Data-driven physics-based digital twins via a library of component-based reduced-order models

M.G. Kapteyn
Massachusetts Institute of Technology, Cambridge, MA 02139

D.J. Knezevic
Akselos Inc., Brookline, MA 02446

D.B.P. Huynh and M. Tran
Akselos S.A., 1015 Lausanne, Switzerland

K.E. Willcox
University of Texas at Austin, Austin, TX 78712

This work proposes an approach that combines a library of component-based reduced-order models with Bayesian state estimation in order to create data-driven physics-based digital twins. Reduced-order modeling produces physics-based computational models that are reliable enough for predictive digital twins, while still being fast to evaluate. In contrast with traditional monolithic techniques for model reduction, the component-based approach scales efficiently to large complex systems, and provides a flexible and expressive framework for rapid model adaptation—both critical features in the digital twin context. Data-driven model adaptation and uncertainty quantification is formulated as a Bayesian state estimation problem, in which sensor data is used to infer which models in the model library are the best candidates for the digital twin. This approach is demonstrated through the development of a digital twin for a 12ft wingspan unmanned aerial vehicle. Offline, we construct a library of pristine and damaged aircraft components. Online, we use structural sensor data to rapidly adapt a physics-based digital twin of the aircraft structure. The data-driven digital twin enables the aircraft to dynamically replan a safe mission in response to structural damage or degradation.

I. Introduction

Computational models are used throughout engineering, but insights depend on the model being an accurate reflection of the underlying physical system. Differences in material properties, manufacturing processes, and operational histories are just some of the many factors that ensure that no two engineering systems are identical, even if they share the same design parameters. Using a single static model to approximate many similar assets ignores these differences, fundamentally limiting their ability to model any particular asset. The *digital twin* paradigm aims to overcome this limitation by providing an adaptive, comprehensive, and authoritative digital model tailored to each unique physical asset. The digital twin paradigm has garnered attention in a range of engineering applications, such as structural health monitoring and aircraft sustainment procedures [1, 2], simulation-based vehicle certification [1, 3], and fleet management [1, 4, 5]. Although the promise of digital twins is recognized, it is also acknowledged that a shift toward digital twins presents a number of complex challenges. Creating, calibrating, and maintaining the models required for even a single digital twin already presents a considerable challenge [1, 6]. This difficulty is only multiplied by the fact that such a model must be maintained for each and every physical asset.

This work develops an approach for creating data-driven physics-based digital twins. At the heart of our approach is a library of physics-based reduced-order models of the system. We adopt a component-based model reduction approach that scales efficiently to large-scale assets, while the construction of a library of model components admits flexible and expressive model adaptation. By sharing a single model library across many assets, this approach also scales to settings in which a large number of digital twins are required. With this model library as a foundation, we create data-driven digital twins by formulating an estimation problem in which online sensor data from a physical asset is used to infer which models in the model library should comprise the digital twin.

In some application areas, a digital twin is thought of as being constructed from data alone. In this work we

argue that, particularly in engineering applications, it is critical that the digital twin also incorporate physics-based models of the asset. A digital twin that incorporates both physics and data is sometimes referred to as a *hybrid twin*[7]. In our work, the physics models underlying the digital twin are computational models based on discretized partial differential equations (PDEs). In contrast with purely data-driven models, physics-based models offer a greater degree of interpretability, reliability, and predictive capability. Such models are already ubiquitous in engineering, and are typically solved using approaches such as finite-element analysis (FEA). Accurately modeling a complete engineering system often requires extremely large computational models that require significant computational resources to evaluate. In many applications, digital twins are required to provide near real-time insights in order for them to be used effectively for operational decision making. This requires the ability to rapidly adapt the computational model in the face of changing model parameters, and rapidly evaluate the model to provide analysis and prediction. Traditional large-scale physics-based models are usually intractable to solve in this type of real-time, many-query context.

Model order reduction [8–11] provides a mathematical foundation for accelerating complex computational models so that they may be operationalized in the digital twin context. Reduced-order modeling involves investing computational time during an offline phase to develop reduced-models; these reduced models can then be rapidly evaluated during an online operational phase. However, many methods for building reduced-order models during the offline phase require many evaluations of the full-order model, which is intractable for large, system-level models. Furthermore, in order for the digital twin to be capable of representing a wide range of asset states and operating conditions, the underlying model needs an expressive parametrization, often involving many parameters, wide parameter ranges, and discontinuous solution dependencies. In this work, we address these challenges by adopting a parametric component-based model reduction methodology [12]. This method scales efficiently to large-scale assets, and admits flexible and expressive model adaptation via parametric modifications and component replacement. Using this method, we create and train a library of reduced-order models, each representing different states of the asset relevant to the purpose of the digital twin. This model library is designed *a priori* to include models that accurately reflect the current asset state and anticipated future states, as well as additional models designed to detect new unexpected or anomalous states. The scalability of the component-based reduced-order modeling strategy enables the model library to scale to large asset state-spaces, and to be further extended and enriched on-the-fly by adding additional component models over the lifetime of an asset.

We then address the challenge of automating the calibration and adaptation of digital twins to ensure that they accurately reflect an evolving physical asset. Motivated by the proliferation of low-cost sensors and increasing connectivity between physical assets, we adopt a data-driven approach in which sensor data gathered by an asset guides adaptation of the digital twin. We show how the pre-specified model library allows us to frame the digital twin adaptation problem as a tractable, yet mathematically rigorous Bayesian state-estimation problem[13, 14]. This state-estimation procedure combines observed sensor data with a state transition probability model (typically derived from historical data or computed via physics-based simulations), to yield a probabilistic classification of the current state of an asset into the model library. The use of reduced-order models in the model library greatly accelerates the Bayesian inference problem[15], making it tractable to solve in near-real time. The resulting probabilistic classification provides a basis for updating the digital twin and quantifying the associated uncertainty in the model. The updated digital twin model can then be used for accurate, up-to-date analysis and prediction.

We demonstrate our methodology and illustrate the benefits of our contributions by means of a case study. We create a digital twin of a fixed-wing unmanned aerial vehicle (UAV). The goal of this digital twin is to enable the UAV to become self-aware, in the sense that it is able to dynamically detect and adapt to changes in its structural health due to either damage or degradation [16–18]. We demonstrate how a component-based reduced-order structural model of the aircraft scales efficiently to the full UAV system, and how we construct a component-based model library designed to enable the rapid adaptation of the digital twin to a wide range of effective damage states. We then demonstrate how the model library enables data-driven model adaptation, and how near real-time estimates of the effect of incurred damage enables dynamic decision-making. We present simulation results for an illustrative UAV mission, in which the UAV estimates its structural state online and uses this estimate to decide whether to perform faster, more aggressive maneuvers, or fall back to more conservative maneuvers to minimize further damage.

The remainder of this paper is organized as follows. Section II presents an overview of the component-based reduced-order modeling methodology we adopt. It then describes how we use this methodology to construct a model library, and the benefits this provides. Section III formulates the problem of data-driven model updating using a component-based model library. Section IV presents the self-aware UAV case study which serves to demonstrate our approach. Finally, Section V concludes the paper.

II. Component-Based Model Libraries

This section describes our methodology for constructing a component-based library of reduced-order models, from which a digital twin can be instantiated. Section II.A describes the component-based reduced-order modeling methodology we adopt. Next, Section II.B describes how we leverage this methodology in order to construct a model library. Finally, Section II.C argues how our approach meets the needs of the digital twin context.

A. Component-based reduced-order models

The component-based reduced-order modeling approach we adopt is the Static-Condensation Reduced-Basis-Element (SCRBE) method, developed in [12, 19–21]. We present a relatively high-level overview of the method herein, and refer the reader to these prior works for a detailed treatment of the underlying theory and procedures for offline training.

Traditional single-domain model reduction techniques such as reduced-basis (RB) methods [22–27], work to reduce a full system-level finite element (FE) approximation space directly. The key limitation in these approaches is that the full system-level problem is typically very large for complex engineering systems for which we require digital twins. So much so, that even a single solution of the full FE system (which is required even for RB methods during offline training) is often intractable. Even if the full system-level model can be solved, the adaptivity and expressivity of a digital twin typically requires a large number of parameters, each with large domains and potentially discontinuous effects on the solution (e.g. geometric parameters). Such parameter spaces are generally not amenable to single-domain model-reduction techniques [28].

The SCRBE method is a component-based model-order reduction strategy that aims to address these challenges. The core idea of SCRBE is to apply the substructuring approach to formulate a system in terms of components [29, 30], and then apply the Certified Reduced Basis (RB) Method within each component. This brings the advantages of the RB method (accuracy, speed, parameters), as well as enhanced scalability and flexibility due to the component-based formulation.

As with all reduced-order modeling methods, the SCRBE approach requires an offline training phase in order to build a dataset for each component, which then enables rapid online evaluation of system-level reduced models. Crucially, the need to solve the costly full system FE problem during the offline stage is circumvented by the "divide-and-conquer" nature of the component-based formulation: the system is decomposed into components and then the training procedure is performed using only individual components and small groups of components.

It is well-known in the context of parametric ROMs in general, and the RB method in particular, that the Offline and Online computational cost of ROMs generally increases rapidly as the number of parameters increases — this is the so-called "curse of dimensionality." However, the SCRBE framework also circumvents this issue because each component in a system typically only requires a few parameters each, since engineering systems are often characterized by many spatially distributed parameters. This means that we can set up large systems assembled from many parametric components in which each component only has a few parameters but the overall system may have many (e.g. thousands) of parameters, all without being affected by the "curse of dimensionality." These features combine to enable the SCRBE approach to scale efficiently to complex, evolving engineering systems—precisely the systems for which digital twins are arguably most beneficial.

Each component in an SCRBE model is defined by a set of parameters, μ_i^c, where i denotes the component index. These parameters can be geometric parameters that affect the spatial domain of the component, or non-geometric parameters such as material properties. A component with a specified set of parameters and associated parameter ranges is referred to as an *archetype component*. Specifying values for these parameters is referred to as *instantiating* the archetype component. The component is based on a computational mesh, including component interface surfaces called *ports* on which a component may be connected to a neighbor component via a common port mesh. Figure 1 shows an example of a typical component from the UAV application considered in this work: a spanwise section of a three dimensional aircraft wing. A full system model is constructed by instantiating a set of components and connecting them at compatible ports. The parameters for the system-level assembly, which we denote by μ, is then simply the union of the component-level parameters, i.e. $\mu = \bigcup \mu_i^c$.

We associate with each component a governing PDE and external boundary conditions where needed. In this work we consider the governing equations of linear elasticity, which provide a physics-based model of an asset's structural response to an applied load. The parametrized weak form at the system level can be written

$$a(u, v; \mu) = f(v; \mu) \quad \forall\, v \in X(\mu), \tag{1}$$

where $u(\mu)$ is the asset's structural displacement field. Details of the bilinear and linear forms $a(u, v; \mu)$ and $f(v; \mu)$ respectively, as well as the function space $X(\mu)$, can be found in [28]. The SCRBE model-reduction approach builds

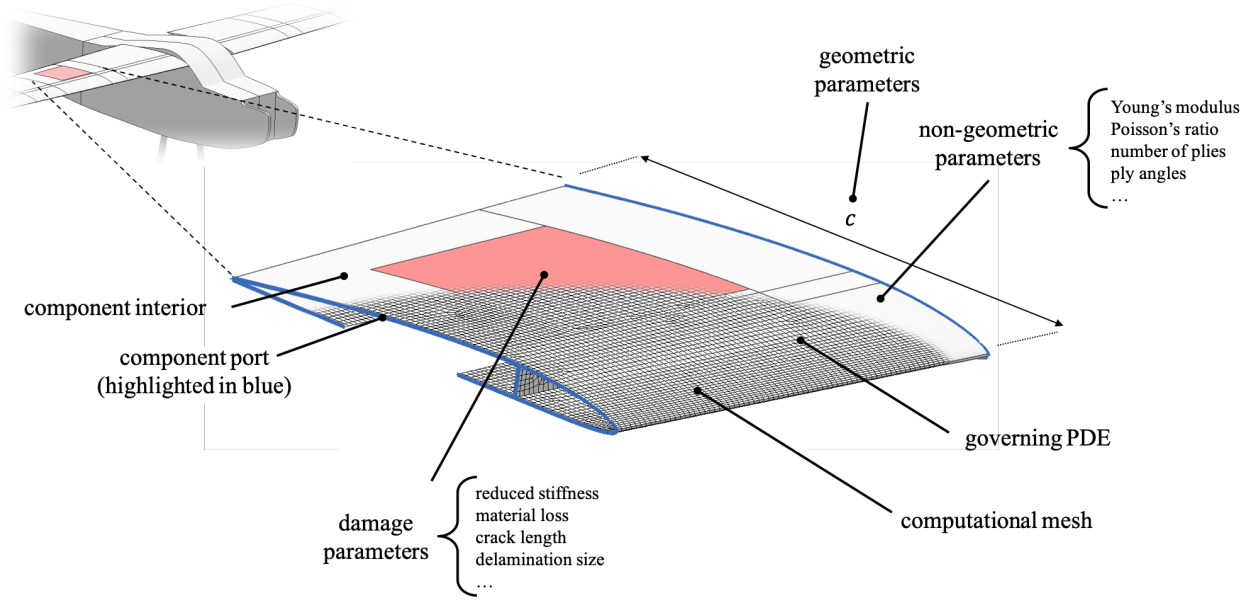


Fig. 1 An example component: A spanwise section of a three dimensional wing. Labels indicate the information required to completely define this component.

upon an underlying FE approximation of the system. In particular, the FE approximation can be stated as seeking a solution $u^h(\mu) \in X^h(\mu)$ such that

$$a(u^h, v; \mu) = f(v; \mu) \quad \forall\, v \in X^h(\mu). \tag{2}$$

where $X^h(\mu) \subset X(\mu)$ is the system-level FE approximation space, corresponding to a system-level mesh created by connecting the meshes of all components in the system. Let $N_{\mathrm{FE}} = \dim(X^h(\mu))$ denote the number of degrees of freedom in the system level FE approximation.

The discretized weak form (2) corresponds to a matrix system

$$K(\mu)U(\mu) = F(\mu) \tag{3}$$

where $K(\mu) \in \mathbb{R}^{N_{\mathrm{FE}} \times N_{\mathrm{FE}}}$ is the (symmetric) stiffness matrix, $F(\mu) \in \mathbb{R}^{N_{\mathrm{FE}}}$ is the load vector, and we seek the displacement solution $U(\mu) \in \mathbb{R}^{N_{\mathrm{FE}}}$. In the context of digital twins of large-scale and/or complex systems, the system (3) may be very large (e.g., of order 10^7 or 10^8 degrees of freedom are typical) and can therefore can be computationally intensive to solve. Therefore, we pursue an alternative approach, which—as indicated above—is to utilize substructuring to bring in a component-based formulation on which we may then apply RB.

We illustrate this idea for the simple case of a two-component system with a single port, with the understanding that the same ideas carry over unchanged to systems with any number of components and ports. We let the subscript p (resp. 1 or 2) denote the degrees of freedom associated with the port (resp. component 1 or 2), and we let N_p denote the number of degrees of freedom on the port. Then we can reformulate (3) as

$$\begin{bmatrix} K_{p,p} & K_{p,1} & K_{p,2} \\ K_{p,1}^T & K_{1,1} & 0 \\ K_{p,2}^T & 0 & K_{2,2} \end{bmatrix} \begin{bmatrix} U_p \\ U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} F_p \\ F_1 \\ F_2 \end{bmatrix}. \tag{4}$$

Note that all quantities in (4) depend on μ, but we omit the μ-dependence from our notation for the sake of simplicity. The matrix structure here suggests a convenient way to proceed: we may solve for U_1 and U_2 in terms of U_p as follows:

$$U_i = K_{i,i}^{-1}(F_i - K_{p,i}^T U_p), \quad i = 1, 2. \tag{5}$$

Substitution of (5) into (4) then yields a system with only U_p as unknown:

$$K_{p,p}U_p + \sum_{i=1}^{2} K_{p,i}K_{i,i}^{-1}(F_i - K_{p,i}^T U_p) = F_p, \tag{6}$$

or equivalently

$$\left(K_{p,p} - \sum_{i=1}^{2} K_{p,i}K_{i,i}^{-1}K_{p,i}^T\right)U_p = F_p - \sum_{i=1}^{2} K_{p,i}K_{i,i}^{-1}F_i. \tag{7}$$

We introduce the notation:

$$\mathbb{K} = \left(K_{p,p} - \sum_{i=1}^{2} K_{p,i}K_{i,i}^{-1}K_{p,i}^T\right), \quad \mathbb{F} = F_p - \sum_{i=1}^{2} K_{p,i}K_{i,i}^{-1}F_i, \tag{8}$$

for the substructured stiffness matrix and load vector, where $\mathbb{K} \in \mathbb{R}^{N_p \times N_p}$, and $\mathbb{F} \in \mathbb{R}^{N_p}$. Hence we have:

$$\mathbb{K}U_p = \mathbb{F}. \tag{9}$$

Here (9) is an exact reformulation of (3), where the key point is that by performing a sequence of component-local solves as in (5) the system is reduced to size $N_p \times N_p$ instead of the original size $N_{\text{FE}} \times N_{\text{FE}}$.

However, there remain two computational difficulties associated with this classical static condensation approach, and the SCRBE method proposes model reduction strategies that address each of these issues in turn. Firstly, to evaluate the matrix inverse in (5) in a practical way, we must perform a sequence (one per port degree of freedom) of component-local FE solves. Thus the formation of the matrix $\mathbb{K}$ will typically be costly, and this must be repeated in component i each time μ_i^c is modified. This is addressed by replacing the FE space within each component with a reduced-basis approximation space of a much smaller dimension, which drastically speeds up the solves required to form $\mathbb{K}$, and also allows parametric changes on component interiors to be incorporated efficiently. As discussed above, the training procedure for this interior reduction can be performed on each component independently.

Secondly, $\mathbb{K}$ is typically much denser than K because each component contributes a dense block to $\mathbb{K}$ based on the number of port degrees of freedom associated with the component. This increased density can, in many or even most cases, undermine any computational advantage provided by substructuring compared to a full order solve. This is a well-known issue with substructuring, and the usual advice to address this is to make sure that ports contain as few nodes as possible (e.g. by locating ports in regions that are small, or coarsely meshed) to limit the size of the dense blocks. In practice these requirements impose very severe limitations on the application of substructuring, and in many cases (depending on the model geometry or mesh density) it is not possible for the requirements to be satisfied. This issue is addressed in the SCRBE framework by applying model reduction to the ports, which is referred to as *port reduction*. With port reduction the goal is to construct a reduced set of $N_{pr} (\ll N_p)$ degrees of freedom on each port, while retaining accuracy compared to the full order solve by ensuring that the dominant information transfer between adjacent components is captured efficiently. The reduction from N_p to N_{pr} typically greatly reduces the overall size of $\mathbb{K}$, and also reduces the size of its dense blocks and hence significantly increases sparsity. This results in a significant computational speedup compared to both the non-port-reduced version of (9), and the full order system (3). Port reduction requires offline training to determine the dominant modes for each port, and here we follow port reduction approaches from the literature, i.e., pairwise training [20, 31], which involves performing proper orthogonal decomposition (POD) of port data obtained from pairs of components, or "optimal modes," which solves a transfer eigenproblem to obtain an optimal set of port modes [32, 33]. These port reduction schemes operate based on small submodels of an overall system, so—as discussed above—this does not require us to perform a full order system-level solve during the offline stage.

In the preceeding formulation the governing equations were linear. Extension to non-linear analysis is also possible within the SCRBE framework using the hybrid SCRBE/FE solution scheme presented in [28]. This framework combines SCRBE in linear regions and FE in nonlinear regions within a fully-coupled global solve. This allows one to handle the full range of nonlinear analysis via the generality of FE, while still benefitting from the SCRBE reduced order modeling approach in linear regions. The SCRBE/FE approach does not accelerate the FE region, but in the case that most of the model is linear (which is often the case when analyzing localized damage scenarios within a large system, for example) then the SCRBE/FE approach still provides a significant speedup compared to a global FE solve.

B. Constructing a model library

We construct a library of component-based reduced-order models, which are trained during an offline phase. This model library can then be used during an online phase to rapidly create, adapt, and evaluate reduced-order models.

Mathematically, we define a *component library* to be a set of archetype components, $\mathcal{C}$. Recall that an archetype component has free parameters, μ_i^c, with specified parameter ranges. Training for each archetype component $C_j \in \mathcal{C}, j = 1, \ldots, |\mathcal{C}|$ is performed during the offline phase. In the online phase, a system model can be constructed by selecting a subset of the component library, instantiating the components by specifying the parameter vector μ, and connecting the components at compatible ports. We define a *model library*, $\mathcal{M}$, to be a finite set of unique models (each corresponding to a unique value of μ), and denote each model in the library by M_j, for $j = 1, \ldots, |\mathcal{M}|$. With this model library in hand, any of the $|\mathcal{M}|$ reduced models can be rapidly evaluated. Figure 2 illustrates the relationship between the component library and model library, using the UAV application considered in this work as an example.

We note that the model library, $\mathcal{M}$ can be enriched in two ways. The first is to add new models by sampling additional values of the SCRBE parameters μ. This enrichment requires no additional offline training, since it utilizes the archetype components that are already trained in the component library, $\mathcal{C}$. The second approach is to first add new archetype components to the component library, thereby expanding the space of possible parameter vectors, μ, and then add new models that utilize the new components to the model library. This type of enrichment is more flexible and expressive, but does require additional offline training for the new archetype components.

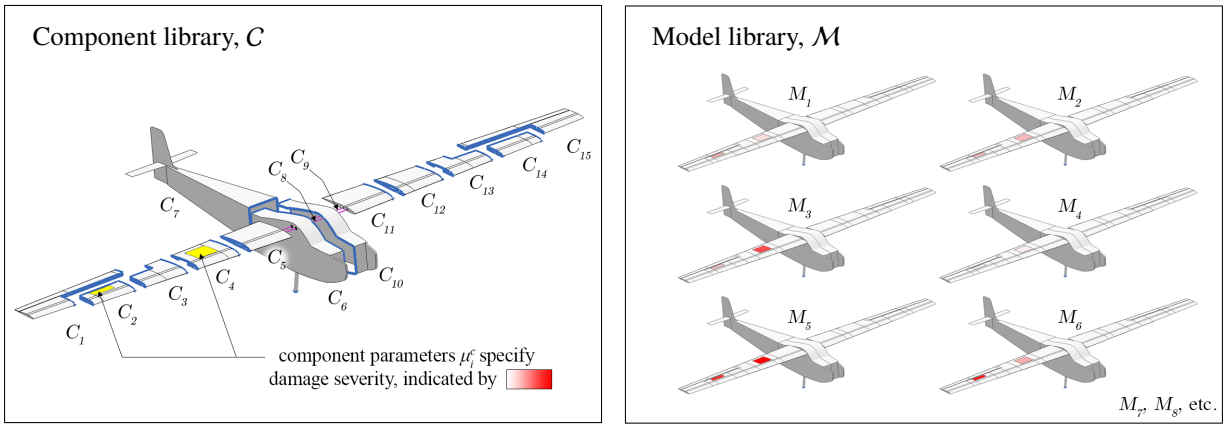


Fig. 2 Component and model libraries for a damaged UAV asset. In this example two of the components have a free parameter to specify the severity of damage in the highlighted damage regions. Each model in the model library has a different setting of these parameters, and thus represents a different UAV damage state.

C. Model libraries as an enabler of digital twins

A library of component-based reduced-order models provides a rich set of physics-based models from which we can derive digital twins. In particular, we show in Sec. III how the digital twin of an asset at a given point in time can represented as a stochastic mixture of multiple models in the library. Utilizing a model library as the foundation for physics-based digital twins has a number of key benefits over constructing a single monolithic digital twin.

Firstly, in addition to being able to accommodate a large number of spatially distributed parameters, component-based models can also accommodate more complex parameters than traditional single-domain techniques. For example, complex geometric parameters can be introduced by including multiple versions of a component in the component library, each with a different geometry (but with identical ports). Topological parameters can be introduced by connecting library components in different topologies, and including each topology in the model library. This paradigm of component instantiation and replacement provides an expressive, efficient, and intuitive way to perform the complex model adaptation required for a digital twin. In particular, in many scenarios one is able to specify a set of potential future states for an asset and monitor these closely. For example, we might know that an asset is at risk for a specific fault or that a specific component is prone to degradation. This knowledge could arise from expert opinion, or from prior experience and observations from other similar assets. In our component-based modeling approach, these known possible future states can be modeled in components to a high degree of accuracy and realism, for example using

specialized damage models and tailored constitutive equations, combined with non-linear regions as described in Section II.A.

Secondly, the model library facilitates life-long adaptation and development of digital twins. This is done by continuously enriching the model library, as described in Sec. II.B. Such library enrichments can be made with only incremental additions to the offline training data, so that the model library can be continuously updated. This ensures that the digital twin is capable of detecting and adapting to unexpected and previously unforeseen asset states. For example, a structural health monitoring application might encounter a damage state that is not represented in the component library. Once such a situation has been detected, one could trigger, for example, a manual inspection of the asset to accurately characterize the damage that has occurred. Once the damage is characterized, it can then be modeled and added to the set of known potential states for the asset, and specialized library components created to accurately model the damage and its anticipated progression.

Finally, the model library is even more powerful in settings where digital twins are required of many similar assets. We refer to a group of similar assets as an *asset class*, For example, a fleet of vehicles sharing a common design might constitute an asset class. In such settings creating, training, and maintaining hundreds or thousands of independent digital twin models could pose a significant practical challenge. Instead, our approach involves creating and maintaining a single model library, with a single collection of training data, which is shared across the entire asset class. This approach leverages the assumption that differences between assets will often be localized differences due to damage, material properties, manufacturing defects, or maintenance histories. In such cases assets within an asset class will be substantially similar, with any differences being limited to only a small number of components. In these settings, our library-based approach promotes model reuse and information transfer between assets. Due to the combinatorial nature of assembling models from components in the library, adding just a single new component can add a large number of possible new models. For example, if an asset is found to have developed a particular defect, a component would be added to the library to model this particular defect in the digital twin. This defect component then becomes available for use in the digital twins of all other assets in the class. This means that if any other asset were to develop a similar defect in the future, the defect could be detected (see Sec. III) and incorporated into the digital twin. This example illustrates model reuse: we have been able to re-use the defect component rather than having to incorporate the defect again from scratch, and information transfer between assets: a defect discovered in one asset is preemptively monitored as a potential defect in all other similar assets.

III. Data-Driven Digital Twins

This section outlines a data-driven approach for maintaining a digital twin of an evolving asset. We leverage sensor data in order to infer the model (or models) from a given model library that best match the physical asset. This approach provides a method for automating the calibration and adaptation of a digital twin during the online operational phase of an asset. In Section III.A, we show how the problem of model adaptation can be formulated as a state estimation problem, and how this state estimation problem can be solved using a sequential Bayesian data assimilation algorithm. Section III.B briefly discusses the computational cost of this approach. Finally, Section III.C shows how the result of this algorithm can be used to update a digital twin, as well as quantify the uncertainty in the digital twin and the inadequacy of the associated model library.

A. Formulating digital twin model updating as Bayesian state estimation

We define the state of a physical asset to be the parameters of its digital twin. This enables us to use state estimation in order to infer these parameters as they evolve, and thus, perform data-driven adaptation of the digital twin model. More concretely, we model the evolution of the physical asset as a hidden Markov model (HMM), where the hidden state that we estimate is the set of digital twin parameters that accurately describe the physical asset.

We define a period of asset monitoring by discrete time steps $t = 0, 1, \ldots, T$. We denote by d_t the digital twin of a given asset at time t. The set of models we have available to represent the physical asset is defined by the model library, $\mathcal{M}$ described in Sec. II. Thus we choose the state-space, i.e., the domain of d_t, to be the model library, $\mathcal{M}$. At each time step t, the control inputs to the physical asset, u_t, are known, and we observe measurements from the physical asset, o_t, such as sensor measurements or inspection data. In this work we focus on the problem of maintaining an up-to-date digital twin, and thus we focus on so-called *filtering* methods [34] in which we aim to estimate d_t at every time step t. We note that our formulation can be readily extended to instead estimate d_τ for some $\tau < t$ or $\tau > t$. These methods are called *smoothing* and *prediction* methods, and aim to estimate past or future states respectively. Such methods could also be of interest in the digital twin context. For example, smoothing could be used to better understand the state

history of an asset while prediction could be used for planning and optimization.

We adopt a Bayesian approach to state estimation in which we maintain a probabilistic estimate for the digital twin, which is updated as new data are acquired [35]. In particular, we treat the digital twin, d_t, as a discrete random variable, with probability mass function given by the posterior distribution

$$b_t(M_j) \coloneqq p(d_t = M_j | u_1, ..., u_t, o_1, ..., o_t), \quad \forall M_j \in \mathcal{M} \tag{10}$$

We refer to b_t as the *belief state* at time t. In our setting the belief state encodes the probability that the true state of the asset is represented by each entry in the model library, given all the evidence acquired until time t. This term can also be referred to as the posterior probability or posterior plausibility of model M_j[36, 37]. Hence, an equivalent way to view this estimation problem is as a probabilistic classification problem: we use observations from the physical asset to perform a probabilistic classification of the true asset state into the model library.

In sequential methods, the belief state is updated recursively at each time step. We begin the monitoring period at $t = 0$ with some initial belief state b_0. For $t > 0$, the belief, b_t can be written in terms of the belief at the previous time step, b_{t-1}, using Bayes' rule as follows:

$$b_t(M_j) \propto \left[\sum_{k=1}^{|\mathcal{M}|} p(d_t = M_j | d_{t-1} = M_k, u_t) b_{t-1}(M_k) \right] p(o_t | d_t = M_j, u_t). \tag{11}$$

Here u_t encapsulates any inputs to the system that affect the state or observation dynamics, for example, operating conditions or applied loads.

The first term on the right-hand side of (11) is a summation over state transition probabilities, multiplied by prior beliefs. The state transition probabilities describe the probability that the asset transitions from one state to another, in this case the transition from model M_k to model M_j, given that the the input to the system was u_t. In our setting, the state transition probability model is treated as a companion to the model library. This model could be derived based on expert opinion or dynamic simulations of the asset, or the transition probabilities can be learned from historical data (see for example [38] or [39]). Learning from historical data is particularly attractive in the digital twin context, since data could be collected from all assets that share the same model library. This approach provides a flexible means of obtaining transition models, and would ensure that the transition model is constantly being updated and improved as more data are obtained.

The second term on the right-hand side of (11) is the observation model describing the likelihood of observing o_t, given that the system was in some state M_j and the input was u_t. This term is sometimes referred to as the model evidence [36, 37]. A common modeling assumption is that observed quantities are some function of the state and input, corrupted by zero-mean Gaussian error term, i.e.,

$$o_t = Z(M_j, u_t) + v_t, \qquad v_t \sim N(0, \sigma^2), \tag{12}$$

where σ^2 is the covariance of the observation noise. The function $Z(M_j, u_t)$ simulates the response of a physical asset represented by model M_j, to an input u_t, and predicts the various observed quantities, o_t. In our setting, this simulation is performed via the rapid evaluation of the reduced-order model $M_j \in \mathcal{M}$. In this way, the model library makes state-estimation tractable by providing a means of rapidly predicting observations for any asset state represented in the model library.

B. Computational cost

It is well-known that the most complex numerical part of classical Bayesian inference is the propagation of uncertainties in order to evaluate the posterior density, as well as the sampling of this density. However, a key feature of the state-estimation problem in our context is that the state-space is discrete and finite, since we include only a finite number of distinct models in the model library. This feature allows us to solve (11) directly for each $M_j \in \mathcal{M}$. In practice, the complexity of this approach is dominated by the $|\mathcal{M}|$ evaluations of the observation model, which consequently requires the evaluation of each of the $|\mathcal{M}|$ different component-based reduced-order models in the model library. Whether or not this is tractable is application dependent, and depends on the size of the model library, the run-time of each reduced-order model, and the required update frequency of the digital twin, i.e., the discretization of time t in the state-estimation problem.

For many applications the speed-up achieved by using reduced-order models, as opposed to full-order models, will be sufficient to make this problem tractable. For other applications, for example those requiring high frequency

model updates, there are a number of ways this algorithm could be accelerated. Firstly, we could choose to perform inference over only a subset of the state-space at each iteration, so as to only consider $m\,(\ll |\mathcal{M}|)$ different models at each time-step. For example, if the models in the library represent different values of some monotonically increasing parameter (for example a damage parameter), we may choose to begin ignoring entries in the library once their parameter value has been passed with high confidence. Another approach is based on the assumption that although the total number of models in the model library might be large, it is often reasonable to expect that the bulk of the probability density will be concentrated on only a handful of different models. In this case, instead of computing the full belief state, we could instead take a greedy approach and perform an approximate belief state update by enumerating only the m most likely state trajectories (see for example [40] for such an approach) In each of these cases, the state estimation would require only m evaluations of library models at each time-step. The value of m could thus be guided by the computational requirements.

Finally, if the the space of possible control inputs u_t in the problem is small and discrete, it may be more efficient to compute the values of $Z(M_j, u_t)$ offline, for all $M_j \in \mathcal{M}$ and all possible values of u_t. These values could then be stored in a database, which could then be used as a look-up table during state-estimation. This circumvents the need to evaluate any reduced models during the online phase. Note, however, that the look-up table approach still benefits from reduced order modeling since the creation, and potential subsequent enrichment, of a look-up table is a many-query problem.

C. Uncertainty and error quantification

A digital twin is typically thought of as a single model (or set of coupled models) that represents a physical asset. In contrast, our data-driven approach instead treats the digital twin, d_t, as a random variable, and estimates the associated probability mass function, given by the belief state, b_t. This approach allows us to quantify the uncertainty in the digital twin. For example, if the belief state assigns relatively high probability to multiple models in the library, then there is significant uncertainty about which model is most likely. This could be due to uninformative observations, or due to the true asset state falling between discrete entries in the model library.

To demonstrate this, we suppose the digital twin is being used to estimate some output quantity of interest, $Q(M_j)$, for the physical asset. Having a distribution over possible models has the benefit that it allows us to propagate the uncertainty in the digital twin in order to compute a distribution over the quantity of interest. We can then use the expectation of this distribution as our estimate, i.e.,

$$\hat{Q}_t = \mathbb{E}_{b_t}[Q(M_j)] = \sum_{M_j \in \mathcal{M}} b_t(M_j) Q(M_j), \tag{13}$$

and we could similarly compute the variance or other statistics of the quantity of interest in order to quantify the uncertainty. Another way to see the benefit of maintaining a stochastic digital twin is that this approach accomodates estimating the quantity of interest using an affine combination of multiple models in the model library, rather than just a single model.

We define a measure of error in the digital twin that utilizes quantities we are able to observe from the physical asset, namely the observations o_t. We define

$$\epsilon_t(b_t, o_t; \mathcal{M}) = |\mathbb{E}_{b_t}[Z(M_j, u_t)] - o_t| \tag{14}$$

Here we explicitly call out the fact that this error depends not only on the belief state b_t, and observation o_t, but also on the model library $\mathcal{M}$ (we assume that the control input, u_t is known exactly, so that this does not contribute explicitly to the error). This highlights the fact that the digital twin may be subject to error due to inadequacies in the model library. If ϵ is small relative to the noise in the observations, it means that the observations are well explained by our digital twin, i.e., by the affine combination of models given by the belief b_t. One might then reasonably expect that quantities of interest are also well approximated by the digital twin. On the other hand, if this error is high, it means that either the belief b_t is not assigning probability to the correct model in the library, or that none of the models in the library are a good representation of the physical asset. Either of these options is cause for concern, and thus this error could be used as an indicator that the digital twin may be unreliable. This situation can be remedied by enriching the model library via the addition of new models. In some cases the belief state alone may provide sufficient guidance on where the library requires enrichment. In other cases, this indicator could be used to trigger a manual inspection of the asset, with the results of the inspection informing the development of new components and/or models which are subsequently added to the respective libraries.

IV. Case Study: Toward a Self-Aware UAV

This section presents a case study that serves to demonstrates the approaches described in the previous two sections. Section IV.A presents the UAV that is the subject of this case study. In Section IV.B we develop a component-based model library for the UAV, and demonstrate how this provides fast, accurate physics-based models by comparison with a full order FE model. Finally, in Section IV.C we demonstrate how this model library enables us to create a data-driven digital twin of the UAV, and how this digital twin enables the UAV to be self-aware. In particular, we demonstrate how structural sensor data is used to perform online data-driven adaptation of the digital twin. The rapidly updating digital twin enables the UAV to respond intelligently to damage or degradation detected in the wing structure.

A. Physical asset

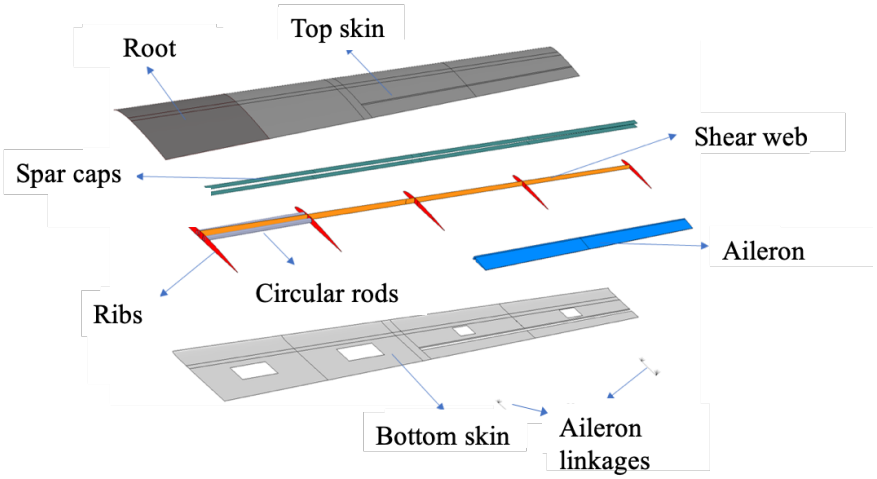
Our physical asset for this research is a 12-foot wingspan, fixed-wing UAV. The fuselage is from an off-the-shelf Telemaster airplane, while the wings are custom-built with a plywood and carbon fiber construction. The top surface of the right wing is outfitted with 24 uniaxial strain gauges distributed in two span-wise rows, either side of the main spar, between 25% and 75% span. The electric motor and avionics are also a custom installation. Photos of the aircraft during a recent series of flight tests are shown in Figure 3. The wings of the aircraft are interchangeable so that the aircraft can

Fig. 3 The custom-built hardware testbed used in this research. We create a digital twin of this 12-foot wingspan aircraft, and update the digital twin in response to online data from structural sensors on the aircraft wings.

be flown with pristine wings or wings inflicted with varying degrees of damage. This will allow us to simulate damage progression, and test whether a digital twin of the aircraft is able to adapt to the damage state using structural sensor data.

B. UAV model library

The goal of this case study is to enable the UAV to detect changes in the structural health of its wings, so that it may adapt its mission accordingly. In order for the digital twin to accurately represent a wide range of structural states, the underlying model must be detailed and expressive enough to accurately capture a range of structural defects, including cracking, delamination, denting, and loss of material. To this end, a component-based reduced-order model for the UAV has been developed using the Akselos *Integra* modeling software, in collaboration with Akselos Inc. This software contains a proprietary implementation of the SCRBE algorithm described in Section II.A, which is called RB-FEA. Figure 4 details the structure of the wing, as represented in our model. The model is divided into 15 components, as shown in Sec. II.B, Figure 2. The number of components chosen for the model takes into consideration factors such as the spatial distribution of parameters, component mesh sizes, etc. Our choice of 15 components provides a good balance between model flexibility and complexity for a system of this scale. We first consider a UAV model with no damage, which could be used as a physics-based digital twin of a pristine UAV. To emphasize the speed-up achieved by the reduced-order model, Table 1 provides a comparison of the number of degrees of freedom (DOFS) between our RB-FEA model, and a traditional FEA model created by stitching component meshes together to form a single domain.



Component	Material	Material Type	Element Type	No. Layers
Root	Carbon Fabric	2D Orthotropic	Quad4	4
Top skin	Carbon Fabric	2D Orthotropic	Quad4	3
Bottom skin	Carbon Fabric	2D Orthotropic	Quad4	3
Spar caps	Carbon Uni	2D Orthotropic	Quad4	8
Shear web	Carbon Fabric	2D Orthotropic	Quad4	3
Ribs	Plywood	Isotropic	Quad4	1
Circular rods	Carbon Iso-Tube	Isotropic	Beam2	-
Ailerons	Foam (HiLoad60)	Isotropic	Hex8	-
Aileron Linkages	-	-	3dof Spring, Rigid	-

Fig. 4 The internal structure of the aircraft wing. We use a combination of material properties and element types in order to capture the level of detail required to accurately model structural health in our digital twin.

	FEA	RB-FEA
# Components	-	15
# DOFS	1,383,234	928

Table 1 DOF comparison between a full-order FEA model of the UAV versus our reduced-order RB-FEA model.

In order for the digital twin of the UAV to be capable of rapidly adapting to different damage states, we construct a model library containing copies of each component inflicted with damage of varying degrees of severity. Rather than a detailed structural damage model (which would be computationally intractable in our setting), we use an effective damage model that is implemented via a reduction in material stiffness for selected regions of the wing skin. Our choice of this effective damage model is motivated by our particular use case of in-flight decision-making: the reduced-stiffness effective damage model is sufficient for detecting the effect of damage on vehicle flight capabilities. In our setting, characterizing precisely what damage has occurred is not required online. Instead this could be done via a subsequent detailed inspection of the aircraft on the ground. This manual inspection would be able to more accurately characterize the damage that has occurred, and, if needed, inform the addition of new components to the model library that model the damage in detail, as well as components representing anticipated damage progression.

Our effective damage model is implemented by creating copies of each archetype wing component, each with a fixed damage region. Within each damage region we introduce a damage parameter corresponding to the percentage reduction

in the Young's modulus. Here we use isotropic constitutive equations so that the structural displacement solution has an affine dependence on the damage parameter, which is assumed by default in the reduced-basis method we employ as part of the component-based modeling framework. Note that a non-affine dependence (and thus anisotropic constitutive laws in the damage region) could be achieved using methods such as the empirical interpolation method (EIM)[41].

The ability of the component-based model to scale to a large number of spatially distributed parameters allows us to have a number of these such regions distributed over the wing, while still maintaining accuracy and efficiency of the vehicle-level reduced model. In particular, our full model library currently contains 28 damage regions across both aircraft wings. For illustrative purposes, in this case study we consider a restricted model library in which only two effective damage regions are included. These regions are located on the top surface of the second and fourth spanwise component on the right wing. We refer to these components by index $i = 1, 2$ respectively. Pristine versions of the remaining 13 components comprising the UAV model are also included in the component library, C, but are excluded from our notation for clarity. To construct the model library, $\mathcal{M}$, we first sample five linearly spaced values of each damage parameter, corresponding to a reduction in stiffness in the damage region of between 0% and 80%. We then take all combinations of the two damage parameters, which gives a model library, $\mathcal{M}$, containing $|\mathcal{M}| = 25$ unique models. This shows that even for our restricted model library with only two damage regions, our digital twin of the UAV is able to adapt to 25 different effective damage states, each of which could represent the effect of a wide range of real damage states. This model library is illustrated in Figure 5.

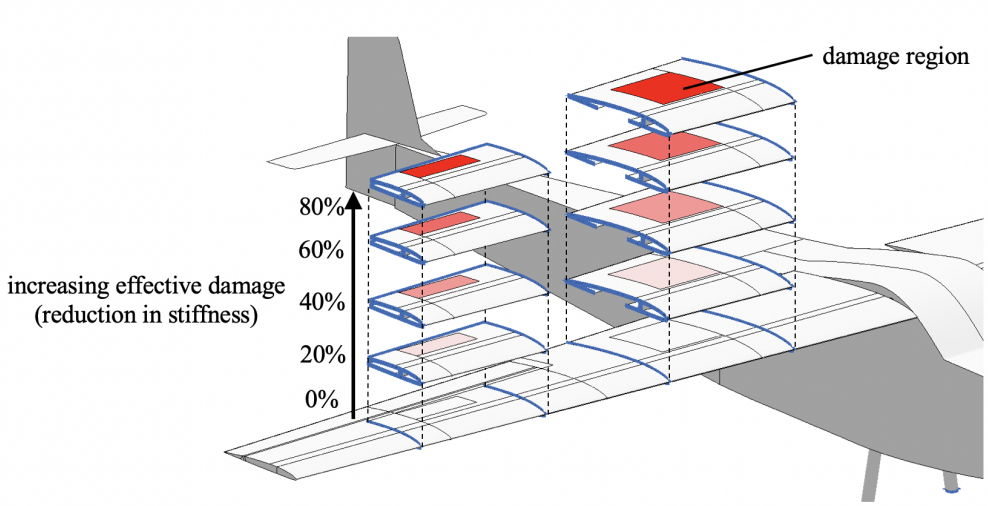


Fig. 5 An illustration of the model library used in this case study. We sample five values of the effective damage parameter in each of the damage regions (highlighted red). The model library is constructed by taking all combinations of damage parameters for the two components.

C. Digital twin model adaptation results

To demonstrate the concept of data-driven model adaptation using online sensor data, we propose an illustrative UAV mission. We suppose that the UAV is flying a mostly predefined mission consisting of successive level turns. However, we allow the UAV some flexibility in the maneuvers it executes in order to complete the mission. Taking the turn at a steeper bank angle makes the path shorter, but also subjects the UAV to an increased aerodynamic load. In particular, we will assume that the UAV can decide to take each turn at a bank angle corresponding to a load factor of either $2g$, or $3g$.

We discretize time such that each time index, $t \in 1, ..., T$ occurs during the steady section of a turn. This allows us to simplify the problem and ignore the transient period between turns. In practice this would mean the UAV would sample structural sensors during the steady section of a turn, and use the transient time between turns to assimilate the data, and decide on a load factor for the subsequent turn. We simulate a mission of length $T = 100$ turns. An example of such a mission is depicted in Figure 6 below.

At each time step t, we estimate the UAV digital twin, d_t, using the approach described in Sec. III. We assume that the UAV is known to begin in pristine condition, and we wish to monitor and estimate the progression of damage in the

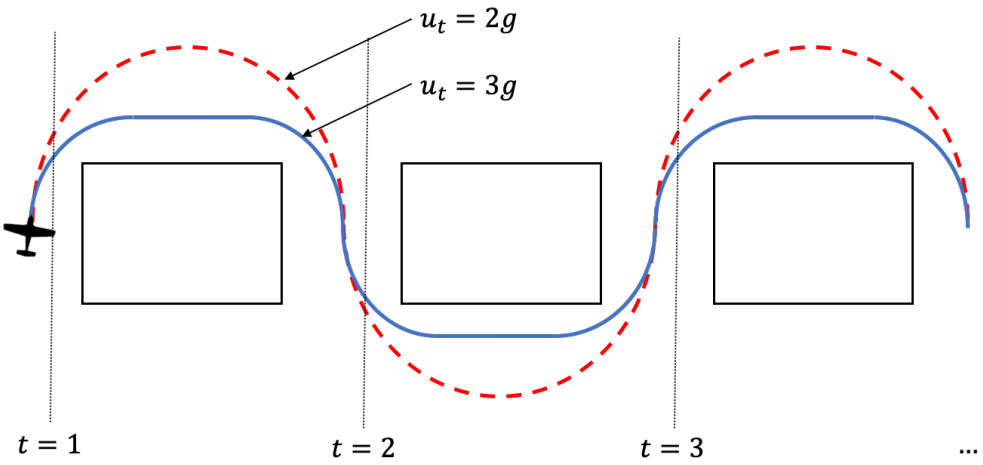


Fig. 6 An illustration of the UAV of mission we consider in this case study. The mission consists of successive steady level turn sections, which the UAV must decide whether to take at a $2g$ or $3g$ load factor.

two damage components. To define the state transition model, namely $p(d_t = M_j | d_{t-1} = M_k, u_t),\ \forall\, j, k$, we assume that the probability of damage progression in each component is known, fixed, and conditionally independent given the load on the aircraft wing (given by the load factor $u_t \in \{2g, 3g\}$). In particular, we suppose that a $2g$ maneuver has a 2% chance to worsen the damage by one level (i.e., 20% reduction in stiffness) in each component, while a $3g$ maneuver has a 5% chance. The probability of damage increasing by more than 20% in a single time-step is set to zero, and the probability of damage worsening beyond 80% reduction in stiffness is also set to zero. In practice this transition probability model could be more complex, and would be estimated from offline experiments, damage progression simulations, or historical data from past flights of similar UAVs.

During each turn, we acquire strain data from each of the 24 strain gauges on the aircraft wing. This data informs the UAV about the current damage state. We assume that the strain measurements, o_t are independent, unbiased, and have a standard deviation of $\sqrt{500}$ microstrain. This gives the measurement model $o_t = Z(M_j, u_t) + v_t$, where $v_t \sim N(\mathbf{0}, 500\mathbb{I})$ and $\mathbb{I} \in \mathbb{R}^{24\times 24}$ is an identity matrix. As described in Sec. III.A, the function $Z(M_j, u_t)$ predicts observations for a given state and input. In this case it returns the predicted strain at strain gauge locations for a given UAV damage state and aerodynamic load. For this function, we compute the aerodynamic load on the aircraft for a steady $2g$ or $3g$ turn using an ASWING [42] model of the aircraft. We then apply this load to the component-based reduced-model defined by $M_j \in \mathcal{M}$ and solve to compute the predicted strain at strain gauge locations.

While in flight, the structural model comprising the digital twin is updated at each time-step, so that the UAV maintains an accurate estimate of its structural damage state. Note that designing a flight control system that combines these damage estimates with specific mission parameters for optimal damage-aware flight planning is outside of the scope of the current paper. Instead we we present a simple illustrative example to demonstrate one way in which the UAV could leverage knowledge about its structural state to inform decision-making. In particular, we show how the UAV could use its up-to-date digital twin to perform predictive simulations in order to determine whether an upcoming maneuver would be safe to perform. In our scenario, if the result of this analysis suggests performing the more aggressive $3g$ maneuver has a high probability of failure, the UAV would instead falls back to the less aggressive $2g$ maneuver.

We define the quantities of interest to be the degree of damage in each damage region. We denote by $Q^i(M_j)$ the stiffness reduction in the damage region of component i of model M_j. While in flight, the up-to-date digital twin provides estimates of these quantities of interest. The structural model in the digital twin can also be used to simulate a $3g$ maneuver with a given reduction in stiffness, and predict whether structural failure will occur (for example due to exceeding allowable stress limits). In this case-study, the UAV digital twin shows that performing the more aggressive $3g$ turn will result in structural failure if the reduction in stiffness in either damage region exceeds 30%. Mathematically, the UAV's control policy is as follows:

$$u_{t+1} = \begin{cases} 3g, & \text{if } \max\limits_{i=1,2} \hat{Q}_t^i < 30\% \\ 2g, & \text{otherwise,} \end{cases} \tag{15}$$

where the digital twin estimate of the quantities of interest, $\hat{Q}_t^i$, are as defined in (13). This control policy allows the UAV to act aggressively when it is in good condition, and more conservatively when damaged, so as to minimize further damage and prevent eventual structural failure.

At each time step we randomly sample from the measurement model to produce a set of noisy strain measurements. We then use these measurements to perform a belief update according to (11). In the first experiment, we simulate stochastic state transitions using the state transition probabilities defined earlier. Note that this means that the true damage state of the UAV is simulated as piece-wise constant, with instantaneous changes in the stiffness reduction. This behaviour mimics sudden damage events, in which the state estimator must quickly detect that damage has occured, and update the digital twin model accordingly. Figure 7 shows the results for one realization of this experiment. These

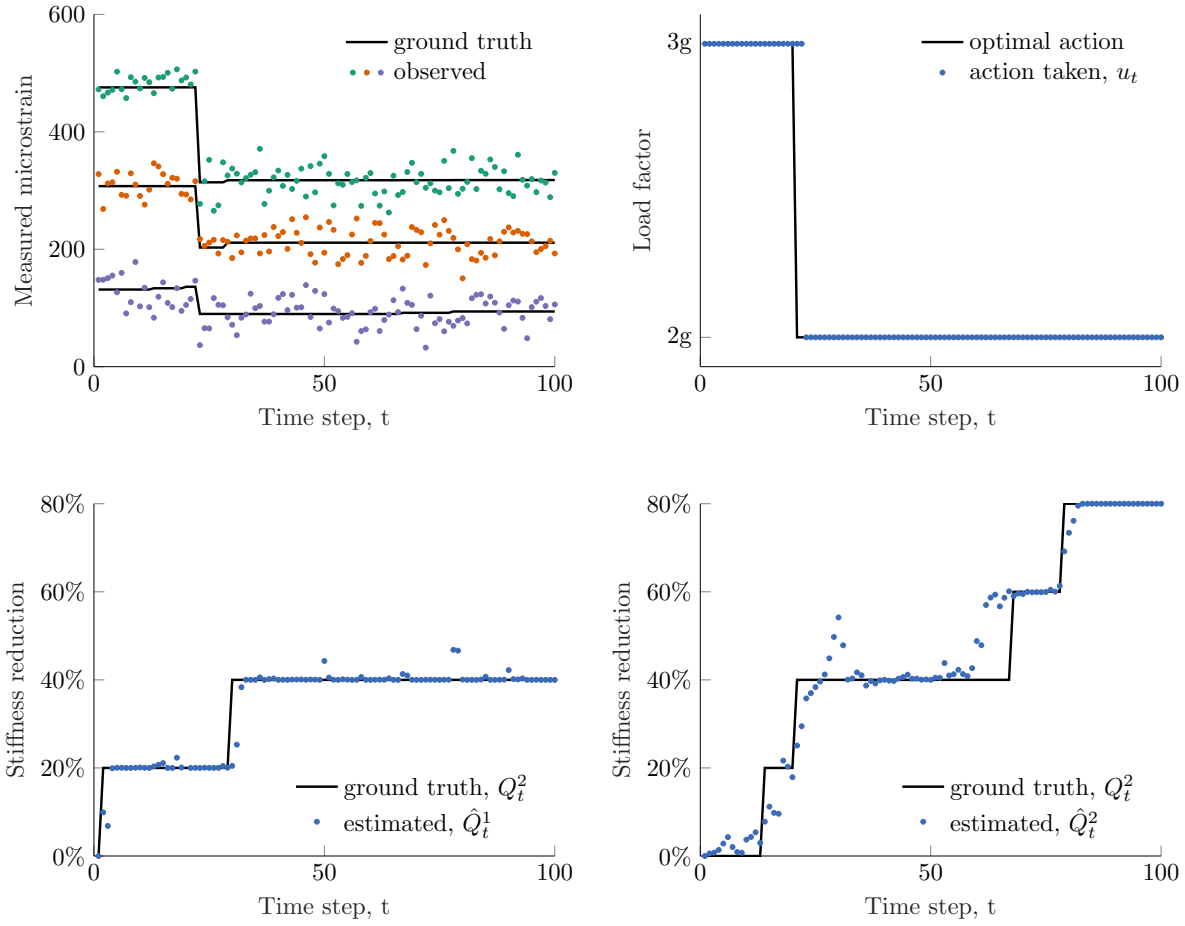


Fig. 7 Simulation results illustrating data-driven model updating. In this experiment the evolution of the ground-truth damage state is piece-wise constant, with stochastic transitions given by the probabilistic transition model described in the text. Top left: Noisy strain measurements used for estimation, (showing a sample of three out of 24 sensors). Top right: Action taken by the UAV at each time-step. Bottom: State estimates for the damage in components $i = 1, 2$ (left and right respectively).

results show that the state estimate, and thus the digital twin model, is able to track the true state of the UAV. Tracking performance is better for component $i = 1$ (the damage component nearest to the wing root). This is because the strain measurements are more sensitive to damage in this component. Damage in this component also has a much greater influence on the structural response under load, and the structural integrity of the wing.

We perform a second experiment, in which the true damage state of the UAV is prescribed to follow a deterministic trajectory, namely, we suppose that each component experiences a linear reduction in stiffness from the pristine state

down to the terminal state (80% reduction in stiffness) over the length of the mission. This type of damage progression mimics slow degradation of the asset, rather than instantaneous damage. This experiment also highlights the fact that the true state of the asset will not always be exactly represented in the model library. Figure 8 shows the result for one execution of this experiment.

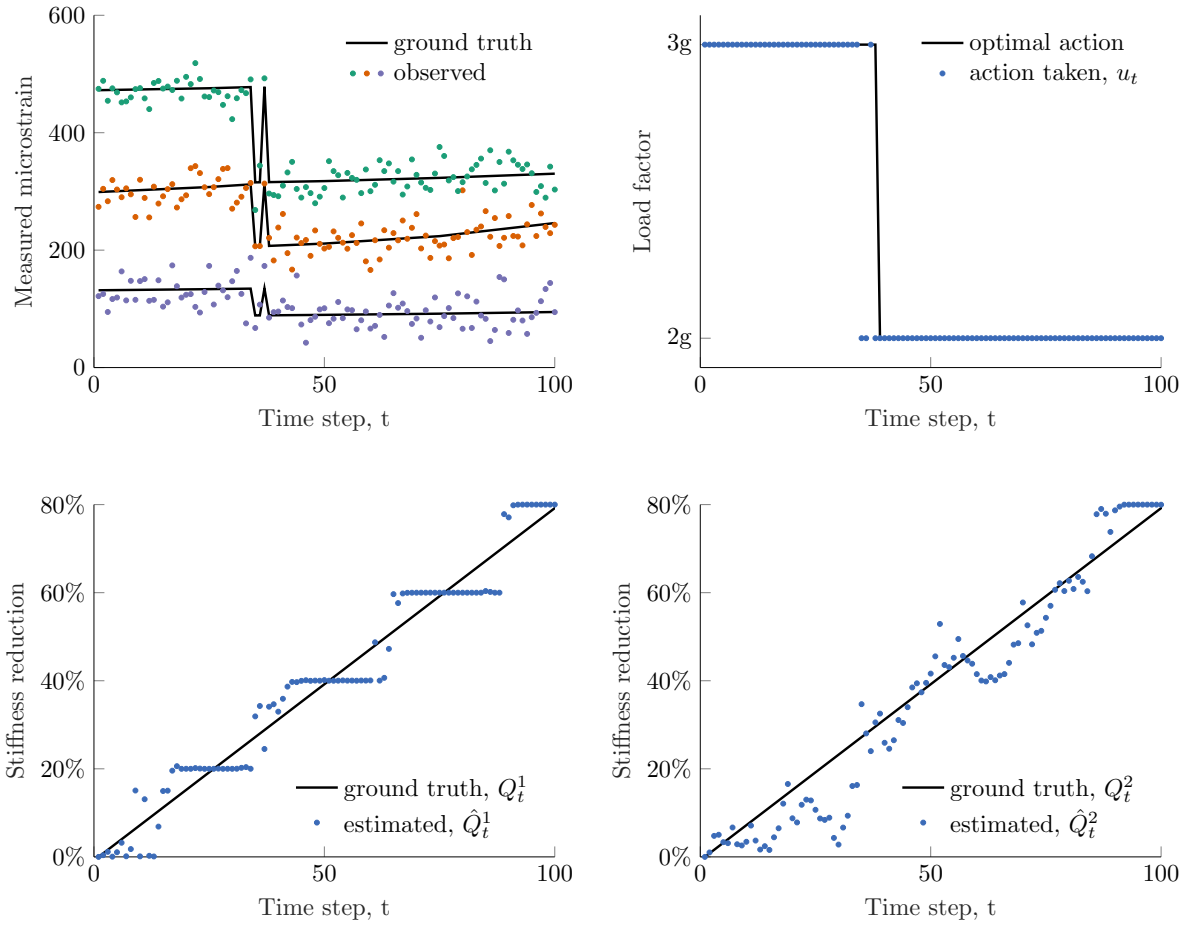


Fig. 8 Simulation results illustrating data-driven model updating via state estimation. In this experiment the evolution of the ground-truth damage state is deterministic and linearly increasing in severity. Top left: Noisy strain measurements used for estimation, (showing a sample of 3 out of 24 sensors). Top right: Action taken by the UAV at each time-step. Bottom: State estimates for the damage in components 1 and 2 (left and right respectively).

These results show that in this case the state estimate still provides an accurate piece-wise constant approximation to the linear underlying damage state. In many applications such a piece-wise constant approximation is likely to be sufficient for significantly improving decision making, provided that a fine enough library discretization is used. Since both of these experiments are stochastic, we repeat each experiment 1000 times. For each realization we compute several performance metrics, which are averaged over all time steps $t = 1, ..., T$. We compute the state estimation error for each component, $i = 1, 2$, as given by the absolute difference between the digital twin estimate, $\hat{Q}_t^i$, and the ground-truth value. We also compute the observation error, ϵ_t, as given in (14). Finally, we report the accuracy of the UAV's decision making, given by the percentage of time steps in which the action taken by the UAV (informed by the state estimate), matches the optimal action (informed by the ground truth). Table 2 reports the mean and standard error of the mean for these quantities over the 1000 realizations.

		$\lvert Q_t^1 - \hat{Q}_t^1 \rvert$	$\lvert Q_t^2 - \hat{Q}_t^2 \rvert$	ϵ_t	Action accuracy (%)
Ground-Truth State	Stochastic	0.78 (0.013)	5.47 (0.063)	23.1 (0.13)	95.8 (0.17)
	Linear	4.59 (0.011)	5.59 (0.039)	32.6 (0.02)	95.0 (0.11)

Table 2 Estimation performance over a sample of 1000 independent realizations. The quantities in the first three columns are averaged over the length of each realization, i.e., $t = 1, ..., T$. The first value in each cell is the sample mean, with the associated standard error following in parentheses.

V. Conclusion

This work has developed an approach for enabling data-driven physics-based digital twins using a library of component-based reduced-order models. The component-based models scale to large, complex assets, while the construction of a model library enables flexible and expressive model adaptation via parametric modification and component replacement. We proposed a method for data-driven model adaptation via Bayesian sequential state estimation. In particular, we showed how online sensor data can be used to solve a probabilistic classification problem in which we infer which entries in the model library best represent a physical asset. A limitation of our approach is that although we can detect model inadequacy or model bias (via the error measure defined in Section III.C.), our inference procedure does not automatically correct for model bias. Future work could involve extending our approach in this direction (see for example [14, 15, 36]). Our methods have been demonstrated using a case study in which a fixed-wing UAV uses structural sensors to detect damage or degradation on one of its wings. The sensor data is used to rapidly update the digital twin of the UAV, which is then used to inform the UAV's decision making about whether to perform an aggressive maneuver or a more conservative one to protect from further damage. Future challenges include investigating the robustness of the digital twin model selection with respect to corrupted sensor data and studying potential methods for improving this robustness [43, 44], and incorporating a wider range of damage models to enable the detection and classification of a wide range of actual damage states. The extreme cost of high-fidelity damage models remains a significant barrier to their implementation in the digital twin setting.

Acknowledgments

This work was supported in part by AFOSR grant FA9550-16-1-0108 under the Dynamic Data Driven Application Systems Program, by the SUTD-MIT International Design Center, and by The Boeing Company. The authors also gratefully acknowledge C. Kays and other collaborators at Aurora Flight Sciences for construction of the physical UAV asset.

References

[1] Glaessgen, E., and Stargel, D., "The Digital Twin Paradigm for future NASA and US Air Force Vehicles," *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA*, 2012, p. 1818.

[2] Li, C., Mahadevan, S., Ling, Y., Choze, S., and Wang, L., "Dynamic Bayesian Network for Aircraft Wing Health Monitoring Digital Twin," *AIAA Journal*, Vol. 55, No. 3, 2017, pp. 930–941.

[3] Tuegel, E. J., Ingraffea, A. R., Eason, T. G., and Spottswood, S. M., "Reengineering Aircraft Structural Life Prediction using a Digital Twin," *International Journal of Aerospace Engineering*, 2011.

[4] Kraft, J., and Kuntzagk, S., "Engine Fleet-Management: The Use of Digital Twins From a MRO Perspective," *ASME Turbo Expo 2017: Turbomachinery Technical Conference and Exposition*, American Society of Mechanical Engineers, 2017.

[5] Reifsnider, K., and Majumdar, P., "Multiphysics stimulated simulation digital twin methods for fleet management," *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2013, p. 1578.

[6] Tuegel, E. J., "The Airframe Digital Twin: Some Challenges to Realization," *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA*, 2012, p. 1812.

[7] Chinesta, F., Cueto, E., Abisset-Chavanne, E., Duval, J. L., and El Khaldi, F., "Virtual, Digital and Hybrid Twins: A New Paradigm in Data-Based Engineering and Engineered Data," *Archives of Computational Methods in Engineering*, 2018, pp. 1–30.

[8] Benner, P., Gugercin, S., and Willcox, K., "A survey of projection-based model reduction methods for parametric dynamical systems," *SIAM review*, Vol. 57, No. 4, 2015, pp. 483–531.

[9] Quarteroni, A., Rozza, G., et al., *Reduced Order Methods for Modeling and Computational Reduction*, Vol. 9, Springer, 2014.

[10] Chinesta, P., Ladeveze, P., and Cueto, E., "A short review on model order reduction based on Proper Generalized Decomposition," *Archives Computational Methods in Engineering*, Vol. 18, 2011, pp. 395–404.

[11] Antoulas, A. C., *Approximation of Large-Scale Dynamical Systems*, Vol. 6, SIAM, 2005.

[12] Huynh, D. B. P., Knezevic, D. J., and Patera, A. T., "A static condensation reduced basis element method: approximation and a posteriori error estimation," *ESAIM: Mathematical Modelling and Numerical Analysis*, Vol. 47, No. 1, 2013, pp. 213–251.

[13] Kennedy, M. C., and O'Hagan, A., "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Vol. 63, No. 3, 2001, pp. 425–464.

[14] Rubio, P.-B., Chamoin, L., and Louf, F., "Real-time Bayesian data assimilation with data selection, correction of model bias, and on-the-fly uncertainty propagation," *Comptes Rendus Mécanique*, Vol. 347, No. 11, 2019, pp. 762–779.

[15] Manzoni, A., Pagani, S., and Lassila, T., "Accurate solution of Bayesian inverse uncertainty quantification problems combining reduced basis methods and reduction error models," *SIAM/ASA Journal on Uncertainty Quantification*, Vol. 4, No. 1, 2016, pp. 380–412.

[16] Allaire, D., Biros, G., Chambers, J., Ghattas, O., Kordonowy, D., and Willcox, K., "Dynamic Data Driven Methods for Self-aware Aerospace Vehicles," *Procedia Computer Science*, Vol. 9, 2012, pp. 1206–1210.

[17] Lecerf, M., Allaire, D., and Willcox, K., "Methodology for dynamic data-driven online flight capability estimation," *AIAA Journal*, Vol. 53, No. 10, 2015, pp. 3073–3087.

[18] Singh, V., and Willcox, K. E., "Methodology for path planning with dynamic data-driven flight capability estimation," *AIAA Journal*, 2017, pp. 2727–2738.

[19] Eftang, J., Huynh, D., Knezevic, D., Ronquist, E., and Patera, A., "Adaptive port reduction in static condensation," *IFAC Proceedings Volumes*, Vol. 45, No. 2, 2012, pp. 695–699.

[20] Eftang, J. L., and Patera, A. T., "Port reduction in parametrized component static condensation: Approximation and a posteriori error estimation," *International Journal for Numerical Methods in Engineering*, Vol. 96, No. 5, 2013, pp. 269–302.

[21] Smetana, K., and Patera, A. T., "Optimal local approximation spaces for component-based static condensation procedures," Vol. 38, No. 5, 2016, pp. A3318–A3356.

[22] Noor, A. K., and Peters, J. M., "Reduced basis technique for nonlinear analysis of structures," *AIAA Journal*, Vol. 18, No. 4, 1980, pp. 455–462.

[23] Almroth, B., Stern, P., and Brogan, F. A., "Automatic choice of global shape functions in structural analysis," *AIAAl Journal*, Vol. 16, No. 5, 1978, pp. 525–528.

[24] Porsching, T., "Estimation of the error in the reduced basis method solution of nonlinear equations," *Mathematics of Computation*, Vol. 45, No. 172, 1985, pp. 487–496.

[25] Rozza, G., Huynh, D. B. P., and Patera, A. T., "Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations," *Archives of Computational Methods in Engineering*, Vol. 15, No. 3, 2007, p. 1.

[26] Veroy, K., Prud'Homme, C., Rovas, D., and Patera, A., "A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations," *16th AIAA Computational Fluid Dynamics Conference*, 2003, p. 3847.

[27] Binev, P., Cohen, A., Dahmen, W., DeVore, R., Petrova, G., and Wojtaszczyk, P., "Convergence rates for greedy algorithms in reduced basis methods," *SIAM Journal on Mathematical Analysis*, Vol. 43, No. 3, 2011, pp. 1457–1472.

[28] Ballani, J., Huynh, D., Knezevic, D., Nguyen, L., and Patera, A., "A component-based hybrid reduced basis/finite element method for solid mechanics with local nonlinearities," Vol. 329, 2018, pp. 498–531.

[29] Turner, M. J., Clough, R. W., Martin, H. C., and Topp, L. J., "Stiffness and deflection analysis of complex structures," *J. Aeronaut. Sci.*, Vol. 23, 1956, pp. 805–823.

[30] Guyan, R. J., "Reduction of stiffness and mass matrices," *AIAA Journal*, Vol. 3, 1965, pp. 380–380.

[31] Eftang, J. L., and Patera, A. T., "A port-reduced static condensation reduced basis element method for large component-synthesized structures: Approximation and a posteriori error estimation," *Advanced Modeling and Simulation in Engineering Sciences*, 2013.

[32] Smetana, K., "A new certification framework for the port reduced static condensation reduced basis element method," *Computer Methods in Applied Mechanics and Engineering*, Vol. 283, 2015, pp. 352–383.

[33] Smetana, K., and Patera, A. T., "Optimal local approximation spaces for component-based static condensation procedures," *SIAM Journal on Scientific Computing*, 2016.

[34] Russell, S. J., and Norvig, P., *Artificial Intelligence: A Modern Approach*, Pearson Education Limited, 2016.

[35] Singh, V., and Willcox, K. E., "Engineering Design with Digital Thread," *AIAA Journal*, Vol. 56, No. 11, 2018, pp. 4515–4528.

[36] Prudencio, E., Bauman, P., Faghihi, D., Ravi-Chandar, K., and Oden, J., "A computational framework for dynamic data-driven material damage control, based on Bayesian inference and model selection," *International Journal for Numerical Methods in Engineering*, Vol. 102, No. 3-4, 2015, pp. 379–403.

[37] El Moselhy, T. A., and Marzouk, Y. M., "Bayesian Inference with Optimal Maps," *Journal of Computational Physics*, Vol. 231, No. 23, 2012, pp. 7815–7850.

[38] Jaulmes, R., Pineau, J., and Precup, D., "Active learning in partially observable Markov decision processes," *European Conference on Machine Learning*, Springer, 2005, pp. 601–608.

[39] Ross, S., Chaib-draa, B., and Pineau, J., "Bayes-adaptive POMDPs," *Advances in Neural Information Processing Systems*, 2008, pp. 1225–1232.

[40] Martin, O. B., Williams, B. C., and Ingham, M. D., "Diagnosis as approximate belief state enumeration for probabilistic concurrent constraint automata," *Proceedings of the National Conference on Artificial Intelligence*, Vol. 20, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005, p. 321.

[41] Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. T., "An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations," *Comptes Rendus Mathematique*, Vol. 339, No. 9, 2004, pp. 667–672.

[42] Drela, M., "Integrated simulation model for preliminary aerodynamic, structural, and control-law design of aircraft," *40th Structures, Structural Dynamics, and Materials Conference and Exhibit*, 1999, p. 1394.

[43] Mishra, S., Shoukry, Y., Karamchandani, N., Diggavi, S. N., and Tabuada, P., "Secure state estimation against sensor attacks in the presence of noise," *IEEE Transactions on Control of Network Systems*, Vol. 4, No. 1, 2016, pp. 49–59.

[44] Pajic, M., Lee, I., and Pappas, G. J., "Attack-resilient state estimation for noisy dynamical systems," *IEEE Transactions on Control of Network Systems*, Vol. 4, No. 1, 2016, pp. 82–92.