Author version

Published as:

van Schie, S. P., & Hwang, J. T. (2023). Model reduction of isogeometric shell structures for large-scale design optimization. In AIAA AVIATION 2023 Forum (Paper No. AIAA 2023-3720). https://doi.org/10.2514/6.2023-3720

BibTeX for this reference:

https://lsdolab.github.io/lsdo-publications/publication/vanschie2023model/

```
@inproceedings{vanschie2023model,
  author = {Sebastiaan P. van Schie and John T. Hwang},
  title = {Model Reduction of Isogeometric Shell Structures For Large-Scale Design
    Optimization},
  booktitle = {AIAA AVIATION 2023 Forum},
  year = {2023},
  doi = {10.2514/6.2023-3720},
  url = {https://doi.org/10.2514/6.2023-3720}
}
```

Model Reduction of Isogeometric Shell Structures For Large-Scale Design Optimization

Sebastiaan P. C. van Schie *, John T. Hwang †
University of California San Diego, 9500 Gilman Dr, La Jolla, CA 92093

Including dynamic gust and flutter constraints in aeroelastic design optimization problems often presents prohibitive computational costs due to their need to be evaluated at many points throughout an aircraft's flight envelope. At the same time various studies have shown that gust response and flutter constraints are often limiting for optimized aircraft designs. A common approach for keeping optimization problems tractable with such constraints is to use lower-fidelity models for the aerodynamic or structural solvers. This fidelity switch can be made either completely or within a multi-fidelity framework, where the low-fidelity methods are used only for evaluating these expensive constraints. The loss of fidelity limits the accuracy of such methods and can make it impossible to include design parameters that lower-fidelity methods cannot take into account well. As alternative approach we use Reduced Modeling to reduce simulation cost in large-scale design optimization problems. This paper describes how a reduced model based on Proper Orthogonal Decomposition and the Matrix Discrete Empirical Interpolation Method is constructed and subsequently applied to speeding up static geometrically nonlinear simulations of isogeometric Kirchhoff—Love shells in large-scale optimization problems. This with the aim of serving as a stepping stone towards implementing a reduced model for dynamic nonlinear multi-shell simulations in such optimization problems. While the speedup per design evaluation obtained for a static test case is fairly modest (1.7 – 1.8 times) the results indicate a possible speedup of the current implementation of 2.5 – 3 times for dynamic simulations, without the use of any offline data acquisition phase. Various ways of improving upon these results are suggested for future work.

I. Introduction

When defining a Multidisciplinary Design Optimization (MDO) problem it is import to strike the right balance between problem scope and the fidelity of the analysis tools that are used to model the system of interest. Under the constraint of limiting total computational cost this translates to trading off the number of design variables versus the fidelity of the analysis tools. The former affects how many design iterations can be required to converge towards a design that is considered optimal, whereas the latter partially dictates the cost of evaluating proposed designs. Other considerations that directly affect the cost of optimization problems are the number and type of constraints. Some (in)equality constraints can be costly to evaluate for a given design point. Prime examples of this in the context of coupled aerostructural design optimization of aircraft are flutter and gust-related constraints. These constraints can depend on parameters that vary throughout the flight envelope, such as aircraft altitude, attitude, flight speed, gust intensity and gust inflow angle. As such these constraints need to be evaluated at many points throughout the flight envelope for each design iteration, since it is not known beforehand which combination of parameters corresponds to the critical load case. As has been pointed out throughout the relevant literature this makes large-scale MDO with such constraints intractable, for example in [1–3]. At the same time various studies have shown that flutter constraints often drive the optimal solutions of aeroelastic MDO problems; some examples of this are [4–6]. Thus there is a clear need to take flutter into account to find realistic optimal designs for aeroelastic MDO, but doing so severely limits the possible scope of tractable optimization problems.

Various approaches have been proposed for dealing with this issue. Jonsson et al. give an extensive overview in Table 1 of [1] of published works that consider flutter in optimization problems. Most works cited there resolve to making the optimization problem with flutter more tractable by opting for lower-fidelity aerodynamic or structural models. Some examples of this include [4, 5, 7, 8]. While that makes it possible to include flutter constraints in

*Ph.D. Student, Department of Mechanical and Aerospace Engineering, AIAA student member, svanschie@ucsd.edu
†Assistant Professor, Department of Mechanical and Aerospace Engineering, AIAA member

aeroelastic MDO, the loss of fidelity limits accuracy when solving the resulting optimization problems. It can also make it difficult to include certain design parameters at all, simply because the lower-fidelity simulation models might not be able to accurately take into account the influence of specific design variables.

The objective of this research is to take an initial step towards reducing the simulation cost in MDO problems of nonlinear parametrized Kirchhoff—Love (KL) shell models that are discretized with Isogeometric Analysis (IGA). We aim to do this by leveraging data-driven Reduced Models (RMs) to construct low-dimensional approximations to the aforementioned KL shell models. As far as we are aware, this has been done in only a few other works [9, 10], with neither considering design optimization problems. The novel contribution of this current work is its reduced modeling approach for large-scale Partial Differential Equation (PDE)-constrained optimization problems. There are four aspects to this novelty:

- The parameter spaces of the optimization problems are high-dimensional and the PDE's parametric dependence is nonaffine;
- No offline training or data acquisition phase is used to construct the reduced models;
- Individual cells are selected for reduced model construction of higher-order isogeometric basis functions;
- The reduced model is applied to optimization problems of nonlinear isogeometric KL shells.

While the main application in this work is single static nonlinear shells the methods that have been implemented are applicable to dynamic problems as well. Extensions to dynamic and multi-shell cases are planned for future work.

The structure of this paper is as follows: First a brief overview of relevant aspects of IGA and KL shell models are given in Sec. II, followed by a general formulation of optimization problems in Sec. III. This defines the high-fidelity optimization problem that we will try to accelerate with the RM methods that are covered in Sec. IV, where a compact overview of alternative RM methods is included as well. We apply our implemented RMs to the test case that is outlined in Sec. V. Lastly we draw conclusions in Sec. VI and, since this work serves as a stepping stone, we identify steps for future work.

II. Isogeometric Shell Modeling

We use this section to give a brief overview of some aspects of Isogeometric Analysis and the well-known Kirchhoff—Love shell model. Further references are given for interested readers in each section. We start with IGA in Sec. II.A, after which KL shell models and their discretization are covered in Sec. II.B.

A. Isogeometric Analysis

Isogeometric Analysis (IGA) was introduced by Hughes et al. in 2005 [11] and constitutes a class of Finite Element Analysis (FEA) methods that use Non-Uniform Rational B-Splines (NURBS) as basis functions. This allows IGA methods to directly use Computer-Aided Design (CAD) geometries for analysis, thereby allowing geometry simplification and (potentially costly) meshing steps to be skipped in lieu of straightforward refinement operations. IGA methods require fewer degrees of freedom to attain similar levels of accuracy to other FEA methods. Degrees of freedom of degree p are supported on $p+1$ quadrilateral cells in each dimension with locally-controllable inter-element continuity up to C^{p-1}. We focus here on only the aspects of IGA that are directly relevant for the current work. For a comprehensive overview of IGA the interested reader is referred to the IGA textbook by Cottrell et al. [12].

Traditional FEA methods use an isoparametric approach to map each element Ω^e in physical space to one single parametrized reference or 'master' element $\tilde{\Omega}$. For IGA on the other hand, the map $F : \hat{\Omega} \to \Omega$ that maps parametric domain $\hat{\Omega}$ to physical domain Ω is bijective: All of $\hat{\Omega}$ is mapped to all of Ω, while F^{-1} maps in the opposite direction. This allows us to pull back weak forms that are posed on Ω to $\hat{\Omega}$ before doing numerical integration in $\hat{\Omega}$. Since $\hat{\Omega}$ is invariant under deformations of Ω this gives us an appropriate domain on which to construct reduced models for problems that involve shape optimization. As mentioned by Hesthaven et al. in section 6.2 of [13], such a parameter-invariant domain is one of the necessary ingredients for using many reduced modeling approaches in shape optimization problems. IGA offers these features as core part of its mathematical definition.

The IGA implementation used in this work leverages the utilities provided by the FEniCS platform [14] through the tIGAr toolkit [15]. While FEniCS does not natively provide support for NURBS basis functions, tIGAr is able to use its provided utilities through the process of Bézier extraction. This allows us to define algebraic maps that express NURBS

bases as linear combinations of Lagrange polynomial bases on structured grids. Let $\hat{\boldsymbol{N}}^{IGA}$ and $\hat{\boldsymbol{N}}^{FE}$ denote vectors of NURBS and Lagrange polynomial basis functions on related structured grids. Bézier extraction allows us to explicitly relate both bases to each other through the extraction operator $\hat{M}$:

$$\hat{\boldsymbol{N}}^{IGA} = \hat{M}^T \hat{\boldsymbol{N}}^{FE}$$

Linear systems resulting from IGA can be computed in a similar way, by first constructing the corresponding FEA system with Lagrange polynomials. We start with the following FEA system:

$$A^{FE}\boldsymbol{x}^{FE} = \boldsymbol{b}^F E$$

The corresponding IGA system is computed by multiplying both sides with extraction matrix $\hat{M}^T$ and by realizing that $\boldsymbol{x}^{FE} = \hat{M}\boldsymbol{x}^{IGA}$. Hence:

$$\underbrace{\hat{M}^T A^{FE} \hat{M}}_{=A^{IGA}} \boldsymbol{x}^{IGA} = \underbrace{\hat{M}^T \boldsymbol{b}^F E}_{=\boldsymbol{b}^I GA}$$
$$A^{IGA}\boldsymbol{x}^{IGA} = \boldsymbol{b}^{IGA}$$

The linear system is considerably smaller when expressed in the IGA basis, as each NURBS is a linear combination of multiple Lagrange polynomials. Note that the NURBS functions are expressed on a Cartesian (quadrilateral) grid, while the FEA basis is expressed on a subdivision of the Cartesian grid into triangles. Fig. 1 is an example of such a subdivision. By using a discontinuous Lagrange basis it is possible to obtain cellwise extraction operators $\hat{M}_{(i)}$ that map the matrix A^{FE} to cellwise IGA matrices $A^{IGA}_{(i)}$. The cellwise extraction operators then form a partition of unity of the total extraction matrix:

$$\sum_{i=1}^{n^{FE}_{cells}} \hat{M}_{(i)} = \hat{M}$$

And naturally, the same holds for the cellwise IGA components and the total IGA matrix:

$$A^{IGA} = \hat{M}^T A^{FE} \hat{M} = \sum_{i=1}^{n^{FE}_{cells}} \hat{M}^T_{(i)} A^{FE} \hat{M} = \sum_{i=1}^{n^{FE}_{cells}} A^{IGA}_{(i)}$$

We will exploit this structure later when constructing reduced models.

B. Kirchhoff—Love Shells

The higher-degree continuity of NURBS basis functions allows us to use IGA to directly solve the KL shell model without any additional assumptions and simplifications, since solving its variational formulation requires at least C^1-continuity. This model was first used in conjunction with IGA by Kiendl et al. [16], and has since been used throughout literature for thin-walled solids in applications ranging from composite shells [17] to biological tissue membranes [18]. While we use the KL shell model in this work, we do not directly depend on its specific formulation: Other shell models are equally amenable to the approach presented here. Readers with an interest in the KL shell model are referred to the aforementioned work of Kiendl et al. for a precise definition.

Applying a variational principle to the KL shell model and discretizing with IGA results in a nonlinear system of equations. Using the Newton—Raphson method to resolve the nonlinearity gives an iterative boundary value problem of the following form:

$$J(\boldsymbol{\mu}, \boldsymbol{u}^k)\Delta\boldsymbol{u}^{k+1} = -\boldsymbol{R}(\boldsymbol{\mu}, \boldsymbol{u}^k) \tag{1}$$

Where $\boldsymbol{\mu}$ is a vector of shell design parameters, J is the (symmetric) Jacobian matrix and $\boldsymbol{R}$ is the residual vector. $\boldsymbol{u}$ is the vector of shell displacement degrees of freedom with solution $\boldsymbol{u}^k$ being used in the k^{th} iteration to compute the displacement update $\Delta\boldsymbol{u}^{k+1}$ such that $\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \Delta\boldsymbol{u}^{k+1}$. Convergence of the iteration scheme can be monitored by computing $||\boldsymbol{R}(\boldsymbol{\mu}, \boldsymbol{u}^k)||_2$, with the Newton iterations being stopped once the residual norm is lower than a predefined tolerance. We split the Jacobian matrix into its linear component J_l and a nonlinear term J_{nl}, such that:

$$\left(J_l(\boldsymbol{\mu}) + J_{nl}(\boldsymbol{\mu}, \boldsymbol{u}^k)\right)\Delta\boldsymbol{u}^{k+1} = -\boldsymbol{R}(\boldsymbol{\mu}, \boldsymbol{u}^k) \tag{2}$$

This decomposition of the Jacobian matrix will be exploited later when constructing reduced models. We note that both terms of the Jacobian matrix inherit the symmetry of the Jacobian matrix itself. In a similar way we can derive the general form of a geometrically nonlinear semi-discrete system:

$$M(\boldsymbol{\mu})\frac{d\boldsymbol{u}}{dt} = (A_l(\boldsymbol{\mu}) + A_{nl}(\boldsymbol{\mu}, \boldsymbol{u}))\,\boldsymbol{u} + \boldsymbol{b}(\boldsymbol{\mu}, \boldsymbol{u}), \tag{3}$$

where now $\boldsymbol{u} = \boldsymbol{u}(t)$ is the time-dependent vector of displacement degrees of freedom, M is the corresponding mass matrix, $\boldsymbol{b}$ the (possibly nonlinear) source term vector and A the matrix that governs the dynamic behavior of $\boldsymbol{u}$. A has been split into its linear and nonlinear components A_l and A_{nl}. A suitable implicit or explicit time stepping scheme has to be applied in order to obtain a series of linearized systems that can be solved to obtain dynamic numerical solutions.

III. Structural Optimization Problem

We recall the general formulation of PDE-constrained optimization problems as can be found in any textbook on optimization, for example [19]. This can be given as:

$$\begin{aligned} \underset{\boldsymbol{\mu}}{\text{minimize}} \quad & \mathcal{J}(\boldsymbol{\mu}, \boldsymbol{u}) \\ \text{subject to} \quad & \mathcal{L}(\boldsymbol{\mu}, \boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{\mu}) \\ & C(\boldsymbol{\mu}, \boldsymbol{u}) \geq \boldsymbol{0} \\ & D(\boldsymbol{\mu}, \boldsymbol{u}) = \boldsymbol{0} \end{aligned} \tag{4}$$

where $\mathcal{J}$ is an objective function that can depend on the vectors of optimization variables $\boldsymbol{\mu}$ and state variables $\boldsymbol{u}$. This system state follows from solving one or more (possibly coupled) Partial Differential Equations (PDEs) $\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{u}) = f(\boldsymbol{\mu})$. $C(\boldsymbol{\mu}, \boldsymbol{u})$ and $D(\boldsymbol{\mu}, \boldsymbol{u})$ indicate inequality and equality constraint operators, which have to be larger than and equal to zero, respectively. Applying any (gradient-based) optimization algorithm to Eq. (4) leads to a sequence of design points $[\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3, \ldots]$ and (after solving the PDE constraints) associated sequence of state vectors $[\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, \ldots]$.

Note that the objective function, PDE model and the constraints all contain dependence on the design variables $\boldsymbol{\mu}$. The PDE model $\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{u}) = f(\boldsymbol{\mu})$ is a static or dynamic KL shell model within the current context, as is given in section II. We keep the objective function $\mathcal{J}$ and the (in)equality constraints general for now, as these will vary from problem to problem. The design variables contained in $\boldsymbol{\mu}$ may be geometric in nature, and can thus affect the PDE operator $\mathcal{L}$ through both the integration domain Ω_{μ} or the integrands themselves. Recall that IGA allows us to pull back PDEs posed on domain Ω_{μ} in physical space to an invariant reference domain $\hat{\Omega}$. As such any change in integration domain shape induced by changes in $\boldsymbol{\mu}$ can be taken into account through the Jacobian of the map $F : \hat{\Omega} \rightarrow \Omega_{\mu}$ and its inverse.

IV. Model Order Reduction

We start our treatment of reduced modeling by discussing its common use-cases and the differences between them in Sec. IV.A. This context motivates some of the choices for the specific reducing modeling approach that we use. In Sec. IV.B we cover the well-known Proper Orthogonal Decomposition method, the first step of our two-step reduced model. As is concluded in that section, reduced models of nonaffinely-parametrized, nonlinear problems need to use a 'hyper-reduction method' to enable significant computational speedups. Hence we start Sec. IV.C with an overview of several hyper-reduction methods before introducing the method that we use in this work and the modifications that we propose to make it more efficient for IGA methods. Following that we detail the online implementation of the reduced model in Sec. IV.D. This online implementation uses an updating Singular Value Decomposition (SVD) approach from literature to minimize the cost of updating the reduced model. It does this by using the known SVD of the data matrix when new data comes in. This updating SVD approach is covered in Sec. IV.E.

A. Applications of Reduced Models

Reduced Modeling, also referred to as model reduction, is a set of techniques that aim to approximate with cheaper analogues the behavior of high-cost, generally high-fidelity simulation models. We will refer to the original (non-reduced) simulation models as Full Models (FMs). Common applications of model reduction generally fall within two general use-cases:

1) Real-time simulation of dynamical systems;
2) Many-query environments.

These applications are related by their need for efficient and repeated evaluations of FMs. Case 1 mainly concerns itself with optimal control problems, in which (near-)real-time simulation allows for the use of predictive models to calculate ahead of time how a system will respond to a given input. This use-case has a clear offline-online decomposition: An overwhelming amount of computing effort can be used during an offline phase to gather data and to construct an efficient RM that accurately captures the system's dynamics while being very cheap to evaluate. RM use then happens during system deployment, where it is often infeasible to use the FM as part of a control system. In this way RMs allow for the significant FM computational cost to be incurred in a 'lab setting' instead of during control system deployment.

Optimization problems are covered by case 2: Numerical solutions of a discretized PDE problem are computed with many different input parameter vectors $\boldsymbol{\mu}$. This application does not lend itself as well to a clear offline-online decomposition since it is often not flat-out infeasible to use the FM in a given optimization problem. Expending an overwhelming amount of computing effort to gather an large amount of data for constructing any RM in optimization settings then raises the question, "Why not use the FM to compute optimal designs in the first place?" As such we limit ourselves to RMs without offline-online decomposition, that are constructed and updated with data gathered while solving an optimization problem.

A wide variety of model reduction approaches can be used to reduce the computational cost of numerical models that are used in optimization problems. Some of these involve using lower-fidelity models and possibly tuning model coefficients such that they approximate the behavior of higher-fidelity models. As previously mentioned, this is one of the approaches that has been used in recent literature to manage the cost of dynamic constraints in aeroelastic optimization problems, with one-dimensional beam models being constructed from higher-fidelity geometrical representations. In this work we focus on Reduced Basis (RB) methods. These align well with FEA-type methods, since they explicitly define finite-dimensional function spaces for their discrete solutions. This allows for theoretical error analysis and theory-based studies of convergence behavior of RB methods [20]. The specifics of the Reduced Basis method that is used in this work are covered in section IV.B.

B. Proper Orthogonal Decomposition

We give a brief overview of the Proper Orthogonal Decomposition (POD) method for model reduction. Consider the linearized iterative system stemming from Newton—Raphson iterations as given in Eq. (2), which we will use as FM:

$$\left(J_l(\boldsymbol{\mu}) + J_{nl}(\boldsymbol{\mu}, \boldsymbol{u}^k)\right) \Delta \boldsymbol{u}^{k+1} = -\boldsymbol{R}(\boldsymbol{\mu}, \boldsymbol{u}^k) \tag{5}$$

where $J_l(\boldsymbol{\mu}), J_{nl}(\boldsymbol{\mu}, \boldsymbol{u}^k) \in \mathbb{R}^{n \times n}$ and $\boldsymbol{u}^{k+1}, \boldsymbol{R}(\boldsymbol{\mu}, \boldsymbol{u}^k) \in \mathbb{R}^n$. As this system has been discretized with IGA, the entries of J_l, J_{nl} and $\boldsymbol{R}$ follow from integrating a weak form over Ω (or equivalently over $\hat{\Omega}$, taking into account the Jacobian of the map F). We define the (orthonormal) reduction operator $V \in \mathbb{R}^{n \times r}$ such that $r << n$, and use V to project the aforementioned linear system to an r-dimensional subspace:

$$\begin{aligned} J_{l,rb}(\boldsymbol{\mu}) &= V^T J_l(\boldsymbol{\mu}) V \\ J_{nl,rb}(\boldsymbol{\mu}, \boldsymbol{u}^k) &= V^T J_{nl}(\boldsymbol{\mu}, \boldsymbol{u}^k) V \\ \boldsymbol{u}_{rb}^{k+1} &= V^T \boldsymbol{u}^{k+1} \\ \boldsymbol{R}_{rb}(\boldsymbol{\mu}, \boldsymbol{u}^k) &= V^T \boldsymbol{R}(\boldsymbol{\mu}, \boldsymbol{u}^k) \end{aligned}$$

Note that $\boldsymbol{u}_{rb}^{k+1} = \boldsymbol{u}_{rb}^k + \Delta \boldsymbol{u}_{rb}^{k+1}$. The related n-dimensional approximation $\tilde{\boldsymbol{u}}^{k+1}$ to $\boldsymbol{u}^{k+1}$ can be retrieved from $\boldsymbol{u}_{rb}^{k+1}$ through the inverse projection:

$$\tilde{\boldsymbol{u}}^{k+1} = V \boldsymbol{u}_{rb}^{k+1}$$

And thus:

$$\left(J_{l,rb}(\boldsymbol{\mu}) + J_{nl,rb}(\boldsymbol{\mu}, V\boldsymbol{u}_{rb}^k)\right) \Delta \boldsymbol{u}_{rb}^{k+1} = -\boldsymbol{R}_{rb}(\boldsymbol{\mu}, V\boldsymbol{u}_{rb}^k) \tag{6}$$

This reduced linear system is only r-dimensional, and is thus much cheaper to solve than Eq. (5). The reduction operator V can be constructed by computing the SVD of a so-called snapshot matrix, i.e. a matrix $X \in \mathbb{R}^{n \times s}$ that contains a

set of s state vectors that are obtained by solving the FM Eq. (5) for various $\boldsymbol{\mu}$. We append the solution $\boldsymbol{u}^{k+1}$ of each Newton iteration to X, which thus has the following structure:

$$X = \begin{bmatrix} \boldsymbol{u}^1(\boldsymbol{\mu}_1) & \boldsymbol{u}^2(\boldsymbol{\mu}_1) & \dots & \boldsymbol{u}^1(\boldsymbol{\mu}_2) & \boldsymbol{u}^2(\boldsymbol{\mu}_2) & \dots & \boldsymbol{u}^1(\boldsymbol{\mu}_3) & \boldsymbol{u}^2(\boldsymbol{\mu}_3) & \dots \end{bmatrix}$$

Section IV.E goes into more detail on the SVD implementation that we use in this work. Reduction operator V follows from the SVD ($X = \Phi\Sigma\Psi^T$) by taking the first r columns of Φ, since these correspond to the dominant modes of X. The number of columns r is chosen by using the Energy Ratio (ER) ϵ_r:

$$\epsilon_r = \frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{s} \sigma_i^2}$$

Where naturally, $\epsilon_r \to 1$ as $r \to s$. An alternative definition is used in some literature as well, where sums of the non-squared singular values are used to calculate ϵ_r. Energy Ratios anywhere between 0.9 and 1 are commonly used in model reduction applications.

The dependence of the FM Jacobian terms on $\boldsymbol{\mu}$ requires us to construct J_l and J_{nl} for each new design point $\boldsymbol{\mu}$ that is to be simulated with the POD-reduced model of Eq. (6). In cases where the parametric dependence of a matrix is said to be affine it is possible to decompose said matrix into a sum of parameter-weighted matrix components. Such a structure would allow us to compute the corresponding reduced matrix components only once, since the POD basis does not act directly on the aforementioned parameter weights. However, since our intended optimization application includes non-affine parametric dependencies (such as e.g. free-form deformation) we cannot leverage such structure.

Moreover, the Jacobian component J_{nl} captures the nonlinear dependence of J on $\boldsymbol{u}$. While J_l only has to be constructed once per design point (during the first Newton—Raphson iteration), each subsequent iteration requires construction of J_{nl}. For nonlinear time-dependent systems of the form shown in Eq. (3), many time steps are needed per design point. While the linear term A_l only has to be constructed once, its nonlinear counterpart A_{nl} has to be constructed at least once per time step (and possibly multiple times, for implicit time marching schemes). As such, the construction cost of A_{nl} dominates the total matrix construction cost, which comprises a significant part of the total compute time per design point. Because of these reasons we opt to focus our efforts on reducing the assembly cost of the nonlinear terms J_{nl} (for static simulations) and A_{nl} (in dynamic cases). This is in line with our goal of constructing accurate and efficient RMs for (dynamic) nonlinear flutter simulations.

C. Hyper-reduction

While we are able to use POD to reduce the time it takes to solve linear(ized) systems, matrix assembly still presents a significant contribution to the total compute time, making the computational gains from applying only POD modest. This and the considerations mentioned in the previous section motivate to speed up assembly of the nonlinear matrices that are used in static and dynamic models. Hyper-reduction models exist specifically for dealing with this issue within the context of nonlinear models. The Empirical Interpolation Method (EIM) [21] constructs approximations by selecting points in the parametric and state spaces in which summations of affine linearized terms interpolate the original nonaffine, nonlinear function. The summation of terms is then discretized in the same way as the FM. EIM's computational speedup comes from the fact that its affine linearized terms only have to be assembled once.

Other methods construct hyper-reduced RMs directly on the discrete level. Some examples of this are Energy-Conserving Sampling and Weighting [22], Missing Point Estimation [23], Gauss-Newton with Approximated Tensors [24, 25], Gappy POD [26, 27] and the Discrete Empirical Interpolation Method (DEIM) [28]. A common thread between these methods is that they attempt to construct vectors or matrices from low-rank approximations, usually a subset of the vector/matrix entries. Various works have compared the performance of these methods when applied to the same problem or discuss their advantages and limitations [29–32]. As hyper-reduction method in this work we opt for a variant of the Matrix Discrete Empirical Interpolation Method (MDEIM), or Matrix DEIM. This method has been applied successfully to various nonlinear structural problems [33–36]. A big advantage of using a DEIM-type method for our intended applications is that it is amenable to nonlinear systems with general (nonaffine) parametric dependence.

Consider the Cartesian grid shown in Fig. 1, where the support of some basis functions has been highlighted with different colors. The cubic NURBS basis functions that are used when simulating KL shells are nonzero on up to

four quadrilateral cells in each direction. Hence it is possible to cover the entire domain with as few as four basis functions. Because of the Bézier extraction process that we use to do IGA in FEniCS this implies that we can be forced to numerically integrate all FE basis functions on the domain when only a handful of selected matrix entries are to be assembled, thereby effectively nullifying any speedup in constructing J_{nl} with MDEIM. This problem is especially prevalent for FEA methods that use basis functions with a large multi-element support, which includes IGA. To circumvent this issue we use the 'unassembled' approach that was suggested for FEA methods by Tiso et al. in [37]: Instead of identifying only matrix entries that should be computed (by integrating over their entire support) we identify the individual cell components that should be computed. In effect this gives us much finer control over how many cells to use for approximating the FM and thus allows us to construct lower-cost RMs. As far as we are aware this unassembled MDEIM approach has not been used before, nor has unassembled DEIM been used in conjunction with IGA methods. Due to the large number of cells that are contained in the support of each NURBS function we expect that the impact of such an unassembled approach is even more significant than what was shown by Tiso et al. in [37]. As was mentioned in Sec. II.A we are able to efficiently obtain decompositions of the Jacobian matrix into its cellwise components by using cellwise Bézier extraction from a discontinuous Lagrange basis.

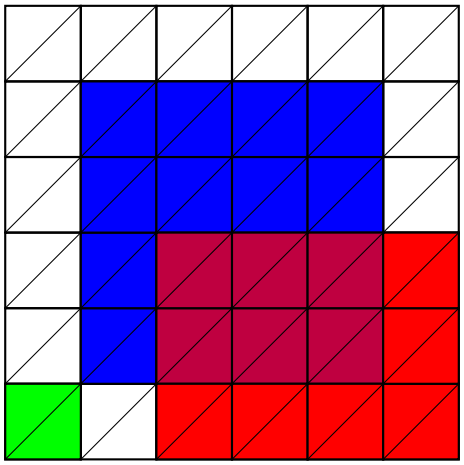

Fig. 1 Example 6 × 6 Cartesian shell mesh with the supports of several cubic NURBS basis functions highlighted in color

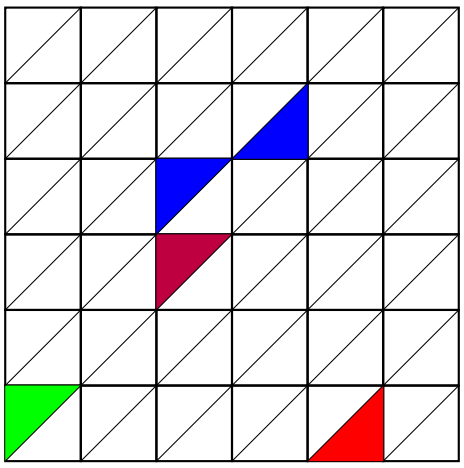

Fig. 2 Example 6 × 6 Cartesian shell mesh with example MDEIM-selected cells

We give a brief overview of our cellwise (or "unassembled") version of MDEIM. Like any data-driven model MDEIM starts with gathering data. We carry out FM simulations for a number of design points $\boldsymbol{\mu}_i$ and store the

nonlinear Jacobian terms of each Newton iteration in a snapshot matrix. To do this for our cellwise approach we need to store each cellwise Jacobian matrix. While this presents an increase in the required storage space each cellwise Jacobian matrix is sparse, and thus the storage requirements are not intractable. We need to do two things to put $J(\boldsymbol{\mu}_i, \boldsymbol{u}^k)$ into a form that can be appended to a snapshot matrix. The order of these two steps is not important, as long as the numbering of matrix entries is consistent within the MDEIM algorithm. Let $J_{nl}^{(q)}(\boldsymbol{\mu}_i, \boldsymbol{u}^k)$ denote the q^{th} cellwise component of $J_{nl}(\boldsymbol{\mu}_i, \boldsymbol{u}^k)$. From now on we omit the subscript nl for clarity and because MDEIM is not limited to nonlinear matrices.

- We denote the columns of cellwise Jacobian matrix $J^{(q)}(\boldsymbol{\mu}_i, \boldsymbol{u}^k)$ as follows:

$$J^{(q)}(\boldsymbol{\mu}_i, \boldsymbol{u}^k) = \begin{bmatrix} \boldsymbol{j}_1^{(q)}(\boldsymbol{\mu}_i, \boldsymbol{u}^k) & \boldsymbol{j}_2^{(q)}(\boldsymbol{\mu}_i, \boldsymbol{u}^k) & \dots & \boldsymbol{j}_{n^{IGA}}^{(q)}(\boldsymbol{\mu}_i, \boldsymbol{u}^k) \end{bmatrix} \in \mathbb{R}^{n^{IGA} \times n^{IGA}}$$

  We can vectorize the matrix by stacking its columns into a single vector:

$$\tilde{\boldsymbol{J}}^{(q)}(\boldsymbol{\mu}_i, \boldsymbol{u}^k) = \begin{bmatrix} \boldsymbol{j}_1^{(q)}(\boldsymbol{\mu}_i, \boldsymbol{u}^k) \\ \boldsymbol{j}_2^{(q)}(\boldsymbol{\mu}_i, \boldsymbol{u}^k) \\ \vdots \\ \boldsymbol{j}_{n^{IGA}}^{(q)}(\boldsymbol{\mu}_i, \boldsymbol{u}^k) \end{bmatrix} \in \mathbb{R}^{\left(n^{IGA}\right)^2}$$

- We then stack the vectorized cellwise components $\tilde{\boldsymbol{J}}^{(q)}(\boldsymbol{\mu}_i, \boldsymbol{u}^k)$ into a single vector. This results in:

$$\tilde{\boldsymbol{J}}(\boldsymbol{\mu}_i, \boldsymbol{u}^k) = \begin{bmatrix} \tilde{\boldsymbol{J}}^{(1)}(\boldsymbol{\mu}_i, \boldsymbol{u}^k) \\ \tilde{\boldsymbol{J}}^{(2)}(\boldsymbol{\mu}_i, \boldsymbol{u}^k) \\ \vdots \\ \tilde{\boldsymbol{J}}^{(n_{cells}^{FE})}(\boldsymbol{\mu}_i, \boldsymbol{u}^k) \end{bmatrix} \in \mathbb{R}^{\left(n^{IGA}\right)^2 n_{cells}^{FE}}$$

The vectorized Jacobian matrices $\tilde{\boldsymbol{J}}(\boldsymbol{\mu}_i, \boldsymbol{u}^k)$ corresponding to multiple design points $\boldsymbol{\mu}_i$ and Newton iteration solution estimates $\boldsymbol{u}^k$ can then be combined into a snapshot matrix $X \in \mathbb{R}^{\left(n^{IGA}\right)^2 n_{cells}^{FE} \times s}$:

$$X = \begin{bmatrix} \tilde{\boldsymbol{J}}(\boldsymbol{\mu}_1, \boldsymbol{u}^1) & \tilde{\boldsymbol{J}}(\boldsymbol{\mu}_1, \boldsymbol{u}^2) & \dots & \tilde{\boldsymbol{J}}(\boldsymbol{\mu}_2, \boldsymbol{u}^1) & \tilde{\boldsymbol{J}}(\boldsymbol{\mu}_2, \boldsymbol{u}^2) & \dots \end{bmatrix}$$

Where the entries of each row of X correspond to the contribution of one cell to one matrix entry. Because of this we can use one of several matrix decompositions to find which rows are most representative for the cellwise decomposition of Jacobian matrices. As we did for POD we compute a Singular Value Decomposition of the snapshot matrix. The SVD approach that is used is explained in more detail in Sec. IV.E. This results in an orthonormal basis $\Phi \in \mathbb{R}^{\left(n^{IGA}\right)^2 n_{cells}^{FE} \times r}$ (with $r \leq s$). The number of columns r of Φ is determined by looking at the Energy Ratio, as was covered in Sec. IV.B.

With this orthonormal low-dimensional basis in hand the next step is to identify the matrix entries that we should compute and the cells on which we should evaluate them to best approximate the columns of Φ. The question of how to select the (M)DEIM matrix indices has received significant attention throughout the relevant literature. We first recall in Alg. 1 the standard DEIM index selection algorithm as originally proposed in [28]. We keep adding indices to the chosen set $\mathcal{I}$ in a greedy way by selecting the index of the maximum entry of residual vector $\boldsymbol{r}$. For an orthonormal basis $\Phi \in \mathbb{R}^{\left(n^{IGA}\right)^2 n_{cells}^{FE} \times r}$ this gives a set of exactly r indices. This set of indices is thus chosen in an attempt to minimize the difference between fully-assembled cellwise vectorized Jacobian matrices $\tilde{\boldsymbol{J}}(\boldsymbol{\mu}, \boldsymbol{u})$ and their projected set of indices $\Phi\Phi_{\mathcal{I}}^{-1}\tilde{\boldsymbol{J}}_{\mathcal{I}}(\boldsymbol{\mu}, \boldsymbol{u})$.

This is the selection approach that is used in this work. Alternative approaches exist as well: One of these is QDEIM [38], which uses a QR decomposition with pivoting to determine the set of indices. Several other approaches can be found in [39–43]. Some of these approaches pick exactly r interpolation indices for a projection operator $\Phi\Phi_{\mathcal{I}}^{-1}$ of rank r whereas others pick more than r indices. This makes matrix $\Phi_{\mathcal{I}}$ non-square and hence its inverse needs to be replaced by the Moore—Penrose pseudo-inverse. The projection operator becomes $\Phi\Phi_{\mathcal{I}}^{\dagger}$ and does regression on the selected indices rather than actually interpolating them. A common theme that ties these models together is that they try to make tractable the combinatorial problem of selecting the set of indices that minimize the projection error over some space.

Algorithm 1 (M)DEIM index selection

Input: Basis $\Phi = [\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2\ \ldots \boldsymbol{\phi}_r] \in \mathbb{R}^{m\times r}$
Output: Set of r selected (M)DEIM indices $\mathcal{I}$

$\mathcal{I} = \arg\max |\boldsymbol{\phi}_1|$
$\Theta = [\boldsymbol{\phi}_1]$
for $k = 2 : r$ **do**
  $\Theta_{\mathcal{I}} = \left[\boldsymbol{\phi}_{1,\mathcal{I}}\ \ldots\ \boldsymbol{\phi}_{k-1,\mathcal{I}}\right] \in \mathbb{R}^{(k-1)\times(k-1)}$
  $\boldsymbol{r} = \boldsymbol{\phi}_k - \Phi\Theta_{\mathcal{I}}^{-1}\boldsymbol{\phi}_{k,\mathcal{I}}$
  $i = \arg\max |\boldsymbol{r}|$
  $\mathcal{I} = \mathcal{I} \cup i$
  $\Theta = [\Theta\ \boldsymbol{\phi}_k]$
end for

After the set of selected indices has been determined we do some straightforward postprocessing to determine the specific cells that require numerical integration to compute the selected matrix entries. This entails division and modulo calculations that depend on the ordering of the entries of the vectorized Jacobian matrices. Fig. 2 shows an example of what such a set of cells looks like. The cells have been colored according to the basis function supports shown in Fig. 1. We can obtain a speedup in FEniCS by restricting to the set of selected cells the integration measure that FEniCS uses to define numerical integration domains. With the DEIM indices and projection operation in hand we can run the RM for given design points with Alg. 2.

Algorithm 2 POD-MDEIM Newton iteration

Input: POD basis V, MDEIM basis Φ, MDEIM indices $\mathcal{I}$, design parameter vector $\boldsymbol{\mu}$, convergence tolerance ϵ, maximum iteration count i_{max}
Output: Approximate solution $\boldsymbol{u}$

$\boldsymbol{u} = \boldsymbol{0}$
Compute initial residual vector $R(\boldsymbol{\mu}, \boldsymbol{u})$
$i = 1$
while $||R(\boldsymbol{\mu}, \boldsymbol{u})||_2 > \epsilon$ and $i \leq i_{max}$ **do**
  if $i = 1$ **then**
    Assemble linear Jacobian term $J_l(\boldsymbol{\mu})$
    $J_{nl}(\boldsymbol{\mu}, \boldsymbol{u}) = 0$
  else if $i > 1$ **then**
    Construct MDEIM-reduced nonlinear Jacobian term $J_{nl,\mathcal{I}}(\boldsymbol{\mu}, \boldsymbol{u})$
    $J_{nl}(\boldsymbol{\mu}, \boldsymbol{u}) = \Phi\Phi_{\mathcal{I}}^{-1}J_{nl,\mathcal{I}}$ ▷ Project reduced term to full dimension
    Reshape J_{nl} into matrix
  end if
  $J = J_l + J_{nl}$ ▷ Add linear and nonlinear Jacobian components
  Apply POD basis, construct Eq. (6)
  Solve Eq. (6) for $\Delta\boldsymbol{u}_{rb}$
  $\Delta\boldsymbol{u} = V\Delta\boldsymbol{u}_{rb}$ ▷ Compute approximate solution update
  $\boldsymbol{u} = \boldsymbol{u} + \Delta\boldsymbol{u}$
  Compute new residual vector $R(\boldsymbol{\mu}, \boldsymbol{u})$
  $i = i + 1$
end while

D. Online implementation

Combining the previous sections gives us the approach shown in Alg. 3 for using the POD-MDEIM RM in an optimization loop. We start the optimization algorithm by using the FM in the first p design iterations. We do this to gather data to construct the RM in lieu of using an offline training phase. In iteration $p + 1$ we construct the initial POD and DEIM projection operators, after which we are immediately able to start using the RM. We run the RM for each

subsequent design iteration and accept its solutions unless its convergence does not meet a predefined tolerance. If this is the case we run the FM for the same design point and accept its solution instead. While running the FM the solution vector and nonlinear Jacobian term of each Newton iteration are stored and used to update the POD and DEIM RMs. This can be done efficiently with the SVD approach that is covered in Sec. IV.E. With the greedy DEIM index selection approach in Alg. 1 we can simply extend the selected set of indices instead of computing from scratch a new index set with the updated DEIM basis. As far as we are aware this is the first use of MDEIM that does not have a straightforward offline-online decomposition.

Algorithm 3 Using the POD-MDEIM RM in an optimization loop

$i = 1$
$\boldsymbol{\mu} = \boldsymbol{\mu}_{initial}$
while $\boldsymbol{\mu}$ is not a local minimum **do**
    if $i = p + 1$ **then**
        Compute POD and DEIM bases, DEIM indices
    else if $i > p$ **then**
        Run RM at $\boldsymbol{\mu}$ with Alg. 2
    end if
    if $i < p$ or $||\boldsymbol{R}||_2 >$ tolerance **then**
        Run FM at $\boldsymbol{\mu}$
        Save solution vector $\boldsymbol{u}$, Jacobian term J_{nl} of each Newton iteration
        Recompute POD and DEIM RMs with new data
        Accept FM solution
    else
        Accept RM solution
    end if
    Compute $\boldsymbol{\mu}_{i+1}$ with gradient-based optimization algorithm
    $\boldsymbol{\mu} = \boldsymbol{\mu}_{i+1}$
    $i = i + 1$
end while

E. Singular Value Decomposition

Since the SVD is used in both our POD and MDEIM implementations we briefly cover it here separately. Computing the SVD of a matrix $X \in \mathbb{R}^{n \times s}$ results in the following matrix decomposition:

$$X = \Phi \Sigma \Psi^T \tag{7}$$

Where (in the case of the 'thin' SVD) $\Phi \in \mathbb{R}^{n \times s}$ and $\Psi, \Sigma \in \mathbb{R}^{s \times s}$. Standard algorithms for computing the SVD of a matrix are well-described in various textbooks [44, 45]. Σ contains the singular values of X on its diagonal, the magnitudes of which describe the relative importance of the modes represented by the columns of the orthonormal matrices Φ and Ψ. The POD and MDEIM bases are taken as the first columns of Φ.

While the aforementioned standard SVD algorithms work for many applications, alternative approaches exist to deal with particular challenges and structure that may arise from specific use-cases. Sirovich introduced the method of snapshots [46], in which the SVD of $X^T X \in \mathbb{R}^{s \times s}$ is computed. The left-singular matrix Φ of X can then be computed directly through various simple matrix products. With $s << n$ this can present significant cost savings for large n. This method is however sensitive to round-off errors due to the loss in precision when calculating $X^T X$. This is problematic for our intended application since the entries of the MDEIM snapshot matrix vary by several orders of magnitude, making them susceptible to round-off errors.

Another consideration for our current application is that we only obtain chunks of X at a time. Standard SVD algorithms would require us to recompute the complete SVD any time we append snapshots to X, which can become prohibitively expensive in terms of computing time and (since the full snapshot matrix must be stored) memory consumption. Several algorithms exist that make use of an existing Singular Value Decomposition when adding to

a snapshot matrix. Brand [47] proposed a widely-used algorithm for updating the SVD of a matrix when appending a single column. Peña and Sauer [48] extended Brand's approach to allow for appending blocks instead of single columns, in a way that is faster than successively adding individual columns. While the output of their algorithm doesn't contain the left-singular matrix its columns can be computed efficiently from a Householder matrix output by using the algorithm proposed by Kaufman [49]. This is the approach to efficiently computing updates to the SVD of the snapshot matrix that is used in this work. As far as we are aware this is the first MDEIM application in which such an updating SVD algorithm is used. This is likely due to applications from literature using a clear offline-online decomposition where all the data is available before computing SVDs.

V. Numerical experiments

We apply the POD-MDEIM RM approach that is covered in Sec. IV to the optimization of a square KL shell surface that is shown in Fig. 3. The shell is clamped along one edge and subject to a vertical distributed load along the opposite edge. We pose a constrained compliance minimization problem over the surface of the shell, with the spatially-varying shell thickness t as design variable:

$$\begin{aligned} \underset{0.01 \le t \le 0.2}{\text{minimize}} \quad & \frac{1}{2}\int_\Omega \boldsymbol{u}\cdot\boldsymbol{u}\; d\Omega + \frac{1}{2}h^2\alpha\int_\Omega \nabla t\cdot\nabla t\; d\Omega \\ \text{subject to} \quad & \mathcal{L}(t,\boldsymbol{u}) = \boldsymbol{f} \\ & \int_\Omega t\; d\Omega = C \end{aligned} \tag{8}$$

where $\mathcal{L}(t,\boldsymbol{u}) = \boldsymbol{f}$ is the nonlinear KL shell model with edge loading and $C \in \mathbb{R}$ is the required volume of the shell, which is computed by integrating the shell thickness over its surface. $\alpha = 0.05$ is a regularization parameter that stabilizes the optimization problem and h is the cell diameter. For non-uniform meshes h will vary spatially, in this example it is a constant value that scales with mesh size. To discretize the minimization problem we expand the shell thickness t in a linear NURBS basis. A 6×6 grid of quadrilateral IGA cells gives 49 design parameters. Note that the design parameters each affect only small parts of the domain, which makes this a challenging case for hyper-reduction methods. The KL shell model is run on this grid with cubic NURBS basis functions, which results in 81 degrees of freedom for each displacement components and hence FM linear systems with 243 rows and columns. The corresponding discontinuous Lagrange basis that we use to generate the IGA basis in FEniCS contains 6 048 basis functions: 84 per triangular FE cell, over a total of 72 such cells.

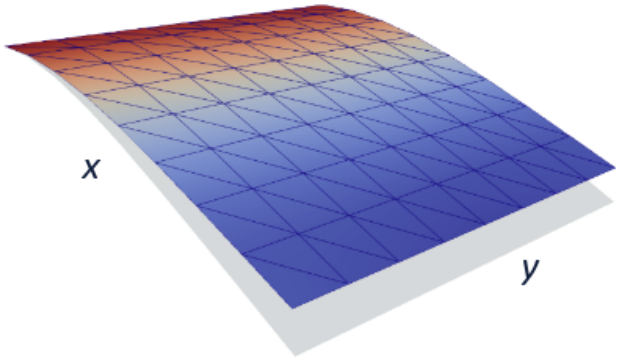

Fig. 3 Cantilever square KL shell under distributed load along the line $\{(x_{max}, y) : y \in [y_{min}, y_{max}]\}$

We use the Sequential Least-Squares Programming (SLSQP) optimization algorithm via OpenMDAO [50]. The FM needs 166 design iterations to converge to an optimal design with a tolerance of 10^{-14} and a nonlinear relative convergence tolerance of 10^{-5}. Next we use the RM in the exact same optimization loop. As was mentioned in section IV.D the first step of our proposed method consists in acquiring FM data to be able to build an effective RM in early iterations. We run the FM (with the aforementioned parameters) for the first 10 iterations, after which we start using the RM as in Alg. 3, deferring to the FM and collecting more data as needed. We use an absolute convergence

tolerance of $\epsilon = 5$ for the RM and Energy Ratios of $1 - 10^{-16}$ for both POD and MDEIM. Since the Jacobian matrices are symmetric we apply MDEIM to only the lower triangular part of the matrices. This reduces the problem size and makes the resulting approximate matrices more accurate, as we can strictly enforce their symmetry. With the RM, we arrive at an optimal design that is nearly indistinguishable to that of the FM; both of these are both shown in Fig. 4.

Tab. 1 summarizes the optimization results obtained with the FM and RM. As can be seen the optimal objective values found with both methods are within 1% of each other. This minor difference is likely due to the RM objective function being evaluated with the RM displacement solution $\boldsymbol{u}$, which is inherently an approximation to the FM displacement. While the RM converges to the optimal solution in fewer design iterations than the FM Fig. 5 shows that both models follow the same convergence behavior. Fig. 6 shows the 2-norm, L_2 and H_2 errors per PDE evaluation, each normalized with the corresponding norm of the FM solution. As can be seen each error norm varies within approximately one order of magnitude. The differences between the errors of subsequent evaluations decrease as the optimization algorithm converges towards an optimum and starts taking smaller steps in the design space. To summarize, using a RM for this optimization problem does not seem to negatively affect the optimization algorithm.

Next we look at the computation speed of the RM and the FM. We start by looking at the total wall time of each evaluation of the nonlinear KL shell model, which is shown in Fig. 7 for the FM, the RM and the FM simulations that are used to gather data for the RM. The data acquisition and storing steps incur wall time costs, which explains why these FM simulations take longer than the simulations from the 'pure FM' optimization run. A naive approach is currently used to store the FM data for the RM, and hence there is still the possibility of reducing this storage overhead further. The average wall time per PDE evaluation is around 1.5 seconds for the RM and 2.6 seconds for the FM.

Additionally we look at the wall time that is incurred per Newton iteration of the nonlinear PDE solution approach. In Fig. 8 these wall times have been binned per Newton iteration number for the FM and the RM. The cost of both models for iteration 0 is similar, since this is when the linear Jacobian term is used in the RM simulations. This term is assembled identically to the FM, and hence there is no significant RM speedup yet. In Newton iteration 1 both models again have similar wall times. While this is the first iteration where MDEIM is used to construct approximate nonlinear Jacobian terms no speedup is observed yet. This is because FEniCS compiles some of the MDEIM routines once per PDE evaluation. Subsequent RM Newton iterations are able to use the exact same approximate Jacobian construction routines and are about $2 - 3$ times faster than their respective FM counterparts. We note that most PDE evaluations with the RM have converged to within tolerance after the first two iterations, making further iterations unnecessary. Hence in this static test case most of the RM speedup relative to the FM is not being leveraged. In dynamic cases, the intended eventual applications of this RM approach, this would not be an issue. Lastly we note that speedups of this magnitude are in line with some of the examples shown with POD-MDEIM in literature [33–35], with the main difference being that in those works an abundance of data over low-dimensional parameter spaces is gathered during offline training phases.

Table 1 Test case optimization results

	FM	**RM**
Optimized objective function value	$1.5465 \cdot 10^{-5}$	$1.5328 \cdot 10^{-5}$
SLSQP iterations	166	132
Total PDE evaluations	166	143
FM evaluations	166	18
RM evaluations	–	125

VI. Conclusions

As mentioned earlier the aim of this work is to speed up the simulation of static nonlinear Kirchhoff—Love shell models in Isogeometric Analysis as a stepping stone towards speeding up dynamic multi-shell models, all with the application of speeding up Multidisciplinary Design Optimization problems. The main contribution of this work is a Reduced Model (RM) that uses Proper Orthogonal Decomposition to project high-dimensional linear(ized) systems to a

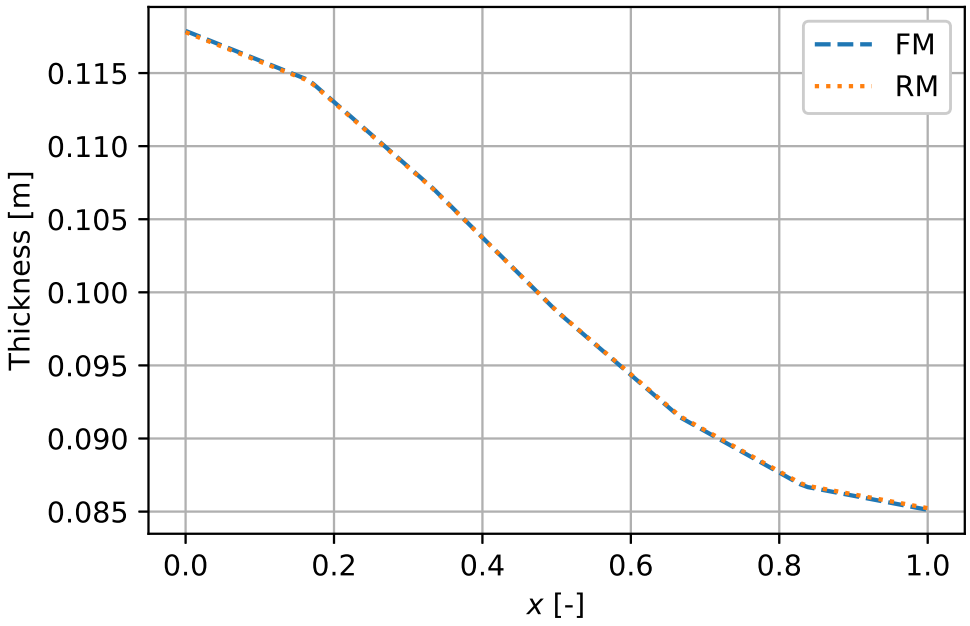

Fig. 4 Optimal solutions obtained with the FM and RM

reduced basis and the Matrix Discrete Empirical Interpolation Method (MDEIM) as hyper-reduction method to speed up construction of the nonlinear components of the Jacobian matrices that are used in Newton—Raphson iteration schemes. We use an 'unassembled' MDEIM approach that allows us to identify individual cells that should be numerically integrated on instead of the entire support of specific matrix entries. This makes the RM more amenable to higher-order basis functions, which can span many cells. An updating Singular Value Decomposition method is used to minimize the cost of updating the RM parameters when new high-fidelity data is made available. This is especially relevant since, in contrast to other works using similar approaches, we do not use an offline training phase to gather high-fidelity data and to construct the RM. The use of this RM approach for design optimization problems is a novel contribution and we will further explore its application to large-scale design optimization in future work.

The RM approach is applied to a test case of a thickness optimization of a single cantilevered nonlinear shell subject to a distributed load along its free edge. The thickness is discretized in a linear basis, resulting in 49 design parameters that cover the entire domain. It is shown for this test case that the reduced and high-fidelity methods both converge to near-identical design points along a similar trajectory. The RM does this by requiring only 12.6% of the design evaluations to be carried out with the high-fidelity model for data gathering purposes; all other design evaluations are handled exclusively by the aforementioned RM approach. Per design evaluation the RM takes an average of about 1.5 seconds whereas the high-fidelity model takes 2.6 seconds. Closer inspection of the wall time per Newton—Raphson iteration reveals that in this specific test case the full speedup potential of the RM compared to the high-fidelity model ($2-3$ times faster) is not being leveraged due to the RM converging to nonlinear solutions within two Newton—Raphson iterations. These speedups are in line with what is reported in literature with the main differences being the lack of offline data acquisition and use of a high-dimensional parameter space in this work.

While the focus of this specific work is on static nonlinear shells the eventual intended application is dynamic multi-shell models. The PENGoLINS framework [51] will be used to couple intersecting pairs of shells through a penalty approach. Thus an appropriate model reduction approach can be applied to these coupling terms separate from the MDEIM method that is used to accelerate nonlinear Jacobian construction. A dynamic multi-shell version of the current RM is intended to be used within larger MDO problems for the design of urban air mobility vehicles. Although extending the current approach to such coupled systems presents the majority of the planned future work, several possible improvements to the current implementation are still possible. As a prime example, the wall time of several RM computation steps can potentially be reduced by converting them to a numerical linear algebra toolbox like PETSc.

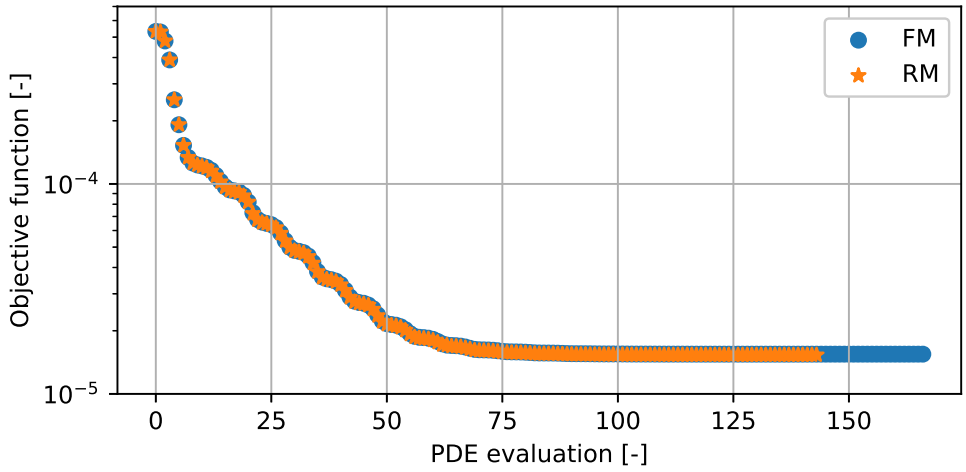

Fig. 5 Objective function values per PDE evaluation of the RM and FM

There are furthermore several ways in which the current MDEIM implementation can be improved for use in parametric problems. The RM wall time per Newton iteration scales directly with the fraction of mesh cells that are selected by MDEIM. MDEIM index selection is currently being done in an unbiased way; including a bias factor in that process to favor reusing information from already-selected cells can prove beneficial in minimizing RM wall time. Online adaptation of the MDEIM index selection step is another thing that has not been considered directly in this work but can prove to be useful for online RM deployment. Peherstorfer and Willcox give a possible approach for this in [52]. In a similar way a Gappy POD approach can be used to transform matrix interpolation into a regression problem. Not all available data of the MDEIM indices is currently being used, leveraging it in a regression problem could increase RM accuracy for negligible computational cost. Lastly we note that it could be beneficial to switch to a CUR decomposition-based version of MDEIM as a more efficient alternative to Singular Value Decompositions, especially for large systems, due to more efficient scaling of the CUR decomposition.

Acknowledgments

This research has been funded through NASA award No. 80NSSC21M0070.

References

[1] Jonsson, E., Riso, C., Lupp, C. A., Cesnik, C. E. S., Martins, J. R. R. A., and Epureanu, B. I., "Flutter and post-flutter constraints in aircraft design optimization," *Progress in Aerospace Sciences*, Vol. 109, 2019, p. 100537.

[2] Rajpal, D., Gillebaart, E., and De Breuker, R., "Preliminary aeroelastic design of composite wings subjected to critical gust loads," *Aerospace Science and Technology*, Vol. 85, 2019, pp. 96–112.

[3] Gray, A. C., Riso, C., Jonsson, E., Martins, J. R. R. A., and Cesnik, C. E. S., "High-Fidelity Aerostructural Optimization with a Geometrically Nonlinear Flutter Constraint," *AIAA Journal*, 2023. To appear.

[4] Lupp, C. A., and Cesnik, C. E. S., "A Gradient-Based Flutter Constraint Including Geometrically Nonlinear Deformations," *AIAA Scitech 2019 Forum*, 2019. AIAA 2019-1212.

[5] Xie, C., Meng, Y., Wang, F., and Wan, Z., "Aeroelastic Optimization Design for High-Aspect-Ratio Wings with Large Deformation," *Shock and Vibration*, Vol. 2017, 2017.

[6] Mallik, W., Kapania, R. K., and Schetz, J. A., "Effect of Flutter on the Multidisciplinary Design Optimization of Truss-Braced-Wing Aircraft," *Journal of Aircraft*, Vol. 52, No. 6, 2015, pp. 1858–1872.

[7] Variyar, A., Economon, T. D., and Alonso, J. J., "Design and Optimization of Unconventional Aircraft Configurations with Aeroelastic Constraints," *AIAA Scitech 2017 Forum*, 2017. AIAA 2017-0463.

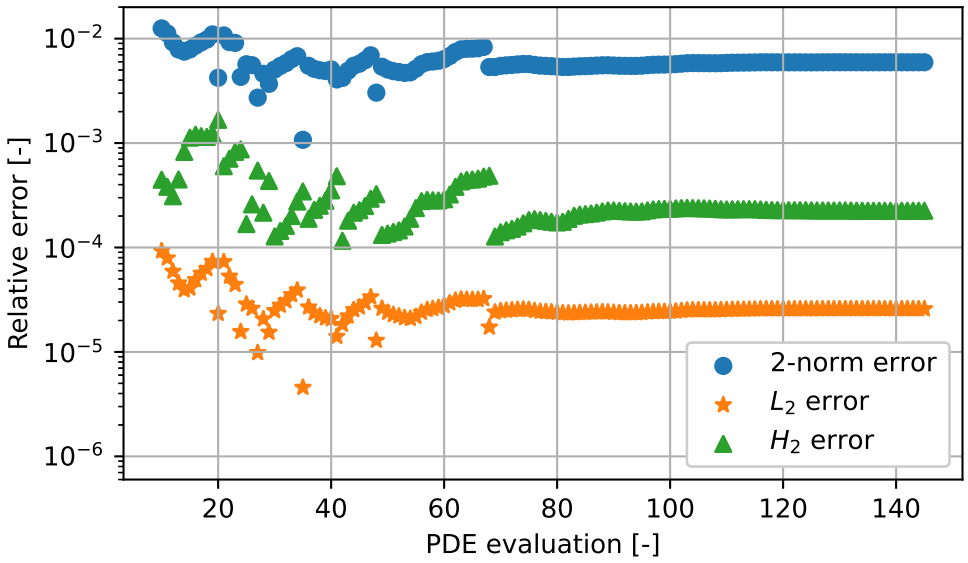

Fig. 6 Relative 2-norm error, L_2 error and H_2 error per PDE evaluation

[8] Kitson, R. C., Lupp, C., and Cesnik, C. E., "Modeling and Simulation of Flexible Jet Transport Aircraft with High-Aspect-Ratio Wings," *AIAA Scitech 2016 Forum*, 2016. AIAA 2016-2046.

[9] Li, W., Chen, Z., and Guo, Y., "Model order reduction for structural nonlinear dynamic analysis based on Isogeometric analysis," *Journal of Physics: Conference Series*, Vol. 2235, No. 1, 2022, p. 012073.

[10] Rinaldi, M., "Reduced Basis Method for Isogeometric Analysis: application to structural problems," Master's thesis, Politecnico di Milano, 2014.

[11] Hughes, T., Cottrell, J., and Bazilevs, Y., "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement," *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, No. 39, 2005, pp. 4135–4195.

[12] Cottrell, J., Hughes, T., and Bazilevs, Y., *Isogeometric Analysis: Toward integration of CAD and FEA*, John Wiley & Sons, Ltd., 2009.

[13] Hesthaven, J., Rozza, G., and Stamm, B., *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, SpringerBriefs in Mathematics, Springer International Publishing, 2015.

[14] Logg, A., Mardal, K.-A., Wells, G. N., et al., *Automated Solution of Differential Equations by the Finite Element Method*, Springer, 2012.

[15] Kamensky, D., and Bazilevs, Y., "tIGAr: Automating isogeometric analysis with FEniCS," *Computer Methods in Applied Mechanics and Engineering*, Vol. 344, 2019, pp. 477–498.

[16] Kiendl, J., Bletzinger, K.-U., Linhard, J., and Wüchner, R., "Isogeometric shell analysis with Kirchhoff–Love elements," *Computer Methods in Applied Mechanics and Engineering*, Vol. 198, No. 49, 2009, pp. 3902–3914.

[17] Schulte, J., Dittmann, M., Eugster, S., Hesch, S., Reinicke, T., Dell'Isola, F., and Hesch, C., "Isogeometric analysis of fiber reinforced composites using Kirchhoff-Love shell elements," *Computer Methods in Applied Mechanics and Engineering*, Vol. 362, 2020, p. 112845.

[18] Debabrata, A., Zhang, X., Rahul, G., Ritvik, V., Krishna, G., Padmini, R., and Shiva, R., "Biomembranes undergo complex, non-axisymmetric deformations governed by Kirchhoff–Love kinematics and revealed by a three-dimensional computational framework," *Proceedings of The Royal Society A*, Vol. 477, No. 2255, 2021.

[19] Martins, J. R. R. A., and Ning, A., *Engineering Design Optimization*, Cambridge University Press, 2021.

[20] Fink, J., and Rheinboldt, W., "On the Error Behavior of the Reduced Basis Technique for Nonlinear Finite Element Approximations," *Zamm-zeitschrift Fur Angewandte Mathematik Und Mechanik*, Vol. 63, 1983, pp. 21–28.

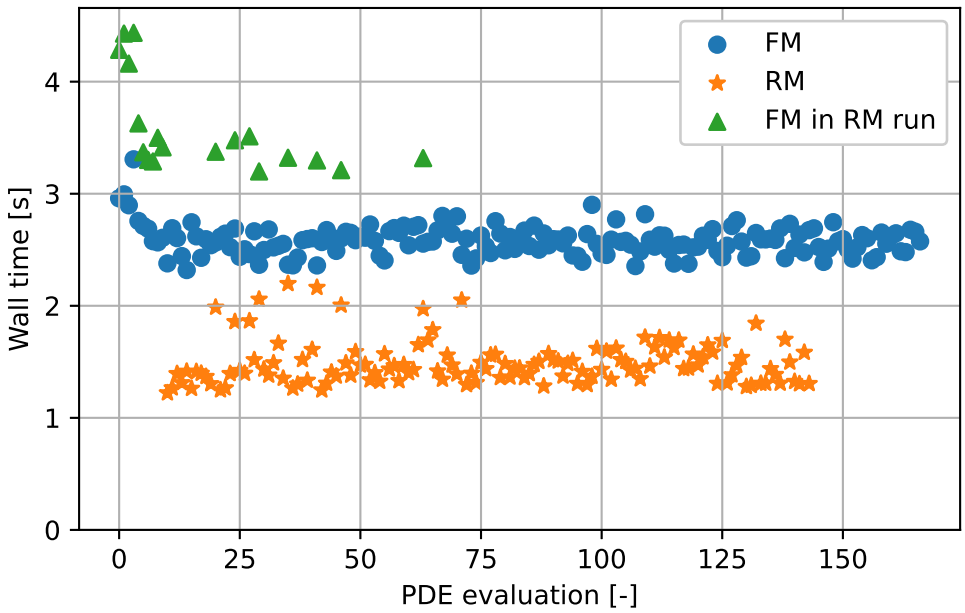

Fig. 7 Wall times per design iteration of the FM and RM

[21] Grepl, M., Maday, Y., Nguyen, N., and Patera, A., "Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations," *ESAIM: Mathematical Modelling and Numerical Analysis*, Vol. 41, No. 3, 2007, pp. 575–605.

[22] Grimberg, S., Farhat, C., Tezaur, R., and Bou-Mosleh, C., "Mesh sampling and weighting for the hyperreduction of nonlinear Petrov–Galerkin reduced-order models with local reduced-order bases," *International Journal for Numerical Methods in Engineering*, Vol. 122, No. 7, 2021, pp. 1846–1874.

[23] Astrid, P., Weiland, S., Willcox, K., and Backx, T., "Missing Point Estimation in Models Described by Proper Orthogonal Decomposition," *IEEE Transactions on Automatic Control*, Vol. 53, No. 10, 2008, pp. 2237–2251.

[24] Carlberg, K., Bou-Mosleh, C., and Farhat, C., "Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations," *International Journal for Numerical Methods in Engineering*, Vol. 86, No. 2, 2011, pp. 155–181.

[25] Carlberg, K., Farhat, C., Cortial, J., and Amsallem, D., "The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows," *Journal of Computational Physics*, Vol. 242, 2013, pp. 623–647.

[26] Everson, R., and Sirovich, L., "Karhunen–Loève procedure for gappy data," *J. Opt. Soc. Am. A*, Vol. 12, No. 8, 1995, pp. 1657–1664.

[27] Peherstorfer, B., Drmač, Z., and Gugercin, S., "Stability of Discrete Empirical Interpolation and Gappy Proper Orthogonal Decomposition with Randomized and Deterministic Sampling Points," *SIAM Journal on Scientific Computing*, Vol. 42, No. 5, 2020, pp. A2837–A2864.

[28] Chaturantabut, S., and Sorensen, D. C., "Nonlinear Model Reduction via Discrete Empirical Interpolation," *SIAM Journal on Scientific Computing*, Vol. 32, No. 5, 2010, pp. 2737–2764.

[29] Dimitriu, G., Ştefănescu, R., and Navon, I. M., "Comparative numerical analysis using reduced-order modeling strategies for nonlinear large-scale systems," *Journal of Computational and Applied Mathematics*, Vol. 310, 2017, pp. 32–43. Numerical Algorithms for Scientific and Engineering Applications.

[30] Brands, B., Davydov, D., Mergheim, J., and Steinmann, P., "Reduced-Order Modelling and Homogenisation in Magneto-Mechanics: A Numerical Comparison of Established Hyper-Reduction Methods," *Mathematical and Computational Applications*, Vol. 24, No. 1, 2019.

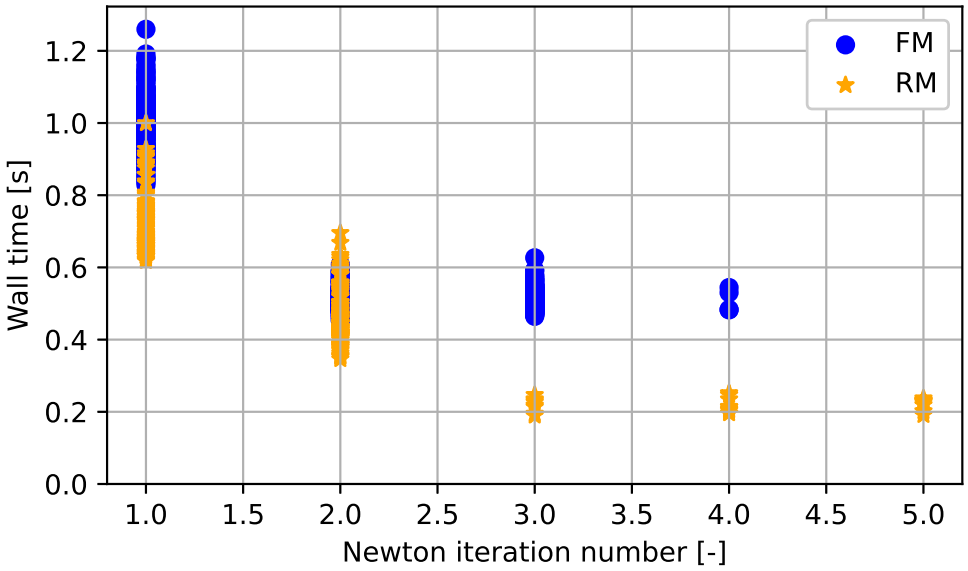

Fig. 8 Wall times per Newton iteration of the FM and RM

[31] Hesthaven, J. S., Pagliantini, C., and Rozza, G., "Reduced basis methods for time-dependent problems," *Acta Numerica*, Vol. 31, 2022, p. 265–345.

[32] Kim, Y., Kang, S.-H., Cho, H., and Shin, S., "Improved Nonlinear Analysis of a Propeller Blade Based on Hyper-Reduction," *AIAA Journal*, Vol. 60, No. 3, 2022, pp. 1909–1922.

[33] Negri, F., Manzoni, A., and Amsallem, D., "Efficient model reduction of parametrized systems by matrix discrete empirical interpolation," *Journal of Computational Physics*, Vol. 303, 2015, pp. 431–454.

[34] Cho, H., Shin, S., Kim, H., and Cho, M., "Enhanced model-order reduction approach via online adaptation for parametrized nonlinear structural problems," *Computational Mechanics*, Vol. 65, No. 2, 2020, pp. 331–353.

[35] Bonomi, D., Manzoni, A., and Quarteroni, A., "A matrix DEIM technique for model reduction of nonlinear parametrized problems in cardiac mechanics," *Computer Methods in Applied Mechanics and Engineering*, Vol. 324, 2017, pp. 300–326.

[36] Wirtz, D., Sorensen, D. C., and Haasdonk, B., "A Posteriori Error Estimation for DEIM Reduced Nonlinear Dynamical Systems," *SIAM Journal on Scientific Computing*, Vol. 36, No. 2, 2014, pp. A311–A338.

[37] Tiso, P., Dedden, R., and Rixen, D., "A Modified Discrete Empirical Interpolation Method for Reducing Non-Linear Structural Finite Element Models," *9th International Conference on Multibody Systems, Nonlinear Dynamics, and Control*, International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 7B, 2013.

[38] Drmač, Z., and Gugercin, S., "A New Selection Operator for the Discrete Empirical Interpolation Method—Improved A Priori Error Bound and Extensions," *SIAM Journal on Scientific Computing*, Vol. 38, No. 2, 2016, pp. A631–A648.

[39] Saibaba, A. K., "Randomized Discrete Empirical Interpolation Method for Nonlinear Model Reduction," *SIAM Journal on Scientific Computing*, Vol. 42, No. 3, 2020, pp. A1582–A1608.

[40] Hendryx, E. P., Rivière, B. M., and Rusin, C. G., "An extended DEIM algorithm for subset selection and class identification," *Machine Learning*, Vol. 110, 2021, pp. 621–650.

[41] Zimmermann, R., and Willcox, K., "An Accelerated Greedy Missing Point Estimation Procedure," *SIAM Journal on Scientific Computing*, Vol. 38, No. 5, 2016, pp. A2827–A2850.

[42] Zimmermann, R., Peherstorfer, B., and Willcox, K., "Geometric Subspace Updates with Applications to Online Adaptive Nonlinear Model Reduction," *SIAM Journal on Matrix Analysis and Applications*, Vol. 39, No. 1, 2018, pp. 234–261.

[43] Zhou, Y. B., "Model reduction for nonlinear dynamical systems with parametric uncertainties," Master's thesis, Massachusetts Institute of Technology, Dept. of Aeronautics and Astronautics, 2012.

[44] Trefethen, L., and Bau III, D., *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, 1997.

[45] Golub, G. H., and Van Loan, C. F., *Matrix Computations*, 4th ed., The Johns Hopkins University Press, 2013.

[46] Sirovich, L., "Turbulence and the dynamics of coherent structures. Part I: Coherent structures," *Quarterly of Applied Mathematics*, Vol. 45, No. 3, 1987, pp. 561–571.

[47] Brand, M., "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra and its Applications*, Vol. 415, No. 1, 2006, pp. 20–30. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.

[48] Peña, J. M., and Sauer, T., "SVD update methods for large matrices and applications," *Linear Algebra and its Applications*, Vol. 561, 2019, pp. 41–62.

[49] Kaufman, L., "Application of Dense Householder Transformation to a Sparse Matrix," *ACM Trans. Math. Softw.*, Vol. 5, No. 4, 1979, p. 442–450.

[50] Gray, J. S., Hwang, J. T., Martins, J. R. R. A., Moore, K. T., and Naylor, B. A., "OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization," *Structural and Multidisciplinary Optimization*, Vol. 59, No. 4, 2019, pp. 1075–1104.

[51] Zhao, H., Liu, X., Fletcher, A. H., Xiang, R., Hwang, J. T., and Kamensky, D., "An open-source framework for coupling non-matching isogeometric shells with application to aerospace structures," *Computers & Mathematics with Applications*, Vol. 111, 2022, pp. 109–123.

[52] Peherstorfer, B., and Willcox, K., "Online Adaptive Model Reduction for Nonlinear Systems via Low-Rank Updates," *SIAM Journal on Scientific Computing*, Vol. 37, No. 4, 2015, pp. A2123–A2150.