Uncertainty-Aware Model-Constrained Neural Digital Twins for Real-Time Forecast, Calibration, and Control of Aeroelastic Airfoils

William Cole Nockolds[1], Thomas A. Scott[2], Tan Bui-Thanh[3], Shatad Purohit[4], Sitaram Ramaswamy[5]

[1,3] *Oden Institute at the University of Texas, Austin, TX, 78712, USA*
cole.nockolds@utexas.edu
tanbui@utexas.edu

[2,3] *Dept of Aerospace Engineering and Engineering Mechanics at the University of Texas, Austin, TX, 78712, USA*
thomas.scott@utexas.edu

[4] *RTX AI Systems Engineering*

[5] *RTX Data and AI*

ABSTRACT

Current structural health monitoring (SHM) in aeroelastic systems struggles with recovering unobservable states from limited sensor data, lacks quantified prediction confidence, and exhibits brittleness under off-nominal conditions. To address these operational gaps, this paper presents a physics-aware neural digital twin (DT) designed for real-time, adaptive monitoring of aerospace structures. The proposed DT framework combines two neural networks for stabilizing control and forward prediction while using Bayesian filtering for nonlinear state estimation and uncertainty quantification. We train the controller using Proximal Policy Optimization (PPO) [1], an actor-critic method, and the forward model using the mcTangent loss [2]. We use Bayesian particle filtering to infer the state and provide uncertainty quantification for assurance. The DT then provides the control input to the physical twin using the mean state estimate. Using only pitch angle measurements from a nonlinear airfoil model, the DT recovers full system state, stabilizes the structure rapidly, and provides uncertainty quantification (UQ) with all predictions. Finally, we compare the results to a more traditional approach of ensemble Kalman filter (EnKF) and model predictive control (MPC) and the base ODE model rather than the surrogate, which preforms worse due to worse early estimation of the hidden state. We also discuss the architecture's transferability to broader digital twin applications across aerospace platforms.

1. INTRODUCTION

Over the past two decades, Digital Twins (DTs) have become increasingly popular tools for monitoring and prediction in complex engineering systems [3]. Fundamentally, a digital twin is a computer simulation of a physical system which seeks to mirror all aspects of its real counterpart through modeling and periodic information exchange, such that it may be used to monitor, optimize, or control its physical twin [4]. Monitoring, or diagnostic, twins may be used to ensure the health of the physical twin, while a twin for optimization and control endeavors to make the operation of the physical twin more resource efficient or guide it toward some user-defined control target [5, 6]. Digital twins are informed by environmental or performance data measured from the physical twin [3, 4]. DT approaches integrate sensor data and physical models to accurately represent a physical system [3]. To be predictive and trustworthy, DTs must be equipped with uncertainty quantification. Reliable DTs must be able to predict or detect when model drift is above prescribed acceptable thresholds or the model is operating in out-of-distribution regimes. They then acquire necessary data to self-update to stay close to their physical counterparts. DTs should be structured such that they could support downstream tasks such as design, optimization, control, and decision making. For applications that require fast actions, machine learning methods are typically deployed to mitigate the high computational expense of many traditional approaches [7, 8].

Digital twins in aerospace applications have been implemented to solve problems such as predictive maintenance, vehicle life prediction, and structural health monitoring (SHM) [9, 10, 11]. These prognostic DTs focus on estimating dam-

age or system states from measurements [12, 13]. However, in many practical aerospace systems, sensors are sparse, making reconstruction of the physical state difficult [14, 15]. For nonlinear dynamical systems, partial observability makes estimation of unobserved states a challenge [16, 17]. Additionally, many predictive DTs are deterministic and therefore lack appropriate uncertainty quantification to be used for decision making in safety-critical environments [18, 19, 20]. Kalman filter variants (Kalman filter, Ensemble Kalman filter) have been used for state estimation with uncertainty quantification, but each relies on strict Gaussian assumptions [21, 22]. Particle filtering methods forgo the Gaussian assumptions by using weighted samples to build a non-Gaussian posterior [23].

While many aerospace DTs focus on diagnostics and prediction rather than closed-loop control, parallel research has explored using DTs for control estimation via model predictive control (MPC) and reinforcement learning (RL) strategies [18, 24, 25]. These works demonstrate the capacity for DTs to move beyond model state reconstruction and toward active system control and optimization. However, MPC requires many model evaluations to facilitate its online optimization procedure, which may make it computationally infeasible for real-time control in the case of expensive nonlinear models [26, 27]. Reinforcement learning approaches, on the other hand, shift the computational burden of optimizing the control policy offline through training [1]. Still, few works integrate state estimation, prediction, control, and UQ in a unified framework, indicating a gap between the research field and deployable DTs for real aerospace systems.

To address the existing challenges and limitations of current DTs, this paper proposes a model-constrained DT for a nonlinear aeroelastic system [28]. The developed framework performs state estimation, prediction, control, and mathematically principled UQ. The twin combines a PPO [1] trained RL agent to control the airfoil with an mcTangent loss [2] trained forward neural ODE model [29]. PPO is an actor-critic algorithm which updates a learned policy using a policy gradient approach and a state value function using gradient descent. The updates are restricted in magnitude in order to induce a trust-region-like behavior in optimization which empirically improves training stability [1]. The mcTangent loss enforces consistency with a ground truth ODE model to promote long-term prediction capability and limit training data requirements [2]. In the problem setting, only one state variable, the pitching angle α, is directly observable. Bayesian particle filtering is used to estimate unobserved states and provide uncertainty quantification for forward neural ODE predictions. The framework successfully performs state estimation which can be used for control and prediction with the trained neural models to reduce airfoil flutter and forecast future system states with uncertainty estimates. Finally, we provide a comparison of our approach to a method using MPC with and ensemble Kalman filter for state estimation in the same problem setting.

2. AEROELASTIC MODEL

We consider a model of a wing undergoing aeroelastic deflection. The second order ODEs describing this model are nonlinear and represent the time evolution of pitching α and plunging h motion. A single control input, flap deflection angle β is featured and only the pitching angle α is observed. The equations of motion of this model, derived from small-angle and other simplifying assumptions in [28, 30], are

$$M\ddot{z} + D\dot{z} + K(\alpha)z = F\beta, \qquad z = \begin{bmatrix} h & \alpha \end{bmatrix}^\top, \tag{1}$$

$$M = \begin{bmatrix} m_t & m_w x_\alpha b \\ m_w x_\alpha b & I_\alpha \end{bmatrix}, \tag{2}$$

$$D = \begin{bmatrix} d_h & 0 \\ 0 & d_\alpha \end{bmatrix} + \rho U b s_p \begin{bmatrix} c_{l_\alpha} & c_{l_\alpha}\left(\frac{1}{2} - a\right)b \\ -c_{m_\alpha} b & -c_{m_\alpha}\left(\frac{1}{2} - a\right)b^2 \end{bmatrix}, \tag{3}$$

$$K(\alpha) = \begin{bmatrix} k_h & \rho U^2 b c_{l_\alpha} s_p \\ 0 & k_\alpha(\alpha) - \rho U^2 b^2 c_{m_\alpha} s_p \end{bmatrix}, \tag{4}$$

$$F = \rho U^2 b s_p \begin{bmatrix} -c_{l_\beta} \\ c_{m_\beta} b \end{bmatrix}. \tag{5}$$

where b is the semichord, s_p is the span, x_α is the non-dimensionalized distance of the center of mass from the elastic axis, ρ is the fluid density, $d_{h,\alpha}$ are damping coefficients, m_t is the total mass, m_w is the mass of the wing, I_α is its pitching moment of inertia and $c_{l,m_{\alpha,\beta}}$ are the lift and moment coefficient slopes with respect to the pitch angle and flap angle respectively. A quartic polynomial is prescribed for the spring constant of the pitching dynamics: $k_\alpha(\alpha) = \sum_{i=0}^{4} c_i \alpha^i$. We use the same constants as defined in [28].

The equations can be expressed in state-space form with a state vector $x = \begin{bmatrix} z & \dot{z} \end{bmatrix}^\top = \begin{bmatrix} h & \alpha & \dot{h} & \dot{\alpha} \end{bmatrix}^\top$ and control input $u = \beta$. The observation is the pitch angle $y = \alpha$. The state-space representation of this system of equations is:

$$\dot{x} = A(x)x + Bu \tag{6a}$$

$$y = Cx \tag{6b}$$

$$A(x) = \begin{bmatrix} 0 & I \\ -M^{-1}K(\alpha) & -M^{-1}D \end{bmatrix} \tag{6c}$$

$$B = \begin{bmatrix} 0 \\ M^{-1}F \end{bmatrix} \tag{6d}$$

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}, \tag{6e}$$

where C is the observation matrix.

3. METHODOLOGY

Practical digital twins require methods that can operate under partial observability, measurement uncertainty, and computational constraints present in aerospace applications. For

systems governed by ODEs, like the aeroelastic system considered in this paper, prediction via numerical integration often requires full knowledge of the system state. However, due to sensor limitations in aerospace applications, often only partial state observations are available, indicating a need for robust state reconstruction. Furthermore, closed-loop control algorithms also necessitate or benefit from full state estimation. In cases in which real-time simulation and control are necessary, traditional nonlinear control schemes and ODE solvers are often impractical due to their computational cost. Faster neural methods are strong candidates as alternatives to traditional approaches for control and forward modeling. We propose a dual component neural architecture for real-time control determination and predictive forward modeling. In addition, we integrate our methodology with particle filtering techniques to facilitate real-time state and uncertainty estimation. The components of the architecture are as follows:

1. RL Trained Control Policy: A neural network which sees the current mean system state estimate and calculates the control that will lead to stabilization.
2. Model-Constrained Neural ODE: A neural ODE network which maps from the current state and control to the next state.
3. Particle Filter: An ensemble of digital twin simulations approximating the belief distribution as Monte Carlo samples with weights updated using Bayes rule after each noisy observation.

Together, these components result in a DT framework which handles state estimation, prediction, control, and uncertainty quantification.

3.1. Optimal Control Problem

The training of the control policy follows the PPO method [1] of reinforcement learning. RL seeks to minimize the expected discounted cumulative reward,

$$\begin{aligned} J(\pi_\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{N_t}\gamma^t r(x_t,u_t)\right], \\ \theta^* &= \arg\max_\theta J(\pi_\theta), \end{aligned} \tag{7}$$

where x is the state (the ODE variables $h,\ \alpha,\ \dot{h},\ \dot{\alpha}$), u is the action (β), γ is the discount factor, T is the time horizon length, r is the reward function, π_θ is the distribution of actions given a policy parameterization θ, and θ parametrizes our controller network, which outputs parameters for a normal distribution we sample during training and use the mean for run-time inference. Actor-critic methods learn both the optimal policy θ and the value function V required to train the policy simultaneously. Both the policy and value function are neural networks with trained parameters. The ensuing loss, with an entropy bonus to encourage exploration, is

$$\mathcal{L}^{\text{Total}} = \mathcal{L}^{\text{CLIP}} + c_1\mathcal{L}^{\text{VF}} - c_2\mathcal{S}[\pi_\theta], \tag{8}$$

where c_1 and c_2 are weighting hyperparameters while S is the differential entropy. And the PPO policy loss, whose clipping is meant to mimic a trust region, is

$$\begin{aligned} \text{pr}_t(\theta) &= \frac{\pi_\theta(u_t|x_t)}{\pi_{\theta_{\text{old}}}(u_t|x_t)} \\ \mathcal{L}^{\text{CLIP}}(\theta) &= -\mathbb{E}_t\left[\hat{A}_t \min\left(\text{pr}_t(\theta), \text{clip}(\text{pr}_t(\theta), 1-\epsilon, 1+\epsilon)\right)\right] \end{aligned} \tag{9}$$

where A_t is the generalized advantage estimate (GAE)

$$\begin{aligned} \delta_t &= r_t + \gamma V(x_{t+1})(1-d_t) - V(x_t) \\ \hat{A}_t &= \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}, \end{aligned} \tag{10}$$

where d_t is a flag indicating if the trajectory has been prematurely terminated (never in our case) and λ is the GAE parameter and finally the critic loss term is

$$\begin{aligned} \mathcal{L}^{\text{VF}} &= \frac{1}{2}\mathbb{E}_t\left[\max\left((V_\theta(x_t) - V_t^{\text{targ}})^2, (V_{\text{clip}} - V_t^{\text{targ}})^2\right)\right], \\ &\text{where} \quad V_t^{\text{targ}} = \hat{A}_t + V_{\theta_{\text{old}}}(s_t), \\ &\text{and} \quad V_{\text{clip}} = V_{\theta_{\text{old}}}(x_t) + \text{clip}(V_\theta(x_t) - V_{\theta_{\text{old}}}(x_t), -\epsilon, \epsilon), \end{aligned} \tag{11}$$

where ϵ is the clipping hyperparameter, V_θ is the value from the current network, and $V_{\theta_{\text{old}}}$ is the value from the network before the optimizer update. The reward for our aeroelastic system encourages stabilized states,

$$r_t(x_t) = \exp\left(-\left(\frac{h^2}{0.015^2} + \frac{dh^2}{0.2^2} + \frac{\alpha^2}{0.15^2} + \frac{d\alpha^2}{1.5^2}\right)\right) \tag{12}$$

At the start of each training epoch, four random initial conditions were generated. From these initial conditions, the ground truth model was used to simulate 128 time step trajectories, with 10 forward Euler steps of model time 10^{-3} taken between each. This dataset is split into training data samples as $\mathcal{D}_{RL}^e = \{(x_t, u_t, r_t, V_{old}(s_t)\}_{t=1}^{512}$, where e represents the training epoch index. This dataset is utilized to optimize the losses in Equations (9) and (11) by updating the policy and value function parameters with four iterations of batch gradient descent. After the networks' parameters are updated, a new set of initial conditions are sampled to create a new batch of data for the next epoch.

3.2. Model-constrained Predictive Forward Mapping

For the forward model, we use the neural ODE architecture [29] which has proven successful in many time series modeling applications for autonomous differential equations [2, 31]. In order to promote consistency with the ground truth model, the forward map loss function is augmented with a

model constrained term. This term necessitates a differentiable ODE integrator. For many models, this is straightforward to implement using standard automatic differentiation libraries. Let the forward model, predicting the next state, be

$$\hat{x}_{t+1} = \Psi_d(x_t, u_t; \theta_d). \tag{13}$$

Also, let the next time step from the ground truth model ODE be

$$\bar{x}_{t+1} = \mathcal{G}(\hat{x}_t). \tag{14}$$

Equation (14) indicates the result of prediction by the ground truth ODE model, given a previous surrogate predicted state as its input. Specifically, the output of the surrogate model is used as an initial condition for integration in the ODE model. The forward model loss compares the predicted $\hat{x}_t$ to generated x_t data. Additionally, it compares further sequential predictions to data from the differentiable ODE solver, i.e. it compares $\hat{x}_t$ and $\bar{x}_t$. This model constrained loss is defined as

$$\mathcal{L}_d(\theta_d) = \frac{1}{N_d N_a} \sum_{s=1}^{N_d} \sum_{t=1}^{N_a} \left[||\hat{x}_t^{(s)} - x_t^{(s)}|| + \gamma \mathcal{L}_{mc}(\hat{x}_t^{(s)}, \theta_d) \right], \tag{15}$$

where N_d denotes the number of sample trajectories, N_a indicates the number of time steps per trajectory, γ determines the strength of the model constraint term and

$$\mathcal{L}_{mc}(\hat{x}_t^{(s)}, \theta_d) = \frac{1}{N_m} \sum_{i=t+1}^{N_m} ||\hat{x}_i^{(s)} - \bar{x}_i^{(s)}||, \tag{16}$$

where N_m is the number of model constrained consistency steps used. The values of N_d, N_a, γ, and N_m used in training are listed in section 4. The parameters θ_d of the prediction network are trained with gradient descent to minimize $\mathcal{L}_d$.

3.3. Uncertainty Quantification

We take a Bayesian approach to UQ where the likelihood based on observed variables is used to update the posterior of the full state. The samples are i.i.d. normally distributed with standard deviation σ, i.e. the likelihood of a pitch observation o_i given the pitch trajectory candidate α_j is

$$p(o_i|\alpha_j) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(o_i - \alpha_j)^2}{2\sigma^2} \right), \tag{17}$$

The posterior for each trajectory candidate is given by Bayes rule

$$p(\alpha_j|\vec{o}) = \frac{p(\vec{o}|\alpha_j)p(\alpha_j)}{p(\vec{o})}. \tag{18}$$

The prior weights $p(\alpha_j)$ begin equally weighted while the MC samples are drawn from the prior distribution and updated with each observation as described in algorithm 1. When resampling occurs, the weights are reset to be equal.

3.4. Orchestration of Models

Algorithm 1 explains how each component of the digital twin is used in sequence. The algorithm specifically describes how predictions and uncertainty estimates are produced between observations of the ground truth model environment. After an observation is made, the weights are updated using Bayes rule. Next, the time stepping procedure begins. In each time step, the mean state estimate given by the Bayesian filter is used as the input to the control policy π_θ. The output of the policy is used as a control to update the state of every particle in the ensemble using the neural ODE surrogate Ψ_d. This procedure is repeated until a new measurement is obtained. Noise is then added to the states to prevent the collapse of the filter. Finally, a new α observation is made to update the filter at the start of the next loop. This autoregressive prediction produces the full time series for state and control with an uncertainty estimate given by the Bayesian filter between each observation.

Algorithm 1 Autoregressive Prediction

Input: Initial state x_0 and weights w, $t \leftarrow 0$
Output: Updated weights w', predictions x_i and controls β_i for complete time series
while $t < t_f$ **do**
  Update weights w to w' using Bayes rule and the sensor likelihood
  If ESS $< 0.3 \cdot$ #MC where ESS=$1/\sum w_i^2$, resample all states using empirical mean and variance
  while *awaiting new observation* **do**
    $u = \mathbb{E}[\pi_\theta(\mathbb{E}[x])]$
    $x_i \leftarrow x_i + \Psi_d(x_i, u)$ for each x_i in the ensemble
    $t \leftarrow t + \Delta t_{ml}$
    If new α observation ready: break while loop
  Record new α
  Inflate filter by adding noise to states to mitigate collapse in face of physical-digital discrepancy

4. NUMERICAL SIMULATION AND RESULTS

4.1. RL with ML surrogate and Particle Filtering

In training the control policy, the RL hyperparameters were $N_t = 128, \gamma = 0.99, \lambda = 0.95, \epsilon = 0.2, c_1 = 0.5, c_2 = 0.01$. The controller network featured 3 deep layers of width 128. The dataset for the DT model consisted of 2000 ground truth trajectories, with 5% withheld from training for testing, and 200 used during each step of gradient descent with learning rate 2.5×10^{-4}. Each trajectory was stored with time steps of $10^{-2}s$ to be consistent with the ML model time step but generated using $10^{-3}s$ forward Euler time steps. The DT RHS was pretrained with state-derivative pairs only, then the mcTangent loss term was introduced. In training, sequential trajectories were incrementally lengthened initially using N_a of 2, then 4, 8, 16, and finally 32. This network featured 3 deep layers of width 512 and was trained using Adam [32] with learning rate 1×10^{-5}. We also selected $N_m = 1$ for

the model-constrained loss term and $N_d = 200$ for as the number of samples per minibatch. The RL controller training took under 30 seconds then the mcTangent training time took approximately 3 hours on an NVIDIA A100. Both were executed using JIT compiled code via JAX [33].

The filtering routine, also JIT compiled via JAX and whose primary cost is propagating the DT model forwards, took 1.3 seconds to simulate a 5 second trajectory with $10,000$ Monte Carlo samples. A filter update was performed at every observation and a prediction was made until the next observation, with observations normally distributed with a standard deviation of 10^{-3} radians occurring at 10Hz. The filtering began with a biased prior whose mean was $10^{-3}m$ above the true h and $10^{-3}rad$ above the true α while being 0 for $\dot{h}$ and $\dot{\alpha}$. The prior standard deviations for $h, \dot{h}, \alpha, \dot{\alpha}$ were $2 \times 10^{-3}, 2 \times 10^{-2}, 2 \times 10^{-2}, 2 \times 10^{-1}$, respectively. After each update, the noise was added to the sample states with $1/10$ the standard deviation of the original priors. This was done to artificially prevent the filter from collapsing earlier than desired due to the discrepancy between the physical and digital twin dynamics. For example, the filter may become tightly concentrated around a state that uniquely explains the observations through the digital twin model but is still incorrect and/or diverges in the future due to model error.

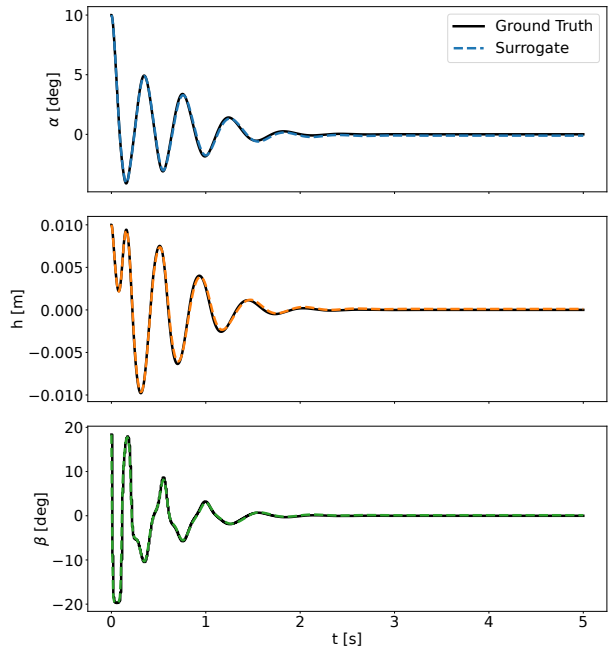

Figure 1. mcTangent surrogate (digital twin) using $dt = 10^{-2}s$ vs GT model (physical twin) using $dt = 10^{-3}s$ for a single, known initial condition; both using forward Euler and using the RL control function every $10^{-2}s$

The DT strategy described in section 3 was implemented and trained for the aeroelastic physics model of interest. Figure 1 depicts a five second prediction by the digital twin. For this case, the initial condition is fully known, but does not come from the training dataset. From the plotted trajectory, two things are observable: the control policy successfully guides the state variables to 0, reducing flutter, and the ML surrogate has high prediction accuracy, as seen by the similarity between dashed (ML) and solid (GT) lines. In fig. 2, the particle filtering strategy is applied to obtain an uncertainty estimate. For this case, the digital twin initial condition begins offset from the true state. As new α observations are taken, the filter is updated. The lightly shaded region around the h trajectory indicates the two standard deviation interval around the mean digital twin forward prediction. This quantifies the uncertainty, but the distribution is not assumed to be Gaussian so it cannot be interpreted as a confidence bound. It should be noted that there is now some oscillation in the GT solution in fig. 2 in contrast to the near perfect stabilization in fig. 1 (beginning just before t=2s). This oscillation stems from the fact that h is unobserved. Small inaccuracies in the recovery of h lead to control responses that are suboptimal for the true state, which prevents the wing from perfectly stabilizing. However, this oscillation is on the scale of millimeters, which is negligible for this airfoil application. Figure 3 presents filtering results for the plunging velocity, $\dot{h}$. Similar to the case of plunge, beginning close to the $t = 2s$ mark, there are slight oscillations around 0, due to suboptimal control inputs. However, the airfoil is very stable relative to an uncontrolled case. Again, it can be observed that the predicted trajectory remains reasonably represented by the hypothesis distribution given by the particle filter, indicating that the uncertainty estimate is reasonable. Finally, in fig. 4, the DT prediction for the rotational velocity $\dot{\alpha}$ of the pitching angle is shown. For $\dot{\alpha}$, the oscillations seen in the cases of h and $\dot{h}$ still appear. However, the two sigma region is much thinner, indicating that most particles followed very similar trajectories between observations. For the example trajectory, the uncertainty estimate provided by the filter is somewhat overconfident for $\dot{\alpha}$ compared to the other state variables. The ground truth trajectory for $\dot{\alpha}$ spends more time outside of the two standard deviation uncertainty region.

4.2. MPC with EnKF and baseline model

To compare the performance of our described approach to a more traditional one, we also consider the ensemble Kalman filter and model predective control. The MPC was implemented in CasADi [34], using the same stabilization objective as the RL. The MPC uses the mean estimate of the EnKF. We do not quantitatively compare computational cost since the methods were executed on different systems, the ML based on on a GPU, and the MPC EnKF on a CPU. The setup was identical to the particle filter, including ensemble size, initial uncertainty etc. but the EnKF filter noise was manually tuned to give good performance, using values of $1 \times 10^{6}, 1 \times 10^{-4}, 1 \times 10^{-6}, 1 \times 10^{-4}$ for $h, \dot{h}, \alpha, \dot{\alpha}$, respec-

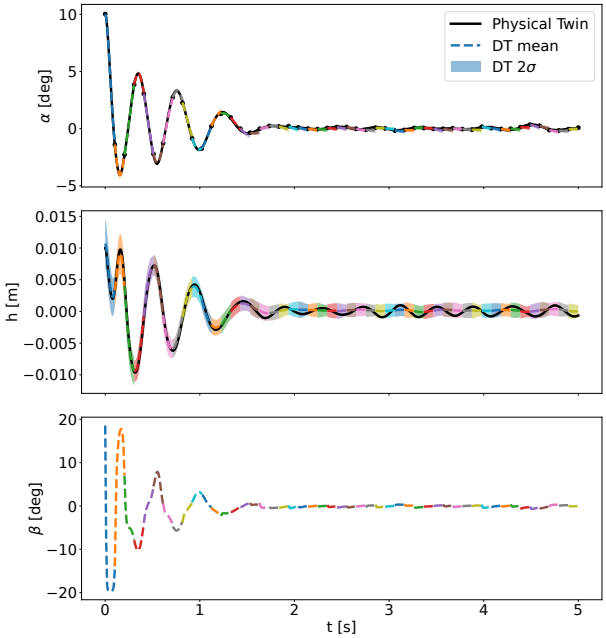

Figure 2. Particle filter using the digital twin as the hypothesis ensemble and providing control to the physical twin by evaluating the RL control function with the mean estimate. The shaded regions indicate the two standard deviation uncertainty region provided by the Bayesian filter.

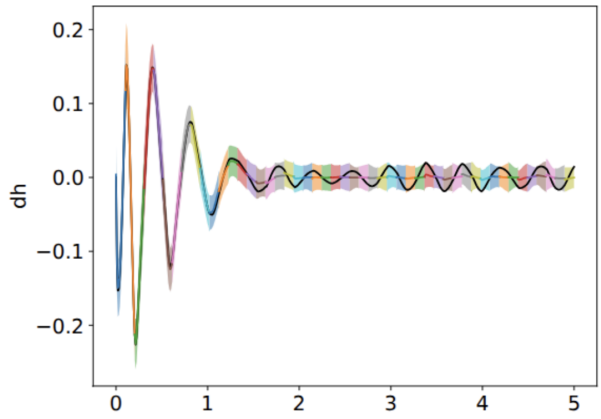

Figure 3. Prediction of plunging velocity by particle filter using the digital twin as the hypothesis ensemble and providing control to the physical twin by evaluating the RL control function with the mean estimate. The shaded regions indicate the two standard deviation uncertainty region provided by the Bayesian filter.

tively. The results are shown in fig. 5, to be compared to fig. 2. Even with the EnKF using the baseline model rather than the surrogate, the estimation for plunge is initially much worse, allowing the displacement to exceed the magnitude of the initial value. Both filtering methods require similar parameters and scale similarly due to use of an ensemble.

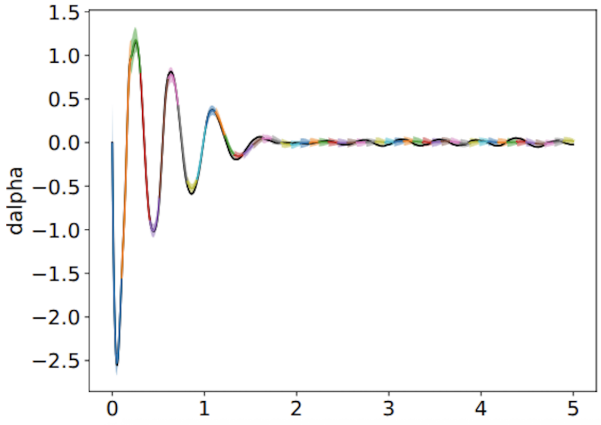

Figure 4. Prediction of angle of attack rotational velocity by particle filter using the digital twin as the hypothesis ensemble and providing control to the physical twin by evaluating the RL control function with the mean estimate. The shaded regions indicate the two standard deviation uncertainty region provided by the Bayesian filter.

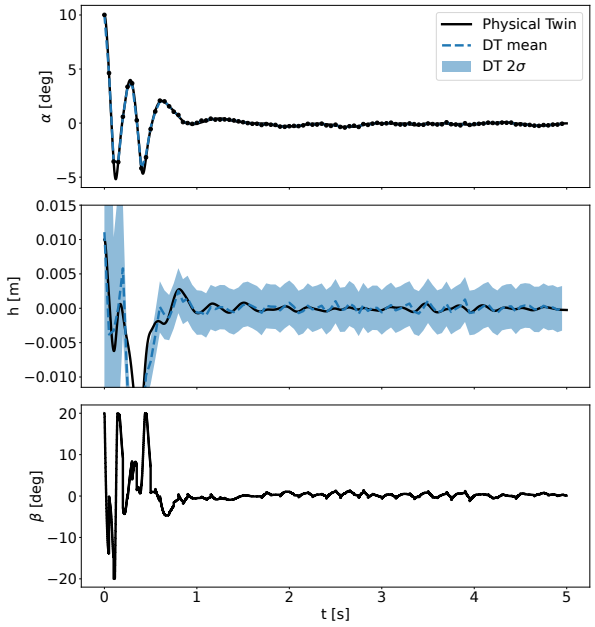

Figure 5. MPC with EnKF and baseline model

5. CONCLUSION

In this paper, we have presented a framework for a digital twin of an aeroelastic system which includes state estimation via Bayesian filtering, control calculation by a trained RL policy, and forward surrogate modeling with a neural ODE approach. The framework was able to provide stabilizing control to the model aeroelastic system, reducing flutter and preventing oscillations of dangerous magnitude. Additionally, the Bayesian approach is successful in state estimation from partial observations and provides valuable uncertainty quan-

tification. The filtering-prediction-control system was around 4x faster to perform on an $A100$ than the elapsed model time for our synthetic example, demonstrating its applicability to real-time applications. The requirements of the given framework are a ground truth model used to generate training data and train a policy network and an ability to collect observations during real-time operation. These basic requirements suggest that the method is more widely applicable to other aerospace models. We emphasize that the components of the framework are not specific to the aeroelastic ODE and therefore may be transferable to other systems. Further extensions of this work would include inference and uncertainty estimation for unknown environmental variables that impact flight dynamics and structural health such as wind speed and direction. In future work, stochastic optimal control methods may be used to improve control efficacy under an uncertain state hypothesis. Additionally, consideration of long-term (past the next observation) predictions and how future filter updates introduce uncertainty is also an important area of investigation. Specifically, quantifying long-horizon predictive uncertainty is relevant to applications that require more elaborate planning than airfoil stabilization.

REFERENCES

[1] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[2] Hai V. Nguyen and Tan Bui-Thanh. A model-constrained tangent slope learning approach for dynamical systems. *International Journal of Computational Fluid Dynamics*, 36(7):655–685, August 2022.

[3] National Academy of Engineering and National Academies of Sciences, Engineering, and Medicine. *Foundational Research Gaps and Future Directions for Digital Twins*. The National Academies Press, Washington, DC, 2024.

[4] Adam Thelen, Xiaoge Zhang, Olga Fink, Yan Lu, Sayan Ghosh, Byeng D. Youn, Michael D. Todd, Sankaran Mahadevan, Chao Hu, and Zhen Hu. A comprehensive review of digital twin – part 1: Modeling and twinning enabling technologies, 2022.

[5] Prasad Talasila, Dmitri Tcherniak, Anders M.D. Jensen, Swarup Mahato, Jakob V. Medom, Martin D. Ulriksen, Giuseppe Abbiati, A. Schörghofer-Queiroz, Peter G. Larsen, and Lars Damkilde. A digital twin platform for structural health monitoring. *Computer-Aided Civil and Infrastructure Engineering*, 50:100086, 2026.

[6] Marcel Menner, Ankush Chakrabarty, Karl Berntorp, and Stefano Di Cairano. Learning optimization-based control policies directly from digital twin simulations. In *2022 IEEE Conference on Control Technology and Applications (CCTA)*, pages 895–900, 2022.

[7] T.G. Ritto and F.A. Rochinha. Digital twin, physics-based model, and machine learning applied to damage detection in structures. *Mechanical Systems and Signal Processing*, 155:107614, 2021.

[8] Raisa Bentay Hossain, Farid Ahmed, Kazuma Kobayashi, Seid Koric, Diab Abueidda, and Syed Bahauddin Alam. Virtual sensing-enabled digital twin framework for real-time monitoring of nuclear systems leveraging deep neural operators, 2024.

[9] Joelle W.Y. Chia, Wim J.C. Verhagen, Jose M. Silva, and Ivan S. Cole. A review and outlook of airframe digital twins for structural prognostics and health management in the aviation industry. *Journal of Manufacturing Systems*, 77:398–417, 2024.

[10] Eric J. Tuegel, Anthony R. Ingraffea, Thomas G. Eason, and S. Michael Spottswood. Reengineering aircraft structural life prediction using a digital twin. *International Journal of Aerospace Engineering*, 2011(1):154798, 2011.

[11] Xiaonan Lai, Liangliang Yang, Xiwang He, Yong Pang, Xueguan Song, and Wei Sun. Digital twin-based

structural health monitoring by combining measurement and computational data: An aircraft wing example. *Journal of Manufacturing Systems*, 69:76–90, 2023.

[12] Jiaqi Xu, Dingqiang Dai, Xuan Zhou, Marco Giglio, Claudio Sbarufatti, and Leiting Dong. Structural damage diagnosis and prognosis with fleet digital twin considering similarity of individual structural features. *Aerospace Science and Technology*, 168:110983, 2026.

[13] M.G. Kapteyn, D.J. Knezevic, D.B.P. Huynh, M. Tran, and K.E. Willcox. Data-driven physics-based digital twins via a library of component-based reduced-order models. *International Journal for Numerical Methods in Engineering*, 123(13):2986–3003, 2022.

[14] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT Press, Cambridge, Mass., 2005.

[15] Yuting Bai, Bin Yan, Chenguang Zhou, Tingli Su, and Xuebo Jin. State of art on state estimation: Kalman filter driven by machine learning. *Annual Reviews in Control*, 56:100909, 2023.

[16] Zheng-Meng Zhai, Mohammadamin Moradi, Bryan Glaz, Mulugeta Haile, and Ying-Cheng Lai. Machine-learning parameter tracking with partial state observation. *Phys. Rev. Res.*, 6:013196, Feb 2024.

[17] Negar Asadi and Seyede Fatemeh Ghoreishi. Bayesian state estimation in partially-observed dynamic multidisciplinary systems. *Frontiers in Aerospace Engineering*, Volume 1 - 2022, 2022.

[18] Lorenzo Schena, Pedro A. Marques, Romain Poletti, Samuel Ahizi, Jan Van den Berghe, and Miguel A. Mendez. Reinforcement twinning: From digital twins to model-based reinforcement learning. *Journal of Computational Science*, 82:102421, 2024.

[19] Ling-Wei Kong, Yang Weng, Bryan Glaz, Mulugeta Haile, and Ying-Cheng Lai. Digital twins of nonlinear dynamical systems, 2022.

[20] T. Hielscher, S. Khalil, N. Virgona, and S.A. Hadigheh. A neural network based digital twin model for the structural health monitoring of reinforced concrete bridges. *Structures*, 57:105248, 2023.

[21] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.

[22] Ian Grooms. A comparison of nonlinear extensions to the ensemble kalman filter: Gaussian anamorphosis and two-step ensemble filters. *Computational Geosciences*, 26(3):633–650, March 2022.

[23] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[24] Andrew McClellan, Joseph Lorenzetti, Marco Pavone, and Charbel Farhat. A physics-based digital twin for model predictive control of autonomous unmanned aerial vehicle landing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 380(2229):20210204, 06 2022.

[25] Yi-Ping Chen, Vispi Karkaria, Ying-Kuan Tsai, Faith Rolark, Daniel Quispe, Robert X. Gao, Jian Cao, and Wei Chen. Real-time decision-making for digital twin in additive manufacturing with model predictive control using time-series deep neural networks. *Journal of Manufacturing Systems*, 80:412–424, 2025.

[26] Matthew J. Tenny, James B. Rawlings, and Stephen J. Wright. Closed-loop behavior of nonlinear model predictive control. *AIChE Journal*, 50(9):2142–2154, 2004.

[27] Rolf Findeisen and Frank Allgöwer. Computational delay in nonlinear model predictive control. *IFAC Proceedings Volumes*, 37(1):427–432, 2004. 7th International Symposium on Advanced Control of Chemical Processes (ADCHEM 2003), Hong-Kong, 11-14 January 2004.

[28] Keum W. Lee and Sahjendra N. Singh. Global Robust Control of an Aeroelastic System Using Output Feedback. *Journal of Guidance, Control, and Dynamics*, 30(1):271–275, 2007.

[29] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2019.

[30] Jeonghwan Ko, Thomas W. Strganac, and Andrew J. Kurdila. Stability and control of a structurally nonlinear aeroelastic system. *Journal of Guidance, Control, and Dynamics*, 21(5):718–725, 1998.

[31] Carlos J. G. Rojas, Andreas Dengel, and Mateus Dias Ribeiro. Reduced-order model for fluid flows via neural ordinary differential equations, 2021.

[32] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[33] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[34] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.