Path Planning: Differential Dynamic Programming and Model Predictive Path Integral Control on VTOL Aircraft

Matthew D. Houghton * and Alex Oshin†
NASA Langley Research Center, Hampton, VA, 23666, USA

Michael J. Acheson‡
NASA Langley Research Center, Hampton, VA, 23666, USA

Evangelos A. Theodorou§
Georgia Institute of Technology, Atlanta, GA, 30332, USA

Irene M. Gregory¶
NASA Langley Research Center, Hampton, VA, 23666, USA

This paper explores two optimal control approaches, widely used in robotics, to establish their viability as real-time trajectory planners for vehicle configurations envisioned for the emerging aviation sector of Urban Air Mobility (UAM). Differential Dynamic Programming (DDP) enables planning over highly nonlinear dynamics using second-order approximations along a nominal trajectory and displays quadratic convergence to a local solution. Model Predictive Path Integral Control (MPPI) is a stochastic sampling-based algorithm that can optimize for general cost criteria, including potentially highly nonlinear formulations, and supports parallel computation through the use of modern GPU hardware. In this work, DDP and MPPI were implemented using model predictive control (MPC), and the results indicate they are able to successfully transition the aircraft over different flight envelopes and generate trajectories unique to UAM vehicles.

I. Introduction

Proliferation of distributed electric propulsion (DEP) based vehicle configurations and their utilization in the emerging Urban Air Mobility (UAM) sector [1, 2] generate new challenges and opportunities for trajectory planners. UAM class vehicles typically exhibit three phases of flight — rotor-borne vertical takeoff/landing, fixed-wing like cruise, and a transition phase between the two. The interactions between multiple propulsors, wings, and flight modes result in complex, highly nonlinear flight dynamics with disparate control effectors whose efficacy changes dramatically across the flight envelope [3]. In addition to complex vehicle flight dynamics, operations in an urban environment provide their own challenges. It is highly likely that UAM vehicles will experience dynamic temporary flight restrictions as part of their normal operations as well as some degree of propulsive failures due to the novel nature of this technology. These factors require dynamic, robust trajectory replanning capabilities in order to ensure safe flight and minimize disruption to the local air traffic management.

Complex vehicle flight dynamics and a potentially crowded operational environment present interesting challenges to optimal control based planning algorithms. A variety of control surfaces and capabilities, such as wing-mounted rotors and rotating wings, presents highly nonlinear systems with complex transition dynamics, making them a compelling test bed for trajectory planning. Aside from the complexity of the dynamics, these vehicles are typically overactuated systems, meaning that multiple valid solutions exist to reach a desired state or flight configuration. The optimal flight path can be impacted by mission objectives such as energy consumption, climb out rate, etc., and can be determined by incorporating outside sensor data and other external factors into the trajectory calculation. The following work assesses

*Research Engineer, Dynamic Systems and Control Branch, Matthew.D.Houghton@nasa.gov.

†Research Engineer, Dynamic Systems and Control Branch, and Machine Learning PhD Student, School of Aerospace Engineering, Georgia Institute of Technology, alexoshin@gatech.edu.

‡Aerospace Technologist, Dynamic Systems and Control Branch, Michael.J.Acheson@nasa.gov, Member AIAA.

§Associate Professor, Georgia Institute of Technology, evangelos.theodorou@gatech.edu.

¶Senior Technologist for Advanced Control Theory and Application, Irene.M.Gregory@nasa.gov, Fellow AIAA.

Differential Dynamic Programming (DDP) and Model Predictive Path Integral Control (MPPI) as potential trajectory planners for this new class of aircraft. DDP [4] utilizes locally-quadratic models of the dynamics and cost function along a nominal trajectory to achieve quadratic convergence to a locally-optimal solution. MPPI [5] is a sampling-based algorithm that can optimize for general cost criteria, including potentially highly nonlinear formulations, in a manner that readily supports parallelized computation. Past work has shown both algorithms are effective trajectory planners for robotic systems and vehicles, including off-road vehicles and aircraft [5–7]. While DDP and MPPI have been compared in the past [8], to the best of the author's knowledge, their capabilities have not been applied to this class of vehicle with such highly nonlinear dynamics.

The paper is organized as follows. Section II discusses the nonlinear vehicle dynamics and the model used for experiments in the paper. The derivations of DDP and MPPI are summarized in Section III, including their use in an MPC fashion. Section IV explains the simulation experiments performed applying DDP and MPPI as trajectory planners on NASA's Revolutionary Vertical Lift Technologies (RVLT) Lift+Cruise UAM vehicle. Finally, Section V discusses the effectiveness of DDP and MPPI for this class of vehicle and offers potential future avenues of research.

II. Nonlinear Vehicle Dynamics and Control

Typical UAM class vehicle configurations are characterized by DEP and entail three phases of flight consisting of vertical takeoff/landing, wing-borne cruise, and transition between the two. Individual phases of flight rely on different sets of predominant control effectors. The transition phase is characterized by complex nonlinear dynamics and rapidly changing effectiveness of different controls. These hover-to-cruise vehicles have historically provided significant challenges to control system designers. Classically, these vehicles implement multiple control schemes to accommodate the various phases of flight. Moreover, these control methods have typically segmented the state variables and control effectors into groups associated with lateral and longitudinal control within each phase. The most recent example of the multi-phase vehicle, the F-35, employs such a strategy for its aerodynamic and propulsion effectors [9]. For example, body axis control states in hover consist of roll (p), pitch (q), and yaw (r) rates while cruise body axis control states consist of p, q, r, α (angle-of-attack), θ (pitch angle), and ϕ (bank angle). Similarly, the aerodynamic and propulsor effectors are segmented for lateral-directional control using a roll control system and aft nozzle lateral deflection at low speeds, while using an asymmetric horizontal tail, asymmetric trailing edge flap, symmetric rudder, roll control system and aft nozzle lateral deflection at high speeds. More recently, controls research has yielded unified control approaches across all flight phases for these vehicles [3, 10, 11] but lateral/longitudinal and control effector segmentation often still exists.

An astute reader will ask, why is the control development history of hover-to-cruise vehicles pertinent to the trajectory optimization problem (e.g., DDP and MPPI)? The answer lies in the open research question of how best to apply trajectory planning tools to UAM transition class vehicles. Since the trajectory optimization problem is at its core a controllability problem which propagates the control solution forward using vehicle dynamics, the historical control of these vehicles is directly applicable to posing the control optimization problem of trajectory optimization. Simply put, is it better to provide the control optimization access to all vehicle state and effector variables or is the historical segmentation advantageous? This research documents the first steps towards addressing this question by applying the opposite of the historical control development to the trajectory replanning problem. In particular, the trajectory algorithms are provided full access to vehicle states while varying the reliance on particular effectors (performed by designer choice of weightings) by flight phase. As noted in Section V, future research will attempt to address the question of whether or not trajectory optimization for UAM vehicles benefits from model simplification and/or the control and effector segmentation that was observed in the historical control development. The ultimate aim is to find the simplest viable model of UAM vehicles required to successfully solve the real-time trajectory replanning problem for these challenging dynamic vehicles.

This research made use of a Generalized Vehicle Simulation (GVS) developed under the NASA Transformation Tools and Technologies (TTT) Intelligent Contingency Management project incorporating the NASA Lift+Cruise vehicle (developed under the RVLT project). This high-fidelity simulation consists of a generic simulation architecture [1] which is readily adaptable to UAM class vehicles. In this research, the high-fidelity aero-propulsive modeling and full nonlinear dynamics of the Lift+Cruise vehicle of the GVS simulation were utilized for both DDP and MPPI trajectory planning. While the details of the implementation of DDP and MPPI for the Lift+Cruise vehicle are described in Section III, some basic details and assumptions are outlined here. The Lift+Cruise vehicle is a hybrid electric Vertical Take-Off and Landing (eVTOL) aircraft that operates as a fixed-wing aircraft with a pusher propeller in forward flight but has eight lifting rotors mounted on the forward wing that provide VTOL capabilities [10]. Throughout the process of DDP

Fig. 1 Rendering of the Lift+Cruise aircraft.

tuning, the effector positions, effector rates, and the effector accelerations were each tested as control variables for the vehicle. The results found that using effector rates as control variables were most effective in providing rapid solutions, enabling enforcement of effector position and rate limits while minimizing effector solution chatter. In total, DDP and MPPI are provided 24 states and 12 controls. These states are: x, y, z positions; x, y, z body velocities, denoted u, v, w; Euler angles ϕ, θ, ψ; body rates p, q, r; the deflection angle of the elevator, aileron, and rudder; the speed of the pusher propeller; and the speed of the eight lifting rotors. The controls are set as the rate of change of the elevator, aileron and rudder, the acceleration of the pusher propeller, and the acceleration of the eight lifting rotors. Fig. 1 provides a rendering of the vehicle.

In this research, the aero-propulsive dynamics and the associated derivatives for the Lift+Cruise in GVS were available via two aerodynamic models. The first model was a first-principles strip theory model that adjusted local angle-of-attack for airfoils and rotors across variable width airfoil sections. This model is a medium-fidelity model [12] which has the advantage that it can be readily adapted to any UAM class vehicle. Moreover, for this strip theory model, the full nonlinear dynamics and derivatives of the aero-propulsive forces and moments are available analytically in the North-East-Down (NED) heading frame. The second model was a statistically rigorous model generated using design of experiments (DOE) applied to computational fluid dynamics [13]. This model consists of aero-propulsive polynomial models computed to fit CFD data in segmented regions of the flight envelope which were then blended to cover the entire flight envelope. This model yields higher fidelity aero-propulsive data (requiring a more substantial database development process) and similarly provides analytic derivative models. The trajectory generation results presented in this paper use the medium-fidelity strip theory modeling. Both analytic and numerical derivatives were shown to yield identical solutions with a faster computational speed using analytic derivatives. Some limitations on the modeling in this preliminary research include a failure to model nonlinear effector-to-effector interactions (using the strip-theory model) and the absence of ground effects and landing dynamics.

III. Control Algorithms

This section introduces the trajectory optimization problem as well as the algorithms that enable planning over nonlinear dynamics. This work considers a general nonlinear discrete-time system with dynamics that evolve according to

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t), \tag{1}$$

where $\mathbf{x}_t \in \mathbb{R}^n$ is the state and $\mathbf{u}_t \in \mathbb{R}^m$ is the control at time t. Trajectories of state and control are denoted as $\mathbf{X} := \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ and $\mathbf{U} := \{\mathbf{u}_1, \ldots \mathbf{u}_{T-1}\}$, respectively, over a finite time horizon $T \in \mathbb{N}^+$.

The discrete-time trajectory optimization problem is to find the control trajectory that minimizes a cost function of the form

$$\mathcal{J}(\mathbf{U}) = \sum_{t=1}^{T-1} \mathcal{L}(\mathbf{x}_t, \mathbf{u}_t) + \phi(\mathbf{x}_T), \tag{2}$$

given an initial state $\mathbf{x}_1$ subject to dynamics in Eq. (1). The functions $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and $\phi : \mathbb{R}^n \to \mathbb{R}$ denote the running cost and terminal cost, respectively. This work compares two algorithms for nonlinear trajectory optimization, which are summarized in the following sections.

A. Differential Dynamic Programming

DDP utilizes linear or quadratic approximations of the dynamics and quadratic approximations of the cost function along a nominal trajectory and produces an update to the controls that results in a reduction of the cost. Repeating this process iteratively results in quadratic convergence to a locally-optimal nominal trajectory. DDP has been successfully applied to high-dimensional nonlinear dynamics, including robot trajectory planning [6]. This section gives a brief overview of the derivation of DDP. There is a large body of knowledge regarding DDP available in the literature. Interested readers are referred to works such as [14, 15] for a more thorough derivation and analysis of the convergence.

DDP solves the discrete-time trajectory optimization problem by rewriting the minimization of Eq. (2) in terms of the value function V describing the cost-to-go when following the optimal control sequence $\mathbf{U}^*$ from a state $\mathbf{x}_i$. Denoting the truncated control sequence from time i onwards as $\mathbf{U}_i := \{\mathbf{u}_i, \dots, \mathbf{u}_{T-1}\}$ and the cost-to-go as the partial sum of costs from time i onwards as $\mathcal{J}_i(\mathbf{x}_i, \mathbf{U}_i) := \sum_{t=i}^{T-1} \mathcal{L}(\mathbf{x}_t, \mathbf{u}_t) + \phi(\mathbf{x}_T)$, the value function is defined as

$$V(\mathbf{x}_i) := \min_{\mathbf{U}_i} \mathcal{J}(\mathbf{x}_i, \mathbf{U}_i). \tag{3}$$

Bellman's principle of optimality allows the minimization in Eq. (3) over the entire control sequence to be rewritten as a sequence of minimizations over each $\mathbf{u}_t$ proceeding backwards in time:

$$V(\mathbf{x}_i) = \min_{\mathbf{u}_i} \Big[\underbrace{\mathcal{L}(\mathbf{x}_i, \mathbf{u}_i) + V(\mathbf{x}_{i+1})}_{Q(\mathbf{x}_i, \mathbf{u}_i)} \Big], \tag{4}$$

with boundary condition $V(\mathbf{x}_T) = \phi(\mathbf{x}_T)$. This is commonly referred to as the *Bellman equation*.

Solving for the value function V in Eq. (4) for all timesteps t is equivalent to solving the optimal control problem that minimizes the cost in Eq. (2). DDP makes this optimization tractable by solving for the optimal variations in state $\delta\mathbf{x}_t$ and control $\delta\mathbf{u}_t$ that minimize the cost. It achieves this by considering quadratic approximations of the value function about a nominal state and control trajectory given by $\bar{\mathbf{x}}_t := \mathbf{x}_t - \delta\mathbf{x}_t$ and $\bar{\mathbf{u}}_t := \mathbf{u}_t - \delta\mathbf{u}_t$:

$$V(\mathbf{x}_i) \approx V_i^0 + (V_i^x)^\top \delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^\top V_i^{xx} \delta\mathbf{x}, \tag{5}$$

where the superscripts denote the partial derivatives of V evaluated at $\bar{\mathbf{x}}_i$. Taking the quadratic expansion of the Q-function defined in Eq. (4) gives

$$\delta Q(\mathbf{x}_i, \mathbf{u}_i) \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta\mathbf{x}_i \\ \delta\mathbf{u}_i \end{bmatrix}^\top \begin{bmatrix} 0 & (Q_i^x)^\top & (Q_i^u)^\top \\ Q_i^x & Q_i^{xx} & (Q_i^{ux})^\top \\ Q_i^u & Q_i^{ux} & Q_i^{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x}_i \\ \delta\mathbf{u}_i \end{bmatrix}, \tag{6}$$

where

$$Q_i^x = \mathcal{L}_i^x + (\mathbf{f}_i^x)^\top V_{i+1}^x, \tag{7a}$$
$$Q_i^u = \mathcal{L}_i^u + (\mathbf{f}_i^u)^\top V_{i+1}^x, \tag{7b}$$
$$Q_i^{xx} = \mathcal{L}_i^{xx} + (\mathbf{f}_i^x)^\top V_{i+1}^{xx} \mathbf{f}_i^x + V_{i+1}^x \cdot \mathbf{f}_i^{xx}, \tag{7c}$$
$$Q_i^{ux} = \mathcal{L}_i^{xu} + (\mathbf{f}_i^u)^\top V_{i+1}^{xx} \mathbf{f}_i^x + V_{i+1}^x \cdot \mathbf{f}_i^{ux}, \tag{7d}$$
$$Q_i^{uu} = \mathcal{L}_i^{uu} + (\mathbf{f}_i^u)^\top V_{i+1}^{xx} \mathbf{f}_i^u + V_{i+1}^x \cdot \mathbf{f}_i^{uu}. \tag{7e}$$

The right-most terms in Eqs. (7c) to (7e) denote contraction with a tensor. The second order derivatives of the dynamics are often neglected, leading to the Iterative Linear Quadratic Regulator (iLQR) algorithm [14]. In practice, ignoring these higher order terms leads to a more computationally efficient algorithm without affecting the quality of the solution [16].

After plugging Eq. (7) into Eq. (4), the optimal control updates can be solved by setting the gradient equal to zero and solving for $\delta\mathbf{u}_i^*$, which is given by

$$\delta\mathbf{u}_i^* = \mathbf{k}_i + \mathbf{K}_i \delta\mathbf{x}_i, \tag{8}$$

with

$$\mathbf{k}_i := -(Q_i^{uu})^{-1} Q_i^u, \tag{9a}$$
$$\mathbf{K}_i := -(Q_i^{uu})^{-1} Q_i^{ux}. \tag{9b}$$

Algorithm 1: Differential Dynamic Programming

Input: Nominal state and control trajectory $\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t$
while *not converged* **do**
  // Backward pass
  $V_T^0 \leftarrow \phi_T^0$
  $V_T^x \leftarrow \phi_T^x$
  $V_T^{xx} \leftarrow \phi_T^{xx}$
  for $t = T-1, \ldots, 1$ **do**
    Calculate $Q_t^x, Q_t^u, Q_t^{xx}, Q_t^{xu}, Q_t^{uu}$ according to Eqs. (7) and (12)
    if Q_t^{uu} *not invertible* **then**
      Increase regularization parameters μ_1, μ_2
      Restart backward pass
    Calculate gains $\mathbf{k}_t, \mathbf{K}_t$ according to Eq. (9)
    Calculate V_t^0, V_t^x, V_t^{xx} according to Eq. (13)
  // End backward pass
  Decrease regularization parameters μ_1, μ_2
  // Line search
  $\epsilon \leftarrow 1$
  while *cost decrease not sufficient* **do**
    // Forward pass
    $\mathbf{x}_1 \leftarrow \bar{\mathbf{x}}_1$
    for $t = 1, \ldots, T-1$ **do**
      $\delta\mathbf{x}_t \leftarrow \mathbf{x}_t - \bar{\mathbf{x}}_t$
      $\mathbf{u}_t \leftarrow \bar{\mathbf{u}}_t + \epsilon\mathbf{k}_t + \mathbf{K}_t\delta\mathbf{x}_t$, from Eq. (11)
      $\mathbf{x}_{t+1} \leftarrow \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$, from Eq. (1)
    // End forward pass
    $\epsilon \leftarrow \rho\epsilon$ // Reduce ϵ
  // End line search
  $\bar{\mathbf{x}}_t \leftarrow \mathbf{x}_t$
  $\bar{\mathbf{u}}_t \leftarrow \mathbf{u}_t$
return *optimal controls* $\mathbf{u}_t$

Substituting this optimal control variation into the quadratic expansion of the value function yields

$$V_i^0 = \mathcal{L}_i^0 + V_{i+1}^0 - \frac{1}{2}(Q_i^u)^\top (Q_i^{uu})^{-1} Q_i^u, \tag{10a}$$

$$V_i^x = Q_i^x - (Q_i^{ux})^\top (Q_i^{uu})^{-1} Q_i^u, \tag{10b}$$

$$V_i^{xx} = Q_i^{xx} - (Q_i^{ux})^\top (Q_i^{uu})^{-1} Q_i^{ux}. \tag{10c}$$

This provides equations for propagating the value function backwards in time to the initial condition with boundary conditions $V_T^0 = \phi_T^0$, $V_T^x = \phi_T^x$, and $V_T^{xx} = \phi_T^{xx}$. Solving for Eqs. (7) and (10) backwards in time up to the initial condition $\mathbf{x}_1$ at time $t = 1$ constitutes the backward pass of DDP.

After the backward pass is complete, the dynamics are forward simulated starting from the initial state and applying the locally-linear feedback policy obtained by Eq. (8) to receive a new nominal state and control trajectory. This is referred to as the forward pass of DDP. The backward and forward passes are repeated until some convergence criteria is met.

Due to the nonlinearities of the dynamics and inaccuracies in the quadratic approximation, applying the control update in Eq. (8) may increase the cost. To resolve this, a line search is performed on the feedforward gain $\mathbf{k}_i$ to ensure that the cost is reduced at every iteration:

$$\delta\mathbf{u}_i^* = \epsilon\mathbf{k}_i + \mathbf{K}_i\delta\mathbf{x}_i. \tag{11}$$

The line search parameter $0 < \epsilon \leq 1$ is initialized to 1 and iteratively reduced if the new trajectory does not reduce the cost. Additionally, regularization is employed to help in the convergence of the algorithm [17]:

$$Q_i^{xx} = \mathcal{L}_i^{xx} + (\mathbf{f}_i^x)^\top (V_{i+1}^{xx} + \mu_1 \mathbf{I}) \mathbf{f}_i^x, \tag{12a}$$
$$Q_i^{xu} = \mathcal{L}_i^{xu} + (\mathbf{f}_i^x)^\top (V_{i+1}^{xx} + \mu_1 \mathbf{I}) \mathbf{f}_i^u, \tag{12b}$$
$$Q_i^{uu} = \mathcal{L}_i^{uu} + (\mathbf{f}_i^u)^\top (V_{i+1}^{xx} + \mu_1 \mathbf{I}) \mathbf{f}_i^u + \mu_2 \mathbf{I}. \tag{12c}$$

The parameter μ_1 can be interpreted as adding a quadratic cost so that the new trajectory stays near the nominal (where the quadratic approximation is accurate), while μ_2 ensures that Q_i^{uu} is positive definite and thus invertible. These modified derivatives are used to calculate the feedforward and feedback gains in Eq. (8). However, the same cancellations of Q_i^{uu} and its inverse are not possible in Eq. (10), so the new value function derivatives are given by

$$V_i^x = Q_i^x + \mathbf{K}_i^\top Q_i^{uu} \mathbf{k}_i + \mathbf{K}_i^\top Q_i^u + (Q_i^{ux})^\top \mathbf{k}_i, \tag{13a}$$
$$V_i^{xx} = Q_i^{xx} + \mathbf{K}_i^\top Q_i^{uu} \mathbf{K}_i + \mathbf{K}_i^\top Q_i^{ux} + (Q_i^{ux})^\top \mathbf{K}_i. \tag{13b}$$

Algorithm 1 summarizes the DDP algorithm in pseudocode.

B. Model Predictive Path Integral Control

MPPI is a sampling-based approach to solving the stochastic alternative to Eq. (2) that utilizes the parallel processing capabilities of modern graphics processing units (GPUs) to propagate the full nonlinear dynamics forward for thousands of noisy trajectories perturbed from some nominal trajectory. An exponential weighted average of these perturbed trajectories based on their respective costs is used to update the nominal trajectory, and the process is repeated until convergence. This section summarizes the MPPI algorithm, and readers are referred to [8, 18] for a more thorough look at the theoretical derivation of MPPI.

In the stochastic optimal control problem, the dynamics are stochastic and the goal is to minimize the expected cost. This problem formulation has deep connections with path integral control, Bayesian inference, and statistical mechanics and can be solved through variational inference [19–21]. Since this work focuses on highly nonlinear but deterministic dynamics, stochasticity is introduced in the dynamics through the controls:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{v}_t), \tag{14}$$

where $\mathbf{v}_t \sim \mathcal{N}(\mathbf{u}_t, \Sigma)$, with $\Sigma \in \mathbb{R}^{m \times m}$ denotes the covariance of the control noise. This amounts to assuming the optimal control is a unimodal Gaussian distribution at each timestep with mean $\mathbf{u}_t^*$ and a fixed covariance Σ [21]. Thus, the goal is to find an iterative update law that moves the current control distribution $p(\mathbf{U})$ towards the optimal $p^*(\mathbf{U})$. This can be formulated through an information theoretic objective by minimizing the Kullback–Leibler (KL) divergence between the current distribution and the optimal distribution:

$$\min_{p(\mathbf{U})} \mathcal{D}_{KL}(p^*(\mathbf{U}) \parallel p(\mathbf{U})). \tag{15}$$

The authors of [18] show that the optimal distribution takes the form

$$p^*(\mathbf{U}) = \frac{\exp(-\lambda^{-1} \mathcal{J}(\mathbf{U})) p(\mathbf{U})}{\int \exp(-\lambda^{-1} \mathcal{J}(\mathbf{U})) p(\mathbf{U}) d\mathbf{U}}, \tag{16}$$

which achieves the free energy lower bound $\mathcal{F} = -\lambda \mathbb{E}_{p(\mathbf{U})}[\exp(-\lambda^{-1} \mathcal{J}(\mathbf{U}))]$ and is the solution to the Hamilton-Jacobi-Bellman (HJB) equation [19]. In general, it is impossible to sample from this optimal distribution directly, which is why an iterative update law is used to drive the controlled distribution towards the optimal by minimizing Eq. (15).

Following [8, 18, 22], this work focuses on the case where the running cost in Eq. (2) can be split into an arbitrary state cost and quadratic control cost given by

$$\mathcal{L}(\mathbf{x}_t, \mathbf{u}_t) := q(\mathbf{x}_t) + \frac{\lambda}{2} \mathbf{u}_t^\top \Sigma^{-1} \mathbf{u}_t, \tag{17}$$

where $q : \mathbb{R}^n \to \mathbb{R}$ denotes the arbitrary state running cost function and $\lambda > 0$ denotes the inverse temperature. Note that this choice of cost function allows the use of arbitrary state constraints through scaled indicator functions if the constraint is violated (e.g. the vehicle has hit an obstacle).

Algorithm 2: Model Predictive Path Integral Control

Input: Nominal control trajectory $\mathbf{u}_t$
while *not converged* **do**
  `// Parallel block`
  for $t = 1, \ldots, T-1$ **do**
    Sample $\mathbf{v}_t^{(k)} \sim \mathcal{N}(\mathbf{u}_t, \Sigma)$
    $\mathbf{x}_{t+1}^{(k)} \leftarrow \mathbf{f}(\mathbf{x}_t, \mathbf{v}_t^{(k)})$ using Eq. (1)
  Compute costs $\mathcal{J}^{(k)} \leftarrow \mathcal{J}(\mathbf{V}^{(k)})$ using Eqs. (2) and (17)
  `// End parallel block`
  $\rho \leftarrow \min_k \mathcal{J}^{(k)}$
  $\eta \leftarrow \sum_{k=1}^{K} \exp(-\frac{1}{\lambda}(\mathcal{J}^{(k)} - \rho))$
  Compute weights $w^{(k)} \leftarrow \frac{1}{\eta} \exp(-\frac{1}{\lambda}(\mathcal{J}^{(k)} - \rho))$ using Eq. (18)
  Update controls $\mathbf{u}_t \leftarrow \sum_{k=1}^{K} w^{(k)} \mathbf{v}_t^{(k)}$ using Eq. (19)
return *optimal controls* $\mathbf{u}_t$

Since it is intractable to sample from the optimal distribution directly, it is approximated using Monte-Carlo estimation and importance sampling. Given a nominal control trajectory $\mathbf{U}$, which is the current estimate of the optimal control distribution, K noisy control trajectories are sampled from $\mathcal{N}(\mathbf{u}_t, \Sigma)$, with each realization denoted $\mathbf{V}^{(k)} := \{\mathbf{v}_1^{(k)}, \ldots, \mathbf{v}_{T-1}^{(k)}\}$. The dynamics are simulated forward using these noisy control trajectories resulting in an associated cost of $\mathcal{J}^{(k)} = \mathcal{J}(\mathbf{V}^{(k)})$. Importance sampling weights are computed per sampled trajectory by

$$w^{(k)} = \frac{1}{\eta} \exp(-\lambda^{-1}(J^{(k)} - \rho)), \tag{18}$$

where the normalization constant $\eta = \sum_{k=1}^{K} \exp(-\lambda^{-1}(J^{(k)} - \rho))$ ensures the weights sum to 1. The offset $\rho := \min_k J^{(k)}$ prevents numerical underflow or overflow errors in the softmax calculation, without changing the resultant weights [7, 23].

The optimal control is given by a cost-weighted average of the sampled trajectories

$$\mathbf{u}_t^* = \sum_{k=1}^{K} w^{(k)} \mathbf{v}_t^{(k)}. \tag{19}$$

Since $\mathbf{u}_t$ was the mean of the control distribution, this provides an iterative update law for bringing the current control distribution closer to the optimal control distribution. The above process is repeated until some convergence criteria is met, often on the cost reduction attained or after some number of iterations. Algorithm 2 summarizes the MPPI algorithm in pseudocode.

Some practical issues of (18) deserve to be mentioned. Performing an exponential mapping of the costs results in trajectories far away in the cost space ending up closer to each other in the exponential space. Additionally, the hyperparameter λ helps control the smoothness of the weightings, and η can be used as an indicator of the performance of the algorithm [23]. A small η suggests that only the best trajectory is being considered, while a large η suggests that only an unweighted average is being used.

The choice of the number of trajectories K to sample is also important. Ideally, one would like to select K to be very large to approximate the optimal distribution as closely as possible. However, this is often not computationally feasible, even on modern CPU architectures. The major advantage of MPPI is that the forward propagation of the dynamics can be performed in parallel, i.e. on a GPU, since each trajectory is independent of one another. This allows MPPI to be used in real-time by taking advantage of the parallel processing capabilities of modern GPUs [5].

C. Model Predictive Control

Model Predictive Control (MPC) is an optimal control technique used in conjunction with a receding horizon that has proven to be effective for real-time control of nonlinear systems [24, 25]. The goal of this approach is to obtain a stable closed-loop control law by calculating and solving a discrete-time optimal control problem for a specified

Algorithm 3: Model Predictive Control

Input: Initial state $\mathbf{x}_1$, planning horizon H, task time horizon T
for $t = 1, \ldots, T$ **do**
  Solve for $\mathbf{u}_t, \ldots, \mathbf{u}_{t+H}$ using Algorithm 1 or Algorithm 2
  Execute $\mathbf{u}_t$ using $\mathbf{x}_{t+1} \leftarrow \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$ from Eq. (1)
  Recede horizon and initialize $\mathbf{u}_{t+H+1} = \mathbf{u}_{t+H}$

planning time horizon $H << T$. MPC offers major computational advantages over trying to optimize for the full time horizon T, since the time complexity of the optimal control solver usually grows linearly (or worse) with the length of the planning horizon.

At time t, the MPC approach consists of solving for the optimal controls $\mathbf{u}_t, \ldots, \mathbf{u}_{t+H-1}$ for the horizon $\{t, \ldots, t+H\}$, executing the first control $\mathbf{u}_t$, and observing the state $\mathbf{x}_{t+1}$. This process is then repeated for the next horizon $\{t+1, \ldots, t+1+H\}$, executing the control $\mathbf{u}_{t+1}$ and observing $\mathbf{x}_{t+2}$, and so on, until the entire task has been solved up to the terminal time T. This is why MPC is often referred to as *receding horizon control*, since the planning horizon is receded by 1 at each timestep.

When an iterative solver such as DDP or MPPI is used in conjunction with MPC, the solver is often run for just 1 iteration due to real-time computation limits, and the first control is executed in an MPC fashion. The full optimal control trajectory returned by the solver can be used to warm start the solver for the next planning horizon, with the final control given using the previous final control value, i.e. $\mathbf{u}_{t+H+1} = \mathbf{u}_{t+H}$ [26].

The MPC algorithm is summarized in Algorithm 3. This work assumes deterministic dynamics so the next state is given directly by Eq. (1), but this is usually never the case in general.

IV. Experiments and Results

The following experiments were proposed and tested to determine the general efficacy of the trajectory optimization algorithms described in Section III. The primary focus was to assess the capabilities of DDP and MPPI as trajectory planning algorithms for aircraft with highly nonlinear aerodynamics. These experiments focus on full trajectory planning and were performed on the GVS Lift+Cruise VTOL model discussed in Section II.

At high forward speeds, the rotors of the vehicle become ineffective as controls. For example, the typical fixed wing cruise speed of this vehicle model is about 220 ft/s, but at these speeds the rotors have substantial undesired edgewise rotor flow and therefore are not used [3]. Because of this, these experiments focus on lower cruise speeds of about 110 ft/s, so that the algorithms can freely use all control effectors as necessary. This allows for the possibility of discovering solutions that may not necessarily be intuitive from a traditional flight perspective.

The following experiments assume a simple quadratic running cost and quadratic terminal cost with the forms

$$\mathcal{L}(\mathbf{x}_t, \mathbf{u}_t) = (\mathbf{x}_{\text{target}} - \mathbf{x}_t)^\top \mathbf{Q} (\mathbf{x}_{\text{target}} - \mathbf{x}_t) + \mathbf{u}_t^\top \mathbf{R} \mathbf{u}_t, \tag{20}$$

$$\phi(\mathbf{x}_T) = (\mathbf{x}_{\text{target}} - \mathbf{x}_T)^\top \mathbf{Q}_f (\mathbf{x}_{\text{target}} - \mathbf{x}_T), \tag{21}$$

where $\mathbf{x}_{\text{target}}$ is the target state for the problem and $\mathbf{Q} \in \mathbb{R}^{n \times n}$, $\mathbf{R} \in \mathbb{R}^{m \times m}$, and $\mathbf{Q}_f \in \mathbb{R}^{n \times n}$ are the running state cost matrix, running control cost matrix, and terminal state cost matrix, respectively. The experiments focus on the case where $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{Q}_f$ are diagonal matrices because they are significantly easier to tune. This is a common choice throughout most prior work on DDP and MPPI. For example, see [6, 8], which use similar cost function design.

In all the following experiments, a NED reference frame is assumed. The positions of the vehicle relative to the origin are given by (x, y, z), the body velocities are denoted (u, v, w), the Euler angles (ϕ, θ, ψ), and the body rates are given as (p, q, r). In the plots, the z position and w velocity components are flipped so that they are more intuitive, i.e. z corresponds to altitude. Positions and velocities are expressed in ft and ft/s, respectively. Body angles and surfaces are given in degrees, while body rates are expressed in degrees per second. Finally, rotor and propeller speeds are expressed in rotations per minute (rpm).

A. Trajectory Planning

The first set of experiments focuses on assessing the trajectories generated by DDP. Since DDP uses first- and second-order information along a nominal trajectory, it achieves scalability which enables planning over long time

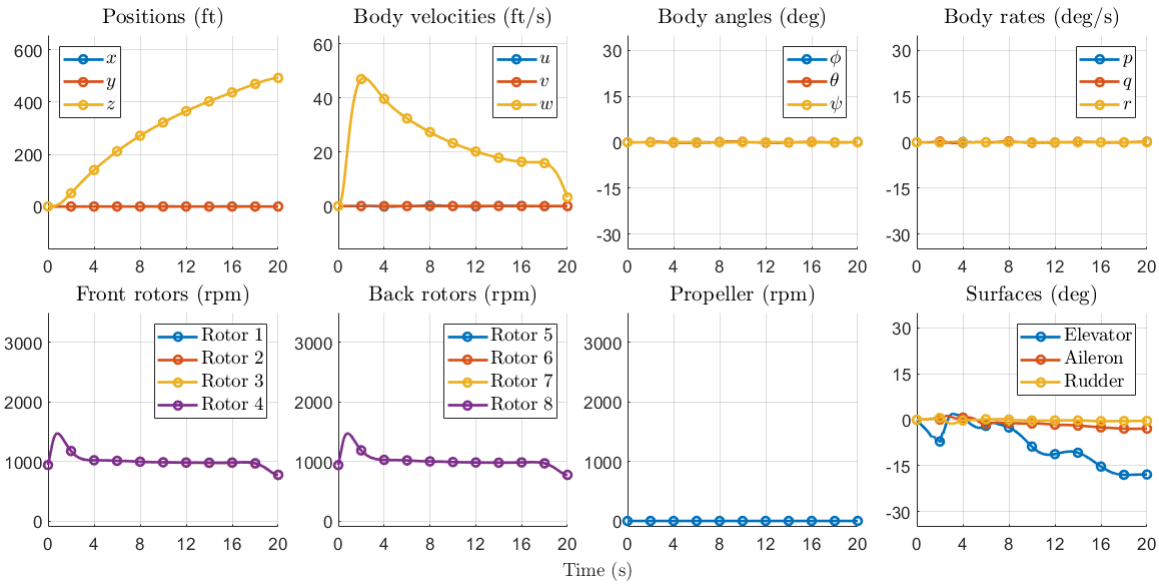


Fig. 2 Vertical ascent trajectory planned by DDP.

horizons. The results show DDP is able to successfully transition between flight regimes and perform various state reaching tasks. For these experiments, a time horizon of $T = 1000$ timesteps was used along with an integration time step of $\Delta t = 0.02$; therefore the planned trajectories are each 20 seconds long. The optimization is terminated if the cost decrease is less than 1e−4, if the regularization parameters μ_1, μ_2 exceed 1e4, or after a max number of 500 iterations.

1. Single Flight Phase Maneuvers

This subsection discusses three preliminary experiments showing the planning capabilities of DDP. These tasks do not require any transition between flight regimes. The trajectories include a vertical ascent, a banking turn, and an altitude change which can be seen in Figs. 2 to 4.

The first maneuver tested was a vertical ascent, common to rotor-based vehicles. Starting from a hover configuration at the origin, the goal is to reach a target altitude of 500 ft while minimizing deviation in the x and y position. For a straight vertical ascent, a cost is applied so that the angles, particularly pitch, remain at zero degrees. Fig. 2 shows the trajectory generated by DDP after convergence in 20 iterations. As expected, vertical acceleration is produced by applying the front and back rotors evenly, and gravity slows the aircraft down as the target altitude is reached. The elevator balances the vehicle to maintain a zero pitch angle.

The second experiment tests the ability for DDP to plan a banking turn. Starting in cruise flight at 110 ft/s and a yaw angle of zero degrees, the goal is to reach a yaw angle of 90 degrees while maintaining forward body velocity. This corresponds to a turn 90 degrees to the right. Fig. 3 shows the trajectory generated by DDP, which converged after only 5 iterations. The turn is accomplished using only the control surfaces, and the vehicle banks in a similar vein as a typical wing-borne aircraft.

The third experiment requires an altitude change of 100 ft while in cruise flight at 110 ft/s. Two different maneuvers generated by DDP are compared and shown in Fig. 4, revealing the presence of multiple solutions due to the overactuation in the vehicle. Fig. 4a shows an ascent using the rotors, where the vehicle pitches forward and uses the thrust generated by the rotors to increase altitude while maintaining its forward velocity. Meanwhile, Fig. 4b details an ascent accomplished without the rotors, where the vehicle pitches back and uses the lift of the wings and the thrust generated by the pusher propeller to increase its altitude, much like a traditional fixed-wing aircraft. Note that a particular type of altitude change maneuver can be encouraged or discouraged by tuning the cost function matrices to penalize high actuation by either the rotors or the propeller and surfaces. These trajectories also required many more iterations from DDP, converging after 123 and 161 iterations, respectively. However, most of the progress is made within the first 30 iterations, with the remainder of the time is spent iteratively improving the trajectories so that they are smoother. This is in large part due to the choice of convergence criteria as well as the regularization scheme being applied.

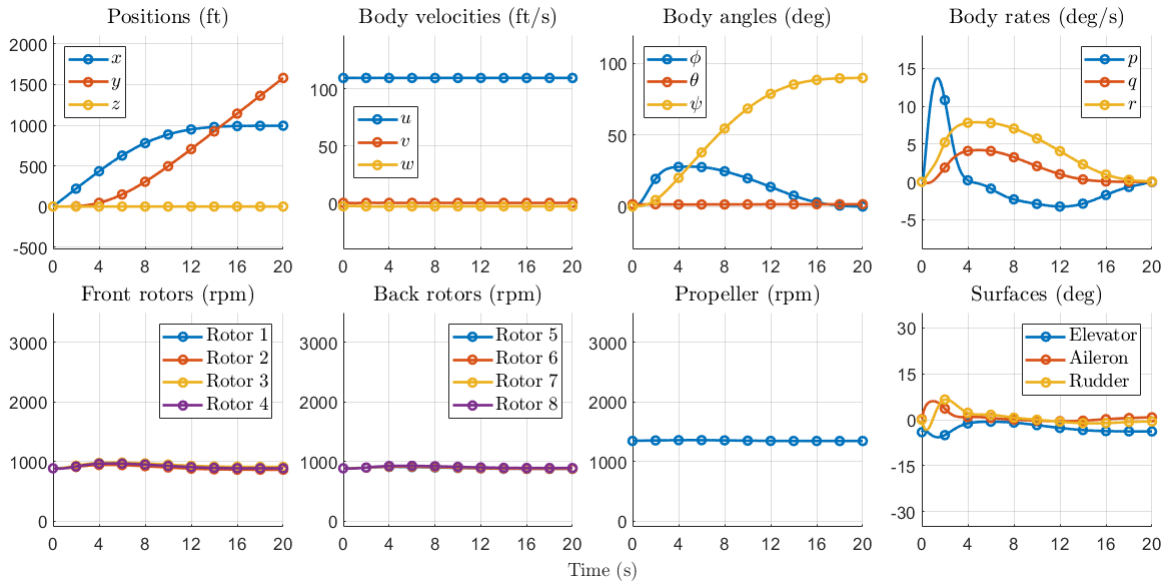


Fig. 3 Banking turn trajectory planned by DDP.

2. Partial Transition Maneuvers

This subsection presents experiments where the vehicle must transition between different flight regimes. The partial transitions include hover to cruise, hover to cruise/climb, ascent to cruise, cruise to hover, and cruise to descent. The trajectories are summarized in Figs. 6 to 10.

The first task requires a partial transition from hover into cruising at 110 ft/s. The maneuver generated by DDP (Fig. 6) accelerates using primarily the push propeller, and the desired cruise speed is reached after 10 seconds. The vehicle pitches forward slightly at the beginning of the trajectory to help accelerate forward and then corrects itself. Note that the z position deviates by 2 ft in either direction, but DDP is able to correct the vehicle so altitude is maintained. DDP hit the max number of iterations (500) to arrive at the solution. However, most of these iterations are spent achieving only a small cost reduction; the task has largely been solved after 80 iterations. Fig. 5 shows the relative contribution of the rotors and the wings to the total force on the aircraft in the z direction during the partial transition. Once the vehicle has accelerated to the target speed, roughly 10% of the load has been transferred from the rotors to the wings. This emphasizes the fact that these experiments focus on an initial partial transition from rotor-borne flight to wing-borne flight.

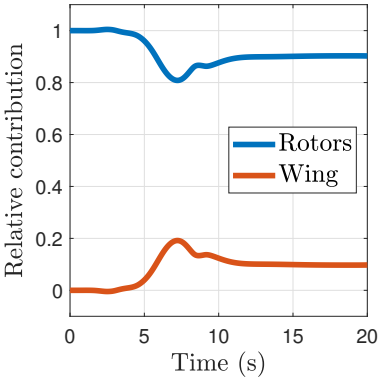
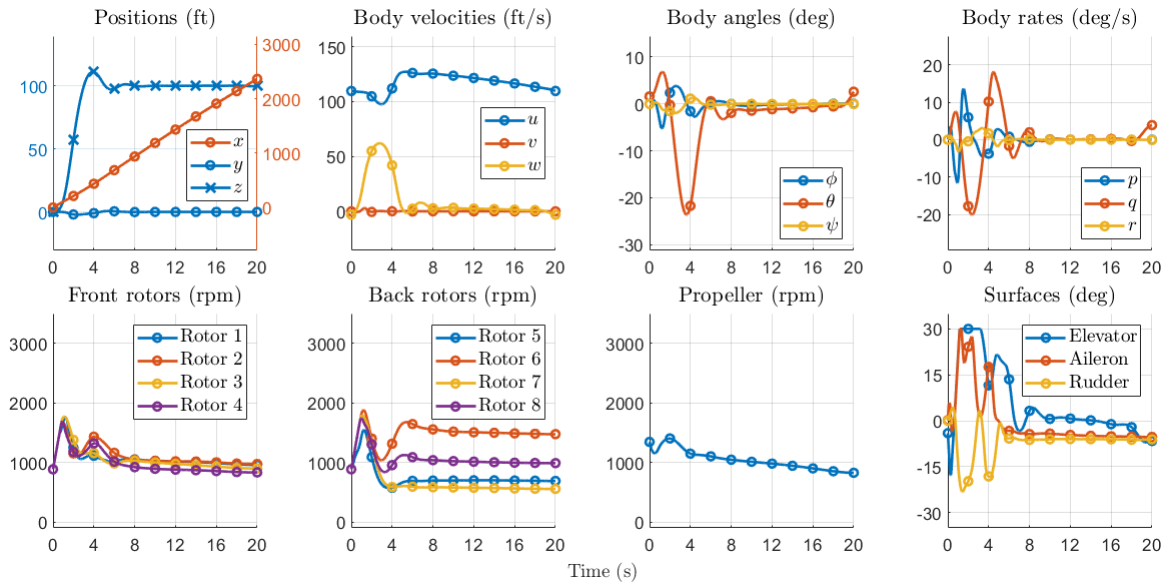


Fig. 5 Contribution to the z force during hover to cruise partial transition

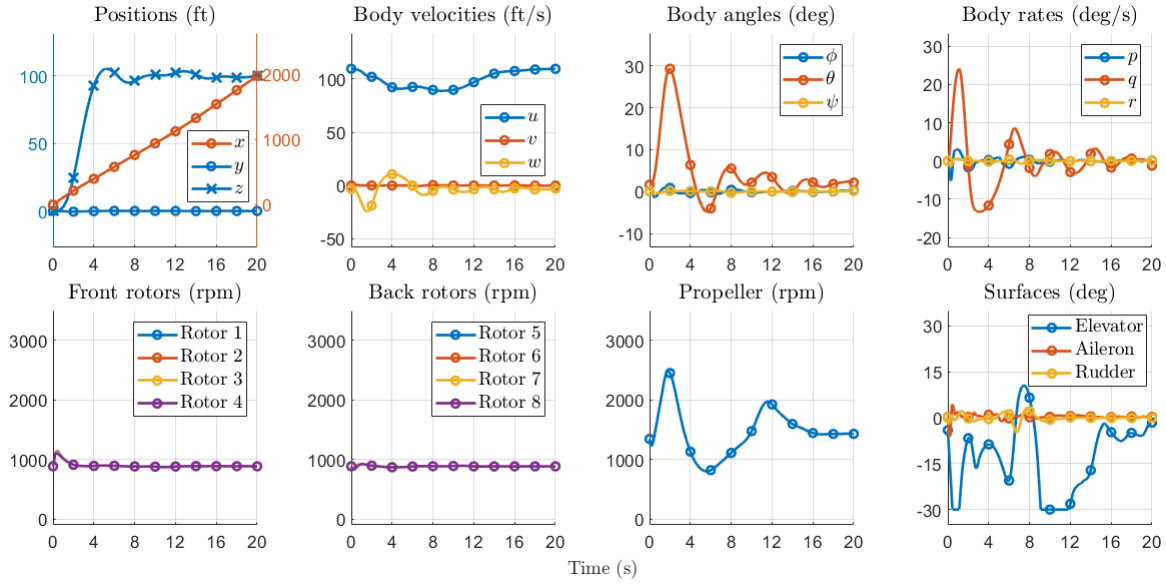
The second partial transition is similar to the first but additionally requires a climb rate of roughly 8.33 ft/s. This corresponds to a body w velocity of about 4.3 ft/s. Fig. 7 shows the trajectory generated by DDP and is very similar to the hover to cruise partial transition. Note the initial pitch forward is harsher at the beginning, and the rotors stay active at a higher rpm once the target velocities have been reached. Most of the vertical velocity is thus generated by the rotors, while the forward velocity is maintained using the push propeller and the lift produced by the wings of the aircraft. This maneuver also required the max number of iterations from DDP, but the task has been solved after only 35 iterations.

Fig. 8 shows the trajectory generated by DDP for the third task, which was an ascent into cruise partial transition. The task requires an altitude increase of 100 ft and a desired forward velocity of 110 ft/s. The trajectory is similar to the first two, except the rotors quickly apply the vertical acceleration necessary to arrive at the desired altitude within the first 8 seconds of the trajectory. Since the vehicle started in a trimmed hover configuration, the rotors have full control authority at the beginning of the time horizon. This task is solved after 33 iterations but runs for the full 500 iterations.

The final two tasks focus on the reverse partial transition from cruising at 110 ft/s into hover. The first of the two



(a) Using rotors to ascend.



(b) Without using rotors to ascend.

Fig. 4 Altitude change maneuvers planned by DDP.

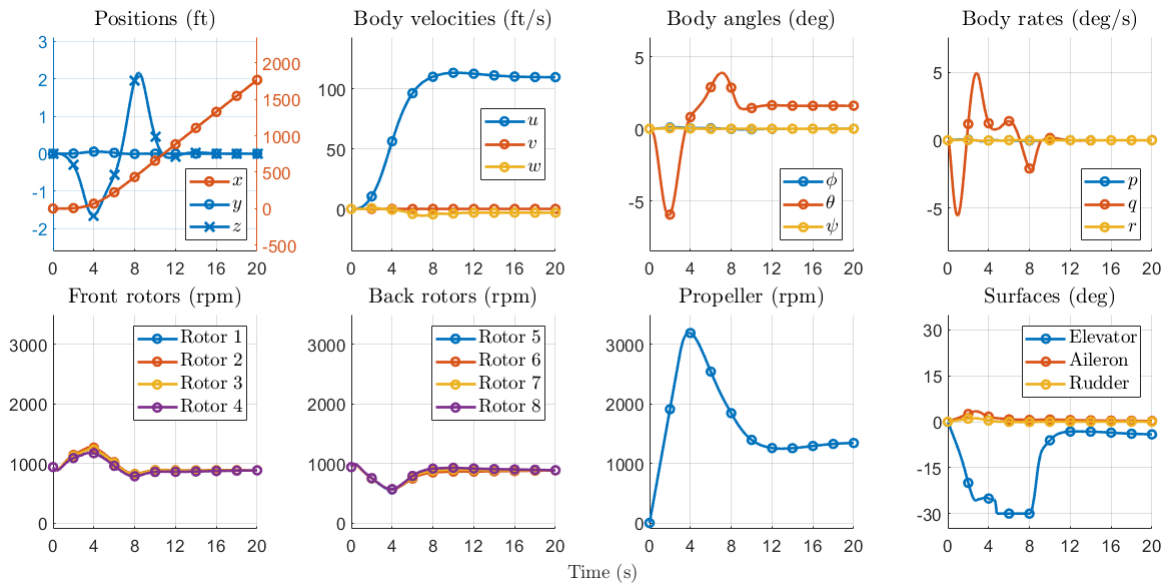


Fig. 6 Hover to cruise trajectory planned by DDP.

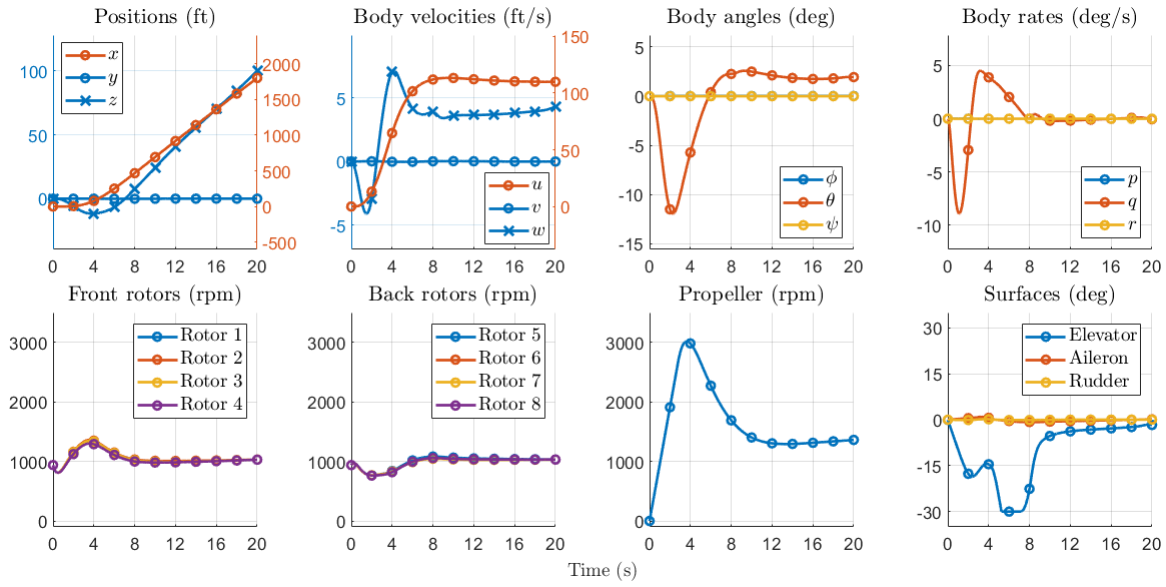


Fig. 7 Hover to cruise and climb trajectory planned by DDP.

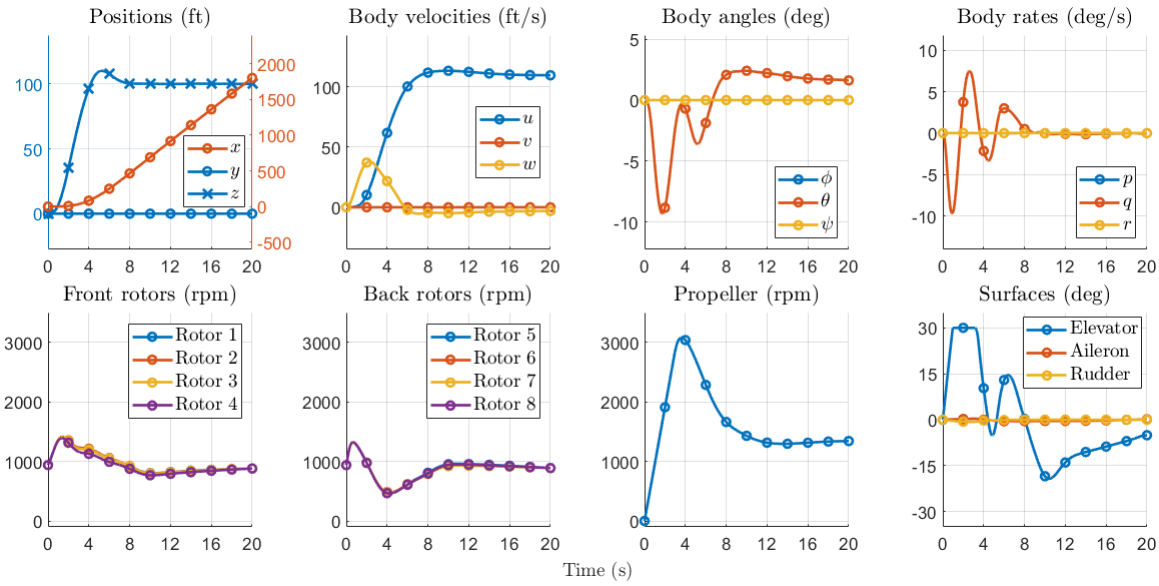


Fig. 8 Vertical ascent into cruise trajectory planned by DDP.

trajectories can be seen in Fig. 9. The vehicle slows down by pitching back so the thrust vector of the rotors points forward. Since the vehicle started in cruise, this causes it to climb for the first 6 seconds, after which it can return to the initial altitude. The second trajectory is shown in Fig. 10, and is similar to the first but requires an altitude decrease of 100 ft — akin to a short descent. Since ground effects and landing dynamics are not modeled, the desired end state is a hover configuration, and the vehicle is able to pass the zero z axis. The partial transition is similar to the first, but the vehicle first dives to the desired altitude and then performs the pitch back to reduce its speed. These trajectories were solved by DDP in 66 and 265 iterations, respectively. However, the majority of the progress is made within the first 20 iterations, with the rest of the time spent smoothing out the trajectories. It is important to note that these changes in altitude to decrease speed are required as the vehicle model does not have speed flaps or any other manner of forward deceleration.

B. Model Predictive Control

In this section, DDP and MPPI are applied as real-time trajectory planners. A subset of the experiments from Section IV.A is repeated using DDP and MPPI in tandem with the MPC algorithm given in Section III.C, and the results are presented below.

For DDP, a planning horizon of $H = 100$ and an integration time step of $\Delta t = 0.02$ is used, so each MPC step plans the next 2 seconds. The number of iterations of DDP per step is limited to 5, but often only one iteration is needed before converging, since the next step is warm started with the solution from the previous step, and the algorithm can take full advantage of the quadratic convergence properties of DDP.

For MPPI, a planning horizon of $H = 50$ is used, but the integration time step is increased to $\Delta t = 0.04$ so the total planning time matches DDP, and the number of iterations per MPC step is limited to 1. The number of samples is chosen based on the difficulty of the problem, but generally a number around 1000 is sufficient for solving the following tasks.

The first MPC experiment was the banking turn to the right. The trajectories executed by DDP and MPPI are given in Fig. 11. Both DDP and MPPI arrive at the target yaw angle of 90 degrees in roughly 16 seconds. A point of interest is that both algorithms overshoot the target, which did not happen when DDP was planning over the entire trajectory (Fig. 3). This is due to the shorter planning horizon of the MPC formulation. Since the algorithms can only plan 2 seconds into the future, they are unable to anticipate the need to correct the state earlier in the trajectory. This myopic behavior is one disadvantage of using MPC. Further discussion on the importance of the planning horizon is provided in Section V.

The second task was the altitude change of 100 ft while in cruise flight at 110 ft/s, and the results are shown in Fig. 12. Both algorithms are able to maintain the vehicle's forward velocity and successfully climb to the target altitude.

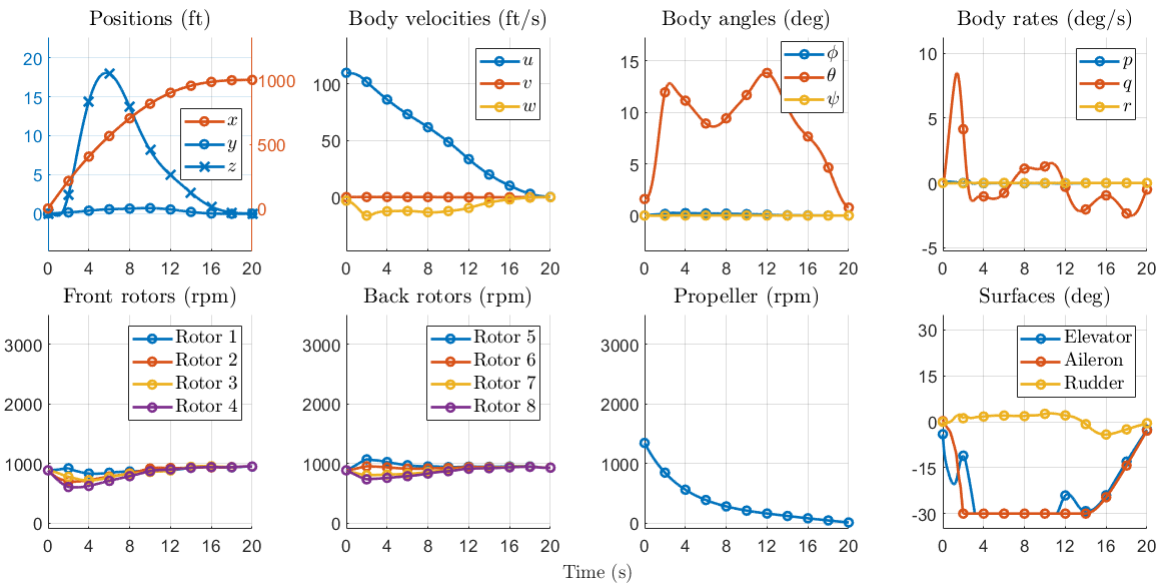


Fig. 9 Cruise to hover trajectory planned by DDP.

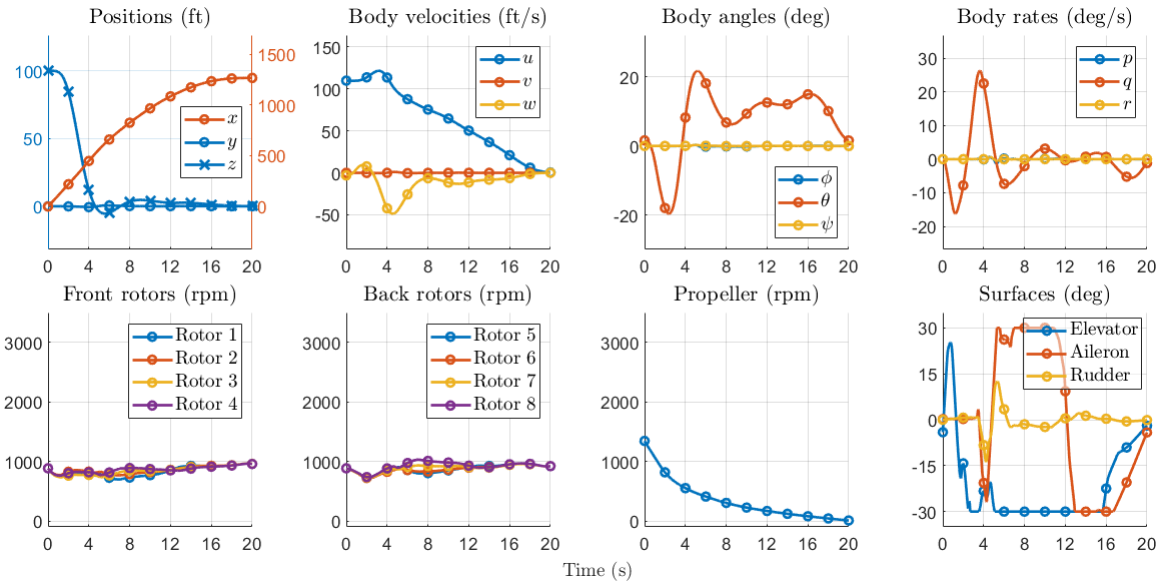
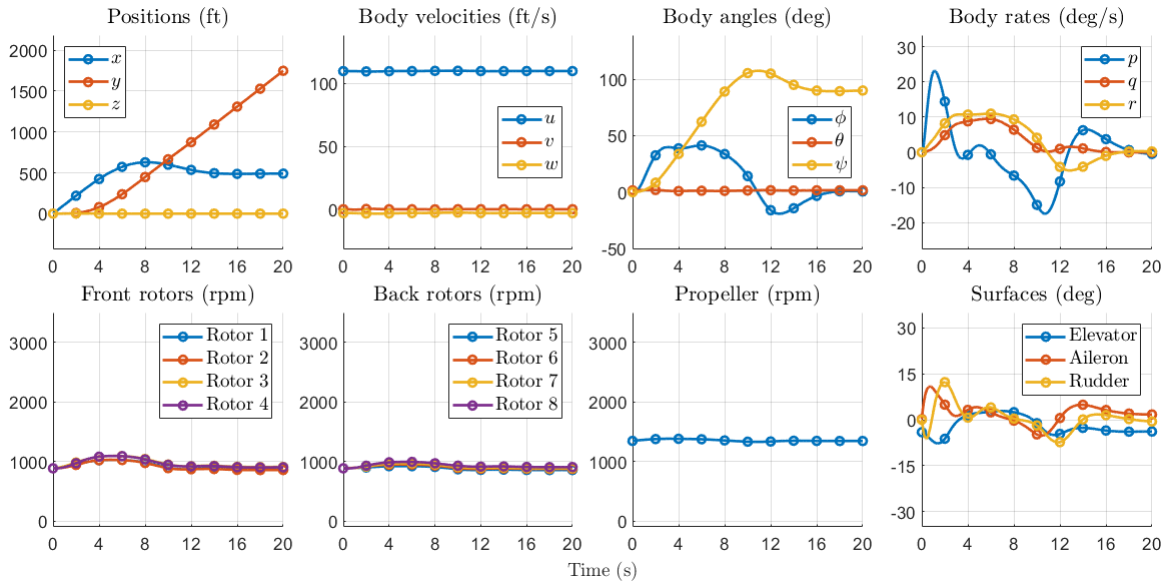


Fig. 10 Cruise to descent trajectory planned by DDP.

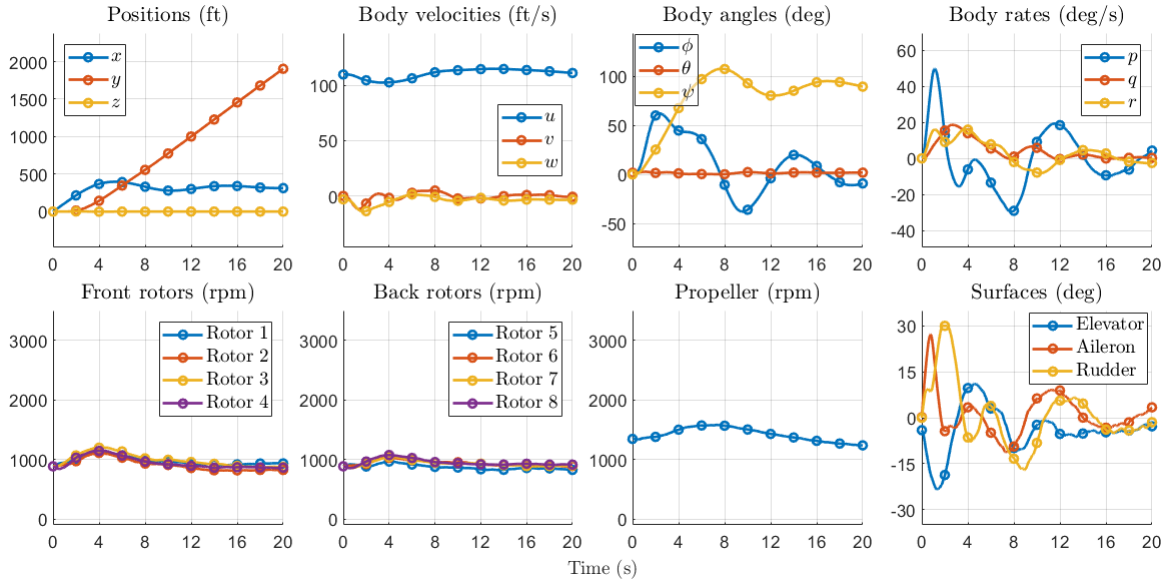
The partial transition made by MPPI is similar to the trajectory planned by DDP without using the rotors (Fig. 4b), while the DDP partial transition uses a balance of all controls. MPPI overshoots the target altitude by a large amount but is able to correct itself by the end of the trajectory. Note that both algorithms were able to solve the task in only 10 seconds.

The third experiment tested was the hover to cruise partial transition, where the goal is to reach a target forward velocity of 110 ft/s starting from hover. See Fig. 13 for the trajectories executed by DDP and MPPI. Forward speed is initially built up by pitching forward, and then the push propeller is used heavily by both algorithms to reach the target speed. The solutions are fairly equivalent, and both are able to reach the desired speed in under 10 seconds. However, DDP hits the actuator limits on the elevator and increases the propeller speed at the maximum rate given by the dynamics, so it is able to arrive at the target speed faster.

The final experiment was the reverse cruise to hover partial transition, with Fig. 14 showing the trajectories generated by the two algorithms. Here, the algorithms required 20 seconds of execution time in order to finish the maneuver. Note that the algorithms arrive at two distinct solutions. MPPI pitches back for the majority of the trajectory while roughly maintaining its altitude. This allows it to reduce its forward velocity, however, staying at the desired altitude introduces some instabilities that can be seen in the deviation of the y position by 10 ft and the non-zero angles at the terminal time. Meanwhile, DDP makes two pitch back maneuvers to slow down in two separate phases — climbing through the first pitch maneuver and then settling back down at the original altitude during the second. This is a unique partial transition and is different from the partial transition given by DDP when it was planning over the full trajectory, though a similar but much less prominent pitch maneuver can be seen in Fig. 9.

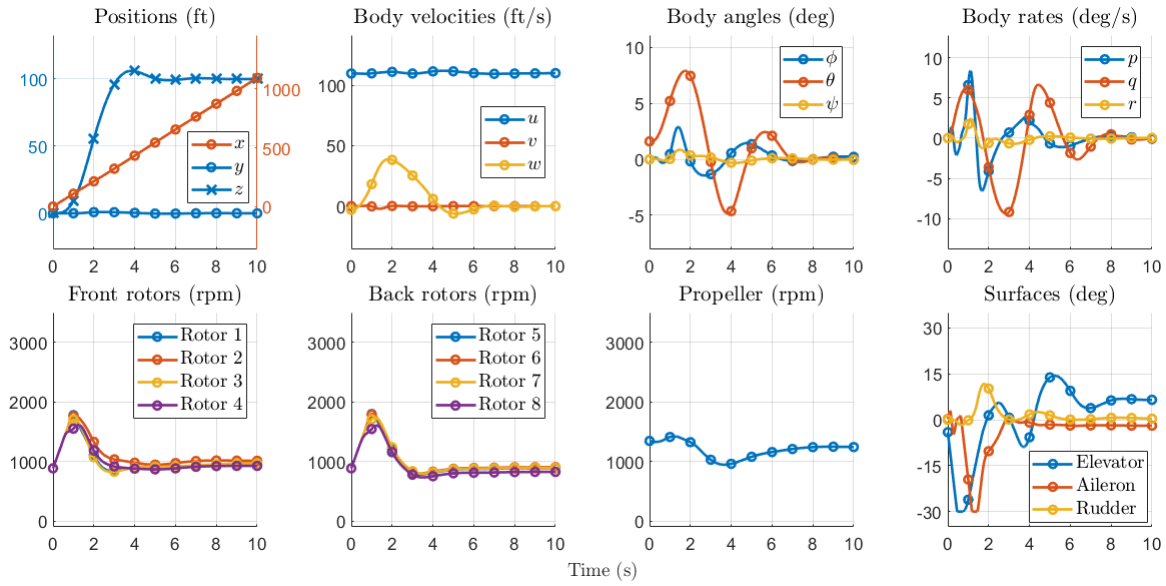


(a) DDP.

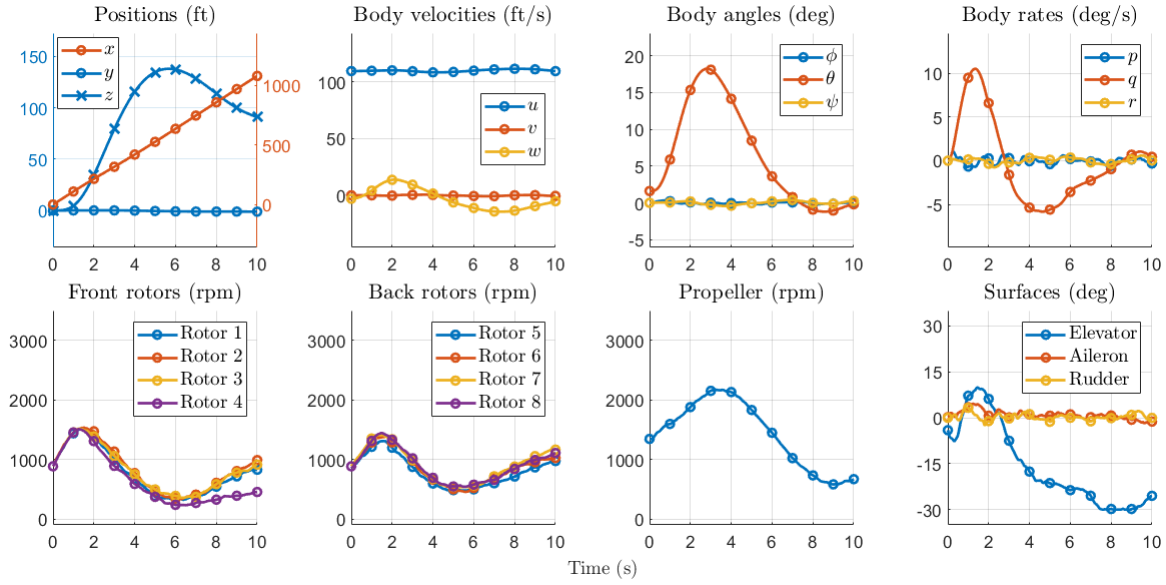


(b) MPPI.

Fig. 11 Banking turn maneuver executed by DDP and MPPI using MPC.

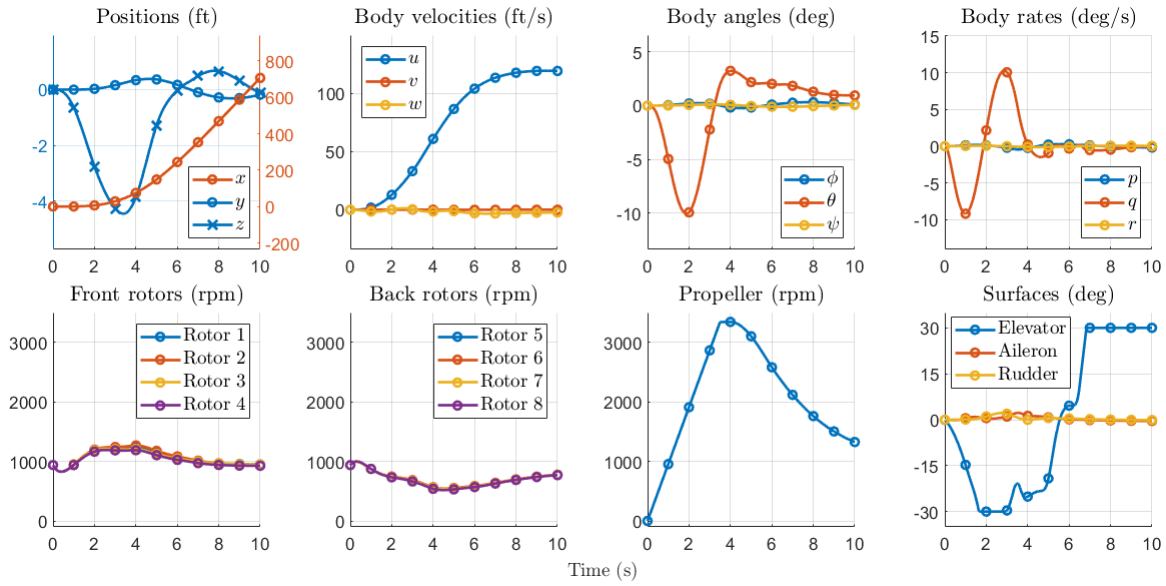


(a) DDP.

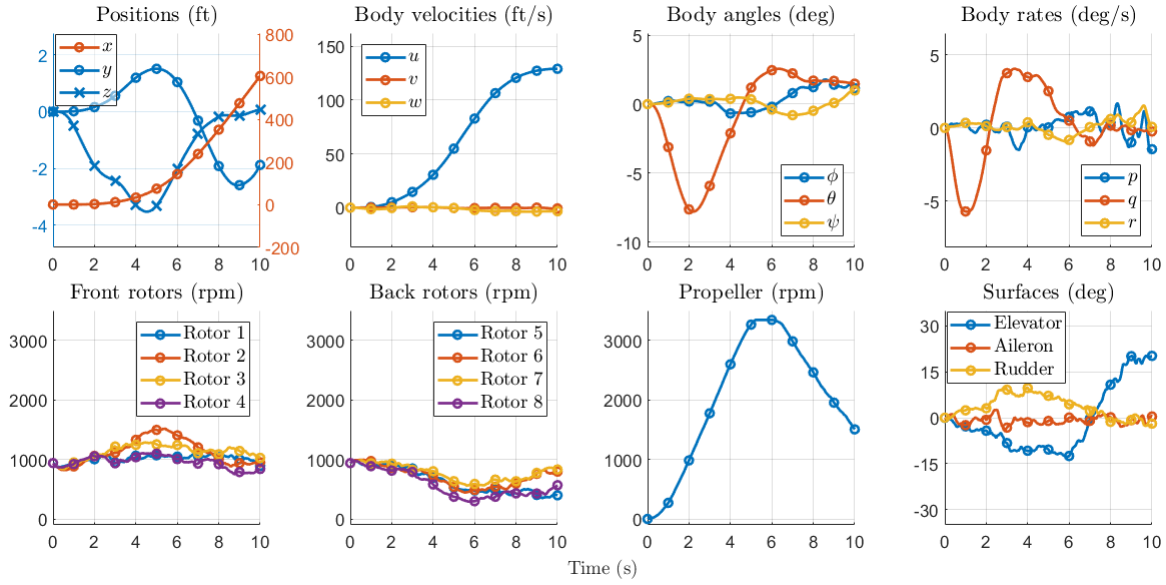


(b) MPPI.

Fig. 12 Altitude change maneuvers executed by DDP and MPPI using MPC.

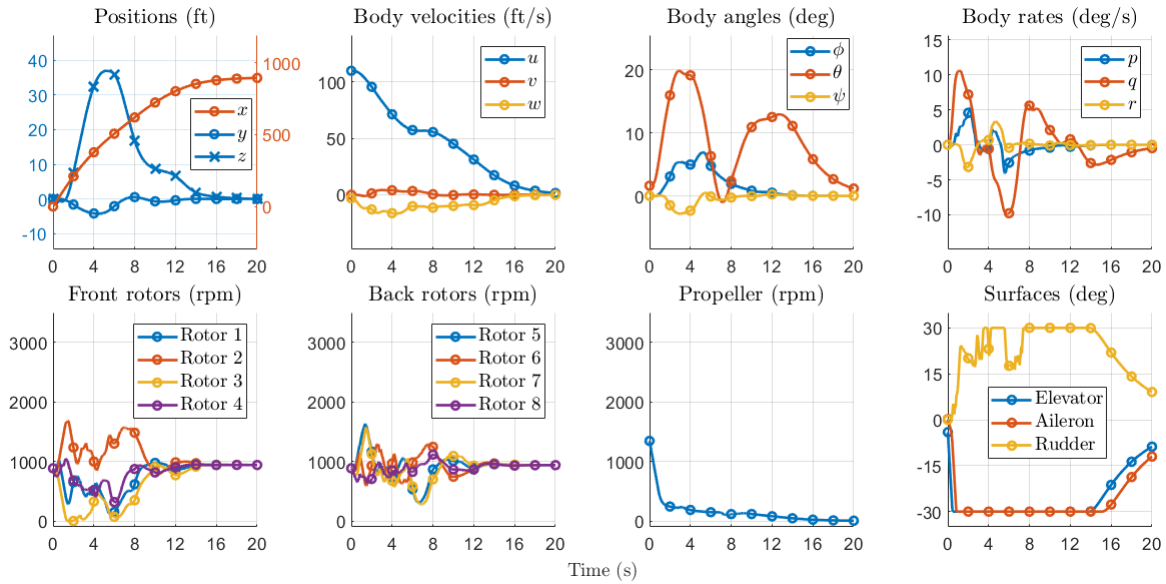


(a) DDP.

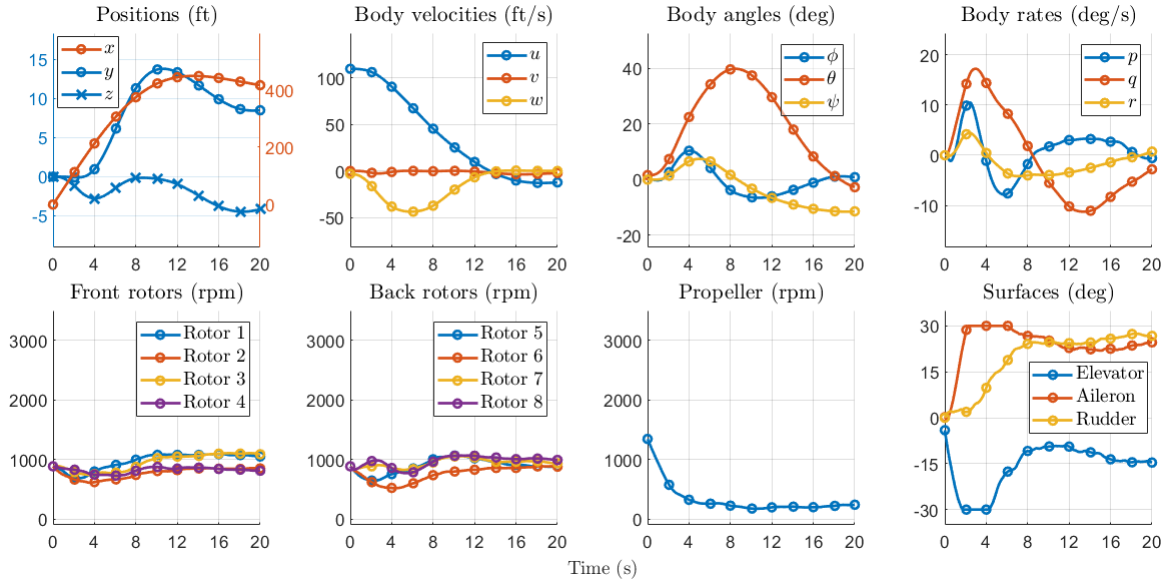


(b) MPPI.

Fig. 13 Hover to cruise transitions executed by DDP and MPPI using MPC.



(a) DDP.



(b) MPPI.

Fig. 14 Cruise to hover transition executed by DDP and MPPI using MPC.

V. Discussion and Future Work

The results of the experiments have demonstrated both DDP and MPPI as capable approaches to planning trajectories for UAM vehicles. Both have the ability to effectively plan trajectories using the complex vehicle dynamics and incorporate higher-level flight parameters into the cost function calculation. It has been shown that the scalability and efficiency of DDP allows it to optimize over long time horizons and can successfully transition a UAM class vehicle between flight regimes. Meanwhile, MPPI offers a speed advantage through its highly parallelized nature and can take full advantage of modern computing hardware. It also provides the ability to include nonlinear costs into the cost function design such as indicator functions for crashing into obstacles or leaving a desired region of the state space. Additionally, through cost function adjustment, DDP was able to adjust its plan and take full advantage of the overactuated vehicle dynamics. Cost functions can be developed to emphasize specific mission conditions and then chosen during flights based on outside sensor data.

Both DDP and MPPI have demonstrated that the planning time horizon length plays a key role in the ability for the algorithm to plan a successful route such as in the banking turn maneuver presented in Fig. 3 and Fig. 11. While both MPC algorithms found a similarly optimal path given a planning horizon of 2 seconds, the question of the tradeoffs between real-time planning vs. optimality comes into play. The particular vehicle dynamics model used throughout this work is unable to respond to fast changes of the state within the 2 second planning horizon of the algorithms. This results in the short-term behavior of DDP and MPPI, which tend to overshoot the target when used in conjunction with the shorter planning horizon (Figs. 11 and 12).

The choice of a 2 second planning horizon balances computational complexity as well as accuracy of the dynamics, since increasing the planning horizon means that either the planning horizon H needs to be increased, resulting in longer computation times, or the integration step Δt needs to be increased, resulting in numerical errors. To add to the complexity, model fidelity must also be considered. While both DDP and MPPI effectively leverage the nonlinear dynamics for path planning, some simplification of the model (e.g., separating longitudinal and lateral dynamics) may provide a computational speed advantage that enables longer trajectories to be optimized over in a similar amount of time. The increase in trajectory horizon must be balanced by the algorithms' capabilities to plan realistic trajectories for the vehicle.

This work focused on verifying the ability of DDP and MPPI to plan for highly nonlinear but ultimately deterministic dynamics. Future research directions can include stochasticity and disturbances into the dynamics, which can be solved using variants of DDP and MPPI that specifically incorporate uncertainty into the optimization [15, 22, 27]. DDP also offers the ability to explicitly handle control and state constraints [6, 28, 29]. Additional future work will focus on applying DDP and MPPI in scenarios where replanning must occur during the MPC optimization process, e.g. an obstacle appears and must be avoided. This type of situation is expected to occur frequently in the typical crowded environments that UAM vehicles will be planning over. Further incorporation of higher-level flight information and outside sensors into the control algorithm will be necessary to determine optimal replanning for situations that are not related to obstacle avoidance. Adjustments in flight plans due to fuel consumption, damage to the vehicle, or other emergencies may require a larger scope of information than the specific time horizon. This type of information is available and able to be provided to the dynamics due to the structure created in [30] and the work done in [31].

Acknowledgments

This research was supported by the NASA Aeronautics Research Mission Directorate (ARMD), Transformative Tools and Technologies (TTT) project, under the Autonomous Systems / Intelligent Contingency Management subproject. Additional thanks to the NASA Langley Research Center MidRange Computer User's Group for supporting experimental processing and analysis on the K-cluster.

References

[1] Gregory, I. M., Neogi, N. A., Campbell, N. H., Holbrook, J., Bacon, B. J., Murphy, P. C., Moerder, D. D., Simmons, B. M., Acheson, M. J., Britton, T. C., and Cook, J., "Intelligent Contingency Management for Urban Air Mobility," *AIAA Scitech 2021 Forum*, January 2021. https://doi.org/10.2514/6.2021-1000.

[2] Gregory, I. M., "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.

[3] Cook, J. W., and Gregory, I. M., "A Robust Uniform Control Approach for VTOL Aircraft," Vertical Flight Society – 2021 Autonomous VTOL Technical Meeting and Electric VTOL Symposium, Virtual, January 27, 2021.

[4] Jacobson, D. H., and Mayne, D. Q., *Differential Dynamic Programming*, American Elsevier Pub. Co., 1970.

[5] Williams, G., Drews, P., Goldfain, B., Rehg, J. M., and Theodorou, E. A., "Aggressive Driving with Model Predictive Path Integral Control," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 1433–1440.

[6] Tassa, Y., Mansard, N., and Todorov, E., "Control-Limited Differential Dynamic Programming," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 1168–1175.

[7] Pravitra, J., Theodorou, E., and Johnson, E. N., "Flying Complex Maneuvers with Model Predictive Path Integral Control," *AIAA Scitech 2021 Forum*, 2021, p. 1957.

[8] Williams, G., Aldrich, A., and Theodorou, E. A., "Model Predictive Path Integral Control: From Theory to Parallel Computation," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 344–357.

[9] Harris, J. J., and Stanford, J. R., "F-35 Flight Control Law Design, Development and Verification," *AIAA Aviation Forum*, June 2018. https://doi.org/10.2514/6.2018-3516.

[10] Acheson, M. J., Gregory, I. M., and Cook, J., "Examination of Unified Control Incorporating Generalized Control Allocation," *AIAA Scitech 2021 Forum*, 2021, p. 0999.

[11] Raab, S., Zhang, J., Bhardwaj, P., and Holzapfel, F., "Proposal of a Unified Control Strategy for Vertical Take-off and Landing Transition Aircraft Configurations," *AIAA Aviation Forum*, June 2018. https://doi.org/10.2514/6.2018-3478.

[12] Cook, J., "A Strip Theory Approach to Dynamic Modeling of eVTOL Aircraft," *AIAA SciTech 2021 Forum*, 2021, p. 1720.

[13] Murphy, P. C., Buning, P. G., and Simmons, B. M., "Rapid Aero Modeling for Urban Air Mobility Aircraft in Computational Experiments," *AIAA SciTech Forum*, January 2021. https://doi.org/10.2514/6.2021-1002.

[14] Todorov, E., and Li, W., "A Generalized Iterative LQG Method for Locally-optimal Feedback Control of Constrained Nonlinear Stochastic Systems," *Proceedings of the 2005, American Control Conference, 2005.*, IEEE, 2005, pp. 300–306.

[15] Pan, Y., Boutselis, G. I., and Theodorou, E. A., "Efficient Reinforcement Learning via Probabilistic Trajectory Optimization," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 29, No. 11, 2018, pp. 5459–5474.

[16] Van Den Berg, J., Patil, S., and Alterovitz, R., "Motion Planning under Uncertainty using Iterative Local Optimization in Belief Space," *The International Journal of Robotics Research*, Vol. 31, No. 11, 2012, pp. 1263–1278.

[17] Tassa, Y., Erez, T., and Todorov, E., "Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 4906–4913.

[18] Williams, G., Drews, P., Goldfain, B., Rehg, J. M., and Theodorou, E. A., "Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving," *IEEE Transactions on Robotics*, Vol. 34, No. 6, 2018, pp. 1603–1622.

[19] Theodorou, E. A., and Todorov, E., "Relative Entropy and Free Energy Dualities: Connections to Path Integral and KL Control," *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, IEEE, 2012, pp. 1466–1473.

[20] Okada, M., and Taniguchi, T., "Variational Inference MPC for Bayesian Model-based Reinforcement Learning," *Conference on Robot Learning*, PMLR, 2020, pp. 258–272.

[21] Wang, Z., So, O., Gibson, J., Vlahov, B., Gandhi, M. S., Liu, G.-H., and Theodorou, E. A., "Variational Inference MPC using Tsallis Divergence," *Robotics: Science and Systems*, 2021.

[22] Gandhi, M. S., Vlahov, B., Gibson, J., Williams, G., and Theodorou, E. A., "Robust Model Predictive Path Integral Control: Analysis and Performance Guarantees," *IEEE Robotics and Automation Letters*, Vol. 6, No. 2, 2021, pp. 1423–1430.

[23] Williams, G. R., "Model Predictive Path Integral Control: Theoretical Foundations and Applications to Autonomous Driving," Ph.D. thesis, Georgia Institute of Technology, 2019.

[24] Garcia, C. E., Prett, D. M., and Morari, M., "Model Predictive Control: Theory and Practice — A Survey," *Automatica*, Vol. 25, No. 3, 1989, pp. 335–348.

[25] Bertsekas, D., *Dynamic Programming and Optimal Control: Volume I*, Vol. 1, Athena Scientific, 2012.

[26] Tassa, Y., Erez, T., and Smart, W. D., "Receding Horizon Differential Dynamic Programming." *NIPS*, Citeseer, 2007, pp. 1465–1472.

[27] Theodorou, E., Tassa, Y., and Todorov, E., "Stochastic Differential Dynamic Programming," *Proceedings of the 2010 American Control Conference*, IEEE, 2010, pp. 1125–1132.

[28] Xie, Z., Liu, C. K., and Hauser, K., "Differential Dynamic Programming with Nonlinear Constraints," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 695–702.

[29] Aoyama, Y., Boutselis, G., Patel, A., and Theodorou, E. A., "Constrained Differential Dynamic Programming Revisited," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 9738–9744.

[30] Acheson, M. J., Bacon, B. J., Britton, T. C., Campbell, N. H., Cook, J., Gregory, I. M., Holbrook, J., Houghton, M., Moerder, D. D., Murphy, P. C., Neogi, N. A., Oshin, A., and Simmons, B. M., "Intelligent Contingency Management: Accomplishments and Benchmark Problem 1," , 10 2021.

[31] Holbrook, J., Neogi, N., Acheson, M., and Gregory, I., "Benchmark Problem Development for Testing Maturity of Intelligent Contingency Management Tools," *AIAA SciTech Forum*, 2022.