $i=k$,

$$\begin{aligned}&\sum_{t=k}^{T}(\nabla\mathbf{u}_t\mathcal{J})^\top\delta\mathbf{u}_t=(\nabla\mathbf{u}_k\mathcal{J})^\top\delta\mathbf{u}_k+\sum_{t=k+1}^{T}(\nabla\mathbf{u}_t\mathcal{J})^\top\delta\mathbf{u}_t\\ &=(\nabla\mathbf{u}_k\mathcal{J})^\top\delta\mathbf{u}_k-\epsilon\sum_{t=k+1}^{T}(\lambda_t+\gamma_t\mathbf{m})\\ &\quad+(V_{k+1}^x-\eta_{k+1})^\top\delta\mathbf{x}_{k+1}+O(\epsilon^2)\\ &=(\mathcal{L}_k^u+(\mathbf{F}_k^u)^\top\eta_{k+1})^\top\delta\mathbf{u}_k-\epsilon\sum_{t=k+1}^{T}(\lambda_t+\gamma_t\mathbf{m})\\ &\quad+(V_{k+1}^x-\eta_{k+1})^\top\Big[\mathbf{F}_k^x\delta\mathbf{x}_k+\mathbf{F}_k^u\delta\mathbf{u}_k\\ &\quad+\underbrace{O(\|\delta\mathbf{x}_k\|^2+\|\delta\mathbf{u}_k\|^2)}_{O(\epsilon^2)}\Big]+O(\epsilon^2)\\ &=(\mathcal{L}_k^u+(\mathbf{F}_k^u)^\top\eta_{k+1})^\top\delta\mathbf{u}_k-\epsilon\sum_{t=k+1}^{T}(\lambda_t+\gamma_t\mathbf{m})\\ &\quad+(V_{k+1}^x-\eta_{k+1})^\top\mathbf{F}_k^x\delta\mathbf{x}_k\\ &\quad+(V_{k+1}^x-\eta_{k+1})^\top\mathbf{F}_k^u\delta\mathbf{u}_k+O(\epsilon^2)\\ &=(\mathcal{L}_k^u+(\mathbf{F}_k^u)^\top\big[\cancel{\eta_{k+1}-\eta_{k+1}}+V_{k+1}^x\big])^\top\delta\mathbf{u}_k\\ &\quad-\epsilon\sum_{t=k+1}^{T}(\lambda_t+\gamma_t\mathbf{m})+(V_{k+1}^x-\eta_{k+1})^\top\mathbf{F}_k^x\delta\mathbf{x}_k+O(\epsilon^2)\\ &=(\underbrace{\mathcal{L}_k^u+(\mathbf{F}_k^u)^\top V_{k+1}^x}_{Q_k^u})^\top\delta\mathbf{u}_k-\epsilon\sum_{t=k+1}^{T}(\lambda_t+\gamma_t\mathbf{m})\\ &\quad+(V_{k+1}^x-\eta_{k+1})^\top\mathbf{F}_k^x\delta\mathbf{x}_k+O(\epsilon^2)\\ &=(Q_k^u)^\top(\epsilon\mathbf{k}_k+\mathbf{K}_k\delta\mathbf{x}_k+\mathbf{M}_k\delta\boldsymbol{\theta})\\ &\quad-\epsilon\sum_{t=k+1}^{T}(\lambda_t+\gamma_t\mathbf{m})\\ &\quad+(V_{k+1}^x-\eta_{k+1})^\top\mathbf{F}_k^x\delta\mathbf{x}_k+O(\epsilon^2)\\ &=\epsilon(Q_k^u)^\top\mathbf{k}_k+(Q_k^u)^\top\mathbf{K}_k\delta\mathbf{x}_k+(Q_k^u)^\top\mathbf{M}_k\delta\boldsymbol{\theta}\\ &\quad-\epsilon\sum_{t=k+1}^{T}(\lambda_t+\gamma_t\mathbf{m})\\ &\quad+(V_{k+1}^x-\eta_{k+1})^\top\mathbf{F}_k^x\delta\mathbf{x}_k+O(\epsilon^2)\\ &=-\epsilon\underbrace{(Q_k^u)^\top(Q_k^{uu})^{-1}Q_k^u}_{\lambda_k}-\epsilon\underbrace{(Q_k^u)^\top(Q_k^{uu})^{-1}Q_k^{u\theta}}_{\gamma_k}\mathbf{m}\\ &\quad-\epsilon\sum_{t=k+1}^{T}(\lambda_t+\gamma_t\mathbf{m})+(V_{k+1}^x-\eta_{k+1})^\top\mathbf{F}_k^x\delta\mathbf{x}_k\\ &\quad+(Q_k^u)^\top\mathbf{K}_k\delta\mathbf{x}_k+O(\epsilon^2)\\ &=-\epsilon\sum_{t=k}^{T}(\lambda_t+\gamma_t\mathbf{m})+\Big[\mathcal{L}_k^x\underbrace{-\mathcal{L}_k^x-(\mathbf{F}_k^x)^\top\eta_{k+1}}_{-\eta_k}\\ &\quad+(\mathbf{F}_k^x)^\top V_{k+1}^x+\mathbf{K}_k^\top Q_k^u\Big]^\top\delta\mathbf{x}_k+O(\epsilon^2)\\ &=-\epsilon\sum_{t=k}^{T}(\lambda_t+\gamma_t\mathbf{m})\\ &\quad+\Big[\underbrace{\mathcal{L}_k^x+(\mathbf{F}_k^x)^\top V_{k+1}^x}_{Q_k^x}+\mathbf{K}_k^\top Q_k^u-\eta_k\Big]^\top\delta\mathbf{x}_k+O(\epsilon^2)\\ &=-\epsilon\sum_{t=k}^{T}(\lambda_t+\gamma_t\mathbf{m})\\ &\quad+\Big[\underbrace{Q_k^x+\mathbf{K}_k^\top Q_k^u}_{V_k^x}-\eta_k\Big]^\top\delta\mathbf{x}_k+O(\epsilon^2)\\ &=-\epsilon\sum_{t=k}^{T}(\lambda_t+\gamma_t\mathbf{m})+(V_k^x-\eta_k)^\top\delta\mathbf{x}_k+O(\epsilon^2)\end{aligned}$$

This implies, by induction, that Eq. (39) holds for all $i=T,\ldots,1$. Taking $i=1$ in Eq. (39) implies Eq. (38) is true, since $\delta\mathbf{x}_1=0$.

Finally, by Proposition 1, the form of Eq. (38) can be

expressed as

$$\sum_{t=1}^{T}(\nabla_{\mathbf{u}_t}\mathcal{J})^\top\delta\mathbf{u}_t = -\epsilon\sum_{t=1}^{T}(\lambda_t+\gamma_t\mathbf{m})+O(\epsilon^2)$$
$$= -\epsilon\sum_{t=1}^{T}\lambda_t+\epsilon\left(\sum_{t=1}^{T}\gamma_t\right)(V_1^{\theta\theta})^{-1}V_1^\theta+O(\epsilon^2).$$

This shows that there exists some ϵ sufficiently small such that Eq. (20) holds. ∎

F. Proof of Lemma 3

Proof: Starting with Eq. (7) and substituting in the expressions for $\delta\mathbf{u}_t^*$ from Eq. (14) and $\delta\boldsymbol{\theta}^*$ from Eq. (15) yields

$$\begin{aligned}
&Q(\mathbf{x}_t,\mathbf{u}_t^*;\boldsymbol{\theta}^*)\\
&\quad\approx\Big[Q_t^0+\epsilon(Q_t^u)^\top\mathbf{k}_t+\epsilon(Q_t^u)^\top\mathbf{M}_t\mathbf{m}+\epsilon(Q_t^\theta)^\top\mathbf{m}\\
&\qquad+\frac{1}{2}\epsilon^2\mathbf{k}_t^\top Q_t^{uu}\mathbf{k}_t+\epsilon^2\mathbf{k}_t^\top Q_t^{uu}\mathbf{M}_t\mathbf{m}\\
&\qquad+\frac{1}{2}\epsilon^2\mathbf{m}^\top\mathbf{M}_t^\top Q_t^{uu}\mathbf{M}_t\mathbf{m}+\epsilon^2\mathbf{k}_tQ_t^{u\theta}\mathbf{m}\\
&\qquad+\epsilon^2\mathbf{m}^\top\mathbf{M}_t^\top Q_t^{u\theta}\mathbf{m}+\frac{1}{2}\epsilon^2\mathbf{m}^\top Q_t^{\theta\theta}\mathbf{m}\Big]\\
&\qquad+\Big[(Q_t^x)^\top+(Q_t^u)^\top\mathbf{K}_t+\epsilon\mathbf{k}_t^\top Q_t^{ux}\\
&\qquad+\epsilon\mathbf{m}^\top\mathbf{M}_t^\top Q_t^{ux}+\epsilon\mathbf{m}^\top Q_t^{\theta x}+\epsilon\mathbf{k}_t^\top Q_t^{uu}\mathbf{K}_t\\
&\qquad+\epsilon\mathbf{m}^\top\mathbf{M}^\top Q_t^{uu}\mathbf{K}_t+\epsilon\mathbf{m}^\top Q_t^{\theta u}\mathbf{K}_t\Big]\delta\mathbf{x}_t\\
&\qquad+\frac{1}{2}\delta\mathbf{x}_t^\top\Big[Q_t^{xx}+2Q_t^{xu}\mathbf{K}_t+\mathbf{K}_t^\top Q_t^{uu}\mathbf{K}_t\Big]\delta\mathbf{x}_t,
\end{aligned}$$

where the zero-, first-, and second-order terms in $\delta\mathbf{x}_t$ have been grouped together. Equating like powers in Eq. (6) gives

$$\begin{aligned}
V_t^0 &= Q_t^0+\epsilon(Q_t^u)^\top\mathbf{k}_t+\epsilon(Q_t^u)^\top\mathbf{M}_t\mathbf{m}+\epsilon(Q_t^\theta)^\top\mathbf{m}\\
&\quad+\frac{1}{2}\epsilon^2\mathbf{k}_t^\top Q_t^{uu}\mathbf{k}_t+\epsilon^2\mathbf{k}_t^\top Q_t^{uu}\mathbf{M}_t\mathbf{m}\\
&\quad+\frac{1}{2}\epsilon^2\mathbf{m}^\top\mathbf{M}_t^\top Q_t^{uu}\mathbf{M}_t\mathbf{m}+\epsilon^2\mathbf{k}_tQ_t^{u\theta}\mathbf{m}\\
&\quad+\epsilon^2\mathbf{m}^\top\mathbf{M}_t^\top Q_t^{u\theta}\mathbf{m}+\frac{1}{2}\epsilon^2\mathbf{m}^\top Q_t^{\theta\theta}\mathbf{m}\\
&= Q_t^0-\epsilon\underbrace{(Q_t^u)^\top(Q_t^{uu})^{-1}Q_t^u}_{\lambda_t}-\epsilon(Q_t^u)^\top(Q_t^{uu})^{-1}Q_t^{u\theta}\mathbf{m}\\
&\quad+\epsilon(Q_t^\theta)^\top\mathbf{m}+\frac{1}{2}\epsilon^2\underbrace{(Q_t^u)^\top(Q_t^{uu})^{-1}Q_t^u}_{\lambda_t}\\
&\quad+\epsilon^2(Q_t^u)^\top(Q_t^{uu})^{-1}Q_t^{u\theta}\mathbf{m}+\frac{1}{2}\epsilon^2\mathbf{m}^\top Q_t^{\theta u}(Q_t^{uu})^{-1}Q_t^{u\theta}\mathbf{m}\\
&\quad-\epsilon^2(Q_t^u)^\top(Q_t^{uu})^{-1}Q_t^{u\theta}\mathbf{m}-\epsilon^2\mathbf{m}^\top Q_t^{\theta u}(Q_t^{uu})^{-1}Q_t^{u\theta}\mathbf{m}\\
&\quad+\frac{1}{2}\epsilon^2\mathbf{m}^\top Q_t^{\theta\theta}\mathbf{m}\\
&= Q_t^0-\epsilon(1-\frac{1}{2}\epsilon)\lambda_t+\epsilon\underbrace{(Q_t^\theta-Q_t^{\theta u}(Q_t^{uu})^{-1}Q_t^u)}_{V_t^\theta}{}^\top\mathbf{m}\\
&\quad+\frac{1}{2}\epsilon^2\mathbf{m}^\top\underbrace{(Q_t^{\theta\theta}-Q_t^{\theta u}(Q_t^{uu})^{-1}Q_t^{u\theta})}_{V_t^{\theta\theta}}\mathbf{m},
\end{aligned}$$

which shows the form of Eq. (21). Now let $t=1$ and substituting in the expression for $\mathbf{m}$ given in Eq. (37) yields

$$\begin{aligned}
V_1^0 &= Q_1^0-\epsilon(1-\frac{1}{2}\epsilon)\lambda_1\\
&\quad-\epsilon\underbrace{(V_1^\theta)^\top(V_1^{\theta\theta})^{-1}V_1^\theta}_{\psi}+\frac{1}{2}\epsilon^2\underbrace{(V_1^\theta)^\top(V_1^{\theta\theta})^{-1}V_1^\theta}_{\psi}\\
&= Q_1^0-\epsilon(1-\frac{1}{2}\epsilon)(\lambda_1+\psi),
\end{aligned}$$

which shows Eq. (22). ∎

G. Proof of Proposition 3

Proof: Let the cost of the nominal trajectory $\bar{\mathbf{x}}_t,\bar{\mathbf{u}}_t,\bar{\boldsymbol{\theta}}$ after iteration k be $\mathcal{J}^{(k)}$. The cost $\mathcal{J}^{(k+1)}$ after iteration $k+1$ when applying the optimal control and parameter update is given by V_1^0 given in Eq. (22) plus higher-order terms, namely

$$\begin{aligned}
\mathcal{J}^{(k+1)} &= V_1^0+O(\epsilon^3)\\
&= Q_1^0-\epsilon(1-\frac{1}{2}\epsilon)(\lambda_1+\psi)+O(\epsilon^3)\\
&= \mathcal{L}_1^0+V_2^0-\epsilon(1-\frac{1}{2}\epsilon)(\lambda_1+\psi)+O(\epsilon^3)\\
&= \mathcal{L}_1^0+Q_2^0-\epsilon(1-\frac{1}{2}\epsilon)\left(\sum_{t=1}^{2}\lambda_t+\psi\right)+O(\epsilon^3)\\
&= \sum_{t=1}^{2}\mathcal{L}_t^0+V_3^0-\epsilon(1-\frac{1}{2}\epsilon)\left(\sum_{t=1}^{2}\lambda_t+\psi\right)+O(\epsilon^3)\\
&\;\;\vdots\\
&= \underbrace{\sum_{t=1}^{T}\mathcal{L}_t^0+\phi_{T+1}^0}_{\mathcal{J}^{(k)}}-\epsilon(1-\frac{1}{2}\epsilon)\left(\sum_{t=1}^{T}\lambda_t+\psi\right)+O(\epsilon^3)\\
&= \mathcal{J}^{(k)}-\epsilon(1-\frac{1}{2}\epsilon)\left(\sum_{t=1}^{T}\lambda_t+\psi\right)+O(\epsilon^3).
\end{aligned}$$

Thus,

$$\begin{aligned}
\Delta\mathcal{J}^{(k+1)} &= \mathcal{J}^{(k+1)}-\mathcal{J}^{(k)}\\
&= -\epsilon(1-\frac{1}{2}\epsilon)\left(\sum_{t=1}^{T}\lambda_t+\psi\right)+O(\epsilon^3).
\end{aligned}$$

∎

H. Proof of Theorem 1

Proof: First, note that for $0 < \epsilon \leq 1$,

$$-(1 - \frac{1}{2}\epsilon) \leq -\frac{1}{2} \implies -\epsilon(1 - \frac{1}{2}\epsilon) \leq -\frac{1}{2}\epsilon.$$

Therefore, by Proposition 3, the cost reduction after the k^{th} iteration is upper bounded by

$$\Delta\mathcal{J}^{(k)} \leq -\frac{1}{2}\epsilon\left(\sum_{t=1}^{T}\lambda_t + \psi\right). \tag{40}$$

Now, by Proposition 1 and Proposition 2, there exists some $0 < \epsilon_1 \leq 1$ such that for all $0 < \epsilon \leq \epsilon_1$,

$$\sum_{t=1}^{T}(\nabla_{\mathbf{u}_t}\mathcal{J})^\top \delta\mathbf{u}_t \leq -\epsilon\sum_{t=1}^{T}\lambda_t + \epsilon\left(\sum_{t=1}^{T}\gamma_t\right)(V_1^{\theta\theta})^{-1}V_1^{\theta}$$
$$\delta\boldsymbol{\theta} = -\epsilon(V_1^{\theta\theta})^{-1}V_1^{\theta}.$$

Likewise, by Eq. (40), there exists some $0 < \epsilon_2 \leq \epsilon_1$ such that for all $0 < \epsilon \leq \epsilon_2$,

$$\Delta\mathcal{J}^{(k)} \leq -\frac{1}{2}\epsilon\left(\sum_{t=1}^{T}\lambda_t + \psi\right),$$

which implies the sequence of costs after successive iterations of PDDP is monotonically decreasing.

To proceed, it is assumed that the space of controls and parameters is compact. Thus, since $\mathcal{J}$ is a continuous function over a compact space, it is bounded, so there exists some $\mathbf{U}^*$ and $\boldsymbol{\theta}^*$ such that

$$\lim_{k\to\infty}\mathcal{J}^{(k)} = \mathcal{J}(\mathbf{U}^*;\boldsymbol{\theta}^*), \tag{41}$$

meaning the sequence of costs converge to a minimum. Finally, it can be shown that $\mathbf{U}^{(k)}$ and $\boldsymbol{\theta}^{(k)}$ converge to the minimizers.

Note that Eq. (41) implies that $\Delta\mathcal{J}^{(k)} \to 0$ as $k \to \infty$. This further implies that for all $t = 1, \ldots, T$, $\lambda_t \to 0$ and $\psi \to 0$; so

$$\begin{aligned}
\lambda_t \to 0 &\implies (Q_t^u)^\top (Q_t^{uu})^{-1} Q_t^u \to 0 \\
&\implies (Q_t^{uu})^{-1} Q_t^u = \mathbf{k}_t \to \mathbf{0} \\
\psi \to 0 &\implies (V_1^\theta)^\top (V_1^{\theta\theta})^{-1} V_1^\theta \to 0 \\
&\implies (V_1^{\theta\theta})^{-1} V_1^\theta = \mathbf{m} \to \mathbf{0}.
\end{aligned}$$

Recall $\delta\boldsymbol{\theta} = \epsilon\mathbf{m}$, so $\delta\boldsymbol{\theta} \to \mathbf{0}$ and $\boldsymbol{\theta}^{(k)} \to \boldsymbol{\theta}^*$ as $k \to \infty$. Using induction, it is proven that for all $t = 1, \ldots, T$,

$$\begin{aligned}
\delta\mathbf{u}_t &\to \mathbf{0} \\
\delta\mathbf{x}_{t+1} &\to \mathbf{0},
\end{aligned} \tag{42}$$

as $k \to \infty$.

At time $t = 1$, since the initial condition is fixed, $\delta\mathbf{x}_1 = \mathbf{0}$; thus

$$\begin{aligned}
\delta\mathbf{u}_1 &= \epsilon\mathbf{k}_1 + \mathbf{M}_1\delta\boldsymbol{\theta} \to \mathbf{0} \\
\delta\mathbf{x}_2 &= \mathbf{F}_1^x\delta\mathbf{x}_1 + \mathbf{F}_1^u\delta\mathbf{u}_1 + \mathbf{F}_1^\theta\delta\boldsymbol{\theta} \\
&\quad + O(\|\delta\mathbf{x}_1\|_2^2 + \|\delta\mathbf{u}_1\|_2^2 + \|\delta\boldsymbol{\theta}\|_2^2) \to \mathbf{0}.
\end{aligned}$$

Now, assume Eq. (42) holds for $t = i$, namely

$$\begin{aligned}
\delta\mathbf{u}_i &\to \mathbf{0} \\
\delta\mathbf{x}_{i+1} &\to \mathbf{0}.
\end{aligned}$$

Then, for time $t = i + 1$,

$$\begin{aligned}
\delta\mathbf{u}_{i+1} &= \epsilon\mathbf{k}_{i+1} + \mathbf{K}_{i+1}\delta\mathbf{x}_{i+1} + \mathbf{M}_{i+1}\delta\boldsymbol{\theta} \to \mathbf{0} \\
\delta\mathbf{x}_{i+2} &= \mathbf{F}_{i+1}^x\delta\mathbf{x}_{i+1} + \mathbf{F}_{i+1}^u\delta\mathbf{u}_{i+1} + \mathbf{F}_{i+1}^\theta\delta\boldsymbol{\theta} \\
&\quad + O(\|\delta\mathbf{x}_{i+1}\|_2^2 + \|\delta\mathbf{u}_{i+1}\|_2^2 + \|\delta\boldsymbol{\theta}\|_2^2) \to \mathbf{0}.
\end{aligned}$$

This proves Eq. (42) holds by induction, implying $\mathbf{U}^{(k)} \to \mathbf{U}^*$. ■