# Predictive Cost Adaptive Control of 3DOF CFD-Simulated Flight

Dennis Serbin*, Krzysztof J. Fidkowski†, and Dennis S. Bernstein‡
*University of Michigan, Ann Arbor, MI, 48109*

**We apply predictive cost adaptive control (PCAC) to CFD-simulated flight of an airfoil. PCAC is a discrete-time, indirect adaptive control technique based on model predictive control. Unlike model-based control, PCAC uses no prior model, and, unlike deep and reinforcement learning, PCAC uses no preflight data or offline learning. For rapid online multiple-input, multiple-output (MIMO) system identification, PCAC uses recursive least squares (RLS) with variable-rate forgetting. To determine control inputs, PCAC uses quadratic programming for receding horizon optimization, which allows PCAC to enforce physically meaningful magnitude and rate-saturation constraints. PCAC is an output-feedback control technique that can be used with a reference model, and unknown disturbances may be either matched or unmatched. The present paper applies PCAC to a high-order finite-element simulation of an airfoil in free flight, where the rigid-body motion of the airfoil is subjected to aerodynamic forces and moments computed from the unsteady computational fluid dynamics (CFD) simulation. To follow a commanded altitude, PCAC requests the flap deflection, which modulates the force and moment on the airfoil. The effect of the flap deflection on the pitch and plunge motion of the airfoil is modeled using an arbitrary Lagrangian-Eulerian formulation. Results demonstrate the ability of PCAC to follow a commanded altitude and horizontal position of the airfoil using the flap actuator for flight with one, two, and three degrees of freedom. With online system identification, PCAC improves its ability to follow position commands as additional commands are given. The results demonstrate the applicability of PCAC to function as a real-time adaptive autopilot, without prior knowledge of the airfoil aerodynamics and flight dynamics.**

## I. Introduction

Advances in computational power have made computational fluid dynamics (CFD) simulations increasingly more accurate and efficient. It is now possible to design, analyze, and simulate aircraft using numerical techniques that greatly reduce the need for wind tunnel experiments. High-order, mesh-adaptive CFD [1, 2] is particularly efficient in yielding high-accuracy results for practical aerodynamic simulations.

Combined with wind-tunnel data collection, CFD can enhance the ability to develop stability augmentation systems (SAS) and autopilots. These design processes rely on aerodynamic databases at various flight conditions and control-surface deflections. CFD can be used to refine these databases to build lookup tables. These tables assume that the forces and moments satisfy superposition, and thus, for example, angle of attack and sideslip angle perturbations can be considered separately. The models based on the resulting lookup tables are thus approximate. Standard control methods, such as gain scheduling and loop shaping, rely on these models, and the development of aircraft control systems is a tedious process that relies on experience and judgment.

In this work, we present an alternative control strategy, called predictive cost adaptive control (PCAC), which does not require the construction of an aerodynamic database and obviates the need for standard control-design techniques, such as loop shaping, dynamic inversion, and gain scheduling. Instead, PCAC identifies a model of the system online during simulated flight of the vehicle. The simulation consists of an unsteady CFD analysis with control surfaces and sensors, coupled with rigid-body dynamics of the vehicle. Following a short learning phase, PCAC responds rapidly and effectively to user-specified commands. Although outside the scope of this paper, PCAC can be implemented onboard a vehicle for real-time adaptive flight control.

The present work is motivated by the fact that simulation studies of this type can be used to quickly evaluate the flight performance of a proposed or modified vehicle design. Since the controller learns directly from the CFD simulation,

*Undergraduate Research Assistant, Department of Aerospace Engineering, dserbin@umich.edu.
†Professor, Department of Aerospace Engineering, kfid@umich.edu.
‡Professor, Department of Aerospace Engineering, dsbaero@umich.edu.

the effort needed to develop an appropriate SAS and autopilot is eliminated, thereby allowing the vehicle designer to quickly evaluate the performance of the vehicle. In addition, for a vehicle that is designed and built, these studies can provide confidence that PCAC can be applied directly to the physical vehicle, with potential savings in time and effort, the ability to facilitate vehicle customization, and reliability in the event of damage and failure.

PCAC is a discrete-time, indirect adaptive control technique based on model predictive control (MPC) [3]. Unlike model-based control, PCAC uses no prior model, and, unlike deep and reinforcement learning, PCAC uses no control-oriented model and no preflight data for offline learning. For rapid online multiple-input, multiple-output (MIMO) system identification, PCAC uses recursive least squares (RLS) with variable-rate forgetting [4–8]. To specify control inputs, PCAC uses quadratic programming for receding horizon optimization, which enforces magnitude and rate-saturation constraints. PCAC is an output-feedback control technique that can be used with a reference model, and unknown disturbances may be either matched or unmatched. During operation, PCAC uses sampled sensor data (e.g., flow state and altitude) and requests system inputs (e.g., surface deflections and jet velocities). Other than specification of the model order, optimization horizon, and forgetting hyperparameters, PCAC operates under cold-start conditions. If a prior control-oriented model is available, PCAC can be implemented under warm-start conditions.

For the CFD simulation in the present paper, we use a high-order discontinuous-finite element method, discontinuous Galerkin (DG) [9, 10], for its accuracy and efficient use of degrees of freedom. The high approximation order $p$ reduces errors associated with approximating the state, and, for smooth states and high accuracy requirements, high $p$ is generally more efficient than mesh refinement. Prospects of output-error estimation and mesh adaptation also motivate the choice of DG, although the calculations in the present paper do not take advantage of these capabilities. The effect of the control surfaces and the vehicle aerodynamics rely on an arbitrary Lagrangian-Eulerian formulation [11, 12], which has been extensively developed for high-order methods [13, 14].

In the present paper, we use PCAC to control the altitude of an airfoil in free-flight. Section II presents the CFD model and DG discretization, Section III presents the PCAC controller, and Section IV presents the results of free-flight airfoil tests.

## II. CFD Model and Discretization

### A. Governing Equations

We assume that the fluid is governed by the compressible Reynolds-Averaged Navier-Stokes (RANS) equations with the Spalart-Allmaras (SA) closure [15]. In conservative form, this system of partial differential equations is given by

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{F}}(\mathbf{u}, \nabla \mathbf{u}) + \mathbf{S}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0}, \tag{1}$$

where $\mathbf{u} \in \mathbb{R}^s$ is the $s$-component state vector containing the conserved quantities, $\vec{\mathbf{F}} \in \mathbb{R}^{\dim \times s}$ is the flux vector, dim is the spatial dimension, and $\mathbf{S}$ is the source term arising from the turbulence model. A detailed exposition of the equations and closure relations of this model is given in [16, 17].

### B. Finite-Element Discretization

We use a discontinuous Galerkin (DG) finite-element spatial discretization [10], with the Roe [18] convective flux and the second form of Bassi and Rebay (BR2) [19] for the viscous treatment. The state is approximated on an unstructured mesh of non-overlapping elements using polynomials of order $p$. The semi-discretized form of the equations is

$$\mathbf{M}\frac{d\mathbf{U}}{dt} + \mathbf{R}(\mathbf{U}) = \mathbf{0}, \tag{2}$$

where $\mathbf{U} \in \mathbb{R}^N$ is the discrete state vector, $N$ is the total number of unknowns including the state rank, $\mathbf{R}(\cdot) \in \mathbb{R}^{N \times N}$ is the nonlinear spatial residual, and $\mathbf{M} \in \mathbb{R}^{N \times N}$ is the block-element sparse mass matrix. For steady simulations, the time-derivative term drops out, although pseudo-time continuation remains in the solver to drive the steady residual to zero [20]. The solver consists of a Newton-Raphson method with the generalized minimum residual (GMRES) [21] linear solver, preconditioned by an element-line Jacobi smoother with a coarse-level ($p = 1$) correction [22, 23]. For unsteady simulations, we use a third-order modified extended backward difference formula [24] applied to the semi-discrete form.

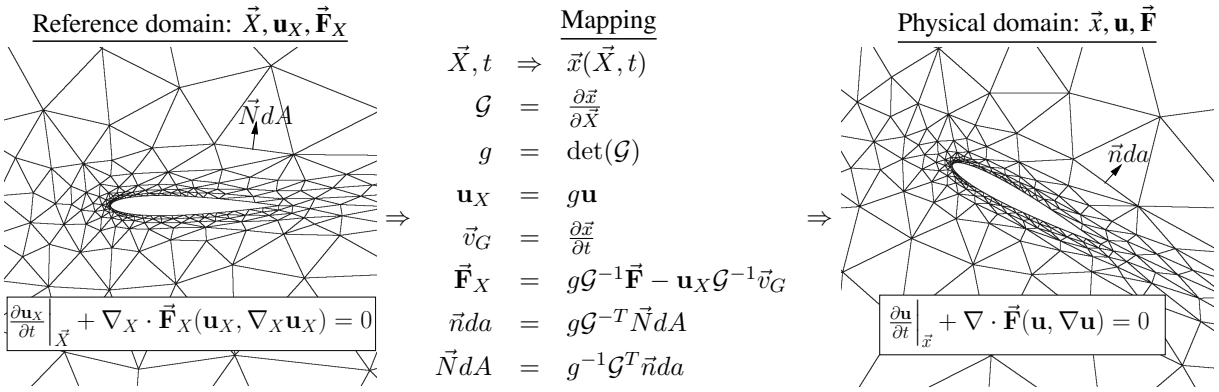



**Fig. 1 Arbitrary Lagrangian-Eulerian formulation (ALE). ALE uses a map between the deforming physical domain and a static reference domain, while solving the transformed equations on the reference domain.**

### C. Arbitrary Lagrangian-Eulerian Formulation

In an arbitrary Lagrangian-Eulerian (ALE) method, the mesh can move at a velocity different from that of the flow, which is useful when objects move or deform. ALE uses a map between the deforming physical domain and a static reference domain, while solving the transformed equations on the reference domain [11, 12, 25]. This transformation is illustrated in Figure 1.

The subscript $()_X$ denotes reference-space quantities, and $\vec{x}/\vec{X}$ are the physical/reference coordinates. The mapping $\vec{x}(\vec{X}, t)$ is analytical, obtained by blending rigid-body motion in the vicinity of the moving object to zero far away from the object [11], as illustrated in Figure 2. The rigid-body motion consists of pitch and plunge, as determined by the dynamics of the airfoil. The blending is a cubic function of the radius with continuous value and first derivative at the edges of the blending region. The resultant mapping Jacobian determinant $g$ may not be polynomial in $\vec{X}$, so that a constant physical state may not be representable with polynomial trial functions in reference space. This leads to slight conservation errors, which can be mitigated with a geometric conservation law [11]. However, as these errors decrease with higher-order approximation and adaptation, in this work we forgo a GCL.

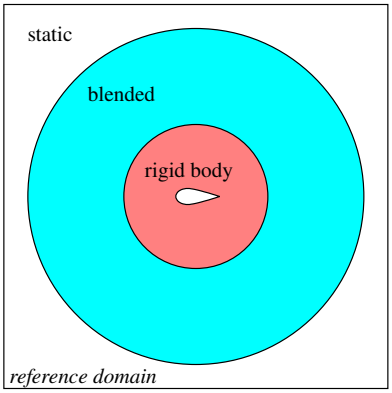



**Fig. 2 Rigid-body motion blending.**

## III. Predictive Cost-Adaptive Control

### A. Overview

As shown in Figure 3, predictive cost adaptive control (PCAC) combines online identification with output-feedback model predictive control (MPC) based on quadratic programming. PCAC uses no prior modeling information aside from a model order for system identification, nor does it use probing signals [3].

### B. Online Identification

The online identification takes the form of the MIMO input-output model

$$\hat{\mathbf{y}}_k = -\sum_{i=1}^{\hat{n}} \hat{\mathbf{F}}_i \mathbf{y}_{k-i} + \sum_{i=1}^{\hat{n}} \hat{\mathbf{G}}_i \mathbf{u}_{k-i}, \tag{3}$$

where $k \geq 0$ is the step, $\hat{n} \geq 1$ is the identification data window, $\hat{\mathbf{F}}_i \in \mathbb{R}^{p \times p}$ and $\hat{\mathbf{G}}_i \in \mathbb{R}^{p \times m}$ are the estimated model coefficients, and $\mathbf{u}_k \in \mathbb{R}^m$, $\mathbf{y}_k \in \mathbb{R}^p$, and $\hat{\mathbf{y}}_k \in \mathbb{R}^p$ are the inputs, outputs, and predicted outputs, respectively. We

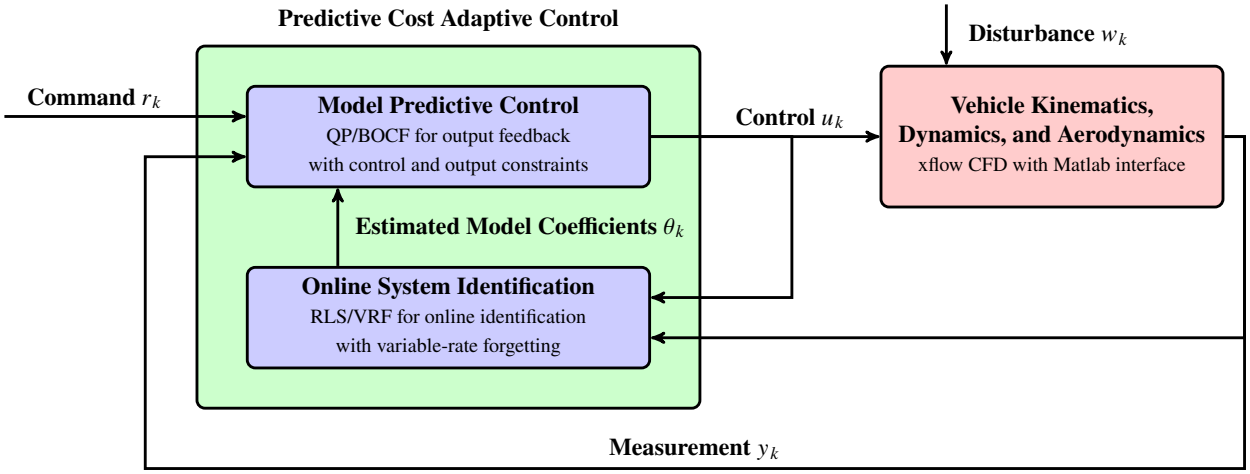



**Fig. 3 PCAC block diagram. The online closed-loop system identification is based on recursive least squares (RLS) with variable-rate forgetting (VRF). The model predictive control (MPC) algorithm, which is based on quadratic programming (QP), uses the estimated model coefficients $\theta_k$ to form the block-observable canonical form (BOCF) state-space model, which is used by QP to determine the control input $\mathbf{u}_k$.**

estimate $\hat{\mathbf{F}}_k$ and $\hat{\mathbf{G}}_k$ online using recursive least squares with variable-rate forgetting (RLS/VRF), by minimizing the cost function [26]

$$J_k(\hat{\boldsymbol{\theta}}) = \sum_{i=0}^{k} \frac{\rho_i}{\rho_k} \mathbf{z}_i^{\mathrm{T}}(\hat{\boldsymbol{\theta}}) \mathbf{z}_i(\hat{\boldsymbol{\theta}}) + \frac{1}{\rho_k} (\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0)^{\mathrm{T}} \mathbf{P}_0^{-1} (\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0), \tag{4}$$

where $\rho_k \equiv \prod_{j=0}^{k} \lambda_j^{-1} \in \mathbb{R}$, $\lambda_k \in (0, 1]$ is the forgetting factor, $\mathbf{P}_0 \in \mathbb{R}^{[\hat{n}p(m+p)] \times [\hat{n}p(m+p)]}$ is positive definite, $\boldsymbol{\theta}_0 \in \mathbb{R}^{[\hat{n}p(m+p)]}$ is the initial estimate of the coefficient vector, and the performance variable $\mathbf{z}_i(\hat{\boldsymbol{\theta}}) \in \mathbb{R}^p$ is defined as

$$\mathbf{z}_k(\hat{\theta}) = \mathbf{y}_k - \boldsymbol{\phi}_k \hat{\boldsymbol{\theta}}. \tag{5}$$

The vector $\hat{\boldsymbol{\theta}} \in \mathbb{R}^{[\hat{n}p(m+p)]}$ of coefficients to be estimated is given by

$$\hat{\boldsymbol{\theta}} \equiv \mathrm{vec} \begin{bmatrix} \hat{\boldsymbol{F}}_1 & \cdots & \hat{\boldsymbol{F}}_{\hat{n}} & \hat{\boldsymbol{G}}_1 & \cdots & \hat{\boldsymbol{G}}_{\hat{n}} \end{bmatrix} = \mathrm{vec} \begin{bmatrix} \hat{\boldsymbol{\theta}}_{\hat{F}} & \hat{\boldsymbol{\theta}}_{\hat{G}} \end{bmatrix},$$

where $\hat{\boldsymbol{\theta}}_{\hat{F}}$ and $\hat{\boldsymbol{\theta}}_{\hat{G}}$ are the estimates of the numerator and denominator coefficients, defined by

$$\hat{\boldsymbol{\theta}}_{\hat{F}} \equiv \mathrm{vec} \begin{bmatrix} \hat{\boldsymbol{F}}_1 & \cdots & \hat{\boldsymbol{F}}_{\hat{n}} \end{bmatrix}, \tag{6}$$

$$\hat{\boldsymbol{\theta}}_{\hat{G}} \equiv \mathrm{vec} \begin{bmatrix} \hat{\boldsymbol{G}}_1 & \cdots & \hat{\boldsymbol{G}}_{\hat{n}} \end{bmatrix}. \tag{7}$$

With the regressor matrix $\boldsymbol{\phi}_k \in \mathbb{R}^{p \times [\hat{n}p(m+p)]}$ defined by

$$\boldsymbol{\phi}_k \equiv \begin{bmatrix} -\boldsymbol{y}_{k-1}^{\mathrm{T}} & \cdots & -\boldsymbol{y}_{k-\hat{n}}^{\mathrm{T}} & \boldsymbol{u}_{k-1}^{\mathrm{T}} & \cdots & \boldsymbol{u}_{k-\hat{n}}^{\mathrm{T}} \end{bmatrix} \otimes \boldsymbol{I}_p,$$

the global minimizer $\boldsymbol{\theta}_{k+1} \equiv \mathrm{argmin}_{\hat{\theta}} J_k(\hat{\boldsymbol{\theta}})$ of (4) is

$$\mathbf{L}_k = \lambda_k^{-1} \mathbf{P}_k, \tag{8}$$

$$\mathbf{P}_{k+1} = \mathbf{L}_k - \mathbf{L}_k \boldsymbol{\phi}_k^{\mathrm{T}} (\boldsymbol{I}_p + \boldsymbol{\phi}_k \mathbf{L}_k \boldsymbol{\phi}_k^{\mathrm{T}})^{-1} \boldsymbol{\phi}_k \mathbf{L}_k, \tag{9}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mathbf{P}_{k+1} \boldsymbol{\phi}_k^{\mathrm{T}} (\boldsymbol{y}_k - \boldsymbol{\phi}_k \boldsymbol{\theta}_k). \tag{10}$$

The variable-rate forgetting factor $\lambda_k$ used in the present paper is given by [27]

$$\lambda_k = \frac{1}{1 + \eta g(\mathbf{z}_{k-\tau_\mathrm{d}}, \ldots, \mathbf{z}_k)\mathbf{1}[g(\mathbf{z}_{k-\tau_\mathrm{d}}, \ldots, \mathbf{z}_k)]}, \tag{11}$$

where $\mathbf{1}\colon \mathbb{R} \to \{0, 1\}$ is the unit step function, and

$$g(\mathbf{z}_{k-\tau_\mathrm{d}}, \ldots, \mathbf{z}_k) \equiv \sqrt{\frac{\tau_\mathrm{n}}{\tau_\mathrm{d}} \frac{\left(\Sigma_{\tau_\mathrm{n}}(\mathbf{z}_{k-\tau_\mathrm{n}}, \ldots, \mathbf{z}_k)\Sigma_{\tau_\mathrm{d}}(\mathbf{z}_{k-\tau_\mathrm{d}}, \ldots, \mathbf{z}_k)^{-1}\right)}{c}} - \sqrt{f},$$

where $\eta > 0$ and $p \leq \tau_\mathrm{n} < \tau_\mathrm{d}$ represent numerator and denominator window lengths. $\Sigma_{\tau_\mathrm{n}}$ and $\Sigma_{\tau_\mathrm{d}}$ are the sample variances of the respective window lengths, and the threshold constant $f$ is described in [27, 28]. The constant $c$, based on the windows lengths is described in [27]. The estimated model coefficients $\hat{\boldsymbol{\theta}}$ can be written in the block observable canonical form with matrices $\hat{\mathbf{A}}_k$, $\hat{\mathbf{B}}_k$, and $\hat{\mathbf{C}}_k$ given by

$$\hat{\mathbf{A}}_k \triangleq \begin{bmatrix} -\hat{\mathbf{F}}_{1,k} & \mathbf{I}_p & \cdots & \cdots & 0_{p\times p} \\ \vdots & \mathbf{0}_{p\times p} & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{0}_{p\times p} \\ \vdots & \vdots & & \ddots & \mathbf{I}_p \\ -\hat{\mathbf{F}}_{\hat{n},k} & \mathbf{0}_{p\times p} & \cdots & \cdots & \mathbf{0}_{p\times p} \end{bmatrix}, \quad \hat{\mathbf{B}}_k \triangleq \begin{bmatrix} \hat{\mathbf{G}}_{1,k} \\ \hat{\mathbf{G}}_{2,k} \\ \vdots \\ \hat{\mathbf{G}}_{\hat{n},k} \end{bmatrix}, \qquad \hat{\mathbf{C}}_k \triangleq \begin{bmatrix} \mathbf{I}_p & \mathbf{0}_{p\times p} & \cdots & \mathbf{0}_{p\times p} \end{bmatrix}. \tag{12}$$

### C. Model Predictive Control

Model predictive control (MPC) uses a model of the system to optimize its performance over a future finite interval of time. The optimization yields a sequence of controls, the first of which is implemented, and the procedure is repeated at subsequent steps. By performing constrained optimization, MPC facilitates the enforcement of constraints on the state and control input. At step $k$, PCAC uses the identified model $\hat{\mathbf{A}}_k$, $\hat{\mathbf{B}}_k$, and $\hat{\mathbf{C}}_k$. As in [29], the receding-horizon optimization is performed using quadratic programming (QP), which is a convex optimization technique. This optimization determines the control input $\mathbf{u}_{k+1}$ at the next time step, while also attempting to satisfy constraints on the state and control input.

Let $\mathcal{R}_{k,\ell} \equiv \begin{bmatrix} \mathbf{r}_{k+1}^\mathrm{T} \cdots \mathbf{r}_{k+\ell}^\mathrm{T} \end{bmatrix}^\mathrm{T} \in \mathbb{R}^{\ell p_\mathrm{t}}$ be the vector of future commands over the horizon $\ell$, let $\mathbf{Y}_{1|k,l}$ be the corresponding $l$-step predicted output for a sequence of $l$ future controls, $\mathbf{U}_{1|k,l}$, and let $\mathbf{Y}_{\mathrm{t},1|k,\ell} \equiv \mathbf{C}_{\mathrm{t},\ell}\mathbf{Y}_{1|k,\ell}$ be the $\ell$-step predicted output, where $\mathbf{C}_{\mathrm{t},\ell} \equiv \mathbf{I}_\ell \otimes \mathbf{C}_\mathrm{t} \in \mathbb{R}^{\ell p_\mathrm{t} \times \ell p}$, $\otimes$ is the Kronecker product, and $\mathbf{C}_\mathrm{t}\mathbf{y}_{i|k}$ computes the commanded outputs from $\mathbf{y}_{i|k}$. Let $\mathbf{C}_\ell \equiv \mathbf{I}_\ell \otimes (\boldsymbol{C}\mathbf{C}_\mathrm{c}) \in \mathbb{R}^{\ell n_\mathrm{c} \times \ell p}$, where $\mathbf{C}_\mathrm{c}\mathbf{y}_{i|k}$ creates the constrained outputs from $\mathbf{y}_{i|k}$, let $\mathbf{D}_\ell \equiv \mathbf{1}_\ell \otimes \mathbf{D} \in \mathbb{R}^{\ell n_\mathrm{c}}$, and define the sequence of differences of control inputs as

$$\Delta\mathbf{U}_{1|k,\ell} \equiv \begin{bmatrix} (\mathbf{u}_{1|k} - \mathbf{u}_k)^\mathrm{T} & \cdots & (\mathbf{u}_{\ell|k} - \mathbf{u}_{\ell-1|k})^\mathrm{T} \end{bmatrix}^\mathrm{T} \in \mathbb{R}^{\ell m}. \tag{13}$$

The QP-based MPC optimization problem is then given by

$$\min_{\mathbf{U}_{1|k,\ell}} \left(\mathbf{Y}_{\mathrm{t},1|k,\ell} - \mathcal{R}_{k,\ell}\right)^\mathrm{T} \mathbf{Q} \left(\mathbf{Y}_{\mathrm{t},1|k,\ell} - \mathcal{R}_{k,\ell}\right) + \Delta\mathbf{U}_{1|k,\ell}^\mathrm{T}\mathbf{R}\Delta\mathbf{U}_{1|k,\ell}, \tag{14}$$

subject to

$$\boldsymbol{C}_\ell\mathbf{Y}_{1|k,\ell} + \boldsymbol{\mathcal{D}}_\ell \leq \mathbf{0}_{\ell n_\mathrm{c}}, \tag{15}$$

$$\mathbf{U}_\mathrm{min} \leq \mathbf{U}_{1|k,\ell} \leq \mathbf{U}_\mathrm{max}, \tag{16}$$

$$\Delta\mathbf{U}_\mathrm{min} \leq \Delta\mathbf{U}_{1|k,\ell} \leq \Delta\mathbf{U}_\mathrm{max}, \tag{17}$$

where $\mathbf{Q} \equiv \begin{bmatrix} \bar{\mathbf{Q}} & \mathbf{0}_{p_\mathrm{t}\times p_\mathrm{t}} \\ \mathbf{0}_{p_\mathrm{t}\times p_\mathrm{t}} & \bar{\mathbf{P}} \end{bmatrix} \in \mathbb{R}^{\ell p_\mathrm{t} \times \ell p_\mathrm{t}}$ is the output weighting, $\bar{\mathbf{Q}} \in \mathbb{R}^{(\ell-1)p_\mathrm{t} \times (\ell-1)p_\mathrm{t}}$ is the cost-to-go output weighting, $\bar{\mathbf{P}} \in \mathbb{R}^{p_\mathrm{t}\times p_\mathrm{t}}$ is the terminal output weighting, $\mathbf{R} \in \mathbb{R}^{\ell m \times \ell m}$ is the control-move-size weighting, $\mathbf{U}_\mathrm{min} \equiv \mathbf{1}_\ell \otimes \mathbf{u}_\mathrm{min} \in \mathbb{R}^{\ell m}$, $\mathbf{U}_\mathrm{max} \equiv \mathbf{1}_\ell \otimes \mathbf{u}_\mathrm{max} \in \mathbb{R}^{\ell m}$, $\Delta\mathbf{U}_\mathrm{min} \equiv \mathbf{1}_\ell \otimes \Delta\mathbf{u}_\mathrm{min} \in \mathbb{R}^{\ell m}$, and $\Delta\mathbf{U}_\mathrm{max} \equiv \mathbf{1}_\ell \otimes \Delta\mathbf{u}_\mathrm{max} \in \mathbb{R}^{\ell m}$.

### D. Code Coupling

We use PCAC to control the xflow CFD simulation. As such, the CFD and PCAC codes need to be coupled together in a synchronous manner. This coupling is complicated by the fact that xflow is written in C, and runs in parallel, whereas PCAC is written in Matlab, and runs serially with updates at fixed time steps. We use the Matlab engine API to link the codes, with xflow driving the simulation and calling PCAC at discrete time steps to obtain the requested actuator settings. Figure 4 illustrates this integration.

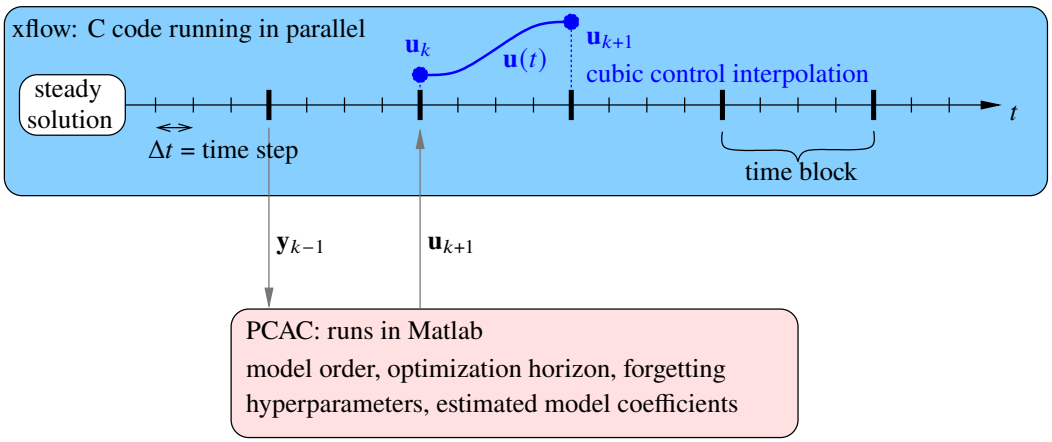



**Fig. 4 Integration of PCAC and xflow. Both codes run concurrently and communicate via the Matlab Engine API for C.**

The starting point in most simulations is a steady-state xflow solution with nominal actuation, e.g., zero flap deflection. Next, xflow begins time integration using time steps of size $\Delta t$. These time steps are grouped into blocks of $n_{\text{tb}}$ time steps each, where in this work we use $n_{\text{tb}} = 4$. After each time block, indexed by $k$, xflow determines simulated sensor values $\mathbf{y}_{k-1}$ and calls PCAC, which retains its own persistent internal state and returns the specified control inputs $\mathbf{u}_{k+1}$. The offset in the indices simulates the practical application of PCAC accounting for computational and actuation time. Between blocks $k-1$ and $k$, PCAC computes the required control input from knowledge of the sensor values at $k-1$, and at $k$ it returns the control inputs, which must then be realized in the system through smooth actuation between $k$ and $k+1$.

Whereas PCAC operates on sampled data in discrete time, the high-order xflow simulation, like a real-world system requires smooth variations of the parameters. For example, when the controlled parameter is a deformation of the geometry, through the ALE formulation, that deformation must be a continuous function of space and time. Hence, the discrete-time control outputs $\mathbf{u}_k$ must be smoothed in time in order for xflow to compute the required derivatives. As shown in Figure 4, this smoothing occurs over the subsequent time block after the controller call. A cubic interpolation in time maps the previous controller outputs, $\mathbf{u}_k$, to the requested values, $\mathbf{u}_k$. This interpolation is clamped to have zero slope at the ends, so that slope continuity is attained.

## IV. Numerical Examples

### A. Airfoil Case Setup

For the present results, we consider the flight of a NACA 0012 airfoil in subsonic flow. Figure 5 shows the setup for this scenario. Sensors measure the altitude $h$ and horizontal position $x$ of the center of gravity of the airfoil, and these values are inputs to the controller. The output of the controller is the flap deflection angle. The flap deflection is modeled as a continuous morphing of the trailing edge, rather than a hinged rotation of a flap. The morphing is performed by specifying a vertical plunge mesh deformation behind the airfoil and blending it to zero over a range that includes the extent of the flap. Figure 6 demonstrates this morphing.

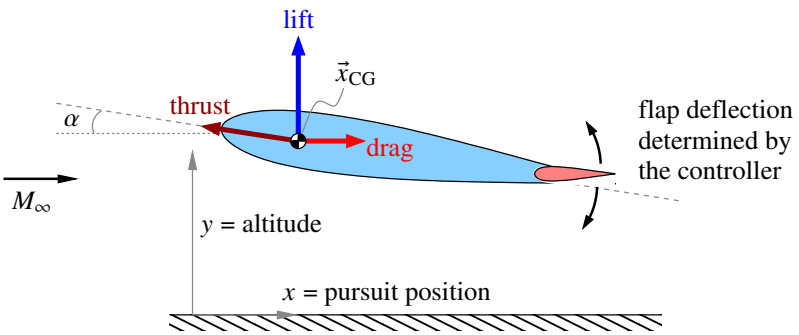
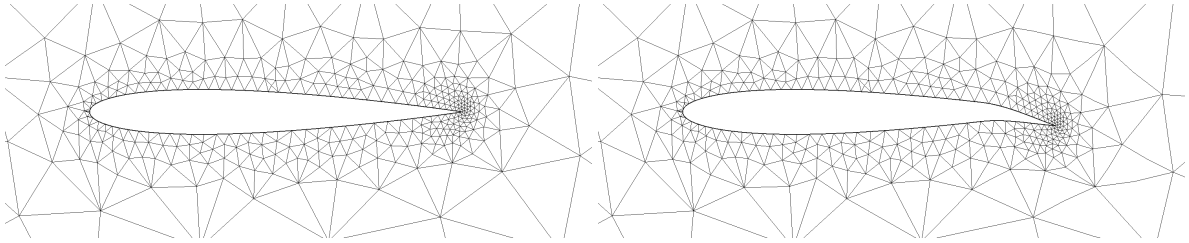



**Fig. 5 Airfoil scenario. The objective of the controller is to achieve and maintain a position setpoint by requesting the deflection of the trailing-edge flap and setting the thrust.**



(a) Undeformed (b) Flap down

**Fig. 6 Continuous flap deflection through a blended plunge mesh motion near the trailing edge of the airfoil. This deformation avoids ALE mapping derivative discontinuities at the hinge location.**

Aside from the flap deflection, the airfoil is assumed to be rigid, so that the equations governing its dynamics are

$$m\frac{d^2\vec{x}_{CG}}{dt^2} = \vec{F}, \tag{18}$$

$$I\frac{d^2\alpha}{dt^2} = M_{CG}, \tag{19}$$

where $\vec{x}_{CG}$ is the location of the center of gravity (CG), $\alpha$ is the pitch angle, $m$ is the mass (per unit depth) of the airfoil, $I$ is the moment of inertia about the out-of-plane axis through the CG, $\vec{F}$ is the sectional force on the airfoil, and $M_{CG}$ is the moment on the airfoil about the CG. The airfoil is made statically stable by placing the center of gravity sufficiently forward, at $0.15c$ from the leading edge, so that $dM_{CG}/d\alpha < 0$ for the flight conditions encountered.

### B. Control of 1DOF Airfoil Dynamics

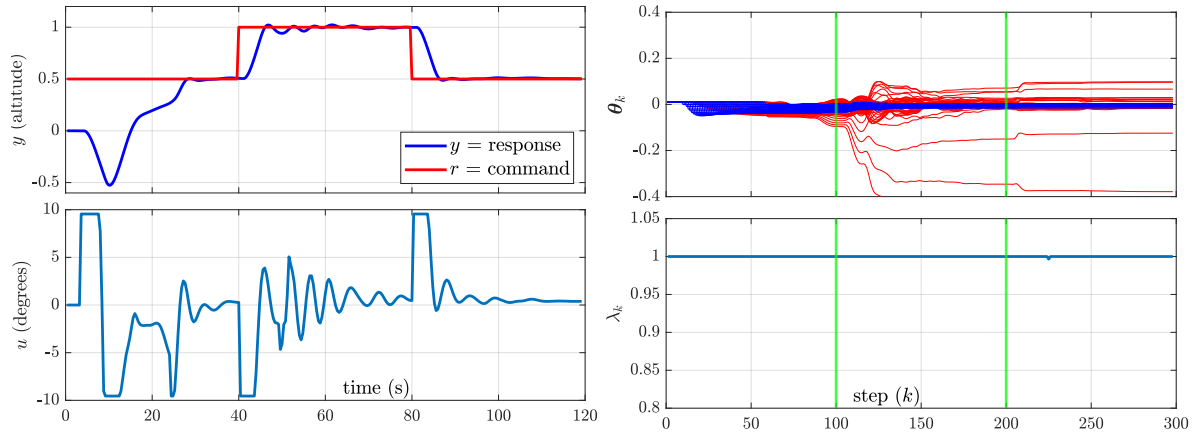
We first study 1DOF dynamics of the airfoil, where the airfoil can only move in the vertical direction as a result of the lift force. The aerodynamic drag and moment on the airfoil are ignored. As a result, the pitch angle remains zero, and only a blended vertical plunge motion is needed to model the motion of the airfoil.

The simulation is performed using a mesh that consists of 834 elements and $p = 2$ (quadratic) solution approximation. The freestream Mach number is set to $M_\infty = 0.2$, and the solved equations are Euler, so that viscous effects are ignored. All length scales are given in units of the airfoil chord, and all time scales are given in units of airfoil chord divided by free-stream speed; for convenience, we call these units seconds (s). The time step is set to 0.05 s.

We are interested in the ability of PCAC to attain and maintain a commanded altitude $y$ by requesting the flap angle $u$. Following a steady-state solution at $y = 0$, the first altitude command is $y = 0.5$. This command occurs at time $t = 0$ and lasts until $t = 40$. Next, the altitude command increases to $y = 1$ until $t = 80$, at which time it drops back to $y = 0.5$ for the remainder of the simulation.

Figure 7(a) shows the time history of the commanded and response altitudes, as well as the flap deflection angle. Without any knowledge of the system, PCAC first requests a flap up deflection (maximum constrained near 10 deg),

which sends the airfoil to negative $y$ values. Recognizing the impact of this control on the output, PCAC then requests a flap down, again at maximum deflection, followed by intermediate values of the flap before homing in on the correct dynamics and reaching the commanded altitude. After the next command to a higher altitude, PCAC immediately moves the flap down and quickly attains the altitude of $y = 1.0$, with some oscillation. On the last command, PCAC moves the flap up to decrease altitude and reaches the $y = 0.5$ altitude with significantly smaller oscillations.

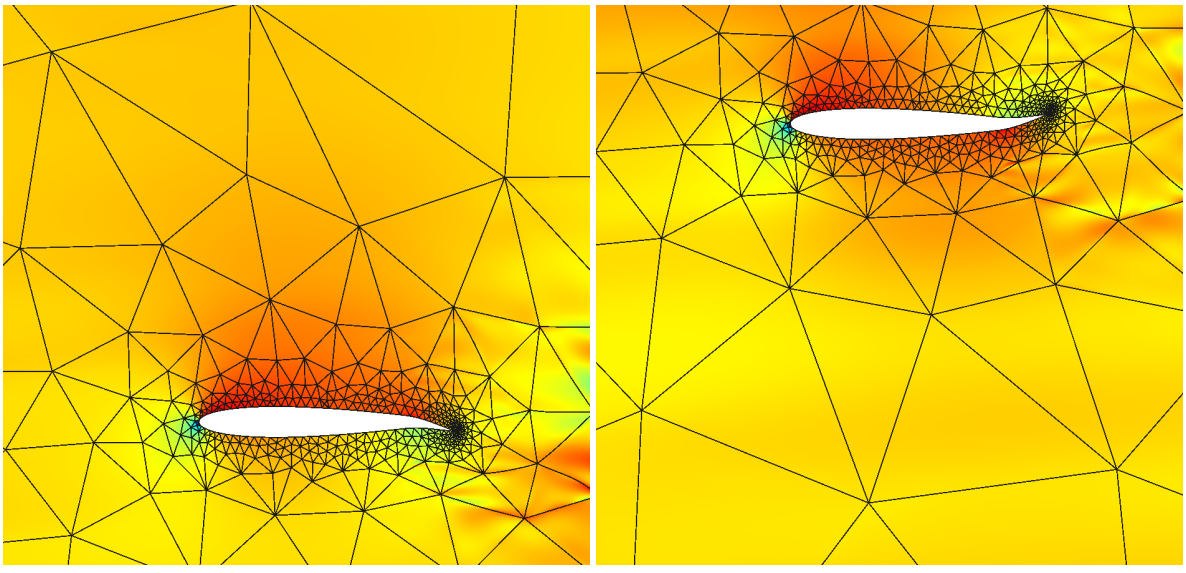



(a) Control input, command, and response

(b) Estimated model coefficients and the forgetting factor

**Fig. 7 PCAC response for the 1DOF scenario. The controller reaches the commanded setpoint more quickly after the first commanded setpoint due to the online closed-loop system identification.**

Throughout the entire simulation, PCAC performs online system identification, as is evident in Figure 7(b), which shows the estimated model coefficients $\boldsymbol{\theta}_k$ and the forgetting factor $\lambda_k$. The parameters keep changing, although not as significantly at the later times compared to the initial time, while the forgetting factor remains mostly at 1, indicating no forgetting.



(a) $t = 10$

(b) $t = 84$

**Fig. 8 Mach number contours (0–0.3) for the controlled 1DOF scenario. The noise is a numerical artifact of the ALE mapping.**

Overall, PCAC effectively pilots the airfoil to the commanded altitudes, with no prior knowledge of the system that

it is controlling. This lack of knowledge is evident in the longer learning period needed to reach the first command compared to subsequent commands, for which the controller is more effective because it has already constructed a representation of the system.

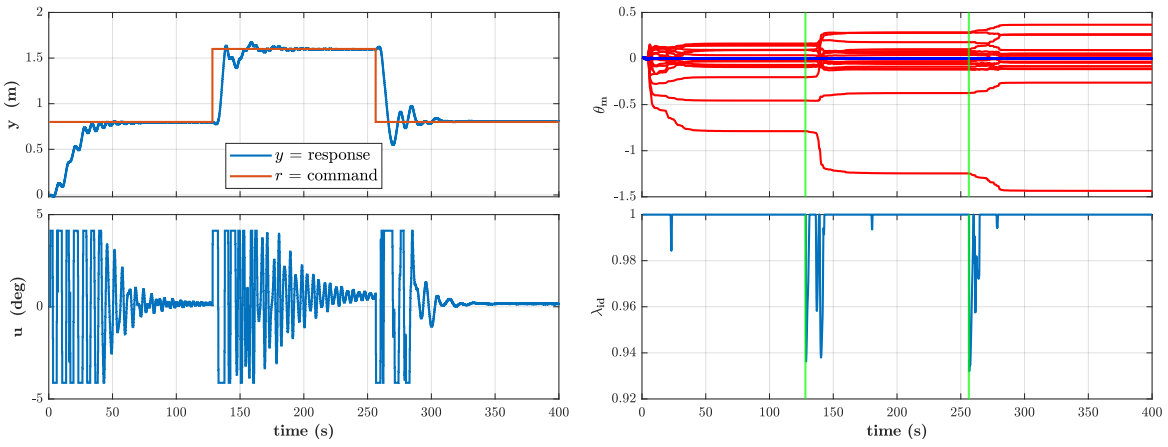
Figure 8 shows Mach-number contours for two snapshots of the unsteady simulation. The first snapshot is at $t = 10$, when the airfoil is on the way to the first commanded altitude, after initially moving downward instead. The second snapshot at $t = 84$ shows the quick response after the last altitude change: the immediate upward flap deflection to drop altitude indicates a correct identification of the system dynamics by PCAC. Animations of this and other simulations will be presented at SciTech.

### C. Control of 2DOF Airfoil Dynamics

The next scenario is 2DOF dynamics of the airfoil, in which the airfoil can pitch in addition to plunging vertically. The lift force dictates the plunging acceleration, whereas the CG moment dictates the pitch angular acceleration. The aerodynamic drag on the airfoil is ignored. The computational mesh, solution approximation order, time step, and freestream Mach number remain the same as in the previous example.

We again investigate the ability of PCAC to attain and maintain a commanded altitude $y$ by actuating the flap angle $u$. Following a steady-state solution at $y = 0$, we increase both the amount of time between setpoint changes as well as the setpoint heights themselves, although the pattern remains similar to the 1DOF scenario: the first setpoint is at $y = 0.8$ until $t = 128$, the second setpoint is at $y = 1.6$ until $t = 256$, and the final setpoint returns to $y = 0.8$ until the end of the simulation at $t = 400$. Figure 9(a) shows the commanded altitude setpoints and responses, as well as the flap deflection state vs time in the unsteady xflow simulation. As we can see, at the beginning of the simulation, PCAC deflects the flap back and forth while building up its model of the physical system. PCAC first requests a positive flap deflection, similar to the 1DOF scenario; however, in the 1DOF scenario, the positive deflection caused a negative camber, producing negative lift and moving away from the setpoint, while in this 2DOF scenario, an upward flap deflection causes a pitch-up moment that increases angle of attack and therefore lift, producing positive lift after a short amount of time and moving the airfoil towards the first altitude setpoint.

As the model is built up, the number of oscillations decreases when the commanded altitude changes; the last setpoint, although having a few oscillations of larger magnitude than at the start of the simulation, ultimately converges to the setpoint faster and with fewer oscillations. We can also see that PCAC improves by looking at the control input that PCAC requests in the same figure. For the first two setpoints in particular, the frequency of flap oscillations is high, and the magnitude is also high relative to the maximum allowed deflection. For the last setpoint, PCAC learns how to control this system using less oscillatory flap deflections.




(a) Control input, command, and response

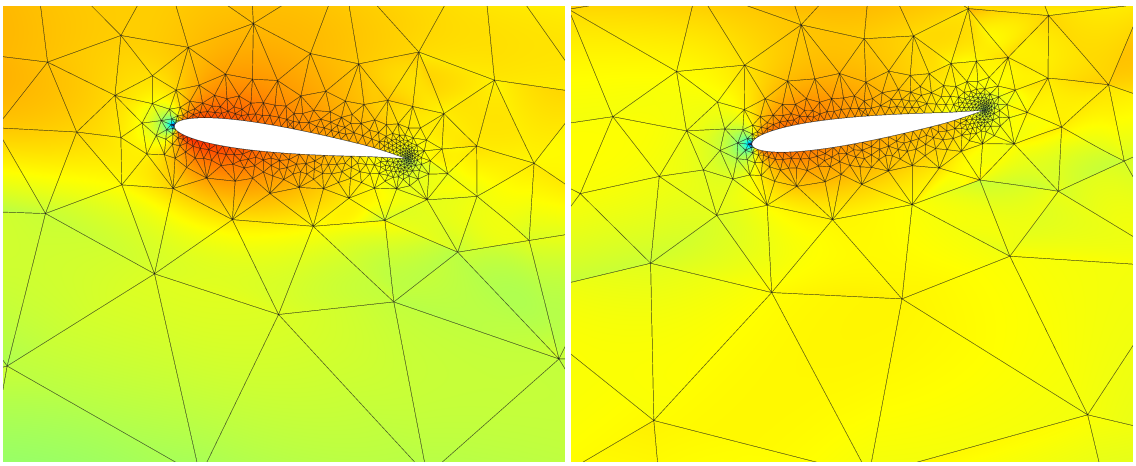
(b) Estimated model coefficients and the forgetting factor

**Fig. 9 PCAC response for the 2DOF scenario. The altitude response is oscillatory during the first setpoint command due to the additional pitch dynamics, but these oscillations diminish with subsequent setpoint commands, as do the flap-control input oscillations. Forgetting is active during command changes.**

The increasingly more efficient responses are, as mentioned, due to PCAC updating its model through online

identification while the simulation is running. In Figure 9(b), we can observe the model coefficients denoted by $\theta_m$ and the forgetting factor $\lambda_{id}$ over time. The vertical green bars represent setpoint changes. We see that within the first 50 s, there is significant change in the model coefficients as the model is identified at the same time the airfoil is controlled, and forgetting is not yet utilized (other than the small spike at about $t = 25$). However, once the altitude setpoint is changed, we see that forgetting is activated and the model coefficients update, allowing the airfoil to experience fewer oscillations with each change in commanded altitude. The model coefficients also update less with each setpoint change, indicating that the model does not require as much improvement with subsequent commands even when the direction of the setpoint values changes, that is, transition from a lower to higher altitude vs vice versa.

After having examined the states of the system over time, a visualization of the mach contours at different times is shown in Figure 10. At time $t = 136$, which is less than 10 s after the first altitude setpoint change is introduced, we see that the flap deflection sequence requested by PCAC caused the airfoil to pitch up, moving towards the setpoint at a higher altitude due to the increased lift production. At time $t = 264$, however, which is less than 10 s after the second altitude setpoint change, we see that the airfoil is pitching down, producing negative lift and therefore moving towards the setpoint at a lower altitude.



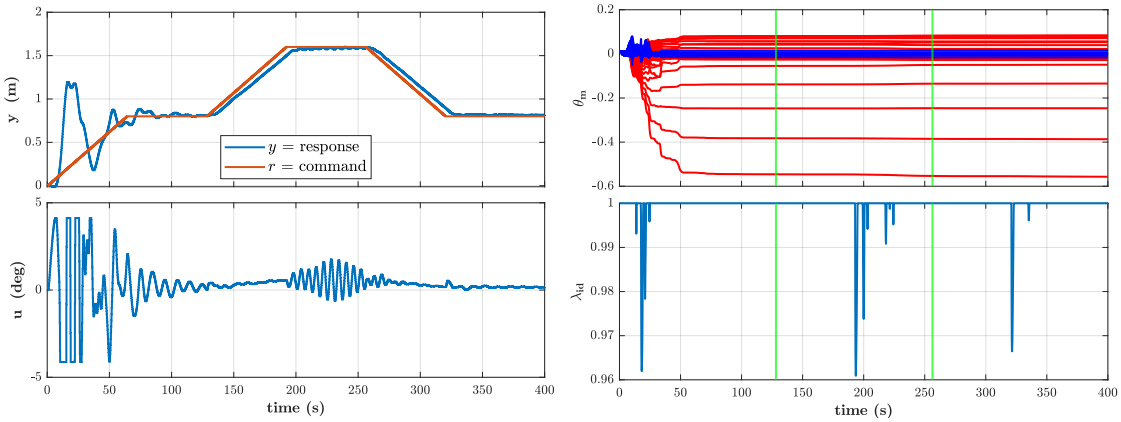
(a) $t = 136$ (b) $t = 264$

**Fig. 10 Mach contours (0–0.35) for the controlled 2DOF scenario. Deflection of the flap excites the plunge and pitch dynamics.**

Although the airfoil did successfully follow the altitude setpoints, there are many oscillations and it took until the 3rd setpoint in order to reduce them significantly. There are a few ways of increasing PCAC's performance to reduce this behavior–increasing model order can be effective, especially in a system where the number of degrees of freedom exceeds the number of outputs (such as this one–a 2DOF simulation with only altitude as an output), as well as replacing the step commands with ramps. Figure 11(a) shows that, compared to the previous setup, the number and strength of oscillations is reduced significantly. Although the initial transients increased in magnitude, the altitude response after the first segment improved overall.

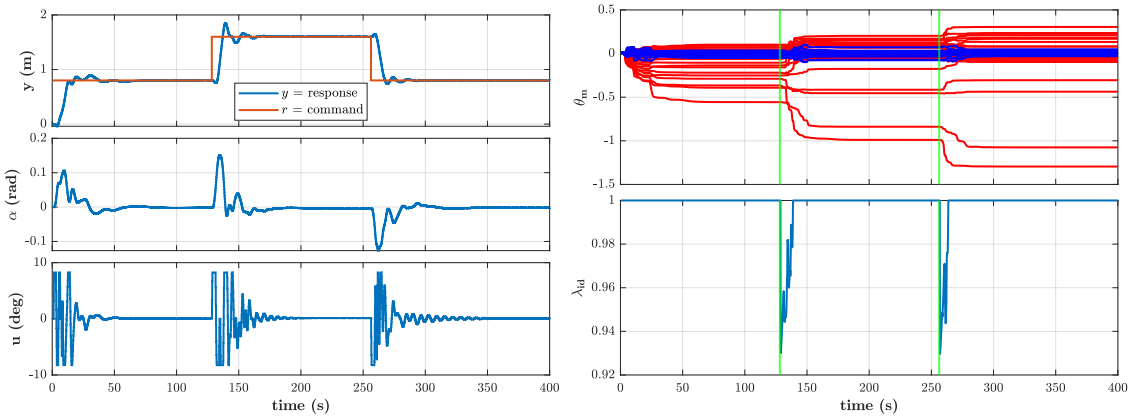
When examining the model coefficients in Figure 11(b), we see that, compared to the step command scenario, there are fewer jumps since there are sudden setpoint changes and the physical configuration of the system is never altered (i.e. the geometry or physics). The forgetting factor is also largely unactivated–this is again due to no jumps in setpoints and no system configuration changes.

Although increasing the model order and adding ramps can improve the response, matching the number of degrees of freedom with the number of outputs results in the most effective solution, also allowing the model order to be decreased. Including angle of attack as an auxiliary (not commanded) output does increase the controller's effectiveness, as shown in Figure 12(a). First, we see that the flap deflections bounce between the maximum deflections only a few times per setpoint change (and only once on the last setpoint), compared to flipping back and forth for about 10 periods per setpoint change in the SISO scenario as shown in Figure 9(a). The amplitude of oscillation also decreases for the flap deflection as the airfoil approaches the commanded altitude, at least for the first two setpoints. Settling time decreases as well, as does percent overshoot for the last setpoint change; overall, the system is controlled more effectively.




(a) Control input, command, and response (b) Estimated model coefficients and the forgetting factor

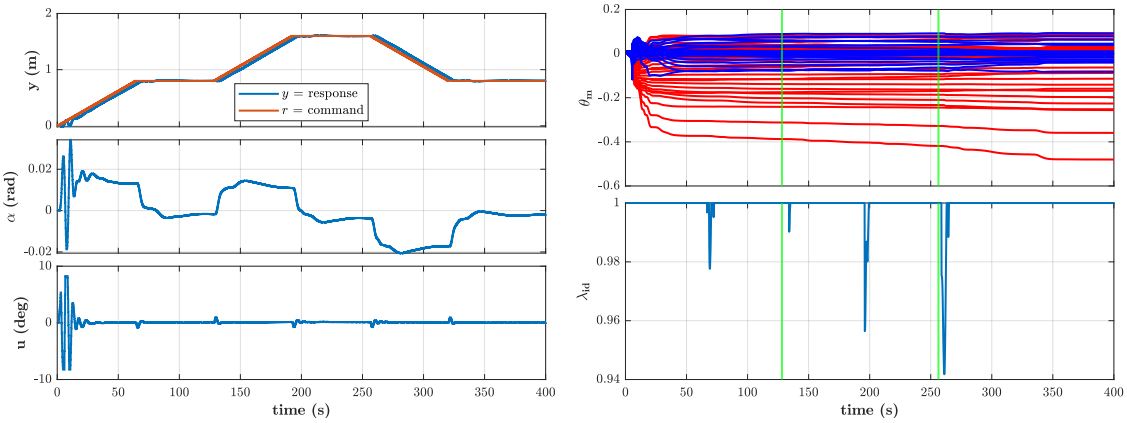
**Fig. 11 PCAC response for the 2DOF scenario with increased model order and ramp command transitions between setpoints.**




(a) Control input, angle of attack, command, and response (b) Estimated model coefficients and the forgetting factor

**Fig. 12 PCAC response for the 2DOF scenario with angle of attack as an auxiliary measurement. Compared to the SISO scenario, the flap oscillations are greatly diminished while improving the ability of the airfoil to follow the setpoint.**

We keep all other parameters constant between the step SISO scenario and this scenario in order to determine how the effectiveness of the controller increases with an extra output state for the system identification. If we also take a look at the model coefficients and forgetting factor in Figure 12(b), we see that, compared to the step SISO scenario shown in Figure 9(b), the process of model identification and forgetting is very similar. At the setpoint changes, we still see forgetting occurring from the $\lambda_{id}$ subplot and a corresponding shift in the model coefficients in the $\theta_m$ subplot, although the forgetting factor activates more uniformly in the second scenario.




(a) Control input, angle of attack, command, and response

(b) Estimated model coefficients and the forgetting factor

**Fig. 13 PCAC response for the 2DOF scenario with an auxiliary angle of attack measurement and ramp transitions.**

Although the system is already controlled quite well, if ramps are used instead of step commands, the airfoil follows altitude setpoints with almost no overshoot or settling time. The initial transients coming from the identification phase at the beginning of the simulation are removed almost entirely, as shown in Figure 13(a).

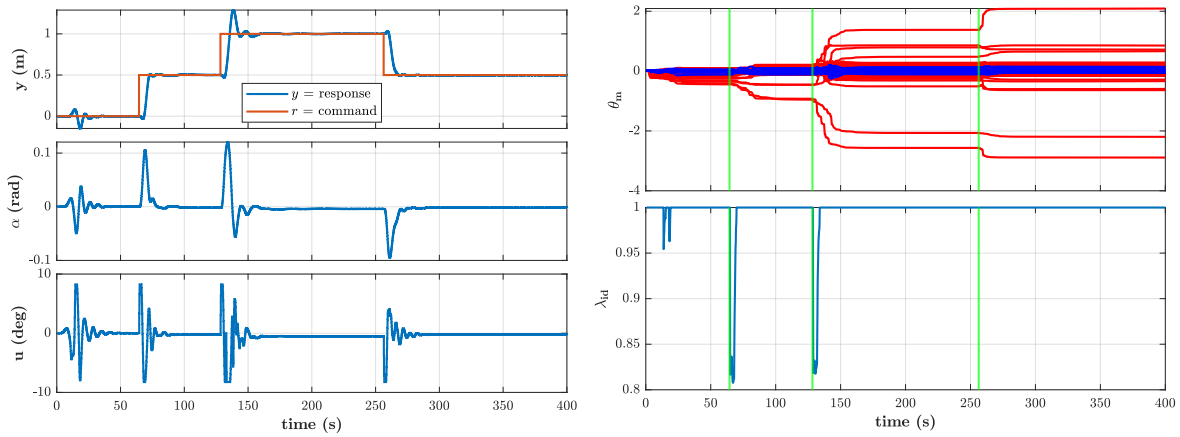
As for the model coefficients, the trends mostly follow the other ramp scenario without sudden jumps in model coefficients. However, they do still update slowly, owing to the gradual setpoint change due to the ramp transitions. There is more forgetting than in the first ramp scenario but still less than in the step scenarios, as shown in Figure 13(b).

### D. Control of 2DOF Airfoil Dynamics in Transonic Flow

The next scenarios introduce a faster flow regime, going from subsonic, mostly incompressible flow to transonic, compressible flow. This enables more nonlinear effects such as shocks to form, testing PCAC more thoroughly in situations with more complicated physics. The airfoil mesh remains the same as in the first two scenarios, but now the Mach number has increased from $M_\infty = 0.2$ to $M_\infty = 0.74$. Since the Euler equations are still used, we are neglecting viscosity for now. Although the Mach number is under 1, shocks are still able to form as the Mach number increases while flow is accelerating around the airfoil, causing shocks to appear on both sides, as shown in Figure 15. Although shocks cause drag, it is ignored in this 2DOF scenario. We have also decided to include gravity in the simulations moving forward, computed from the physical value 9.81 $\mathrm{m/s^2}$ by assuming a 1m chord airfoil moving at 251.7m/s.

The model order and prediction horizon are kept the same as the previous 2DOF scenarios. The initial covariance is increased by one order of magnitude, which makes the model coefficients less resistant to change as the simulation is started.

We once again investigate the ability of PCAC to follow an altitude setpoint with cold start, that is, with no prior model. Following a steady-state solution at $y = 0$, the controller is given free reign in the transonic flow regime by requesting flap deflection while plunging and rotating due to the interaction of the airfoil with the flow stream. This time, the airfoil starts with the altitude setpoint $y = 0$. Since gravity is now present, the airfoil cannot remain in steady flight without flap deflection, although PCAC deflects control surfaces in order to start the system identification process even with no command-following error. We see from Figure 14(a) that, after an initial flap deflection, the pitch angle and altitude of the airfoil oscillate for a short period of time until approaching a steady state at the altitude setpoint. At the next two setpoints ($t = 64$ and $t = 128$), we see that PCAC has constructed an accurate model since there is not much overshoot and the settling time is quick for the commanded altitudes. When the system encounters an altitude




(a) Control input, angle of attack, command, and response

(b) Estimated model coefficients and the forgetting factor

**Fig. 14 PCAC response for the 2DOF transonic scenario with gravity.**

setpoint that is in the opposite direction at $t = 256$, the controller uses the identified model to quickly approach and settle at the correct altitude.

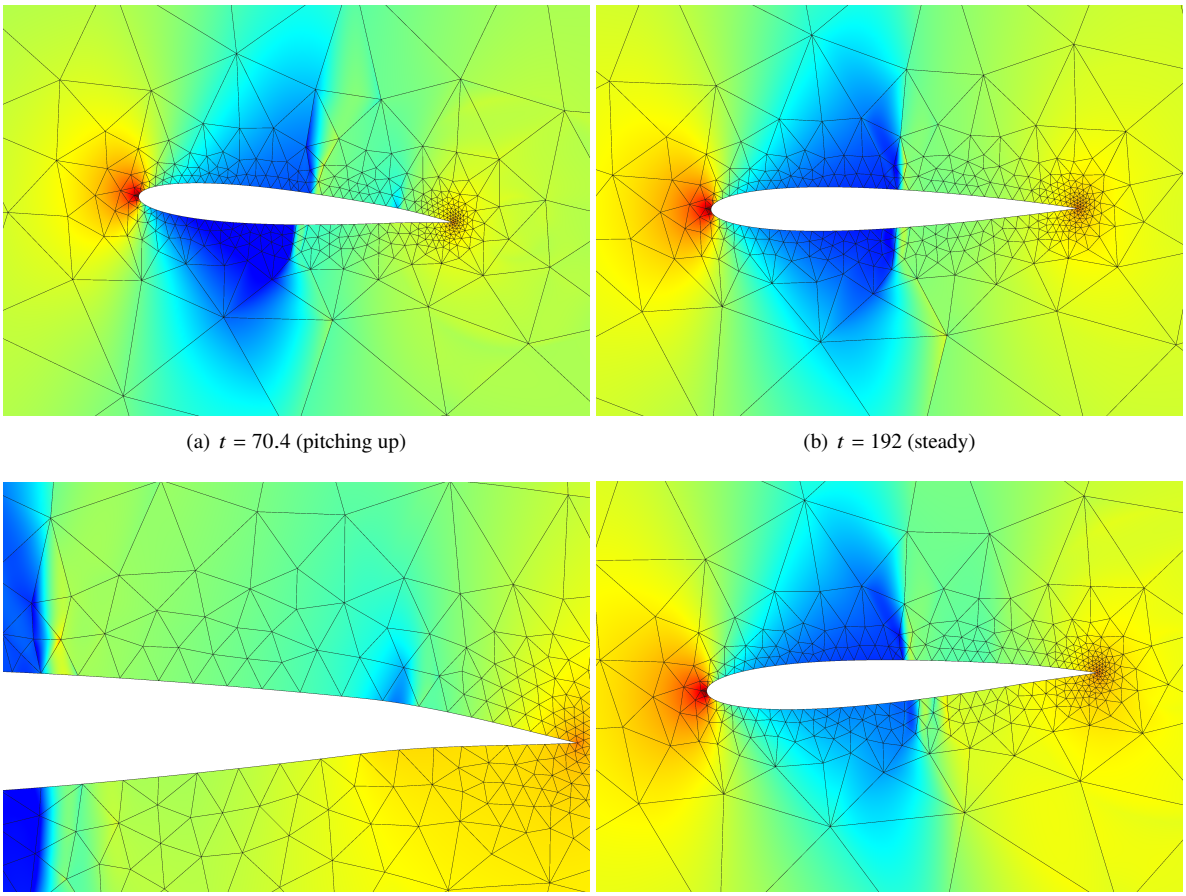
Figure 14(b) shows us that at the first setpoint change, although a high magnitude of forgetting is activated, the model coefficients change less that usual. However, at the second setpoint change, we observe a more expected update in the PCAC parameters, along with a similar magnitude of activated forgetting. In this situation, excessive forgetting can cause a slight momentary decrease in model accuracy; we see the higher overshoot as the airfoil approaches the commanded altitude after the second setpoint change. This is not always the case, but since the physics of the system and the nature of requested commands have not changed drastically, we do not see a huge benefit from forgetting at this point in the simulation. However, due to the forgetting factor causing the model coefficients to update, PCAC is much more effective at handling a negative altitude change.

Examining Figure 15, we can observe visualizations of the flow field during pitch up, steady, and pitch down maneuvers. Due to the presence of gravity, the shocks are not symmetric over the chord line for the symmetric airfoil during steady flight in order to offset the gravity and remain at the same altitude. During pitch down and pitch up maneuvers, this is shown more clearly, although this time the shocks are offset such that the airfoil is producing positive and negative lift, respectively. There is also more of a shock location difference between the upper and lower surfaces during the pitch up maneuver than the pitch down maneuver as the airfoil is fighting gravity. We can also see that not only do the shocks on the upper and lower surfaces move depending on pitch angle, expansion waves are created when the flap is deflected, further adding to the complexity of the physical system–yet PCAC handles the nonlinear dynamics effectively.

Finally, a curious observation is that the steady state of the system is not unique–we see this especially by observing the flap deflection at the third and fourth setpoints once the system has stabilized (between $t \approx 175$ to $t = 264$, and $t \approx 300$ to $t = 400$ in Figure 14(a)). The flap deflections are very close, but the flap is deflected more at the third setpoint and the angle of attack is also different, allowing both combinations to result in steady flight. Another interesting observation is that the steady solution with gravity is yielding a very slightly negative angle of attack, but this causes positive lift due to the slightly negative flap deflection which shifts the shock imbalance such that the shock forms earlier on the bottom surface of the airfoil, providing a larger sectional area of higher pressure behind the shock than on the top surface and contributing enough lift to counteract gravity.

Another feature of PCAC is the ability to impose constraints on outputs. For example, the angle of attack has been an output that, up until this point, was used only for more effective model buildup, but if we constrain it by imposing limits on the maximum and minimum angles of attack, it could be helpful for simulations that rely on reasonable angles of attack for accuracy. We are introducing constraints in the 2DOF transonic scenario as these constraints will be used in all later scenarios, as well as in order to show results for the scenario with the standard altitude command pattern that has been used for previous tests, as shown in Figure 16.

As Figure 16(a) shows, PCAC satisfies the constraint imposed on the angle of attack at $\pm 0.12$ rad. PCAC quickly learns and is able to follow commands on the altitude, especially at the third setpoint. The model coefficients and the



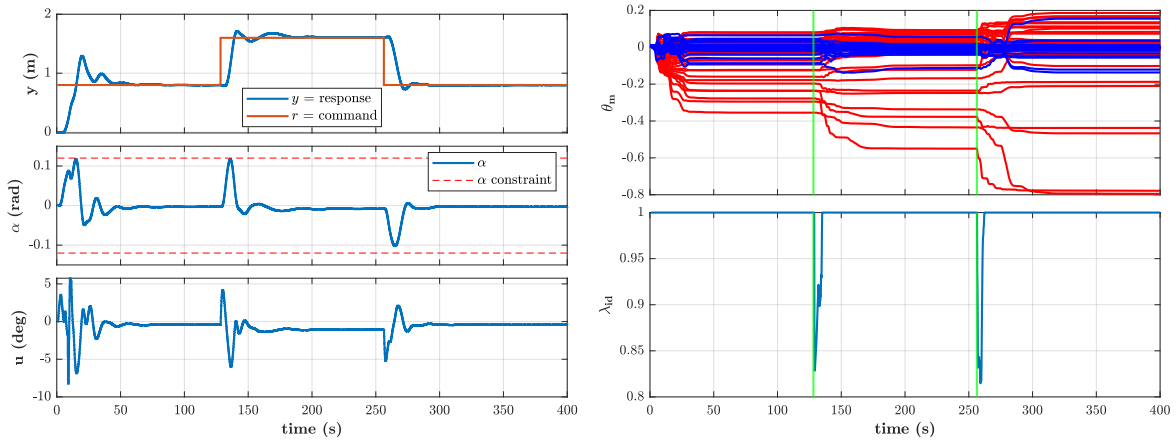
(a) $t$ = 70.4 (pitching up)

(b) $t$ = 192 (steady)

(c) $t$ = 257.6 (close-up of flap expansion wave)

(d) $t$ = 262.4 (pitching down)

**Fig. 15 Density contours (0.5484–1.36) for the controlled 2DOF transonic scenario. Deflection of the flap excites the plunge and pitch dynamics, causing shocks to move along the upper and lower surfaces of the airfoil.**




(a) Control input, angle of attack with constraints, command, and response

(b) Estimated model coefficients and the forgetting factor

**Fig. 16 PCAC response for the 2DOF transonic scenario with angle-of-attack constraints and gravity.**

forgetting factor also follow the well-established pattern of approaching a steady value and updating at each setpoint (with forgetting occurring at setpoints) as shown in Figure 16(b).

### E. Control of 3DOF Airfoil Dynamics in Transonic Flow

After testing PCAC in nonlinear systems with a single controlled input (flap) and single commanded output (altitude), we can move on to stressing the controller more by introducing the last degree of freedom possible in two dimensions–horizontal position. We have implemented testing horizontal position by using a control volume that moves at the same speed and direction as the freestream and measuring the airfoil's displacement relative to this frame that is stationary in the CFD simulation. This horizontal position can also be thought of as a pursuit metric–if there were an aircraft moving at constant speed in front of the airfoil, "horizontal position" would be measured as the distance from such an imaginary aircraft, with $x = 0$ defined as the starting position, $x > 0$ when the airfoil is farther away (moves to the right) while pursuing the aircraft, and $x < 0$ when the airfoil is closer to the imaginary aircraft (moves to the left).

Since there is an added commanded output on top of the altitude and constrained angle of attack, we are now up to three commanded outputs using only one control input. Furthermore, since the airfoil can now move horizontally in a relative frame, it has no way to counteract the wave drag to control movement in the relative horizontal position, specifically to move to the left. This is why we have added a thrust input that acts as if it were placed collinear with the chord, so the force due to thrust acts parallel to the chord in the direction of the vector pointing from the trailing to the leading edge.

Now that there are two commanded outputs (altitude and pursuit position) and two inputs (flap and thrust), plus a constrained output (angle of attack), the system can be controlled. In this scenario, we are stressing the ability of PCAC to perform MIMO control in a nonlinear dynamics environment. Again, the model order and prediction horizon are kept constant, but this time the initial covariance is decreased substantially. The reasoning is that at the start of the simulation, the model is less resistant to change, and the controller has a tendency to maximize control inputs in order to build up the model. This can be counteracted by changing some of the model predictive control optimization weights–specifically, decreasing the cost-to-go weight and increasing the controller output change weight.

These changes increase penalties during the optimization process to larger changes in control input deflections, which counteracts the effects of a high initial covariance (and also smooths out flap deflections later in the simulation), but we find that decreasing the initial covariance in the scenarios going forward is preferable in conjunction with modifying the optimization parameters. The forgetting factor is also decreased in order to avoid model changes that can momentarily cause undesirable effects, such as the increased overshoot for one of the setpoints we saw in the unconstrained 2DOF transonic scenario (Figure 14(a)).

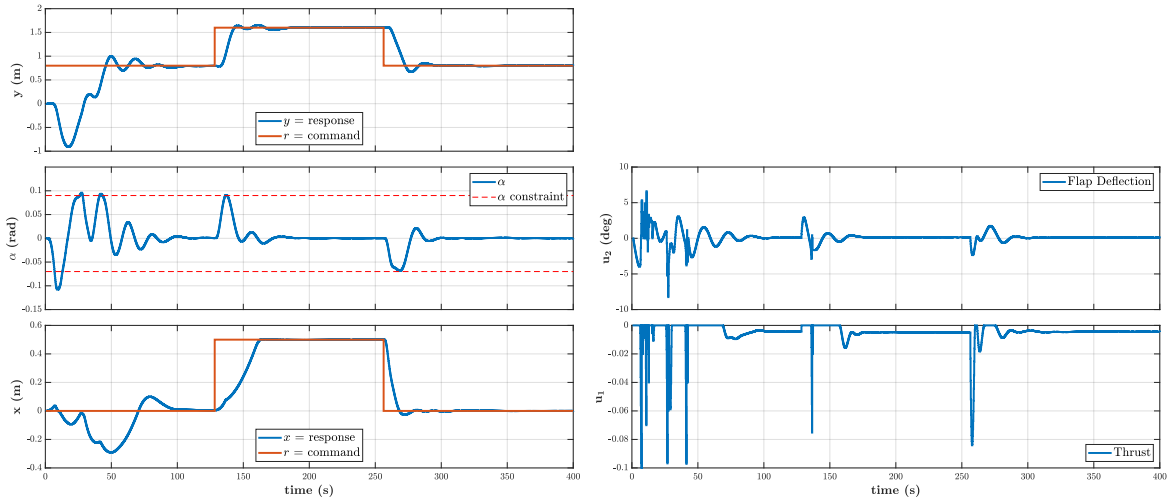



**Fig. 17 PCAC step transition response for the 3DOF transonic scenario with gravity.**

As shown in Figure 17, there are constraints imposed on the angle of attack; however, in the first segment (first setpoint, up until $t \approx 128$ s), the airfoil exceeds the minimum and maximum angles of attack, although this is somewhat

expected. The first instance occurs at $t \approx 10$ s, which is very early in the simulation, so the model, as shown in Figure 19(a), has not yet approached an accurate representation of the system. However, because of the constraint, even with an initially imperfect "awareness" of the physical system, PCAC quickly determines that reversing the flap deflection will allow the airfoil to return to the set angle of attack margin.

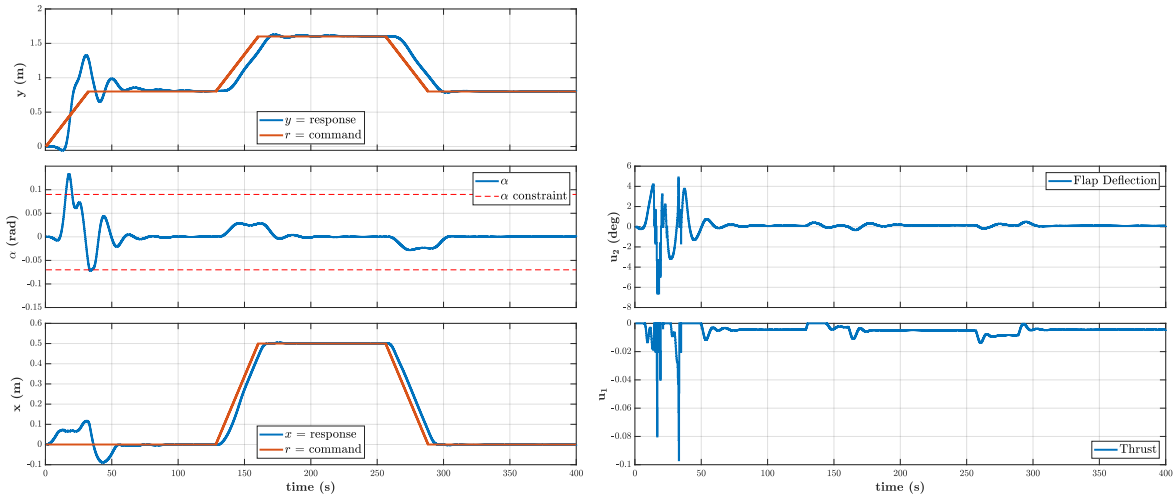
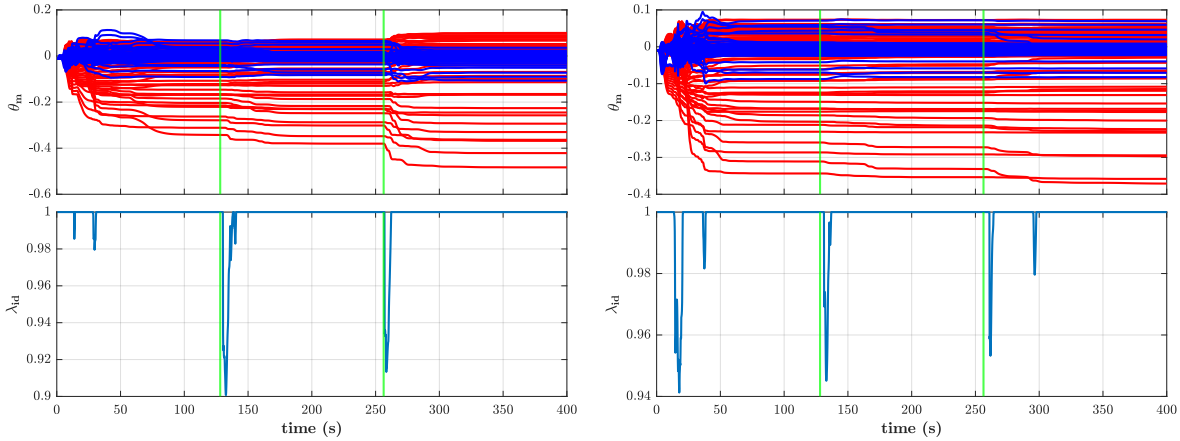



**Fig. 18 PCAC ramp transition response for the 3DOF transonic scenario with gravity.**

We also see that at $t \approx 27$ s and $t \approx 42$ s, the angle of attack of the airfoil barely exceeds the constraint, although this is still within the initial identification phase and PCAC successfully keeps to both the minimum and maximum constraints for the rest of the simulation.



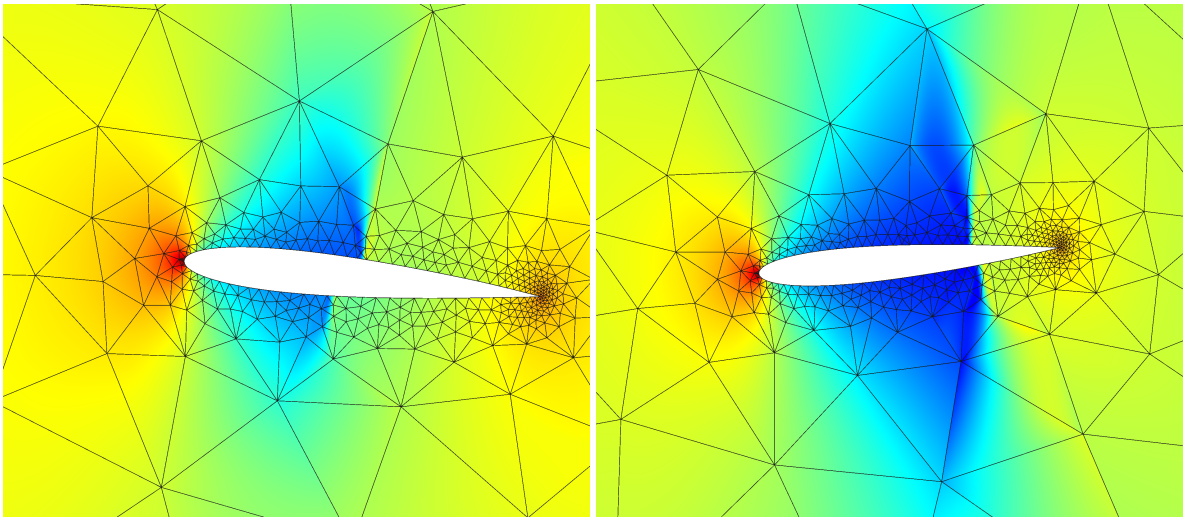

(a) Step transition scenario

(b) Ramp transition scenario

**Fig. 19 Model coefficients and forgetting factor for the 3DOF transonic scenario with gravity for step and ramp command transitions.**

Moving on to analyzing the behavior of the commanded outputs, PCAC prioritizes approaching the altitude setpoint before attempting to converge to the horizontal position setpoint, although this is mostly a behavior for the first segment. For the second setpoint, PCAC has learned to juggle both requests, converging to both the altitude and horizontal pursuit setpoints simultaneously. Something to note is that the airfoil does not have thrust reversing capability, which means that it is limited in how quickly it can approach positive horizontal position setpoints by the amount of drag the airfoil can experience, which is why we see faster convergence for the third setpoint than the second, where it can leverage the available thrust.

However, even though the airfoil does not have thrust reversing capability, it can still quickly converge to horizontal pursuit setpoints in both directions, which is shown in the ramp transition scenario in Figure 18. Again, we see the smoother command changes allowing PCAC to respond more effectively, cutting down on the initial transients and almost completely removing overshoot. Settling time is also quick, and the angle of attack constraints are, similar to the step transition scenario, only exceeded once at the beginning of the simulation when the model is not yet representative of the physical system.



(a) $t = 136$ (b) $t = 264$

**Fig. 20 Density contours (0.5–1.339) for the step response 3DOF transonic scenario. Snapshots of the airfoil in the flowfield chosen to highlight additional dynamics due to the 3rd degree of freedom.**

From the model coefficient and forgetting parameter plots in Figure 19, we see that the trends in behavior follow what we have seen for previous scenarios–forgetting and model coefficient updates after the setpoints, with model coefficients approaching steady values in each segment. For the step transition scenario, we see larger changes in model coefficients and the forgetting parameters due to the setpoint jumps, with more gradual transitions for the ramp transition scenario.

Figure 20 shows that the extra degree of freedom significantly changes the dynamics of the system. We see that, when the airfoil is falling back and ascending, e.g. at $t = 136$ s, the shocks move closer to the leading edge, since the airfoil is effectively decelerating. At time $t = 264$ s, however, the airfoil is descending and using thrust, so it is accelerating and we see the shock has moved back and increased in size and strength. Overall, PCAC is successful in controlling this system throughout the simulation, especially after the model has been identified near the end of the first segment.

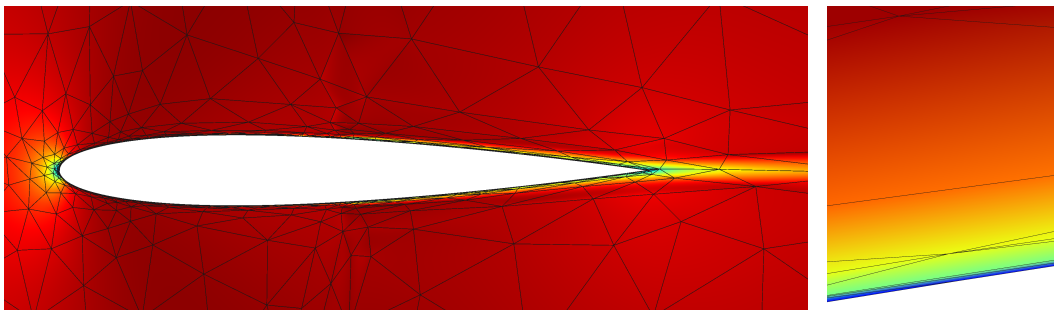
### F. Control of 3DOF Airfoil Dynamics in Transonic Flow with Turbulence

Our final scenario for testing a 2D airfoil is with all 3 possible degrees of freedom in the transonic flow regime with viscosity turned on in the form of turbulence. We have chosen a very high Reynolds number of $Re = 1.7 \times 10^7$ for this scenario, as it is representative of cruise conditions for most aircraft–for this scenario specifically, the Reynolds number is calculated from sea level conditions with a 1m chord at $M = 0.74$; however, at a similar Mach number and cruise altitude of $\approx$ 35000 ft, the Reynolds number is also around 20 million.

The Reynolds-averaged Navier-Stokes turbulence model is the Spalart-Allmaras (SA-neg) closure. This is a relatively simple model that allows for fairly accurate simulation within reasonable angles of attack, something we can enforce using pitch constraints with PCAC.

Because this is now a viscous simulation with a high Reynolds number, we needed to update the mesh to be much more resolved close to the surface of the airfoil to accurately capture the boundary layer. To achieve this, we use xflow's steady adjoint-based adaptive capability that minimized the effect of the discretization error on the drag, and we increased the number of elements in the mesh to 2544, while keeping the solution approximation order at $p = 2$. The updated mesh is shown in Figure 21, where contours of $x$-momentum are shown–it is difficult to see the boundary layer

near the leading edge without zooming in substantially since it is very thin, but closer to the trailing edge we see how the increased element density close to the airfoil surface allows the boundary layer to be resolved.



(a) Full View (b) Boundary Layer Closeup

**Fig. 21 X-Momentum Contours on refined airfoil mesh for the turbulent scenario.**

This time, the model order is marginally increased from 16 to 18, as is the prediction horizon to about 1/4 of the total simulation time from 1/5. This is done in order to give PCAC more "foresight" for the more physically demanding viscous scenario. The initial covariance is slightly increased, and the optimization cost-to-go weight is decreased substantially–too rapid of control deflections threatened to stall the airfoil, which would be unrecoverable in our simulations. The forgetting factor is kept the same as for the inviscid 3DOF scenario.

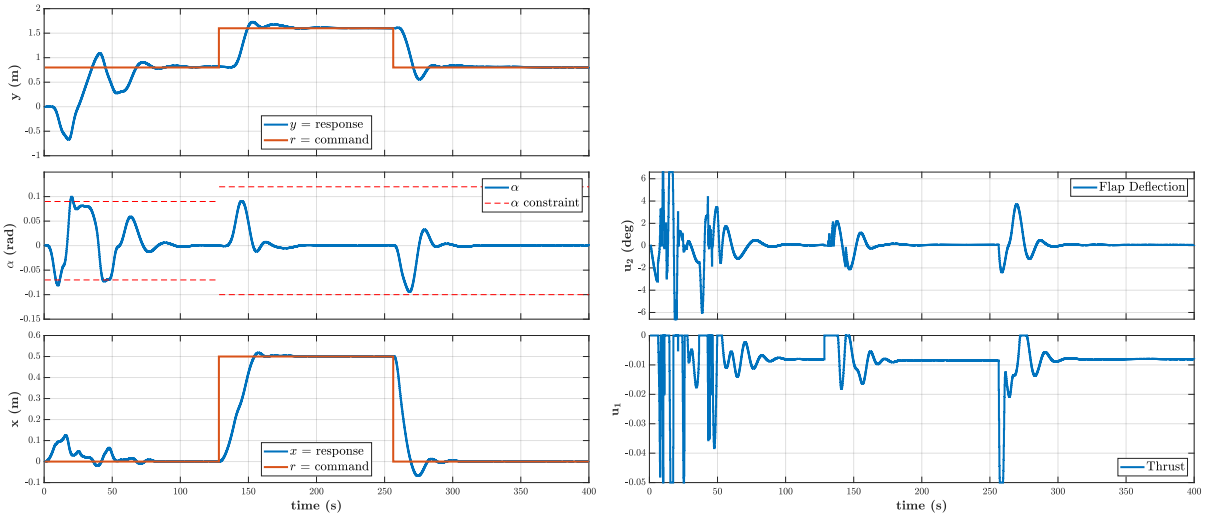



**Fig. 22 PCAC response for the 3DOF transonic scenario with turbulence and gravity.**

Testing PCAC using the same setpoint pattern yields favorable results from a high-fidelity model in Figure 22. Again, we see similar behavior with slight encroachment over the limits on the angle of attack in the first segment. However, in the later segments, with the second and third setpoints, we see that the angle of attack is successfully constrained within expanded bounds. In PCAC, we can change the constraint boundaries while the simulation is running–this is useful in scenarios such as this one, as it puts more pressure on PCAC to stay within reasonable bounds sooner when the model is not yet converged, forcing the controller to act sooner in the beginning to counteract what would otherwise be more serious infringements on the constraints that could or could not be recoverable. With this approach, we can keep the system stable at all times as well as allowing more leniency during the later stages once the model is close to convergence in order to improve command-following performance.

Otherwise, we see that, apart from the first setpoint, where PCAC is rapidly constructing the model, the airfoil follows the commands for both horizontal pursuit as well as altitude while keeping to the angle of attack constraint and

using reasonable control surface deflections. The model coefficients and forgetting factor, as shown in Figure 23, keep to the same pattern, although something to note is that there is slightly more deviation after the first setpoint than we saw in the inviscid scenario, likely because of the more complex physics and change of constraint limits.

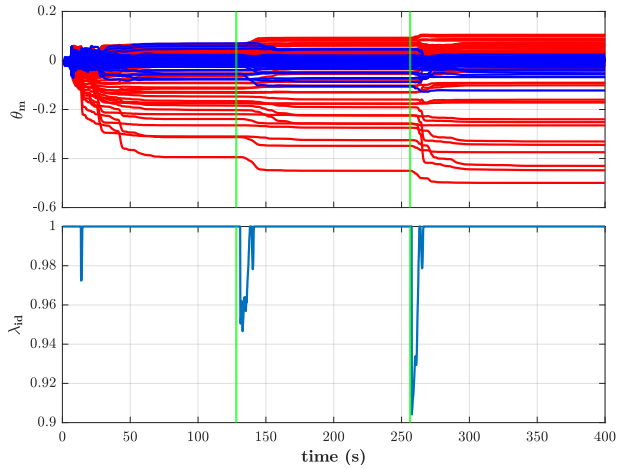



**Fig. 23 Model coefficients and forgetting factor for the 3DOF transonic scenario with turbulence and gravity.**

## V. Conclusions

This paper used predictive cost adaptive control (PCAC) to control an airfoil in two-dimensional flight. PCAC is a discrete-time (sampled-data) controller that requires only online sensor measurements to learn the vehicle dynamics. In particular, PCAC requires no prior knowledge of the system being controlled and instead identifies the system dynamics online during flight. In the present scenarios, the system is a high-order finite-element simulation of an airfoil in free flight, and the sensors measure the altitude, angle of attack, and horizontal pursuit position of the airfoil. The controller actuates a flap and turns on thrust, which change the force and moment on the airfoil, with both quantities predicted by the simultaneously running fluid simulation. The vertical motion of the flap and the pitch and plunge motion of the airfoil are implemented via an arbitrary Lagrangian-Eulerian formulation, which consists of rigid-body motion surrounded by cubic blending.

These results demonstrate the ability of PCAC to follow commanded altitudes and horizontal pursuit positions as well as to keep within angle of attack constraints for one, two, and three degree-of-freedom flight in subsonic and transonic flight regimes in both inviscid and turbulent flows. With online closed-loop system identification, PCAC improves its ability to follow the commanded altitude as the setpoint command changes. Ramp commands instead of setpoint commands can improve the effectiveness of PCAC, reducing overshoot and initial transients, as well as matching the number of degrees of freedom with the number of outputs. Different parameters can be adjusted to modify the behavior of the controller, allowing users to impose constraints on both inputs and outputs and tune its aggressiveness. Current and future tests using PCAC include a three-dimensional wing with up to six degrees of freedom in its flight dynamics.

# References


[1] Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P.-O., van Leer, B., and Visbal, M., "High-Order CFD Methods: Current Status and Perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, 2013, pp. 811–845. https://doi.org/10.1002/fld.3767.

[2] Fidkowski, K. J., and Darmofal, D. L., "Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics," *AIAA Journal*, Vol. 49, No. 4, 2011, pp. 673–694. https://doi.org/10.2514/1.J050073.

[3] Nguyen, T. W., Islam, S. A. U., Bernstein, D. S., and Kolmanovsky, I. V., "Predictive Cost Adaptive Control: A Numerical Investigation of Persistency, Consistency, and Exigency," *IEEE Contr. Sys. Mag.*, Vol. 41, 2021, pp. 64–96.

[4] Islam, S. A. U., and Bernstein, D. S., "Recursive Least Squares for Real-Time Implementation," *IEEE Contr. Syst. Mag.*, Vol. 39, No. 3, 2019, pp. 82–85.

[5] Bruce, A. L., Goel, A., and Bernstein, D. S., "Convergence and consistency of recursive least squares with variable-rate forgetting," *Automatica*, Vol. 119, 2020, p. 109052.

[6] Goel, A., Bruce, A. L., and Bernstein, D. S., "Recursive Least Squares With Variable-Direction Forgetting: Compensating for the Loss of Persistency," *IEEE Contr. Sys. Mag.*, Vol. 40, No. 4, 2020, pp. 80–102.

[7] Bruce, A. L., Goel, A., and Bernstein, D. S., "Necessary and Sufficient Regressor Conditions for the Global Asymptotic Stability of Recursive Least Squares," *Sys. Contr. Lett.*, Vol. 157, 2021, pp. 1–7. Article 105005.

[8] Mohseni, N., and Bernstein, D. S., "Recursive least squares with variable-rate forgetting based on the F-test," *Proc. Amer. Contr. Conf.*, 2022, pp. 3937–3942.

[9] Cockburn, B., and Shu, C.-W., "Runge-Kutta discontinuous Galerkin methods for convection-dominated problems," *Journal of Scientific Computing*, Vol. 16, No. 3, 2001, pp. 173–261. https://doi.org/https://doi.org/10.1023/A:1012873910884.

[10] Fidkowski, K. J., "Output error estimation strategies for discontinuous Galerkin discretizations of unsteady convection-dominated flows," *International Journal for Numerical Methods in Engineering*, Vol. 88, No. 12, 2011, pp. 1297–1322. https://doi.org/10.1002/nme.3224.

[11] Persson, P.-O., Bonet, J., and Peraire, J., "Discontinuous Galerkin Solution of the Navier-Stokes Equations on Deformable Domains," *Computer Methods in Applied Mechanics and Engineering*, Vol. 198, 2009, pp. 1585–1595.

[12] Kast, S. M., and Fidkowski, K. J., "Output-based Mesh Adaptation for High Order Navier-Stokes Simulations on Deformable Domains," *Journal of Computational Physics*, Vol. 252, No. 1, 2013, pp. 468–494. https://doi.org/10.1016/j.jcp.2013.06.007.

[13] Persson, P.-O., Fidkowski, K. J., and Wukie, N. A., "High-Fidelity CFD Workshop 2022: Mesh Motion," AIAA Paper 2021–1551, 2021. https://doi.org/10.2514/6.2021-1551.

[14] Wukie, N. A., Fidkowski, K., Per-Olof, and Wang, Z., "High-Fidelity CFD Verification Workshop 2023: Mesh Motion," AIAA Paper 2023–1243, 2023. https://doi.org/10.2514/6.2023-1243.

[15] Allmaras, S., Johnson, F., and Spalart, P., "Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model," Seventh International Conference on Computational Fluid Dynamics (ICCFD7) 1902, 2012.

[16] Ceze, M. A., and Fidkowski, K. J., "High-Order Output-Based Adaptive Simulations of Turbulent Flow in Two Dimensions," AIAA Paper 2015–1532, 2015. https://doi.org/10.2514/6.2015-1532.

[17] Fidkowski, K. J., "Three-Dimensional Benchmark RANS Computations Using Discontinuous Finite Elements on Solution-Adapted Meshes," AIAA Paper 2018–1104, 2018. https://doi.org/10.2514/6.2018-1104.

[18] Roe, P., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372. https://doi.org/https://doi.org/10.1016/0021-9991(81)90128-5.

[19] Bassi, F., and Rebay, S., "Numerical evaluation of two discontinuous Galerkin methods for the compressible Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, Vol. 40, 2002, pp. 197–207. https://doi.org/https://doi.org/10.1002/fld.338.

[20] Ceze, M. A., and Fidkowski, K. J., "Constrained pseudo-transient continuation," *International Journal for Numerical Methods in Engineering*, Vol. 102, 2015, pp. 1683–1703. https://doi.org/10.1002/nme.4858.

[21] Saad, Y., and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific Computing*, Vol. 7, No. 3, 1986, pp. 856–869. https://doi.org/https://doi.org/10.1137/0907058.

[22] Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L., "*p*-Multigrid solution of high–order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 207, 2005, pp. 92–113. https://doi.org/10.1016/j.jcp.2005.01.005.

[23] Persson, P.-O., and Peraire, J., "Newton-GMRES Preconditioning for Discontinuous Galerkin Discretizations of the Navier-Stokes Equations," *SIAM Journal on Scientific Computing*, Vol. 30, No. 6, 2008, pp. 2709–2733. https://doi.org/https://doi.org/10.1137/070692108.

[24] Cash, J., "The integration of stiff initial value problems in ODEs using modified extended backward differentiation formulae," *Computers & mathematics with applications*, Vol. 9, No. 5, 1983, pp. 645–657. https://doi.org/https://doi.org/10.1016/0898-1221(83)90122-0.

[25] Fidkowski, K. J., "An Output-Based Adaptive Hybridized Discontinuous Galerkin Method on Deforming Domains," AIAA Paper 2015–2602, 2015. https://doi.org/10.2514/6.2015-2602.

[26] Bruce, A. L., Goel, A., and Bernstein, D. S., "Convergence and consistency of recursive least squares with variable-rate forgetting," *Automatica*, Vol. 119, 2020, p. 109052.

[27] Mohseni, N., and Bernstein, D. S., "Recursive least squares with variable-rate forgetting based on the F-test," *Proc. Amer. Contr. Conf.*, 2022, pp. 3937–3942.

[28] McKeon, J. J., "F approximations to the distribution of Hotelling's $T_0^2$," *Biometrika*, Vol. 61, No. 2, 1974, pp. 381–383.

[29] Nguyen, T. W., Islam, S. A. U., Bernstein, D. S., and Kolmanovsky, I. V., "Predictive Cost Adaptive Control: A Numerical Investigation of Persistency, Consistency, and Exigency," *IEEE Contr. Sys. Mag.*, Vol. 41, 2021, pp. 64–96.