PHILOSOPHICAL TRANSACTIONS A

rsta.royalsocietypublishing.org

Research

Article submitted to journal

Subject Areas:
computational modelling and simulation; computational mathematics

Keywords:
autonomous landing; digital twin; flight test; fluid-structure interaction; model predictive control; model reduction

Author for correspondence:
Charbel Farhat, Department of Aeronautics and Astronautics, Stanford University, Stanford, California 94305, United States of America
e-mail: cfarhat@stanford.edu

A Physics-Based Digital Twin for Model Predictive Control of Autonomous Unmanned Aerial Vehicle Landing

McClellan, Andrew[1], Lorenzetti, Joseph[1], Pavone, Marco[1], and Farhat, Charbel[1,2,3]

[1]Department of Aeronautics and Astronautics, Stanford University, Stanford, California, United States of America
[2]Department of Mechanical Engineering, Stanford University, Stanford, California, United States of America
[3]Institute for Computational and Mathematical Engineering, Stanford University, Stanford, California, United States of America

This paper proposes a two-level, data-driven, digital twin concept for the autonomous landing of aircraft, under some assumptions. It features a digital twin instance for model predictive control; and an innovative, real-time, digital twin prototype for fluid-structure interaction and flight dynamics to inform it. The latter digital twin is based on the linearization about a pre-designed glideslope trajectory of a high-fidelity, viscous, nonlinear computational model for flight dynamics; and its projection onto a low-dimensional approximation subspace to achieve real-time performance, while maintaining accuracy. Its main purpose is to predict in real-time, during flight, the state of an aircraft and the aerodynamic forces and moments acting on it. Unlike static lookup tables or regression-based surrogate models based on steady-state wind tunnel data, the aforementioned real-time digital twin prototype allows the digital twin instance for model predictive control to be informed by a truly dynamic flight model, rather than a less accurate set of steady-state aerodynamic force and moment data points. The paper describes in details the construction of the proposed two-level digital twin concept and its verification by numerical simulation. It also reports on its preliminary flight validation in autonomous mode for an off-the-shelf unmanned aerial vehicle instrumented at Stanford University.

THE ROYAL SOCIETY PUBLISHING

1 Introduction

In a fully autonomous mode, the control of an aircraft is usually performed via an autonomous flight control system (AFCS), together with auxiliary information that allows a pilot to monitor the maneuver's progress. Historically, the design of an AFCS has followed two main approaches. In one of them, the equations of motion of the aircraft are linearized about equilibrium (trim) conditions – such as unaccelerated level flight conditions, steady coordinated turns, and steady roll [1]. Then, stability and control augmentation systems as well as autopilots are designed using techniques from linear control theory [2], modern multivariable state-space control [3], or robust control (H_∞ – and μ – synthesis) [4,5]. In the second approach, tools from nonlinear control and stability theory are used. In this context, a popular control technique is dynamic inversion or, alternatively, feedback linearization: such a technique inverts the nonlinear dynamics of the aircraft to yield a multivariable linear dynamical system that may then be stabilized using linear control tools [6]. All of these control techniques require either the linearization or the inversion of the aircraft dynamics at some operating point. Consequently, they parameterize control laws based on the operating conditions and rely on lookup tables and interpolation or adaptive control [7] to switch between different operating points. However, the aforementioned control techniques neither allow the direct inclusion of hard performance *constraints* (e.g. sink rate below a given desired threshold), nor incorporate a state prediction capability to increase the robustness of the control process. More critically, they neither leverage high-fidelity computational models to predict the aerodynamics loads acting on an aircraft during the fluid-structure interaction (FSI), nor include a provision for autonomous emergency abort maneuvers.

On the other hand, model predictive control (MPC) is a principled method in optimal control theory that accounts for complex state/control constraints while optimizing performance criteria. At the core, it generates *online* (i.e. real-time) optimal control inputs for the feedback control of a system. For this purpose, MPC relies on a model of the system to *predict* and *optimize* its future behavior. Its key advantage is that it allows a system such as an aircraft to adapt to a large set of operating conditions. As in this approach the control inputs are obtained in *real-time* by repeatedly solving an optimization problem, updates can be made to the cost function to reflect changes in: mission requirements (e.g. wave-off versus continued approach); in the model to reflect parameter changes; and in the constraints to reflect new limitations due to exogenous events (e.g. gust wind). Most importantly, the predictive nature of MPC enables control at the performance limits, as required during contingency maneuvers and in the presence of uncertainty in both the sensor measurements and environmental conditions. These characteristics have made MPC a promising technology for a variety of autonomous systems, including autonomous cars, aircraft, and space vehicles [8].

Nevertheless, the successful application of MPC to the problem of autonomous aircraft landing requires the availability of: a low-dimensional computational model capable of accurately predicting in real-time the aerodynamic forces and moments over a large set of operating conditions; a control algorithm equipped with correctness guarantees for solving online the MPC optimization problem; and a real-time implementation of this algorithm on a miniature computer based on, for example, graphics processing units (GPUs). This paper considers the case where the aircraft of interest is an unmanned aerial vehicle (UAV). It focuses on the part of the final approach that occurs roughly two minutes prior to touchdown, where the altitude drops, for example, from $90\,\mathrm{m}$ to ground level. It addresses the aforementioned requirements in the form of a *digital twin* (DT) for the MPC of autonomous UAV landing.

A DT is a virtual replica of a physical object such as a UAV, or a process such as its manufacturing and/or certification. The underlying concept typically integrates artificial intelligence, machine learning, and/or software analytics with physics-based modeling, to create a digital simulation model that can mirror the states and behaviors of the DT's physical counterpart. For this purpose, such a model may have to be continuously trained and/or updated using sensor data and a probabilistic computational framework that accounts for uncertainties

[9,10]. Most recently, DTs were divided in three types [11]: the DT Prototype (DTP); the DT Instance (DTI); and the DT Aggregate (DTA). A DTP consists of the designs, analyses, and processes used to realize the physical product; hence, it exists before there is a physical product and data generated by sensors mounted on the product. On the other hand, a DTI is the DT of each individual instance of the product, once it is manufactured and equipped with sensors that generate data for it. As for the DTA, it is an aggregation of DTIs that allows for a larger set of data to be collected and processed for interrogation about the physical product. In other words, a DTP is often a rebranding of important but nevertheless conventional, well-established technologies such as computer-aided design (CAD) and simulation-based engineering. As a former managing director at McLaren Applied Technologies wrote "we used a digital twin – though we just called it a computer simulation" [12]. A DTI however embodies the concept of customization which is central to its vision and involves three distinct parts: a physical asset (or a process); the corresponding virtual product; and data that flows between them. The same holds true for a DTA, which furthermore offers the opportunity to enhance prognostics and learning processes beyond what can be achieved using a DTI.

Using the nomenclature highlighted above, the DT proposed in this paper is a DTI for the MPC of autonomous UAV landing, where the computational model for state prediction and aerodynamic forces and moments post-processing is a low-dimensional surrogate of a high-fidelity, physics-based, computational model and therefore a special form of a DTP that can operate in real-time. Hence, this DT can be described as a two-level DT, where a DTI for MPC of autonomous landing (AL) is wrapped around an innovative, real-time DTP (RT-DTP) for FSI (see Figure 1). The design of this DT features three innovations that constitute the main contributions of this paper: the formulation of the governing coupled FSI equations in an arbitrary Lagrangian-Eulerian (ALE) setting and their linearization about a pre-designed glideslope trajectory, rather than a mere equilibrium point, albeit using a reasonable simplifying assumption; the construction of a linearized, computational fluid dynamics (CFD)-based, fluid PROM [13] and its coupling with a linearized six-degrees-of-freedom (6dof) representation of the rigid dynamics of the UAV; and the training of the resulting FSI PROM for any deflection of any of the UAV's control surfaces. Hence, in this work, the UAV is assumed to be sufficiently stiff to justify its representation by a 6dof rigid body equipped with deployable control surfaces. This assumption is mild, as it is satisfied by many UAVs, particularly in the context of landing. Linearization is also justified by the fact that the controller can be expected to maintain the UAV within small perturbations about a pre-designed glideslope, except if it has to deal with an exogenous event just before touchdown – a scenario that is not considered in this work. On the other hand, linearization about a steady-state equilibrium point would not be effective in this case, because the fluid state at such a point is time-dependent. For this reason, a more suitable linearization is performed instead. As for relying on a low-dimensional, *physics-based* DTP instead of a lookup table or a regression-based surrogate model based on steady-state wind tunnel data for predicting the aerodynamic forces and moments needed by the MPC controller, it offers the following advantage. It allows the DTI for MPC to be informed by a truly dynamic FSI model, rather than a less accurate set of steady-state aerodynamic force and moment data points, and interpolation/extrapolation between/away from these points. Other contributions of this paper include: the description of a high-dimensional DTP for FSI and controlled flight dynamics, and its application to the verification by simulation of the proposed DT; and a brief report on the preliminary flight validation in autonomous mode of the proposed DT using a commercial off-the-shelf UAV instrumented at Stanford University.

To this end, the remainder of this paper is organized as follows. Section 2 presents a physics-based, high-dimensional, high-fidelity computational framework for the simulation of FSI problems that is suitable for: constructing the low-dimensional, linearized, RT-DTP appropriate for informing an MPC controller; and performing a class of nonlinear flight dynamics computations including those relevant to the verification of the proposed two-level DT for

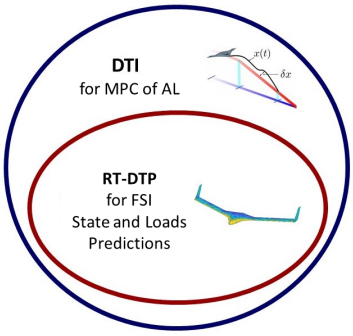


Figure 1: Schematic of a two-level DT for autonomous UAV landing: DTI for MPC (outerset); and RT-DTP for FSI (innerset).

autonomous UAV landing. Section 3 describes the computational technology underlying the RT-DTP proposed for state prediction and aerodynamic forces and moments post-processing, which is grounded in robust, linear, projection-based model order reduction (PMOR). Hence, this section also presents a novel approach for training a linear FSI PROM for arbitrary motions and control inputs. Section 4 describes the proposed DTI for MPC. Section 5 describes the verification process of the proposed DT for autonomous UAV landing using a high-dimensional DTP for nonlinear flight dynamics. Section 6 reports on the in-flight validation of this DT and Section 7 concludes this paper.

2 High-Dimensional Computational Models for FSI

The DT computational technology presented in this paper is grounded in a physics-based, high-dimensional, high-fidelity, ALE computational framework for the simulation of a class of FSI and flight dynamics problems characterized by small to moderate structural rotations, deformations, and/or topological changes – such as those induced by the opening of a control surface [14]. This framework is associated with the three-field formulation [15] of the aforementioned class of FSI problems, which represents a body-fitted CFD mesh as a pseudo-structural system and models all interactions of the fluid, structural, and dynamic CFD mesh subsystems. For this purpose, this framework is typically equipped with advanced mesh motion algorithms that maintain at all times body-fitted an initially body-fitted CFD mesh.

Numerical stability and higher-order spatio-temporal accuracy can be guaranteed [16,17] for this computational framework that furthermore facilitates linearization about many entities [18] and eases linear PMOR [13]. In the linearized case, it accounts for structural motions and deformations via transpiration conditions rather than CFD mesh motion and therefore avoids altogether the potential issue of mesh entanglement associated with control surface deflections. In the nonlinear case, for a UAV system undergoing small to moderate rotations and deflections due to the flight control system, advanced mesh motion algorithms such as those presented in [19,20] are capable of circumventing mesh entanglements. Hence, for all these reasons, the ALE computational framework associated with the three-field formulation [15] of FSI problems and flight dynamics is chosen here for both: building a linearized, low-dimensional DTP capable of accurately predicting in real-time aerodynamic forces and moments, and therefore informing an MPC controller; and verifying the proposed two-level DT for autonomous UAV landing.

The rigid UAV is represented by a 6dof system equipped with deployable control surfaces. Specifically, this system is modeled by an assembly of rigid and phantom[1] finite elements that leads to: six, and only six, *unconstrained*, translational and rotational dofs; as many internal Lagrange multiplier dofs as needed to enforce the constraints arising from the rigid elements and

[1]Phantom elements are zero mass and stiffness elements that serve no other purpose but transferring to a computational structural dynamics (CSD) model the flow-induced loads computed using CFD.

their assembly – if the Lagrange multiplier method is used to enforce these constraints; and for each modeled deployable control surface, a rotation variable around its hinge line to be *prescribed* by the MPC controller. Such a finite element (FE) representation of a rigid dynamic system is easily extendible to flexible aircraft and facilitates a CFD / computational structural dynamic (CSD)-based FSI analysis.

Due to space limitation, the computational framework outlined above are described below at the semi-discrete level: while they can be equipped with any preferred temporal approximation, the discretization of the corresponding final product is discussed in Section 4(a).

(a) Nonlinear Semi-Discrete Base Models

i Reference Frames and Rigid Body Dynamics

i.1 Inertial, Relative, and Body-Fixed Frames

For the stated application of interest, three reference frames are relevant (assuming the flat-earth approximation):

(i) An inertial frame $\mathcal{I}$ fixed to the flat-Earth, described by its unit vectors $(\hat{i}, \hat{j}, \hat{k})$.
(ii) A non-accelerating relative frame $\mathcal{R}$ traveling at the constant free-stream velocity vector $\hat{v}_\infty$ along a nominal glideslope (constructed in Section 2(b).i), described by its unit vectors $(\hat{x}, \hat{y}, \hat{z})$.
(iii) A body-fixed frame $\mathcal{B}$, described by its unit vectors $(\hat{b}_x, \hat{b}_y, \hat{b}_z)$.

The unit vectors of $\mathcal{I}$ and those of $\mathcal{R}$ are related by a fixed rotation of magnitude ψ_0 about the axis $\hat{j} = \hat{y}$. A natural choice of this angle is

$$\psi_0 = \theta_{y,0} = \gamma_0 + \alpha_0$$

where $\theta_{y,0}$ is the initial pitch of the UAV, γ_0 is the glideslope flight path angle, and α_0 is the initial angle of attack. When $\psi_0 = 0°$ and the roll, pitch, and yaw of the UAV relative to $\mathcal{I}$ are also zero, the three sets of unit vectors become identical – with $\hat{b}_x$ pointing nose to tail, $\hat{b}_y$ pointing out the right wing, and $\hat{b}_z$ pointing upwards.

Throughout this paper, the equations governing the dynamic equilibrium of the assumed rigid UAV are written in $\mathcal{I}$ for the translational dofs and in $\mathcal{B}$ for the rotational ones; however, in both cases, the governing equations are explicitly expressed in terms of the unit vectors $(\hat{x}, \hat{y}, \hat{z})$.

i.2 FE Representation of a 6dof System

As stated above, for the purpose of a CFD / CSD-based FSI analysis, a rigid UAV can be modeled by a $2N_{\text{CSD}}$-dimensional FE model comprising an arbitrary number of rigid and phantom elements leading to 6 unconstrained and $(2N_{\text{CSD}} - 6)$ *constrained* dofs. In this case, the governing nonlinear semi-discrete equations of dynamic equilibrium can be written as [21]

$$M_c(\theta)\ddot{\varphi} + D_c(\theta, \dot{\theta})\dot{\varphi} + f^{\text{int}}(\lambda, \varphi) - f_c^{\text{aero}}(w, \theta) - f^g - f_c^{\text{thrust}}(c_T, \theta) = 0 \tag{2.1}$$

where

$$M_c(\theta) = \begin{bmatrix} M_{\text{CM}} & 0 \\ 0 & T^{a,T}(\theta) J_{\text{CM}} T^a(\theta) \end{bmatrix}$$

$$D_c(\theta, \dot{\theta}) = \begin{bmatrix} 0 & 0 \\ 0 & T^{a,T}(\theta) J_{\text{CM}} \dot{T}^a(\theta, \dot{\theta}) + T^{a,T}(\theta) [T^a(\theta)\dot{\theta}]_\times J_{\text{CM}} T^a(\theta) \end{bmatrix}$$

$$M_{\text{CM}} = \text{block\_diag}\left(\begin{bmatrix} 0 & \cdots & m\mathbb{I} & \cdots & 0 \end{bmatrix} \right), \; J_{\text{CM}} = \text{block\_diag}\left(\begin{bmatrix} 0 & \cdots & J & \cdots & 0 \end{bmatrix} \right)$$

$$f_c^{\text{aero}}(w, \theta) = \begin{bmatrix} \mathbb{I} & 0 \\ 0 & T^a(\theta) \end{bmatrix} \begin{bmatrix} f_{\text{for}}^{\text{aero}}(w, \theta) \\ f_{\text{mom}}^{\text{aero}}(w, \theta) \end{bmatrix}, \; f_c^{\text{thrust}}(c_T, \theta) = \begin{bmatrix} \mathbb{I} & 0 \\ 0 & T^a(\theta) \end{bmatrix} \begin{bmatrix} f_{\text{for}}^{\text{thrust}}(c_T, \theta) \\ f_{\text{mom}}^{\text{thrust}}(c_T, \theta) \end{bmatrix} \tag{2.2}$$

and:

- A dot designates a time derivative.
- The $2N_{\text{CSD}}$-dimensional solution of Eq. (2.1) can be written in vector form as

$$\varphi = \begin{bmatrix} u^T & \theta^T \end{bmatrix}^T$$

where u and θ denote the vectors of translational and rotational dofs attached to the nodes of the FE model, respectively, and the superscript T designates here, above, and throughout the remainder of this paper the transpose operation.
- $T^{a,T}(\theta) = T^a(\theta)\Lambda^a(\theta)$.
- All 2×2 partitionings in (2.2) correspond to the partitioning of the dofs of the $2N_{\text{CSD}}$-dimensional system in N_{CSD} translational and N_{CSD} rotational dofs.
- The mass of the UAV m and its moment of inertia tensor J (in $\mathcal{B}$) about the center of mass CM are lumped at CM, where a FE node can be conveniently positioned: this results in the N_{CSD}-dimensional FE matrices M_{CM} and J_{CM}.
- $\mathbb{I}$ is the identity matrix of dimension N_{CSD}, except in the definition of M_{CM}, where it is of dimension 3.
- D_c is the configuration-dependent matrix associated with the velocity-dependent inertial forces.
- $T^a(\theta)$ is the matrix defined by

$$T^a(\theta) = \begin{bmatrix} T_1^a & 0 & \dots & 0 \\ 0 & T_2^a & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & T_{n_{\text{FE}}}^a \end{bmatrix}$$

where: $T_i^a \in \mathbb{R}^{3\times 3} = T^a(\theta_i) = \dfrac{\sin\psi_i}{\psi_i}\mathbb{I} - \dfrac{1-\cos\psi_i}{\psi_i^2}[\theta_i]_\times + \dfrac{\psi_i - \sin\psi_i}{\psi_i^3}\left(\theta_i\theta_i^T\right),\ i = 1, \cdots, n_{\text{FE}}$, is the *material tangential transformation* operator [22] that arises when the rotational components of the dynamic equations of equilibrium at a node i of the FE model are written in terms of $\dot{\theta}_i$ and $\ddot{\theta}_i$ (instead of the *convected* angular velocities and accelerations); $\psi_i = \|\theta_i\|$; $[(\cdot)]_\times$, where $(\cdot)$ is a generic vector in $\mathbb{R}^3$, is the skew-symmetric matrix in $\mathbb{R}^{3\times 3}$ defined by $([(\cdot)]_\times)\,\eta = (\cdot)\times\eta$, where η is an arbitrary vector in $\mathbb{R}^3$ and $\times$ designates in this case the vector (cross) product; n_{FE} is the total number of nodes in the FE model; the pre-multiplication of the second N_{CSD} equations of (2.1) by $T^a(\theta)$ is performed to ensure the symmetry of $M_c(\theta)$ (see Section 4.2 of [21]); and $[T^a(\theta)\dot{\theta}]_\times$ in (2.2) is to be understood as the block diagonal matrix $[T_i^a\dot{\theta}_i]_\times$, $i = 1, \cdots, n_{\text{FE}}$.
- $\begin{bmatrix} f_{\text{for}}^{\text{aero}} & f_{\text{mom}}^{\text{aero}} \end{bmatrix}^T$ is the vector of aerodynamic forces and moments (expressed in $\mathcal{R}$).
- f^g is the vector of gravitational forces (which, considering that the aircraft mass m is lumped at the CM, do not create a moment about the CM).
- $\begin{bmatrix} f_{\text{for}}^{\text{thrust}} & f_{\text{mom}}^{\text{thrust}} \end{bmatrix}^T$ is the vector of thrust-induced forces and moments, and c_T is the magnitude of $f_{\text{for}}^{\text{thrust}}$.
- Propulsion is not explicitly modelled and therefore f_c^{thrust} is considered to be independent of the fluid state vector w.
- f_c^{thrust} is applied at a single node j of the FE model, along $y = 0$, and in the direction of $-\hat{b}_x$: hence, it can be written as

$$f^{\text{thrust}}(c_T, \theta) = \begin{bmatrix} 0 & \cdots & \begin{bmatrix} -c_T & 0 & 0 \end{bmatrix}\Lambda^T(\theta_j) & \begin{bmatrix} 0 & -c_T\Delta z_{T,CM} & 0 \end{bmatrix}\Lambda^T(\theta_j) & \cdots & 0 \end{bmatrix}^T$$

where

$$\Lambda(\theta_j) = \exp[\theta_j]_\times = \mathbb{I} + \frac{\sin\psi_j}{\psi_j}[\theta_j]_\times + \frac{1-\cos\psi_j}{\psi_j^2}[\theta_j]_\times^2$$

is the rotation matrix representation of the rotation vector and $\Delta z_{T,CM}$ is the vertical offset of the thrust from the center of mass (see Figure 2).

- When the Lagrange multiplier method is used to enforce the constraints arising from the FE modeling and assembly of rigid elements, λ is the vector of Lagrange multiplier dofs and f^{int} is the associated vector of internal forces. If the penalty method is used however for this purpose, $f^{\text{int}} = 0$ and several of the terms of Eq. (2.1) are affected by a penalty parameter but otherwise keep the same physical meaning.

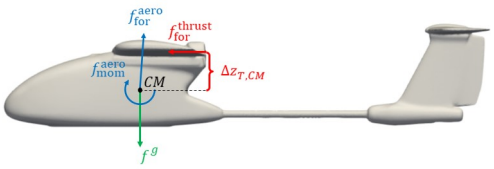


Figure 2: Free-body diagram of a (rigid) UAV.

During the solution of the discrete counterpart of Eq. (2.1) – in this case, using either the midpoint rule or the generalized-α method [23] – the $(2N_{\text{CSD}} - 6)$ constrained dofs are eliminated (in terms of the 6 *unconstrained* dofs) to obtain the discrete solution of the classical equations governing the dynamics of a rigid body subsystem with 6 dofs.

ii RANS-based CFD Prediction of Aerodynamic Loads in the ALE Setting

Air is modeled as a perfect gas and its flow past the UAV is assumed to be compressible, and governed by the three-dimensional (3D) Reynolds-averaged Navier-Stokes (RANS) equations equipped with the Spalart-Allmaras turbulence model [24]. The convective terms of these equations are semi-discretized on unstructured CFD meshes by a vertex-based, second-order, finite volume (FV) method with dual cells (or control volumes); and the diffusive and source terms (due to turbulence modeling) are semi-discretized on the primal cells of the CFD mesh by a piecewise linear FE method.

In the ALE setting, the FV-FE semi-discretization of the governing RANS equations is performed on a dynamic, body-fitted, CFD mesh. It leads to the following system of ordinary differential equations of dimension N_{CFD}

$$\begin{cases} \dot{\overbrace{\left(A(\mathcal{X})[w^T \;\; w_t^T]^T\right)}} + \Phi^{\text{cv}}\left([w^T \;\; w_t^T]^T, \mathcal{X}, \dot{\mathcal{X}}\right) - \Phi^{\text{ds}}\left([w^T \;\; w_t^T]^T, \mathcal{X}\right) & = 0 \qquad \text{in } \Omega^F(t) \\ \widetilde{K}\mathcal{X} - \widetilde{K}_c u & = 0 \qquad \text{on } \Gamma^{F/S}(t) \end{cases} \tag{2.3}$$

where $\Omega^F(t)$ and $\Gamma^{F/S}(t)$ denote the computational fluid domain and fluid / structure interface, respectively, and their dependence on time t is emphasized; and the second equation models the ALE CFD mesh as a quasi-static pseudo-structural system and enforces a semi-discrete version of the no-penetration condition on $\Gamma^{F/S}(t)$ [25]. In Eq. (2.3) above, $\mathcal{X} = (x, y, z)^T$ denotes the semi-discrete, time-dependent, position vector of the ALE, body-fitted, CFD mesh; A is a diagonal matrix containing the time-dependent cell volumes of this mesh; w_t is the vector of semi-discrete, conservative, time-dependent, turbulence model variables; Φ^{cv} denotes the vector of numerical convective flux functions; Φ^{ds} is the vector of numerical diffusive flux functions and source terms due to turbulence modeling; $\widetilde{K}$ is in general a time-dependent, FE stiffness matrix arising from the structural analogy chosen for representing the CFD mesh as a pseudo-structural system; $\widetilde{K}_c$ is a time-dependent, transfer matrix representing the effect of the motion of the real structural system at $\Gamma^{F/S}(t)$ on that of the ALE CFD mesh; and u denotes as before the vector of semi-discrete, time-dependent structural displacements.

From here on, the vectors of semi-discrete, conservative, time-dependent, fluid state and turbulence model variables w and w_t are merged and denoted simply by w.

It is noted that:

- Here and throughout the remainder of this paper, the numerical convective flux function at any vertex i of the CFD mesh is based on the second-order extension of Roe's approximate Riemann solver and thus can be written as

$$\Phi_i^{\text{cv}} = \sum_{j \in \mathcal{N}(i)} \left|\partial\Omega_{ij}^F(t)\right| \phi_{ij}^{\text{Roe}} \left(w_{ij}, w_{ji}, \nu_{ij}(\mathcal{X}), \kappa_{ij}(\mathcal{X}, \dot{\mathcal{X}})\right)$$

  where $\mathcal{N}(i)$ denotes the set of vertices of the CFD mesh connected to vertex i; for each $j \in \mathcal{N}(i)$, $\left|\partial\Omega_{ij}^F(t)\right|$ denotes the area of the facet defined by the intersection of the dual cells $\Omega_i^F(t)$ and $\Omega_j^F(t)$ – that is, $\left|\Omega_{ij}^F(t)\right| = \left|\partial\Omega_i^F(t) \cap \partial\Omega_j^F(t)\right|$; ϕ_{ij}^{Roe} is the ALE version of the Roe numerical flux function; w_{ij} and w_{ji} are the linearly reconstructed values of w_i and w_j at $\Omega_{ij}^F(t)$, respectively; and ν_{ij} and κ_{ij} are defined as

$$\nu_{ij}(\mathcal{X}) = \frac{1}{\left|\partial\Omega_{ij}^F(t)\right|} \int_{\partial\Omega_{ij}^F(t)} \mu_{ij}(\mathcal{X}) ds$$

$$\kappa_{ij}(\mathcal{X}, \dot{\mathcal{X}}) = \frac{1}{\left|\partial\Omega_{ij}^F(t)\right|} \int_{\partial\Omega_{ij}^F(t)} \dot{\mathcal{X}} \cdot \mu_{ij}(\mathcal{X}) ds$$

  where $\mu_{ij}(\mathcal{X})$ is the unitary normal to $\partial\Omega_{ij}^F(t)$ and thus is a time-dependent quantity.
- $\widetilde{K}_c$ can be constructed whether the discrete representation of the wet surface of the UAV matches or not that of the wall boundary of Ω_F [25].
- The time-discretization of Eq. (2.3) is performed using a DGCL (discrete geometric conservation law)-preserving – and thus numerically stable – and provably second-order time-accurate scheme [17].
- The deflection of a control surface shears the CFD mesh along the edges between the lateral sides of the control surface and the parts of the main wing facing them. When these deflections are small to moderate, the motion of the dynamic CFD mesh is sustainable by advanced mesh motion algorithms [19,20]; and the inability of the ALE computational framework to account for the air flow crossing underneath the opened control surface, from its lateral sides, does not significantly affect accuracy.

(b) Linearized Semi-Discrete Base Models

Next, the physics-based coupled equations (2.1) and (2.3) governing an FSI problem are linearized around a glideslope constructed as a planned equilibrium trajectory, as motivated and justified in Section 1. For this purpose and the sake of notational simplicity, the variable θ is redefined below for various purposes designated by various subscripts.

i Linear Perturbation Analysis About a Dynamic Equilibrium Trajectory

The linearization procedure described in [18] is adopted here, except for the following major differences: as stated above, linearization is performed about a *planned* equilibrium trajectory rather than a mere equilibrium point; and structural flexibility effects are not accounted for, since the UAV is reasonably assumed to be rigid. Here, the planned equilibrium trajectory is the glideslope defined in the inertial frame $\mathcal{I}$ and the absence of a cross wind by: the free-stream velocity magnitude v_∞; the constant flight-path angle γ_0; zero roll and yaw angles; a time-dependent angle of attack α^p; and the associated pitch angle $\theta_{R,y}^{\mathcal{I},p} = \gamma_0 + \alpha^p$, where the superscripts p and $\mathcal{I}$ designate the planned aspect of the trajectory and a quantity measured in the inertial frame $\mathcal{I}$, respectively. Specifically, the glideslope is determined so that each point of

this trajectory is a trim point where the following conditions hold

$$\ddot{u}^p_R = \dot{u}^p_R = u^p_R = 0, \quad \ddot{\theta}^p_{R,x} = \dot{\theta}^p_{R,x} = \theta^p_{R,x} = 0, \quad \ddot{\theta}^p_{R,z} = \dot{\theta}^p_{R,z} = \theta^p_{R,z} = 0$$
$$\dot{\theta}^p_{\mathrm{CS},a} = \theta^p_{\mathrm{CS},a} = 0, \quad \dot{\theta}^p_{\mathrm{CS},r} = \theta^p_{\mathrm{CS},r} = 0 \tag{2.4}$$

Here, the subscript R designates the translations and rotations due *only* to rigid body motion – and therefore not to control surface deflections; the subscript CS designates a quantity pertaining to a control surface or their entire set; and $\theta_{\mathrm{CS},a}$ and $\theta_{\mathrm{CS},r}$ designate the deflections of the ailerons and rudders, respectively. Observe that this choice of glideslope implies that if the UAV has ailerons, elevators, and rudders, the pitch angle at each point of this planned equilibrium trajectory is determined by deflecting only the elevators appropriately. Observe also that this choice of glideslope incorporates all necessary information for tracking the evolution of the relative reference frame $\mathcal{R}$ with respect to the inertial reference frame $\mathcal{I}$ (see Section 2(a).i.1).

During descent, when the altitude drops from $90\,\mathrm{m}$ to ground level, the atmospheric density [26] changes only by $\sim 1\%$. For this reason, along the glideslope, the variations of the pitch angle and control surface deflections can be expected to be small, and the following relations

$$\ddot{\theta}^p_{R,y} \approx 0, \quad \left|\dot{\theta}^p_{R,y}\right| \ll 1, \quad \left|\theta^p_{R,y}\right| \ll 1, \quad \left|\dot{\theta}^p_{\mathrm{CS},e}\right| \ll \left|\theta^p_{\mathrm{CS},e}\right| \ll 1 \tag{2.5}$$

where $\theta_{\mathrm{CS},e}$ designates the deflections of the elevators, can be expected to hold. Similarly, the rate of change of the fluid state along the glideslope can be expected to be negligible – that is,

$$\left|\dot{w}^p\right| \ll 1 \tag{2.6}$$

Reference [27] provides an algorithm for generating the glideslope described above.

Then, the structural and fluid perturbations are defined as

$$\delta\phi(t) := \phi(t) - \phi^p(t), \quad \delta w(t) := w(t) - w^p(t), \quad \delta c(t) := c(t) - c^p(t)$$

where

$$c = \begin{bmatrix} c_T & \dot{\theta}_{\mathrm{CS},1} & \cdots & \dot{\theta}_{\mathrm{CS},n_{\mathrm{CS}}} \end{bmatrix}^T \tag{2.7}$$

$\dot{\theta}_{\mathrm{CS},i}$, $i = 1, \cdots, n_{\mathrm{CS}}$, is the deflection rate of each $i-$th control surface, and c is the vector of control inputs.

After some calculus, the linearization of Eq. (2.1) about the planned glideslope leads to

$$\begin{aligned} M_c\left(\theta^p\right)\delta\ddot{\phi} \quad &- \quad \frac{\partial f_c^{\mathrm{aero}}}{\partial w}\left(w^p, \theta^p_R\right)\delta w - \frac{\partial f_c^{\mathrm{aero}}}{\partial \phi}\left(w^p, \theta^p_R\right)\delta\phi \\ &- \quad \frac{\partial f_c^{\mathrm{thrust}}}{\partial \phi}\left(c^p_T, \theta^p_R\right)\delta\phi - \frac{\partial f_c^{\mathrm{thrust}}}{\partial c_T}\left(c^p_T, \theta^p_R\right)\delta c_T = 0 \end{aligned} \tag{2.8}$$

Again, the dimension $2N_{\mathrm{CSD}}$ of problem (2.8) is artificially large due to the convenient representation of the UAV as an assembly of rigid and phantom finite elements. However, in the linearized setting, the computational overhead associated with eliminating the $(2N_{\mathrm{CSD}} - 6)$ constrained dofs in terms of the 6 unconstrained dofs can be avoided altogether without losing accuracy, simply by performing the Galerkin projection of the above problem onto the basis

$$X = \begin{bmatrix} X_R & X_{\mathrm{CS}} \end{bmatrix}$$

where the columns of $X_R \in \mathbb{R}^{2N_{\mathrm{CSD}} \times 6}$ are the rigid body modes of the UAV and those of $X_{\mathrm{CS}} \in \mathbb{R}^{2N_{\mathrm{CSD}} \times n_{\mathrm{CS}}}$ are its control surface modes[2]. This leads to re-writing the solution $\delta\phi$ as

$$\delta\phi = X\delta y = \begin{bmatrix} X_R & X_{\mathrm{CS}} \end{bmatrix} \begin{bmatrix} \delta y_R \\ \delta y_{\mathrm{CS}} \end{bmatrix} \tag{2.9}$$

[2] A control surface mode is defined here as a mode that is zero everywhere except at the dofs attached to the nodes located on a particular control surface, where its values are given by a deflection of that surface.

which transforms problem (2.8) into

$$M^p_{c,r}\delta\ddot{y}_R - P^p_R\delta y_R - P^p_w\delta w - p^{\text{thrust},p}_{c_T}\delta c_T - P^p_{\text{CS}}\delta y_{\text{CS}} = 0 \tag{2.10}$$

where (noting that the thrust is not applied on any control surface)

$$M^p_{c,r} = X_R^T M_c\left(\theta^p\right) X_R,\; P^p_R = \left(P^{\text{aero},p}_R + P^{\text{thrust},p}_R\right),\; P^{\text{aero},p}_R = X_R^T\frac{\partial f_c^{\text{aero}}}{\partial\phi}\left(w^p,\theta^p_R\right)X_R$$

$$P^{\text{thrust},p}_R = X_R^T\frac{\partial f_c^{\text{thrust}}}{\partial\phi}\left(c^p_T,\theta^p_R\right)X_R,\; p^{\text{thrust},p}_{c_T} = X_R^T\frac{\partial f_c^{\text{thrust}}}{\partial c_T}\left(c^p_T,\theta^p_R\right)$$

$$P^p_{\text{CS}} = X_R^T\frac{\partial f_c^{\text{aero}}}{\partial\phi}\left(w^p,\theta^p_R\right)X_{\text{CS}}$$

Next, the linearization of Eq. (2.3) about the glideslope and the considerations expressed in (2.4–2.6) lead to

$$A\left(\mathcal{X}^p\right)\delta\dot{w} + H^p\delta w + B^p\delta\dot{\mathcal{X}} + C^p\delta\mathcal{X} = 0 \tag{2.11}$$

where $E(w)$ is the matrix with components $e_{ik} = \dfrac{\partial a_{ij}}{\partial\mathcal{X}_k}w_j$ [18], a_{ij} denotes the entries of the matrix A, and

$$H^p = \frac{\partial\left(\Phi^{\text{cv}} - \Phi^{\text{ds}}\right)}{\partial w}\left(w^p,\mathcal{X}^p,\dot{\mathcal{X}}^p\right),\; B^p = E\left(w^p\right) + \frac{\partial\left(\Phi^{\text{cv}} - \Phi^{\text{ds}}\right)}{\partial\dot{\mathcal{X}}}\left(w^p,\mathcal{X}^p,\dot{\mathcal{X}}^p\right)$$

$$C^p = \frac{\partial(\Phi^{\text{cv}} - \Phi^{\text{ds}})}{\partial\mathcal{X}}\left(w^p,\mathcal{X}^p,\dot{\mathcal{X}}^p\right)$$

Furthermore, the substitution of the second of Eqs. (2.3) and (2.9) into the linearized equations (2.11) transforms these into

$$A\left(\mathcal{X}^p\right)\delta\dot{w} + H^p\delta w + B^p_R\delta\dot{y}_R + B^p_{\text{CS}}\delta\dot{y}_{\text{CS}} + C^p_R\delta y_R + C^p_{\text{CS}}\delta y_{\text{CS}} = 0 \tag{2.12}$$

where

$$B^p_R = B^p\widetilde{K}^{-1}\widetilde{K}_cX_R,\; C^p_R = C^p\widetilde{K}^{-1}\widetilde{K}_cX_R,\; B^p_{\text{CS}} = B^p\widetilde{K}^{-1}\widetilde{K}_cX_{\text{CS}},\; C^p_{\text{CS}} = C^p\widetilde{K}^{-1}\widetilde{K}_cX_{\text{CS}}$$

In summary, from Eqs. (2.12) and (2.10), it follows that the linearization of the governing coupled fluid-structure equations (2.3) and (2.1) can be written in first-order form as follows

$$\begin{bmatrix} A^p & 0 & 0 & 0\\ 0 & M^p_{c,r} & 0 & 0\\ 0 & 0 & \mathbb{I} & 0\\ 0 & 0 & 0 & \mathbb{I}\end{bmatrix}\delta\dot{q} - \begin{bmatrix} -H^p & -B^p_R & -C^p_R & -C^p_{\text{CS}}\\ P^p_w & 0 & P^p_R & P^p_{\text{CS}}\\ 0 & \mathbb{I} & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\delta q - \begin{bmatrix} 0 & -B^p_{\text{CS}}\\ p^{\text{thrust}}_{c_T} & 0\\ 0 & 0\\ 0 & \mathbb{I}\end{bmatrix}\delta c = 0 \tag{2.13}$$

where

$$A^p = A\left(\mathcal{X}^p\right),\quad \delta q = \begin{bmatrix}\delta w^T & \delta\dot{y}_R^T & \delta y_R^T & \delta y_{\text{CS}}^T\end{bmatrix}^T \in \mathbb{R}^{N_{\text{CFD}}+12+n_{\text{CS}}}$$

ii Constant Operator Approximation and Justification

In principle, the building blocks of Eq. (2.13) are altitude-dependent and therefore require updating during the descent trajectory, which unfortunately cannot be performed analytically and therefore increases computational complexity. However, as previously mentioned, the free-stream atmospheric conditions vary slowly with altitude. Hence, the aforementioned building

blocks are approximated here with their initial descent values, thereby transforming (2.13) into

$$\begin{bmatrix} A^0 & 0 & 0 & 0 \\ 0 & M^0_{c,r} & 0 & 0 \\ 0 & 0 & \mathbb{I} & 0 \\ 0 & 0 & 0 & \mathbb{I} \end{bmatrix} \delta\dot{q} - \begin{bmatrix} -H^0 & -B^0_R & -C^0_R & -C^0_{\mathrm{CS}} \\ P^0_w & 0 & P^0_R & P^0_{\mathrm{CS}} \\ 0 & \mathbb{I} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \delta q - \begin{bmatrix} 0 & -B^0_{\mathrm{CS}} \\ p^{\mathrm{thrust}}_{c_T} & 0 \\ 0 & 0 \\ 0 & \mathbb{I} \end{bmatrix} \delta c = 0 \quad (2.14)$$

Unless deemed necessary for the sake of clarity, the subscript c, r introduced in (2.10) and superscript 0 introduced in (2.14) are dropped in the remainder of this paper for the sake of notational simplicity.

3 Real-Time DTP for State Prediction Based on Linear PMOR

Due to its fluid component, the DTP for FSI (2.14) is a linearized high-dimensional model (HDM) – specifically, of dimension $N_{\mathrm{HDM}} = N_{\mathrm{CFD}} + n_{\mathrm{CS}} + 12$. As such, this DTP cannot perform in real-time. Linear PMOR is performed here to reduce the dimension of its fluid component from N_{CFD} to $n_{\mathrm{CFD}} \ll N_{\mathrm{CFD}}$ – and therefore reduce N_{HDM} to $(n_{\mathrm{CFD}} + n_{\mathrm{CS}} + 12) \ll N_{\mathrm{HDM}}$ – and obtain an RT-DTP for FSI state and loads predictions.

(a) Subspace Approximation

Fundamental to PMOR is the assumption that the solution – in this case, the flow perturbation solution – may be well-approximated in a low-dimensional affine subspace $\mathcal{V}$ of dimension $n_{\mathrm{CFD}} \ll N_{\mathrm{CFD}}$, represented by a *right* reduced-order basis (ROB) $V \in \mathbb{R}^{N_{\mathrm{CFD}} \times n_{\mathrm{CFD}}}$. This can be written as

$$\delta w \approx \delta w_{\mathrm{ref}} + V \delta w_r \quad (3.1)$$

where $\delta w_r \in \mathbb{R}^{n_{\mathrm{CFD}}}$ is the vector of reduced coordinates of the fluid state perturbation and $\delta w_{\mathrm{ref}} \in \mathbb{R}^{N_{\mathrm{CFD}}}$ is a reference fluid state perturbation vector typically chosen so that the approximation (3.1) captures well the initial condition $\delta w(0)$. In the context of the application discussed in this paper, $\delta w(0) = 0$ and therefore δw_{ref} is set to zero.

Substituting the approximation (3.1) (with $\delta w_{\mathrm{ref}} = 0$) into the HDM (2.14) and applying a subscript r to δc to emphasize in this case the dependence of its optimal value (see Section 4(c)) on the reconstructed approximation $V\delta w_r$ rather than δw leads to

$$\begin{bmatrix} AV & 0 & 0 & 0 \\ 0 & M & 0 & 0 \\ 0 & 0 & \mathbb{I} & 0 \\ 0 & 0 & 0 & \mathbb{I} \end{bmatrix} \delta\dot{q}_r - \begin{bmatrix} -HV & -B_R & -C_R & -C_{\mathrm{CS}} \\ P_w V & 0 & P_R & P_{\mathrm{CS}} \\ 0 & \mathbb{I} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \delta q_r - \begin{bmatrix} 0 & -B_{\mathrm{CS}} \\ p^{\mathrm{thrust}}_{c_T} & 0 \\ 0 & 0 \\ 0 & \mathbb{I} \end{bmatrix} \delta c_r \quad (3.2)$$

where

$$\delta q_r = \begin{bmatrix} \delta w_r^T & \delta \dot{y}_R^T & \delta y_R^T & \delta y_{\mathrm{CS}}^T \end{bmatrix}^T \in \mathbb{R}^{n_{\mathrm{CFD}}+12+n_{\mathrm{CS}}}$$

(b) Training for Arbitrary Control Inputs

From its expression given in (2.14), it follows that the linearized HDM is parameterized only by rigid body motions and control inputs. Hence, the right ROB V to be constructed for transforming this HDM into a PROM – that is, an RT-DTP – needs to be trained only for these parameters. Since in this work c_T is assumed to be independent of w and therefore influences the fluid subsystem implicitly via its effort on the dynamics of the rigid body subsystem, it follows from (2.7) that it suffices to train V specifically arbitrary rigid body motions and control surface deflections. This is performed here by tailoring the frequency domain procedure developed in [13] for constructing and training a right ROB to aircraft rigid body and control surface modes, as follows:

(i) Consider the harmonic expansions $\delta w = \delta w_a e^{\mathcal{J}\omega t}$ and $\delta y = \delta y_a e^{\mathcal{J}\omega t}$, where $\mathcal{J}$ is the purely imaginary number ($\mathcal{J}^2 = -1$), ω is a circular frequency, and the subscript a designates the amplitude.

(ii) Substitute the above expansions into the fluid subsystem to obtain

$$\delta w_a = -\Big(\mathcal{J}\omega A + H\Big)^{-1}\Big(\mathcal{J}\omega\begin{bmatrix}B_R & B_{\text{CS}}\end{bmatrix} + \begin{bmatrix}C_R & C_{\text{CS}}\end{bmatrix}\Big)\delta y_a \tag{3.3}$$

(iii) Select an appropriate frequency band $\mathcal{B}_\omega = \begin{bmatrix}0 & \omega_{\text{max}}\end{bmatrix}$ – e.g. choose ω_{max} to be slightly higher than the highest frequency of the aircraft's dynamic stability modes (see [28]) – and sample $\mathcal{B}_\omega$ (e.g. uniformly) to obtain $(k+1)$ circular frequencies $\{0, \omega_1, \ldots, \omega_k\}$.

(iv) Let $n_{\text{modes}} = n_{\text{RB}} + n_{\text{CS}} = 6 + n_{\text{CS}}$ denote the total number of rigid body and control surface modes. For each sampled frequency $\omega_m \in \mathcal{B}_\omega$, $m = 1, \cdots, k$, and each amplitude $\delta y_a \in \{e_1, \ldots, e_{n_{\text{modes}}}\}$, where e_l denotes l-th column of the identity matrix of dimension n_{modes}, solve Eq. (3.3) (i.e. train for each rigid body motion and control surface deflection separately).

(v) Collect the real and imaginary parts of the computed solutions of (3.3) into the snapshot matrix $S \in \mathbb{R}^{N_{\text{CFD}} \times n_{\text{modes}}(2k+1)}$ – that is,

$$S = \begin{bmatrix}\ldots & \delta w_a^{l0} & \Re\{\delta w_a^{l1}\} & \Im\{\delta w_a^{l1}\} & \ldots & \Re\{\delta w_a^{lk}\} & \Im\{\delta w_a^{lk}\} & \ldots\end{bmatrix}$$

where $\Re$ and $\Im$ designate the real and imaginary parts of a complex number, respectively, and

$$\delta w_a^{lm} = -\Big(\mathcal{J}\omega_m A + H\Big)^{-1}\Big(\mathcal{J}\omega_m\begin{bmatrix}B_R & B_{\text{CS}}\end{bmatrix} + \begin{bmatrix}C_R & C_{\text{CS}}\end{bmatrix}\Big)e_l\,,$$
$$l = 1, \cdots n_{\text{modes}};\ \ m = 0, \cdots k;\ \ \text{and}\ \omega_0 = 0$$

(vi) Compress the snapshot matrix S using the singular value decomposition method (SVD) – that is, perform $S = U\Sigma Z^T$ – and truncate $U \in \mathbb{R}^{N_{\text{CFD}} \times n_{\text{modes}}(2k+1)}$ using the relative singular value energy criterion [29] to obtain a right ROB $V \in \mathbb{R}^{N_{\text{CFD}} \times n_{\text{CFD}}}$, where $n_{\text{CFD}} \leq n_{\text{modes}(2k+1)}$.

The right ROB construction and training procedure described above is equivalent to a proper orthogonal decomposition (POD) method of snapshots [30]. It can also be viewed as approximating the transfer function between the states of the structure and fluid subsystems at the set of sampled frequencies. Thus, as long as this set includes the frequencies that dominate the response of the fluid subsystem, the computed solution snapshots will contain the information necessary for constructing an accurate right ROB V.

(c) Stabilization via Petrov-Galerkin Projection

To complete the construction of the RT-DTP for FSI state and load predictions, the first row of Eq. (3.2), which is associated with the fluid component of this HDM, must be projected onto a *left* ROB $W \in \mathbb{R}^{N_{\text{CFD}} \times n_{\text{CFD}}}$ representing a subspace of constraints $\mathcal{W}$, to transform this HDM component, in general, into a Petrov-Galerkin fluid PROM. Indeed, this leads to a low-dimensional DTP for FSI that can be written as

$$\delta\dot{q}_r - \underbrace{\begin{bmatrix}-H_r & -B_{R,r} & -C_{R,r} & -C_{\text{CS},r}\\ M^{-1}P_{w,r} & 0 & -M^{-1}P_R & M^{-1}P_{\text{CS}}\\ 0 & \mathbb{I} & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}}_{A_{\text{RT-DTP}}}\delta q_r - \underbrace{\begin{bmatrix}0 & -B_{\text{CS},r}\\ M^{-1}p_{cT}^{\text{thrust}} & 0\\ 0 & 0\\ 0 & \mathbb{I}\end{bmatrix}}_{B_{\text{RT-DTP}}}\delta c_r = 0 \tag{3.4}$$

where

$$H_r = A_r^{-1}W^T H V,\ A_r = W^T A V,\ B_{R,r} = A_r^{-1}W^T B_R,\ C_{R,r} = A_r^{-1}W^T C_R$$
$$B_{\text{CS},r} = A_r^{-1}W^T B_{\text{CS}},\ C_{\text{CS},r} = A_r^{-1}W^T C_{\text{CS}},\ P_{w,r} = P_w V$$

Specifically, the dimension of the linear fluid PROM represented by the first row of Eq. (3.4) is $n_{\text{CFD}} \ll N_{\text{CFD}}$ and that of the RT-DTP for FSI is $n_{\text{RT-DTP}} = (n_{\text{CFD}} + n_{\text{CS}} + 12) \ll N_{\text{HDM}}$. However,

the numerical stability of the fluid PROM represented by the building block H_r is not guaranteed by default – that is, due to projection, $-H_r$ may have eigenvalues in the right-half of the complex plane even when $-H$ does not have such eigenvalues. This is particularly true when W is chosen to be $W = V$, in which case the linear fluid PROM is known as a Galerkin PROM. However, a procedure was presented in [31] for constructing a left ROB $W \neq V$ that guarantees the numerical stability of the fluid PROM, which is necessary for guaranteeing the stability of the RT-DTP (3.4) for FSI state and load predictions. This procedure is adopted here for constructing the left ROB W.

4 DTI for MPC

The MPC controller adopted in this work is that described in details in [32]. Here, it is overviewed in order to keep this paper as self-contained as possible. Because this MPC controller is informed by the RT-DTP (3.4) for FSI state and load predictions, it is referred to in this paper as a digital twin. Because it interacts with sensor data about the rigid body state and control surface deflections of the individual UAV it is applied to, it is more specifically referred to as a DTI.

(a) Discretization

To implement the control scheme in a digital flight computer, the semi-discrete model (3.4) is discretized by the matrix exponential method under the assumption of a zero-order hold (ZOH) control (see Appendix W8.7 of [33]), where ZOH control refers to the hypothesis of constant control inputs during a single time-step. Given a time-step size Δt, this method updates δq_r from time-index k to time-index $k+1$ as follows

$$\delta q_r^{k+1} = A^d_{\text{RT-DTP}}\, \delta q_r^k + B^d_{\text{RT-DTP}}\, \delta c_r^{k,\star} \tag{4.1}$$

where

$$A^d_{\text{RT-DTP}} = e^{\Delta t A_{\text{RT-DTP}}},\ B^d_{\text{RT-DTP}} = \left(\int_0^{\Delta t} e^{\eta A_{\text{RT-DTP}}} d\eta \right) B_{\text{RT-DTP}}$$

and the superscript $\star$ in $\delta c_r^{k,\star}$ emphasizes that this control input is optimal (an determined in Section 4(c)).

As pointed out in [34], the matrix exponential method under the assumption of a ZOH control utilizes the *exact* solution of a first-order system of linear ordinary differential equations over a single sample period. Therefore, it represents a *discrete equivalent* of the exact solution, provided that the applied control is constant over the sample period.

(b) State-Estimator for State Feedback

To form a *closed-loop* system, the control scheme is equipped with a *state-estimator* and the vector of control inputs δc_r^k is adjusted appropriately. Typically, the state-estimator is chosen of the form

$$\delta \hat{\dot{q}}_r^{k+1} = A^{\text{SE},d}_{\text{RT-DTP}}\, \delta \hat{q}_r^k + B^{\text{SE},d}_{\text{RT-DTP}}\, \delta c^k + L^d \delta z^k \tag{4.2}$$

where $L^d \in \mathbb{R}^{n_{\text{RT-DTP}} \times (n_{\text{CS}}+12)}$ is the state-estimator gain matrix whose evaluation is mentioned below, $\delta z^k \in \mathbb{R}^{n_{\text{CS}}+12}$ is the vector of measured rigid body state perturbations (displacements and rates) and control surface deflection perturbations at time-index k,

$$A^{\text{SE},d}_{\text{RT-DTP}} = A^d_{\text{RT-DTP}} - L^d C,\ C = \begin{bmatrix} 0 & \mathbb{I} & 0 & 0 \\ 0 & 0 & \mathbb{I} & 0 \\ 0 & 0 & 0 & \mathbb{I} \end{bmatrix},\ \text{and}\ B^{\text{SE},d}_{\text{RT-DTP}} = B^d_{\text{RT-DTP}}$$

Hence, C is the output matrix that extracts from $\delta \hat{q}_r^k$ only the rigid body states and control surface deflections – that is, *no* measurements of the fluid state are assumed to be available. As for the vector of adjusted control inputs δc^k – that is, the inputs that are actually applied to the UAV and

the state-estimator – it can be written as

$$\delta c^k = \delta c_r^{k,\star} + K_c^d(\delta \hat{q}_r^k - \delta q_r^{k,\star}) \tag{4.3}$$

where $\delta c_r^{k,\star}$ is, again, the vector of optimal control inputs, K_c^d is the controller gain matrix mentioned below, and $\delta q_r^{k,\star}$ is computed using the RT-DTP (4.1) as well as $\delta c_r^{k,\star}$. Therefore, the state-estimator is designed to track both the UAV, via feedback from sensor measurements, and the trajectory produced by the optimal control, via the reduced-order state feedback.

Ideally, the gain matrices L^d and K_c^d would be computed based on a discretized (2.14) using standard techniques from control theory, then reduced according to the subspace approximation (3.1). However, given the typical size of a CFD mesh, such techniques are not computationally tractable. Therefore, L^d and K_c^d are computed in this work using the reduced-order Riccati method for time-discrete systems described in [32].

(c) Robust Reduced-Order MPC Controller

All that remains to fully describe the MPC controller is to describe the computation of $\delta c_r^{k,\star}$. For this purpose, it is noted that at the core of the controller is the real-time *open-loop* trajectory δq_r^j, $j = 1, 2, \cdots$, computed using the time-discrete RT-DTP (4.1). This trajectory is generated by solving a *receding horizon* optimal control problem (OCP), which may be written as

$$\begin{aligned}
\delta c_r^{k,\star} = \min_{\delta q_r^j,\, \delta c_r^j} \; & \left\| \delta q_r^{k+n_T|k} \right\|_{Q_T}^2 + \sum_{j=k}^{k+n_T-1} \left\| \delta q_r^{j|k} \right\|_Q^2 + \left\| \delta c_r^{j|k} \right\|_R^2 \\
\text{subject to } & \delta q_r^{i+1|k} = A^d_{\text{RT-DTP}}\, \delta q_r^{i|k} + B^d_{\text{RT-DTP}}\, \delta c_r^{i|k} \\
& Q_c \delta q_r^{i|k} \in \mathcal{Q}, \quad \delta c_r^{i|k} \in \mathcal{C} \\
& \delta q_r^{k+n_T|k} \in \mathcal{Q}_T, \quad \delta q_r^{k|k} = \delta q_r^k
\end{aligned} \tag{4.4}$$

where: the superscript $k, \star$ emphasizes that even though the above OCP computes n_T optimal solutions, only that indexed by k (the first one in the *time horizon*) is retained by the MPC controller; n_T denotes the length of the time horizon; Q and R are two positive-definite weighting matrices for the state and control, respectively; Q_T is the positive-definite terminal cost weighting matrix; the matrix Q_c describes the state constraints; the sets $\mathcal{Q}$ and $\mathcal{C}$ describe the state and control constraint sets, respectively; and $\mathcal{Q}_T$ describes the terminal set. The superscript $i|k$ designates the dependence of a quantity on the state at the time-index k and therefore in general, $\delta q_r^{k+i|k} \neq \delta q_r^{k+i}$. The solution of the above OCP consists of the optimal vector of control inputs computed at time-index k, $\delta c_r^{k,\star}$. The optimal reduced-order state at time-index $k+1$ is generated by advancing (4.1) using $\delta c_r^{k,\star}$ and $\delta q_r^{k,\star} = \delta q_r^k$. Hence, to generate a trajectory, problem (4.4) is first solved using the initial condition δq_r^0. Then, using the computed optimal input $\delta c^{0,\star}$, δq_r^1 is computed by time-advancing the RT-DTP (4.1) and the OCP (4.4) is solved again for $k = 1$. This process is repeated until the trajectory returns to the glideslope – that is, all perturbations are eliminated (see below). Because each instance of the OCP (4.4) has different start and end times, the resulting control scheme is referred to as a receding horizon control scheme.

The OCP (4.4), the discrete state-estimator (4.2), and the control adaptation (4.3) are collectively referred to as the control scheme. Because this scheme is informed by the RT-DTP (4.1), which is a PROM, it can be described as a reduced-order MPC (ROMPC) control scheme.

Solving the OCP (4.4) is significantly more computationally intensive than computing the updates (4.1) and (4.2). For this reason, two different time-step sizes are used: 1) one time-step size Δt_{OCP} for the solution of problem (4.4) that is ideally roughly equal to the wall-clock time required to solve this problem (though it may be chosen to be larger to obtain a sufficiently long time horizon without an excessive number of time steps); and 2) a finer time-step size $\Delta t_{\text{SE}} = \Delta t_{\text{OCP}}/m$ for performing the updates (4.1) and (4.2), where m is an integer that can be tuned for real-time performance. Additionally, the solution of the OCP (4.4) may be accelerated

by observing that the variables $\delta q_r^{i|k}$ are entirely dependent upon the initial condition δq_r^k and the optimal control inputs $\delta c_r^{i|k,\star}$: therefore, $\delta q_r^{i|k}$ may be eliminated and (4.4) may be rewritten entirely in terms of the variables $\delta c_r^{i|k}$ and the *known* initial condition.

5 Verification and Demonstration

Here, the two-level DT for AL graphically depicted in Figure 1 is verified by numerical simulation. Specifically, the objective is set to tracking a constructed glideslope and correcting lineup errors during the approach induced by a specified initial offset. In this context, the DTI for MPC receives "sensor" measurements from a *high-dimensional*, *nonlinear* DTP for flight dynamics based on the ALE computational framework described in Section 2(a).ii; and applies the control inputs determined by the ROMPC scheme defined by (4.2)–(4.4) and informed by the RT-DTP (3.4). Note that even when the side-slip angle is zero and the CFD mesh is symmetric, numerical imperfections (errors) lead to a nonzero lateral force and nonzero roll and yaw moments. Thus, regardless of the specific initial conditions, the numerical simulation includes the following disturbances:

(i) The presence in the high-dimensional, nonlinear DTP of nonlinear effects that are not accounted for in the construction of the RT-DTP, which is a linear PROM.
(ii) Projection and modeling errors due to PMOR.
(iii) Nonzero lateral force and roll and yaw moments, even when the side-slip angle is zero.

In all numerical experiments discussed below, the glideslope is defined by: the flight-path angle $\gamma_0 = -3.5°$; the free-stream velocity $v_\infty = 15\,\mathrm{m\,s^{-1}}$; the initial altitude of $90\,\mathrm{m}$; and the slowly time-varying pitch angle $\theta_{R,y}^{\mathcal{I},p}$, thrust c_T^p, and elevator deflection $\theta_{CS,e}^p$ computed as described in [27]. The initial angle of attack is set to $\alpha_0 = \alpha_0^p = 1.79°$.

(a) High-Dimensional and Low-Dimensional Computational Models

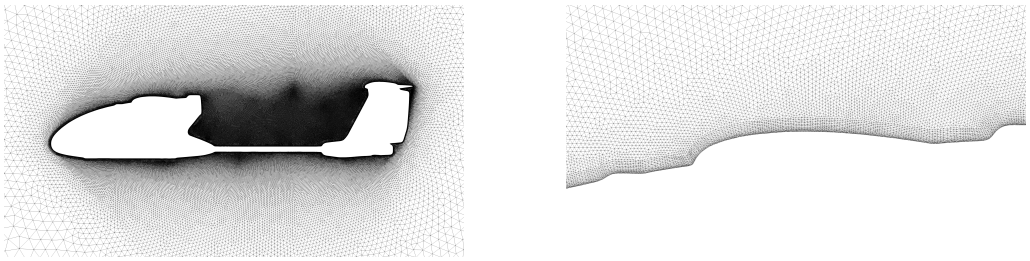
i Nonlinear High-Dimensional CFD and 6dof CSD Models

The UAV considered in this work is an off-the-shelf hobby UAV that the authors have laser-scanned to obtain a discrete representation of its surface. It has a single propeller and four control surfaces (see Figure 3): two ailerons that are coupled to deflect opposite one another; one elevator; and one rudder. Treating the coupled ailerons as a single control surface mode results in $n_{\mathrm{CS}} = 3$. The nonlinear ALE CFD HDM underlying the construction of the linearized RT-DTP for FSI – that is, the nonlinear DTP for flight dynamics – is based on a CFD mesh with 2,930,804 grid points and 17,223,270 tetrahedral primal elements, extending in a sphere of radius $30\,\mathrm{m}$ that encloses the aircraft: this mesh leads to the high dimension $N_{\mathrm{CFD}} = 17,584,824$. Figure 4a depicts a slice of the CFD mesh at $y = 0$ (note the absence of the propeller) and Figure 4b zooms on a region of the boundary layer (BL). The CFD mesh is built so that $y^+ \approx 30$ and the BL is discretized by twelve prism layers. The Reichardt nonlinear wall function [35] is used to model the wall boundary effects below $y^+ \approx 30$. The resulting nonlinear, semi-discrete fluid HDM is discretized in time using the three-point backward difference (BDF2) scheme [36]. On the other hand, the 6dof semi-discrete CSD model representing the rigid body dynamics of this UAV is time-discretized using the generalized-α method [23]. The FSI coupling is performed using the improved serial staggered (ISS) algorithm described in [37]. The time-step size for the structural subsystem and that for the coupled FSI system are set to Δt_{SE} (see Section 5(b)). Hence, at each coupled time-step, the time-step size for the fluid subsystem is either set to Δt_{SE} or the fluid subsystem is subcycled to meet nonlinear numerical stability requirements.

ii Low-Dimensional Linearized CFD Model and Accuracy Verification

The linearized RT-DTP (3.4) is trained in the frequency band $\mathcal{B}_\omega = \begin{bmatrix} 0 & 10\pi \end{bmatrix}\,\mathrm{rad\,s^{-1}}$ that is uniformly sampled for this purpose in $m = 10$ steps of size $\pi\,\mathrm{rad\,s^{-1}}$. This results in 189 snapshots (126 corresponding to the group of rigid body modes and 63 to that of control surface modes). In order to ensure that during their compression using SVD, the snapshots corresponding to the

Figure 3: Off-the-shelf hobby UAV with a single propeller and four control surfaces.



(a) (b)

Figure 4: RANS CFD mesh: slice at $y = 0$ (a); and zoomed view on the region of the boundary layer (b).

group of rigid body modes do not dominate their counterparts corresponding to the group of control surface modes which typically have much smaller magnitudes, each group is normalized by its average two-norm. Compressing the computed solution snapshots and truncating the resulting left singular vectors according to the relative singular value energy criterion with the tolerance of 99.9999% (i.e. only 10^{-6} of the relative energy is truncated) leads to a fluid right ROB and then a fluid PROM of dimension $n_{\text{CFD}} = 88$ (which represents a reduction of the dimension of the fluid HDM by a factor of 2×10^5). Thus, the dimension of the constructed RT-DTP (3.4) is $n_{\text{CFD}} + 12 + n_{\text{CS}} = 103$.

As mentioned in Section 3(c), the left ROB W is computed using the stabilization algorithm presented in [31], which is implemented in MATLAB [38] using the convex optimization solver package CVX [39,40].

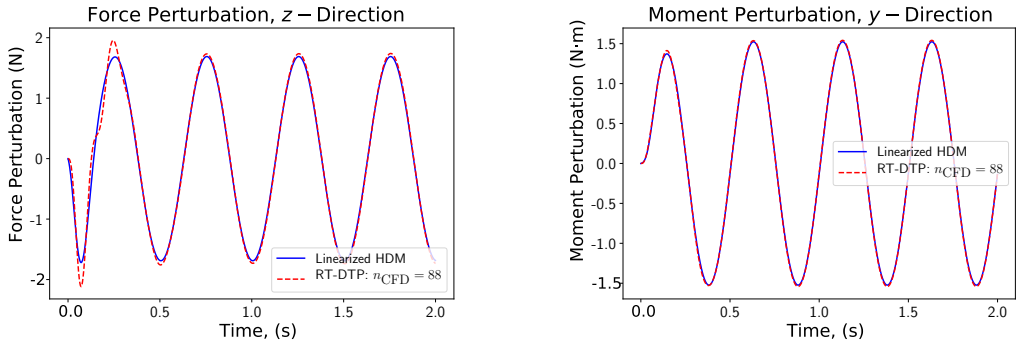
The accuracy of the constructed linear RT-DTP is verified here for the flow response problem to the following prescribed rigid body motion of the UAV

$$\delta y_R = \delta y_{R,a} \left(1 - e^{-\omega^2 t^2}\right) \sin(\omega t), \quad \delta y_{\text{CS}} = \delta y_{CS,a} \left(1 - e^{-\omega^2 t^2}\right) \sin(\omega t)$$

where

$$\delta y_{R,a} = \left[\begin{bmatrix} 0.01 & 0.01 & 0.01 \end{bmatrix} \text{m} \quad \begin{bmatrix} 8.73 & 8.73 & 8.73 \end{bmatrix} \times 10^{-3} \text{rad}\right]^T$$

$$\delta y_{CS,a} = \begin{bmatrix} 8.73 & 8.73 & 8.73 \end{bmatrix}^T \times 10^{-2} \text{rad}, \quad \omega = 4\pi \, \text{rad}\, \text{s}^{-1}$$



(a) (b)

Figure 5: Numerical predictions of the vertical force (a) and lateral moment (b) perturbations using the linearized fluid HDM (solid) and linearized RT-DTP (dashed).

The amplitudes $\delta y_{R,a}$ and $\delta y_{CS,a}$ are set such that the amplitude of the resulting force and moment perturbations are of similar magnitudes. Given in this case the continuous nature of the forcing function, the ZOH is not appropriate for the treatment of the control surface deflection rate. For this reason, the RT-DTP (3.4) is also time-discretized using BDF2 [36] and the constant time-step $\Delta t = 0.001\,\text{s}$. The flow response is computed for $n_T = 2,000$ time-steps – that is, four periods of the excitation – using both the linearized version of the fluid HDM and the linearized RT-DTP.

Figures 5a and 5b report on the computed perturbations to the aerodynamic vertical force and lateral (i.e. pitch) moment, respectively. They show that for this problem at least, the linearized RT-DTP reproduces well the solution computed using the linearized HDM, particularly in the later part of the numerical simulation where $\left(1 - e^{-\omega^2 t^2}\right) \approx 1$.

To measure the accuracy delivered by the linearized RT-DTP, the following definitions leading to a force / moment-based measure of the RT-DTP-based global solution error are introduced

$$\upsilon_{\text{RT-DTP}} = \sum_{i=0}^{n_T} \xi_{\text{RT-DTP},i} \left| P_{w,r}\delta w_r^i + P_R \delta y_R^i + P_{\text{CS}}\delta y_{\text{CS}}^i \right|$$

$$\upsilon_{\text{HDM}} = \sum_{i=0}^{n_T} \xi_{\text{HDM},i} \left| P_w \delta w^i + P_R \delta y_R^i + P_{\text{CS}}\delta y_{\text{CS}}^i \right|$$

$$\varepsilon_{\text{RT-DTP},j} = \left| \upsilon_{\text{RT-DTP},j} - \upsilon_{\text{HDM},j} \right| / \upsilon_{\text{HDM},j},\ j = 1,\ldots,6$$

where for $(\cdot) \in \{\text{RT-DTP}, \text{HDM}\}$, $\upsilon_{(\cdot)}$ is the discrete approximation of the integral

$$\int_0^{n_T \Delta t} \left| \Delta f_{c,(\cdot)}^{\text{aero}} \right| dt,$$

the weights $\xi_{(\cdot),i}$ are determined using Simpson's Rule [41], Δf_c^{aero} is the vector of perturbations to the aerodynamic forces and moments, and $\varepsilon_{\text{RT-DTP},j}$ is the relative error of the approximated integral for each component of the aerodynamic forces and moments.

Tables 1 and 2 report the values of $\upsilon_{(\cdot)}$ and $\varepsilon_{\text{RT-DTP}}$ for all forces and moments, respectively. The largest relative error is in the lateral force, at 4.4%; the smallest one is in the horizontal force, at 0.3%. These values demonstrate that the RT-DTP is sufficiently accurate and therefore may be used as part of the proposed two-level DT for AL.

	Force$-x$		Force$-y$		Force$-z$	
$(\cdot)$	HDM	RT-DTP	HDM	RT-DTP	HDM	RT-DTP
$v_{(\cdot),j}$ (N)	0.239	0.238	0.910	0.950	2.136	2.179
$\varepsilon_{\text{RT-DTP},j}$ (%)	–	0.287	–	4.357	–	2.029

Table 1: Approximate integrals of the force perturbation magnitudes predicted using the RT-DTP and corresponding relative errors.

	Moment$-x$		Moment$-y$		Moment$-z$	
$(\cdot)$	HDM	RT-DTP	HDM	RT-DTP	HDM	RT-DTP
$v_{(\cdot),j}$ (N m)	1.069	1.078	1.891	1.912	0.478	0.493
$\varepsilon_{\text{RT-DTP},j}$ (%)	–	0.865	–	1.112	–	3.270

Table 2: Approximate integrals of the moment perturbation magnitudes predicted using the RT-DTP and corresponding relative errors.

(b) OCP Constraints, Parameters, Gains, and Solution

To completely specify the controller – that is, the OCP (4.4) and gain matrices L^d and K_c^d, the constraints and cost function parameters must be specified. Table 3 lists the constraints based on the *total* values of the various variables rather than their perturbations. To obtain the constraints on the perturbations (which are needed for the solution of the OCP (4.4)), the minimum and maximum values of the variables along the glideslope are subtracted from their lower and upper bounds, respectively; these values are listed in Table 4. Using these constraints, the state constraint matrix becomes $Q_c = C$; and the state constraint set $\mathcal{Q}$ and control constraint set $\mathcal{C}$ become polyhedra.

The parameters Q and R of the cost function are set to

$$Q = C^T W_z^T W_z C + 10^{-5}\mathbb{I}, \quad R = W_u^T W_u \tag{5.1}$$

where $\mathbb{I} \in \mathbb{R}^{(n_{\text{CFD}}+12+n_{\text{CS}})\times(n_{\text{CFD}}+12+n_{\text{CS}})}$,

$$W_z = \text{diag}\left(\begin{bmatrix} 50 & 50 & 50 & 100 & 100 & 100 & 25 & 25 & 25 & 750 & 750 & 750 & 1 & 1 & 1 \end{bmatrix}\right),$$

$$W_u = \text{diag}\left(\begin{bmatrix} 10 & 150 & 150 & 150 \end{bmatrix}\right),$$

and the role of the regularization term in Q is to ensure that Q is symmetric positive definite, given that the assumption of no fluid measurements leads to $C^T W_z^T W_z C$ being symmetric positive *semi*-definite. The terminal cost Q_T is computed using its definition, namely, the solution of the time-discrete algebraic Riccati equation using $A^d_{\text{RT-DTP}}$ and $B^d_{\text{RT-DTP}}$, as well as the parameters Q and R specified in (5.1). To this end: both $A^d_{\text{RT-DTP}}$ and $B^d_{\text{RT-DTP}}$ are evaluated using the larger time-step size Δt_{OCP} rather than its smaller counterpart Δt_{SE}, and Δt_{OCP} and Δt_{SE} are set to $\Delta t_{\text{OCP}} = 0.1\,\text{s}$ and $\Delta t_{\text{SE}} = 0.01\,\text{s}$, respectively; and Q_T is computed using the SciPy [42] implementation of the Riccati equation solver.

To compute the gain matrix K_c^d, the same matrices Q and R used in the definition of the OCP (4.4) are used in the time-discrete reduced-order Riccati method [32]. To compute L^d, two new cost matrices are introduced

$$Q_w = 100\mathbb{I} \quad \text{and} \quad Qy = W_y^T W_y$$

where $\mathbb{I} \in \mathbb{R}^{(n_{\text{CFD}}+12+n_{\text{CS}})\times(n_{\text{CFD}}+12+n_{\text{CS}})}$ and

$$W_y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 10^{-5} & 10^{-5} & 10^{-5} \end{bmatrix}$$

Again, SciPy is applied to compute the solution of the time-discrete algebraic Riccati equations involved in the process of computing the gain matrices K_c^d and L^d.

As for the terminal constraint set $\mathcal{Q}_T$, it is generated as follows. First, the matrix $A^{d,K}_{\text{RT-DTP}} = A^d_{\text{RT-DTP}} + B^d_{\text{RT-DTP}} K_c^d$ is formed. If the polyhedron control constraints are expressed in the form $C_c \delta c_r^k \leq b_c$, they may be appended to the state constraints as $C_c K_c^d \delta q_r^k \leq b_c$. Then, $\mathcal{Q}_T$ is computed using the algorithm presented in [43], $A^{d,K}_{\text{RT-DTP}}$, and the combined state / control constraints (assuming an identity output matrix). The optimization problem underlying the aforementioned algorithm is solved in MATLAB using YALMIP [44] – which in turn invokes CPLEX [45] and MPT3 [46].

Finally, the OCP (4.4) is solved using a fork of qpOASES [47] that can be found at [48].

$\dot{u}_R$ ($\mathrm{m\,s^{-1}}$)	u_R (m)	$\dot{\theta}_R$ ($^\circ\,\mathrm{s^{-1}}$)	θ_R ($^\circ$)	$\dot{\theta}_{\text{CS}}$ ($^\circ\,\mathrm{s^{-1}}$)	θ_{CS} ($^\circ$)	c_T (N)
± 2	± 10	± 1.0	± 2.5	± 150	± 30	[0 10]

Table 3: Total state and control constraints.

	$\dot{\theta}^p_{R,y}$ ($^\circ\,\mathrm{s^{-1}}$)	$\theta^p_{R,y}$ ($^\circ$)	$\dot{\theta}^p_{\text{CS},e}$ ($^\circ\,\mathrm{s^{-1}}$)	$\theta^p_{\text{CS},e}$ ($^\circ$)	c^p_T (N)
Minimum	-1.24×10^{-3}	-0.11	-7.79×10^{-4}	0.24	1.07
Maximum	-1.23×10^{-3}	0	-7.56×10^{-4}	0.31	1.09

Table 4: Minimum and maximum values of the *nonzero* components of the state along the glideslope.

(c) Numerical Simulation Results

The state of the structural subsystem is initialized with the perturbation offset defined by

$$\delta u_R(0) = \begin{bmatrix} \mathbb{I} & 0 \end{bmatrix} \delta y_R(0) = \begin{bmatrix} 1.0 & -1.0 & 1.0 \end{bmatrix} \mathrm{m},$$

$$\delta \dot{\theta}_{R,y}(0) = 1.24 \times 10^{-3}\,^\circ\,\mathrm{s^{-1}}, \quad \theta_{\text{CS},e}(0) = -0.307\,^\circ$$

and zero initial perturbations for all other state variables. The initial free-stream velocity and angle of attack are kept unchanged: however, the initial altitude of the UAV after application of the above offset becomes $91.03\,\mathrm{m}$. The state of the fluid subsystem is initialized by the HDM-based steady-state solution (or in other words, the steady-state solution produced by the nonlinear DTP for flight dynamics) of the flow problem around the UAV after the the above offset is applied. The initial perturbation of the state of the fluid subsystem is then determined by subtracting from the aforementioned steady-state solution the state associated with the initial condition for the glideslope. Using this FSI initialization, the simulation of the controlled flight dynamics of the UAV is performed in the time-interval $\begin{bmatrix} 0 & 25 \end{bmatrix}\mathrm{s}$, which is approximately the time it takes for the designed controller to eliminate the perturbations.

Figures 6 and 7 report on the longitudinal (δx, δz and $\delta\theta_{R,y}$) and lateral (δy, $\delta\theta_{R,x}$ and $\delta\theta_{R,z}$) components of the dynamic state of the UAV, respectively. Specifically, they display: the time-histories of the optimal open loop state corrections – measured with respect to the counterpart states along the glideslope – computed by the RT-DTP (using the finer time-step size Δt_{SE}, to have the same resolution used to compute δc^k); and the corresponding optimal closed-loop deviations predicted by the nonlinear DTP for flight dynamics. These figures reveal that as desired, the MPC controller reduces the initial offset and stabilizes the UAV's state components at final values close to zero. The most notable discrepancies are the small, steady-state offsets in the rotation parameters and a corresponding steady-state offset in the vertical direction. This

is unsurprising, because of the additional intrinsic disturbances listed at the beginning of this section – and in particular, the nonzero lateral force and nonzero roll and yaw moments that arise from numerical errors at a zero side-slip angle.

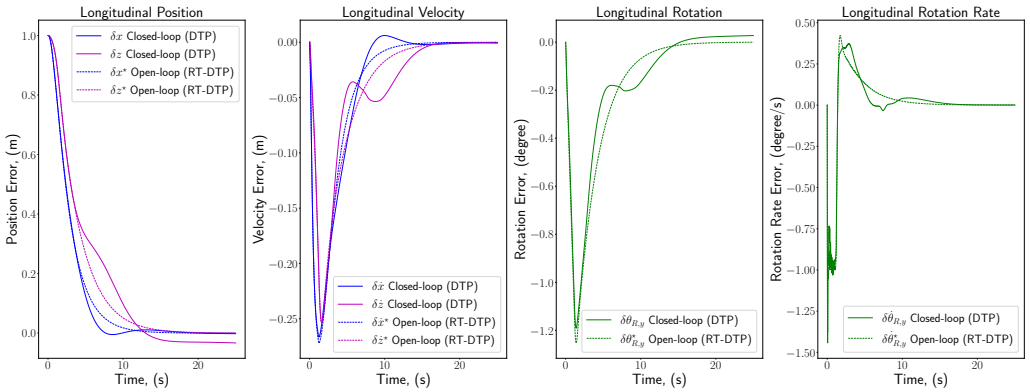


Figure 6: Longitudinal dynamics: optimal open-loop corrections computed by the RT-DTP (dashed); and achieved closed-loop deviations as predicted by the nonlinear DTP for flight dynamics (solid) – corrections and deviations are measured with respect to the glideslope.

Note that in the computation of the glideslope and control inputs, trim is achieved using the thrust, angle of attack, and elevator deflection as parameters. Specifically, due to the symmetry of the considered UAV, its CFD mesh, and therefore the flow with respect to the x-z plane, only the equations of equilibrium associated with the $x-$ translation, $z-$ translation, and $y-$ rotation are considered for computing the trim point. However, because of round-off and other numerical errors, the other force and moments are relatively negligible but nonzero. Hence, they generate disturbances that simulate noise and therefore render the simulation even more realistic. Thus, it is unsurprising that at any time $t \in \begin{bmatrix} 0 & 25 \end{bmatrix}$s: the largest discrepancies between the optimal open-loop corrections and optimal closed-loop deviations occur in the lateral components of the UAV's state; and both $\delta\theta_{R,x}$ and $\delta\theta_{R,z}$ end with relatively negligible but nonzero steady-state offsets as a result of the aforementioned spurious forces and moment. The similar steady-state offsets for δz and $\delta\theta_{R,y}$ can be explained by observing that the small steady-state offsets in the other rotations affect both the angle of attack and side-slip angle: thus, they result in less lift, which in turn requires a small pitch correction to achieve equilibrium. However, it is more desirable to have no deviations from the glideslope: this can be achieved by modifying the control scheme to include, for example, an integrator that helps reducing steady-state offsets in the presence of disturbances.

The time-histories of the optimal open-loop control input corrections $\delta c_r^{k,\star}$ and closed-loop deviations δc^k are reported in Figure 8: here, the discrepancies between the two signals are much smaller and the convergence of each signal to zero is faster that in Figures (6)–(7). The time-histories of the optimal open-loop control surface deflections are displayed in Figure 9: the reader can observe that for the deflections (which are the integral of the deflection rates), the aforementioned discrepancies are larger; however they are only of the order of a small fraction of a degree and therefore of not much practical significance. The reader can also observe that the steady-state offsets seen in Figures (6)–(7) also appear here: this is expected, given that any offset in the trajectory would either be caused by offsets in the control surface deflections, or counteracted by offsets in these deflections to prevent them from amplifying in time.

Table (5) highlights the real-time capability of the proposed two-level DT for AL. It reports the execution time of both: the control updates consisting of evaluating (4.1)–(4.3) (where, as mentioned above, (4.1) is evaluated using Δt_{SE}); and the solution the OCP (4.4). More specifically,

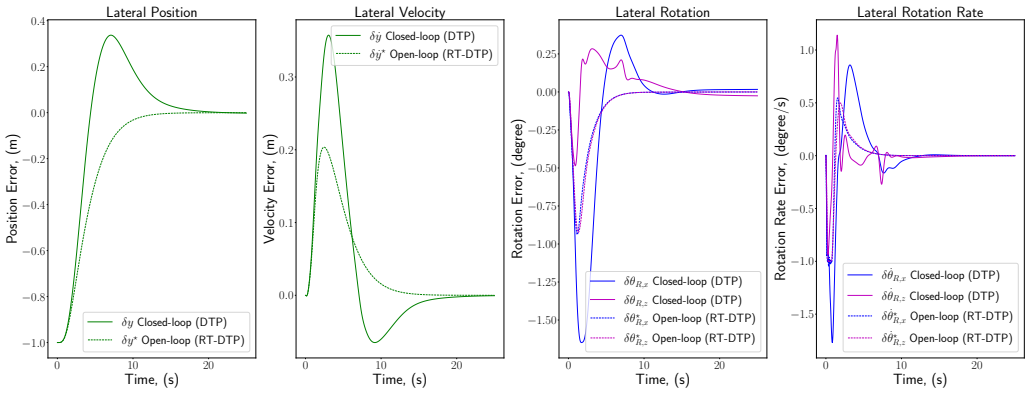


Figure 7: Lateral dynamics: optimal open-loop corrections computed by the RT-DTP (dashed); and achieved closed-loop deviations as predicted by the nonlinear DTP for flight dynamics (solid) – corrections and deviations are measured with respect to the glideslope.

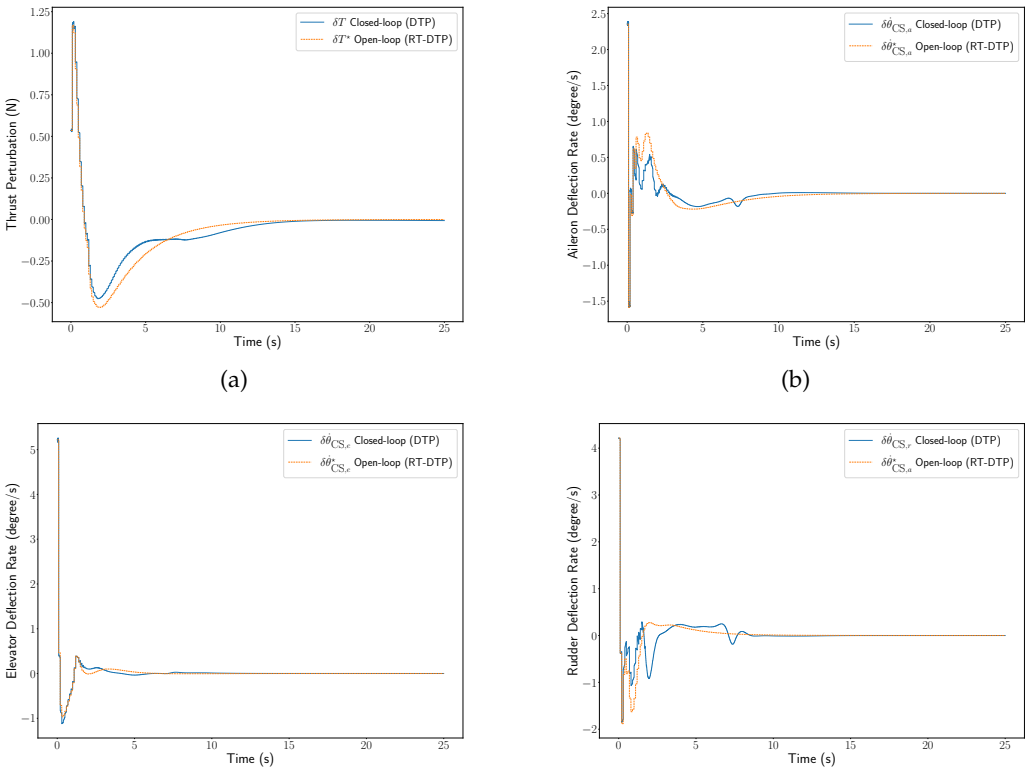


Figure 8: Optimal open-loop control input corrections computed by the RT-DTP (orange) and achieved closed-loop deviations as predicted by the nonlinear DTP for flight dynamics (blue): (a) thrust; (b) aileron deflection rate; (c) elevator deflection rate; and (d) rudder deflection rate – corrections and deviations are measured with respect to the glideslope.

it reports the number of function calls for both sets of control updates and the corresponding average execution wall clock time. For reference, the numerical simulation described above was

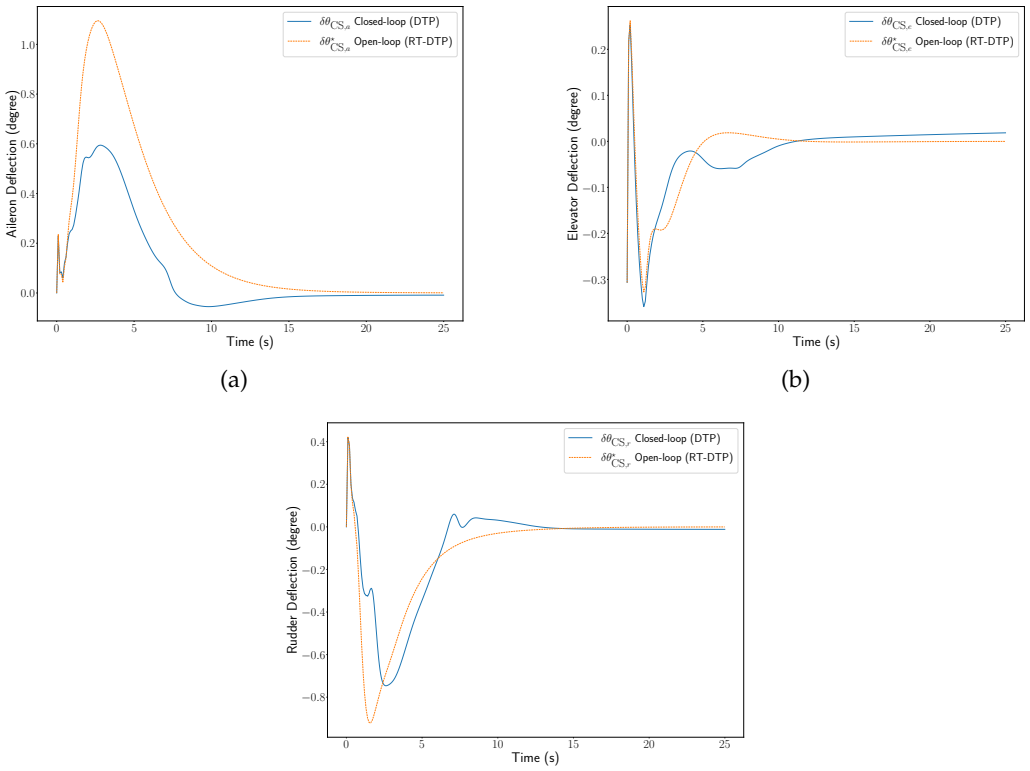


Figure 9: Optimal open-loop control input corrections computed by the RT-DTP (orange) and achieved closed-loop deviations as predicted by the nonlinear DTP for flight dynamics (blue): (a) aileron deflection; (b) elevator deflection; and (c) rudder deflection – corrections and deviations are measured with respect to the glideslope.

performed on fourteen Intel© Xeon© Gold 5118 processors [49]. Each of these has twelve physical cores, with two hyperthreads per core and therefore a total of 336 cores: 335 hyperthreads were allocated to the fluid subsystem and one to the structural subsystem. The MPC controller, which is managed by the analysis software for the structural subsystem, was executed on the same hyperthread. As expected, the overall two-level DT delivered a real-time performance. This implies that: the updates of the state estimator and all other control-relevant updates perform extremely fast, which is not surprising considering that they incur mostly GAXPY (General matrix A multiplied by a vector X plus a vector Y) operations; and the chosen OCP solver is also very fast.

Model	Average wall clock time (ms)	Number of calls
(4.1), (4.2) and (4.3)	6.7×10^{-2}	2,490
(4.4)	8.4	250
Total	8.467	2,940

Table 5: Average two-level DT (DTI + RT-DTP) wall clock timings, where the average is computed over the number of function calls: closed-loop and RT-DTP equipped with the finer time-integration (1st row); and OCP solver (2nd row).

6 Preliminary Flight Validation

A preliminary flight validation of the proposed two-level DT for AL was conducted at Stanford University in the simpler context of steady-level flight (i.e $\gamma_0 = 0°$). For this purpose, the thrust and rigid body rotation rates were used as control parameters – instead of the thrust and control surface deflection rates – and an autopilot with a low-level PID controller was used to convert the rotation rate commands into control surface deflections [50]. The obtained preliminary results established the feasibility of the proposed two-level DT concept. In [50], Figures 5.14–5.17 report the time-histories of the velocity, position, Euler angle (instead of rotation vector components), thrust, and rotation rate (in terms of the roll, pitch, and yaw rates) errors of the hobby UAV. They show similar results to those discussed in Section 5, namely: steady-state offsets are noticeable in the position and rotation of the UAV, but much smaller in the rates. Again, this is unsurprising given the presence of various disturbances. Chief among these is the the presence of a wind, which, as previously mentioned, is not accounted for in the computational model guiding the MPC controller. This can be seen in the thrust, which is reported in Figure 5.17 of [50] to be significantly higher than its counterpart determined during the computation of trim. This issue can be mitigated by adding an integrator to the control scheme, as mentioned above. Alternatively, it can be addressed by adopting a database $\mathcal{DB}$ of pre-computed RT-DTPs associated with a parameter space $\mathcal{D}$ representing flight conditions (for example, see [51–53]), where each RT-DTP is constructed at a parameter point adaptively sampled in $\mathcal{D}$ by a greedy procedure (design of experiments). Then, at any queried but unsampled parameter point in $\mathcal{D}$, a linear RT-DTP is constructed in real-time by interpolation on matrix manifolds [52] of the content of $\mathcal{DB}$ and used to inform the MPC controller.

7 Summary and Conclusions

A two-level digital twin (DT) in which a digital twin instance (DTI) for model predictive control (MPC) of autonomous landing (AL) is wrapped around an innovative, real-time digital twin prototype (RT-DTP) for fluid-structure interaction (FSI) and flight dynamics (see Figure 1) is proposed in this paper. Its design features three innovations: the formulation of the governing coupled FSI equations in an arbitrary Lagrangian-Eulerian (ALE) setting and their linearization about a pre-designed glideslope trajectory, rather than a mere equilibrium point; the construction of a linearized, computational fluid dynamics (CFD)-based, projection-based reduced-order model (PROM) and its coupling with a linearized six-degrees-of-freedom (6dof) representation of the rigid dynamics of an aircraft; and the training of the resulting FSI PROM for any deflection of any control surface. The main purpose of the RT-DTP is to predict in real-time, during flight, the state of an aircraft and the aerodynamic forces and moments acting on it. It is an alternative to static lookup tables or regression-based surrogate models based on steady-state wind tunnel data that allows the DTI for MPC to be informed by a truly dynamic FSI / flight model, rather than a less accurate set of steady-state aerodynamic force and moment data points (and interpolation/extrapolation between/away from these points). Both components of the proposed DT for AL are data-driven: the DTI for MPC is driven by sensor data about the rigid body state and control surface deflections of the aircraft it pertains to; and being essentially a PROM, the RT-DTP is built from the knowledge of solution snapshots collected in a parameter space of interest then compressed. The proposed two-level DT for AL has been amply demonstrated and verified by numerical simulation. Even though its RT-DTP component is based on a three-dimensional Reynolds-averaged Navier-Stokes (RANS) computational model, it delivers real-time performance. The potential for flight control of the overall two-level DT concept has also been demonstrated in autonomous mode during a preliminary flight test.

Data Accessibility. The data required for the simulation of the controller is located at [54] (the temporary link for use in the submission/review process is https://datadryad.org/stash/share/XlCmljroFllTBxiu73Wo2xxa-awkt0eoO3t4ByqYVIA). The submission also includes the scripts used for: stabilizing the PROM; interfacing with the code for finding the terminal set (see below); and generating

as well as utilizing the controller. Other needed utility scripts may be found at [55]. The flow solver and structural analyzer used throughout this work are AERO-F [56] and AERO-S [57], respectively. Associated pre- and post-processing auxiliary codes are SOWER [58], CD2TET [59], and MATCHER [60]. Metis [61], version 5.1.0 (modified to accept the mesh format expected by SOWER), was used to decompose the CFD mesh into subdomains; the mesh partitioning file is included in the Dryad dataset. Other used software packages are SciPy [42], NumPy [62], Matplotlib [63], Matlab [38], YALMIP [44], CPLEX [45], MPT3 [46], and a fork of qpOASES [47] that can be found at [48]. The code for finding the terminal set can be found at [64] and that for interfacing with qpOASES can be found at [65].

Authors' Contributions. Andrew McClellan has contributed to the design of the proposed two-level digital twin and performed all numerical experiments, with significant assistance from Prof. Charbel Farhat. Joseph Lorenzetti has provided the code for building the controller and interfacing it with the OCP solver, and has contributed to the flight test, with significant advice and assistance from Prof. Marco Pavone. All authors have contributed to producing the manuscript.

Competing Interests. The authors declare that they have no competing interests.

Funding. The authors acknowledge the support by the Office of Naval Research under Grant N00014-17-1-2749, Andrew McClellan acknowledges support by the Stanford Graduate Fellowship (SGF), and Joseph Lorenzetti acknowledges support by the U.S. Department of Defense (DoD) through the National Defense Science and Engineering Fellowship (NDSEG).

Acknowledgements. The authors acknowledge Dr. Charbel Bou-Mosleh who has generated the body-fitted CFD mesh used in this study and Dr. Philip Avery who has assisted in the appropriate and efficient usage of AERO-F and AERO-S.

Disclaimer. This document does not necessarily reflect the position of the Office of Naval Research, the U.S. Department of Defense, or Stanford University, and therefore no official endorsement should be inferred.

References

1. Stevens BL, Lewis FL, Johnson EN. 2015 *Aircraft control and simulation: dynamics, controls design, and autonomous systems*.
   John Wiley & Sons.
2. Lock JHB.
   Automatic control of aircraft and missiles.
3. Gangsaas D, Bruce K, Blight J, Ly UL. 1986 Application of modem synthesis to aircraft control: Three case studies.
   IEEE Transactions on Automatic Control **31**, 995–1014.
4. Nichols RA, Reichert RT, Rugh WJ. 1993 Gain scheduling for h-infinity controllers: A flight control example.
   IEEE Transactions on Control systems technology **1**, 69–79.
5. Niewoehner R, Kaminer II. 1996 Design of an autoland controller for an f-14 aircraft using h-infinity synthesis.
   Journal of Guidance, Control, and Dynamics **19**, 656–663.
6. Enns D, Bugajski D, Hendrick R, Stein G. 1994 Dynamic inversion: an evolving methodology for flight control design.
   International Journal of control **59**, 71–91.
7. Åström KJ, Wittenmark B. 2013 *Adaptive control*.
   Courier Corporation.
8. Borrelli F, Falcone P, Keviczky T, Asgari J, Hrovat D. 2005 Mpc-based approach to active steering for autonomous vehicle systems.
   International journal of vehicle autonomous systems **3**, 265–291.
9. Soize C, Farhat C. 2017 A nonparametric probabilistic approach for quantifying uncertainties in low-dimensional and high-dimensional nonlinear models.
   International Journal for Numerical Methods in Engineering **109**, 837–888.
10. Farhat C, Bos A, Avery P, Soize C. 2018 Modeling and quantification of model-form uncertainties in eigenvalue computations using a stochastic reduced model.
    AIAA Journal **56**, 1198–1210.

11. https://en.wikipedia.org/wiki/Digital_twin.
12. Todd F. 2019.
    Digital twin examples: Simulating formula 1, singapore and wind farms to improve results.
    https://www.ns-businesshub.com/technology/digital-twin-examples-formula1-singapore.
13. Lieu T, Farhat C, Lesoinne M. 2006 Reduced-order fluid/structure modeling of a complete aircraft configuration.
    Computer methods in applied mechanics and engineering **195**, 5730–5742.
14. Farhat C, Lesoinne M, Maman N. 1995 Mixed explicit/implicit time integration of coupled aeroelastic problems: Three-field formulation, geometric conservation and distributed solution.
    International Journal for Numerical Methods in Fluids **21**, 807–835.
15. Farhat C, Geuzaine P, Brown G. 2003 Application of a three-field nonlinear fluid–structure formulation to the prediction of the aeroelastic parameters of an f-16 fighter.
    Computers & Fluids **32**, 3–29.
16. Farhat C, Geuzaine P, Grandmont C. 2001 The discrete geometric conservation law and the nonlinear stability of ale schemes for the solution of flow problems on moving grids.
    Journal of Computational Physics **174**, 669–694.
17. Geuzaine P, Grandmont C, Farhat C. 2003 Design and analysis of ale schemes with provable second-order time-accuracy for inviscid and viscous flow simulations.
    Journal of Computational Physics **191**, 206–227.
18. Lesoinne M, Sarkis M, Hetmaniuk U, Farhat C. 2001 A linearized method for the frequency analysis of three-dimensional fluid/structure interaction problems in all flow regimes.
    Computer Methods in Applied Mechanics and Engineering **190**, 3121–3146.
19. Farhat C, Degand C, Koobus B, Lesoinne M. 1998 Torsional springs for two-dimensional dynamic unstructured fluid meshes.
    Computer methods in applied mechanics and engineering **163**, 231–245.
20. Degand C, Farhat C. 2002 A three-dimensional torsional spring analogy method for unstructured dynamic meshes.
    Computers & structures **80**, 305–316.
21. Farhat C, Avery P, Chapman T, Cortial J. 2014 Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency.
    International Journal for Numerical Methods in Engineering **98**, 625–662.
22. Mäkinen J. 2004 *A Formulation for Flexible Multibody Mechanics*, volume Research Report 2004:3.
    Tampere University of Technology.
23. Chung J, Hulbert G. 1993 A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized-α method.
    Journal of Applied Mechanics **60**, 371–375.
24. Spalart P, Allmaras S. 1992 A one-equation turbulence model for aerodynamic flows.
    In *Proceedings of the 30th AIAA Aerospace Sciences Meeting and Exhibit*. Reno, NV.
25. Farhat C, Rallu A, Wang K, Belytschko T. 2010 Robust and provably second-order explicit–explicit and implicit–explicit staggered time-integrators for highly non-linear compressible fluid–structure interaction problems.
    International Journal for Numerical Methods in Engineering **84**, 73–107.
26. National Oceanic and Atmospheric Administration, National Aeronautics and Space Administration, United States Air Force. 1976 (accessed 26 October 2018) U.S. Standard Atmosphere, 1976.
    NASA Technical Reports Server **NASA-TM-X-74335**.
    https://ntrs.nasa.gov/search.jsp?R=19770009539
27. McClellan AR. 2021 *Projection-based Model Order Reduction for Model Predictive Control of a Descending Aircraft*.
    Stanford University.
    (In progress)
28. McCormick BW. 1979 *Aerodynamics, Aeronautics, and Flight Mechanics*.
    New York: John Wiley & Sons, 1st edition.
29. Benner P, Gugercin S, Willcox K. 2015 A survey of projection-based model reduction methods for parametric dynamical systems.
    SIAM review **57**, 483–531.

30. Sirovich L. 1987 Turbulence and the dynamics of coherent structures. i. coherent structures. *Quarterly of applied mathematics* **45**, 561–571.
31. Amsallem D, Farhat C. 2012 Stabilization of projection-based reduced-order models. *International Journal for Numerical Methods in Engineering* **91**, 358–377.
32. Lorenzetti J, McClellan A, Farhat C, Pavone M. 2020 Linear reduced order model predictive control. *arXiv preprint arXiv:2012.03384* .
33. Franklin GF, Powel JD, Emami-Naeini A. 2015 *Feedback Control of Dynamic Systems*. Upper Saddle River, New Jersey: Pearson, 7th edition.
34. Franklin GF, Powell JD, Workman ML, al et. 1998 *Digital Control of Dynamic Systems*. Half Moon Bay, California: Ellis-Kagle Press, 3rd edition.
35. Reichardt H. 1951 Vollständige darstellung der turbulenten geschwindigkeitsverteilung in glatten leitungen. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik* **31**, 208–219.
36. Iserles A. 2008 *A First Course in the Numerical Analysis of Differential Equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2 edition.
37. Lesoinne M, Farhat C. 1998 Higher-order subiteration-free staggered algorithm for nonlinear transient aeroelastic problems. *AIAA journal* **36**, 1754–1757.
38. MATLAB. 2020 *version 9.8.0 (2020a)*. Natick, Massachusetts: The MathWorks Inc.
39. Grant M, Boyd S. 2014. CVX: Matlab software for disciplined convex programming, version 2.2. http://cvxr.com/cvx.
40. Grant M, Boyd S. 2008 Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control* (ed. V Blondel, S Boyd, H Kimura), Lecture Notes in Control and Information Sciences, pp. 95–110. Springer-Verlag Limited. http://stanford.edu/~boyd/graph_dcp.html
41. Abramowitz M, Stegun I. 1972 *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing*. New York: Dover.
42. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat I, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, SciPy 10 Contributors. 2020 SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* **17**, 261–272.
43. Kolmanovsky I, Gilbert EG. 1998 Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical problems in engineering* **4**, 317–367.
44. Löfberg J. 2004 YALMIP : A toolbox for modeling and optimization in MATLAB. In *In Proceedings of the CACSD Conference*. Taipei, Taiwan.
45. IBM. 2020 *IBM ILOG CPLEX Optimization Studio Version 12.10 User's Manual*. IBM.
46. Herceg M, Kvasnica M, Jones C, Morari M. 2013 Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pp. 502–510. Zürich, Switzerland. http://control.ee.ethz.ch/~mpt
47. Ferreau H, Kirches C, Potschka A, Bock H, Diehl M. 2014 qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation* **6**, 327–363.
48. Lorenzetti J. 2021. qpOASES. GitHub. GitHub repository. https://github.com/jlorenze/qpOASES. This repository was forked from https://github.com/coin-or/qpOASES; the asl-v3.2.0 branch was used.
49. Intel Corporation.

Intel© Xeon© Gold 5118 Processor Specifications Sheet.
https://ark.intel.com/content/www/us/en/ark/products/120473/intel-xeon-gold-5118-processor-16-5m-cache-2-30-ghz.html.
Accessed October 2021

50. Lorenzetti J. 2021 *Reduced Order Model Predictive Control of High-Dimensional Systems*. Stanford University.
51. Amsallem D, Cortial J, Carlberg K, Farhat C. 2009 A method for interpolating on manifolds structural dynamics reduced-order models.
International journal for numerical methods in engineering **80**, 1241–1258.
52. Amsallem D, Farhat C. 2011 An online method for interpolating linear parametric reduced-order models.
SIAM Journal on Scientific Computing **33**, 2169–2198.
53. Choi Y, Boncoraglio G, Anderson S, Amsallem D, Farhat C. 2020 Gradient-based constrained optimization using a database of linear reduced-order models.
Journal of Computational Physics **423**, 109787.
54. McClellan A, Lorenzetti J, Pavone M, Farhat C. 2021.
Data from: A physics-based digital twin for model predictive control of autonomous unmanned aerial vehicle landing.
Dryad. Dataset. https://doi.org/10.5061/dryad.34tmpg4mh.
55. McClellan A. 2021.
pyaeroutils.
GitHub. GitHub repository. https://github.com/amcclell/pyaeroutils.
56. Farhat Research Group.
AERO-F.
Bitbucket. Bitbucket repository. https://bitbucket.org/frg/aero-f.
57. Farhat Research Group.
AERO-S.
Bitbucket. Bitbucket repository. https://bitbucket.org/frg/aero-s.
58. Farhat Research Group.
SOWER.
Bitbucket. Bitbucket repository. https://bitbucket.org/frg/sower.
59. Farhat Research Group.
CD2TET.
Bitbucket. Bitbucket repository. https://bitbucket.org/frg/cd2tet.
60. Farhat Research Group.
MATCHER.
Bitbucket. Bitbucket repository. https://bitbucket.org/frg/matcher.
61. Karypis G, Kumar V. 1999 A fast and highly quality multilevel scheme for partitioning irregular graphs.
SIAM Journal on Scientific Computing **20**, 359—392.
62. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, Fernández del Río J, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE. 2020 Array programming with NumPy.
Nature **585**, 357–362.
63. Hunter JD. 2007 Matplotlib: A 2d graphics environment.
Computing in Science & Engineering **9**, 90–95.
64. Lorenzetti J. 2021.
rompc.
GitHub. GitHub repository. https://github.com/StanfordASL/rompc.
65. Lorenzetti J. 2021.
asl_fixedwing.
GitHub. GitHub repository. https://github.com/StanfordASL/asl_fixedwing.