

An Adaptive POD-Krylov Reduced-Order Model for Structural Optimization

Kevin Carlberg, Charbel Farhat

Stanford University, Stanford, CA, 94305, USA.

Email: carlberg@stanford.edu, cfarhat@stanford.edu

1. Abstract

We present an adaptive proper orthogonal decomposition (POD)-Krylov reduced-order model (ROM) for structural optimization. At each step of the optimization loop, we compute approximate solutions to the structural state and sensitivity equations using a novel POD-augmented conjugate gradient (CG) algorithm. This algorithm consists of three stages. In the first two stages, the solution component in the POD subspace is computed using a CG algorithm. Here, fast convergence is ensured due to well-conditioned reduced equations. This property results from using an energy inner product to compute the POD basis. In the third stage, the solution is refined in an adaptively-computed Krylov subspace using an augmented preconditioned CG algorithm. The dimension of the Krylov subspace is increased until the prescribed tolerance is satisfied. This methodology can be considered a reduced-order modeling technique, as it efficiently computes approximate states and sensitivities in the sum of two subspaces. The ROM is well-suited for optimization settings because its accuracy is continually improved as the optimum is approached. We report on the benchmarking of the proposed method for a direct sensitivity analysis of a parameterized V-22 tiltrotor wing panel. The results highlight the ability of the proposed method to compute approximate solutions to the structural state and sensitivity equations at a significantly lower cost than the typical augmented preconditioned CG method.

2. Keywords: augmented Krylov subspace method, proper orthogonal decomposition, reduced-order model, surrogate-based optimization

3. Introduction

Optimization techniques have become indispensable tools for the design, damage detection, and control of structures. However, executing an optimization procedure can be prohibitively expensive, as it often incurs repeated analyses of a high-fidelity (e.g. detailed finite element) model of the structure in different configurations. Consequently, much attention has been given to decreasing the computational cost of solving structural optimization problems. In particular, two fields have made progress to this end using different approaches: surrogate-based optimization (SBO) and Krylov subspace recycling methods.

SBO algorithms employ a surrogate model of the structure in lieu of the high-fidelity model to execute the optimization. A surrogate model is typically required to both approximate the behavior of the high-fidelity model and be inexpensive to evaluate. Common choices include data fits of the high-fidelity response (e.g. Kriging, radial-basis functions) and lower-fidelity models (e.g. reduced-order models, lower element order). Surrogates are often constructed from high-fidelity simulations executed at reference configurations determined either from a design of experiments or from previous iterates of the SBO algorithm. In general, such models are accurate near these reference configurations; however, they often lack global accuracy. To this effect, basis updating and subspace/operator interpolation methods have been developed to improve the robustness of proper orthogonal decomposition (POD)-based [1][2] and modal decomposition-based [3] reduced-order surrogate models. Yet, such improvements do not guarantee global accuracy. As a result, SBO algorithms can be ineffective. Specifically, inaccurate surrogates usually result in small trust regions and incorrectly estimated minima, which lead to (many) costly high-fidelity simulations to correct the surrogate. This slows the convergence of SBO algorithms and limits their potential for cost savings. To mitigate these problems, a method to incrementally improve the surrogate (without re-evaluating the high-fidelity model) until it becomes sufficiently accurate would be advantageous. This idea can be refined by noting that surrogates should capture only trends in most regions of the design space, as high accuracy is important only near the optimum [4]. These observations suggest that the “optimal surrogate” satisfies the abstract problem

$$\underset{\text{available surrogates}}{\text{minimize}} \quad \alpha \times \text{error} + (1 - \alpha) \times \text{cost}, \quad (1)$$

where $\alpha \in [0, 1]$ and $\alpha \rightarrow 1$ as the optimum is approached.

Krylov subspace recycling methods have also been developed to decrease the cost of solving repeated analyses problems such as those arising in structural optimization. Such algorithms seek to accelerate convergence when solving multiple linear systems of equations. They accomplish this objective by reusing information generated during the solution of previous systems. In structural optimization, these linear systems arise from a finite element analysis at each optimization iteration and are characterized by non-invariant symmetric positive-definite (SPD) matrices. First, block Krylov [5] and successive right-hand side [6][7][8] methods were developed to treat multiple right-hand sides with an invariant matrix. These methods can be considered ‘augmented Krylov subspace methods’ [9] because they project the linear system onto the subspace

$$\mathcal{K} = \mathcal{K}_m + \mathcal{Y}. \quad (2)$$

Here, the standard Krylov subspace \mathcal{K}_m has been augmented with another subspace \mathcal{Y} , which in this case is the Krylov subspace generated during the solution for previous right-hand sides. These ideas were also extended to solve multiple systems with non-invariant matrices. First, approximate orthogonalization techniques and projection methods were introduced in [10][11] and [12][13], respectively. An efficient full orthogonalization method was then proposed in [14]. However, because the Krylov subspaces from previous systems can have relatively large dimensions, retaining the accumulation of the corresponding bases can be costly, particularly when the iterative solver is not based on a domain decomposition approach. Thus, truncation methods that retain a subset of the old Krylov vectors were investigated. First, deflation techniques for systems with invariant [15][16] and non-invariant [17][18] matrices were considered. For these methods, \mathcal{Y} corresponds to the approximated eigenvectors associated with the extreme eigenvalues of the governing matrices. Recently, a method was developed that computes \mathcal{Y} as the subspace of small dimension that most accurately represents the Krylov subspace in the orthogonalization step of conjugate gradient (CG) [19].

Existing Krylov subspace recycling techniques appear to be tailored to improve convergence toward the “exact” solution of a linear system of equations. As such, they are not necessarily the most efficient solvers within optimization loops, where computing solutions that are only *sufficiently accurate* (e.g. satisfy relaxed tolerances) in the sense of Eq.(1) often result in the fastest computational strategies.

In this paper, we present a method for accelerating the solution of structural optimization problems that combines concepts from surrogate modeling and Krylov subspace recycling. The approach efficiently computes solutions of the structural state and sensitivity equations to any prescribed tolerance. The proposed methodology relies on a novel POD-augmented CG algorithm where \mathcal{Y} in Eq.(2) is a POD basis that is computed to approximately minimize the solution error at the new configuration. Therefore, in contrast to other Krylov subspace recycling methods, this approach directly aims to efficiently compute solutions that satisfy (loose) tolerances appropriate for optimization. Additionally, the method can be combined with existing Krylov subspace recycling methods to further accelerate convergence. The proposed approach can be considered an adaptive POD-Krylov reduced-order modeling technique, as it computes approximations to the structural state and sensitivities that are contained in the sum of POD and Krylov subspaces. The reduced-order model (ROM) can be used within an optimization framework to generate significant cost savings, as it aims to satisfy the “optimal surrogate” problem stated in Eq.(1). That is, the tolerance satisfied by the approximation can be made more rigorous as the optimum is approached.

4. Structural optimization

Structural optimization problems can be mathematically formulated as

$$\begin{aligned} & \underset{\boldsymbol{\mu} \in \mathcal{D}}{\text{minimize}} && J(u(\boldsymbol{\mu}), \boldsymbol{\mu}) \\ & \text{subject to} && c_i(u(\boldsymbol{\mu}), \boldsymbol{\mu}) = 0, \quad 1 \leq i \leq n_{ec} \\ & && d_j(u(\boldsymbol{\mu}), \boldsymbol{\mu}) \geq 0, \quad 1 \leq j \leq n_{ic}. \end{aligned} \quad (3)$$

Here, $\mathcal{D} \subset \mathbb{R}^{n_\mu}$ is the parameter domain, $\boldsymbol{\mu} = (\mu_1, \dots, \mu_{n_\mu}) \in \mathcal{D}$ are the parameters, $J : \mathbb{R}^N \times \mathcal{D} \rightarrow \mathbb{R}$ is the objective function, and $c_i : \mathbb{R}^N \times \mathcal{D} \rightarrow \mathbb{R}$ and $d_j : \mathbb{R}^N \times \mathcal{D} \rightarrow \mathbb{R}$ represent equality and inequality constraints, respectively. The dependence of $u : \mathcal{D} \rightarrow \mathbb{R}^N$ on $\boldsymbol{\mu}$ is provided by the state equations

$$K(\boldsymbol{\mu})u(\boldsymbol{\mu}) = f(\boldsymbol{\mu}), \quad (4)$$

which in this work constitute the finite element discretization of the structural configuration. Hence, $K : \mathcal{D} \rightarrow \mathbb{R}^N \times \mathbb{R}^N$ is the SPD stiffness matrix, $f : \mathcal{D} \rightarrow \mathbb{R}^N$ is the force vector, and the state variables u represent the static displacement of the structure. When a sequential quadratic programming

(SQP) algorithm is employed to solve Eqs.(3), this is considered a reduced space Sequential Quadratic Programming (rSQP) formulation [20]. We assume N to be large.

In order to compute the total derivatives $\frac{dJ}{d\boldsymbol{\mu}}$, $\{\frac{dc_i}{d\boldsymbol{\mu}}\}_{i=1}^{n_{ec}}$, and $\{\frac{d(d_j)}{d\boldsymbol{\mu}}\}_{j=1}^{n_{ic}}$ required by the SQP algorithm, we must solve Eq.(4) and execute a sensitivity analysis (SA). The direct method for sensitivity analysis requires solving the direct sensitivity equations

$$K(\boldsymbol{\mu}) \frac{du}{d\mu_i}(\boldsymbol{\mu}) = \left. \frac{\partial f}{\partial \mu_i} \right|_{\boldsymbol{\mu}} - \left. \frac{\partial K}{\partial \mu_i} \right|_{\boldsymbol{\mu}} u(\boldsymbol{\mu}), \quad 1 \leq i \leq n_{\mu}. \quad (5)$$

The adjoint method for SA involves solving the adjoint sensitivity equations

$$K(\boldsymbol{\mu}) \psi_i(\boldsymbol{\mu}) = \left. \frac{\partial \gamma_i}{\partial u} \right|_{\boldsymbol{\mu}}^T, \quad 1 \leq i \leq 1 + n_{ec} + n_{ic}. \quad (6)$$

Here, $\psi_i : \mathcal{D} \rightarrow \mathbb{R}^N$ are adjoint solutions and $\gamma = \{J, c_1, \dots, c_{n_{ec}}, d_1, \dots, d_{n_{ic}}\}$. Either choice results in solving n_{RHS} systems of equations with an invariant matrix at each SQP iteration

$$K(\boldsymbol{\mu}) u_i(\boldsymbol{\mu}) = f_i(\boldsymbol{\mu}), \quad 1 \leq i \leq n_{\text{RHS}}, \quad (7)$$

where $n_{\text{RHS}} = 1 + n_{\mu}$ for direct SA and $n_{\text{RHS}} = 2 + n_{ec} + n_{ic}$ for adjoint SA. An augmented CG algorithm for systems with an invariant matrix can be used to solve Eqs.(7).

5. Augmented conjugate gradient background

This section provides a brief overview of augmented CG. Assume we are interested in solving $K(\bar{\boldsymbol{\mu}})u(\bar{\boldsymbol{\mu}}) = f(\bar{\boldsymbol{\mu}})$, where $\bar{\boldsymbol{\mu}}$ represents the ‘‘target’’ configuration. Augmented CG algorithms [9] employ a previously-computed basis $Y = [y_1, \dots, y_{n_y}]$ with $\mathcal{Y} \equiv \text{span}\{Y\}$ to accelerate convergence of solving this system when $K(\bar{\boldsymbol{\mu}})$ is SPD. This is done by computing a sequence of approximations \tilde{u}_k that satisfy

$$\tilde{u}_k = \arg \min_{\tilde{u} \in \mathcal{Y} + \mathcal{K}_k} \|u(\bar{\boldsymbol{\mu}}) - \tilde{u}\|_{K(\bar{\boldsymbol{\mu}})}, \quad (8)$$

where \mathcal{K}_k is the Krylov subspace of dimension k and $\|x\|_K \equiv \sqrt{x^T K x}$ is the energy norm defined from $(x, y)_K \equiv x^T K y$.

It can be shown [21] that any \tilde{u} satisfying $\tilde{u} = \arg \min_{\tilde{u} \in \mathcal{A}} \|u - \tilde{u}\|_K$ for a subspace $\mathcal{A} \subset \mathbb{R}^N$ can be expressed as the result of a projection process

$$\tilde{u} = \mathbf{P}_{\mathcal{A}}^K u, \quad (9)$$

where $\mathbf{P}_{\mathcal{A}}^K$ is the K -orthogonal projector onto \mathcal{A} . That is,

$$\mathbf{P}_{\mathcal{A}}^K u \in \mathcal{A}, \quad ((I - \mathbf{P}_{\mathcal{A}}^K) u, v)_K = 0 \quad \forall v \in \mathcal{A}. \quad (10)$$

Furthermore, \tilde{u} can be computed from a Galerkin projection of $Ku = f$ onto $\mathcal{A} = \text{span}\{A\}$ as

$$A^T K A \hat{u} = A^T f, \quad \tilde{u} = A \hat{u}. \quad (11)$$

Since $\tilde{u}_k = \mathbf{P}_{\mathcal{Y} + \mathcal{K}_k}^K u$ and we have no direct control of the Krylov subspace \mathcal{K}_k , we should choose \mathcal{Y} such that $\|u(\bar{\boldsymbol{\mu}}) - \tilde{u}_0\|_{K(\bar{\boldsymbol{\mu}})} = \|(I - \mathbf{P}_{\mathcal{Y}}^K)u(\bar{\boldsymbol{\mu}})\|_{K(\bar{\boldsymbol{\mu}})}$ is approximately minimized. Fortunately, POD enables this. In the following, we denote this quantity

$$e_{\mathcal{Y}}^K(u) \equiv \|(I - \mathbf{P}_{\mathcal{Y}}^K)u\|_K. \quad (12)$$

6. Proper orthogonal decomposition and optimal subspaces

Here, we present a methodology for choosing snapshots and weights to compute a POD basis that is aligned with an augmented CG algorithm. We also outline salient properties of the POD basis.

6.1. POD snapshot weighting and optimality

POD computes a basis that optimally represents a given set of snapshots in a certain sense. The snapshots typically correspond to state vector realizations of the system at various times, frequencies, or configurations. Since we intend to use the POD subspace as \mathcal{Y} in an augmented CG algorithm, the optimality of the POD subspace should be linked with minimizing $e_{\mathcal{Y}}^K(u(\bar{\boldsymbol{\mu}}))$.

Of course, the optimal subspace in this setting is $\mathcal{Y} = \text{span}\{u(\bar{\boldsymbol{\mu}})\}$, as this results in $e_{\mathcal{Y}}^{K(\bar{\boldsymbol{\mu}})}(u(\bar{\boldsymbol{\mu}})) = 0$. However, $u(\bar{\boldsymbol{\mu}})$ is not available, so this approach is not possible.

Instead, assume we have a collection of ‘‘snapshots’’ $\{w_i\}_{i=1}^{n_w} \in (\mathbb{R}^N)^{n_w}$ that provides information concerning the behavior of $u(\boldsymbol{\mu})$ in the parameter space \mathcal{D} . Using these snapshots, we can (crudely) estimate $u(\bar{\boldsymbol{\mu}})$ as $u_{\text{est}}(\bar{\boldsymbol{\mu}}) \approx u(\bar{\boldsymbol{\mu}})$ with $u_{\text{est}}(\bar{\boldsymbol{\mu}}) \in \text{span}\{w_i\}_{i=1}^{n_w}$, for example, via a response surface. Since this estimated solution is our best guess for the exact solution, we could simply use $\mathcal{Y} = \text{span}\{u_{\text{est}}(\bar{\boldsymbol{\mu}})\}$. However, it is unlikely that the one-dimensional subspace will significantly improve convergence. Thus, we seek a subspace that is of higher dimension (so we are less restrictive on the subspace containing \tilde{u}_0) and is still related to the minimization of $e_{\mathcal{Y}}^{K(\bar{\boldsymbol{\mu}})}(u(\bar{\boldsymbol{\mu}}))$.

Since $u_{\text{est}}(\bar{\boldsymbol{\mu}}) \in \text{span}\{w_i\}_{i=1}^{n_w}$, we can express the estimated solution as

$$u_{\text{est}}(\bar{\boldsymbol{\mu}}) = \sum_{i=1}^{n_w} \gamma_i w_i, \quad (13)$$

where $\gamma_i \in \mathbb{R}$, $1 \leq i \leq n_w$ are snapshot weights. We can now write

$$e_{\mathcal{A}}^{K(\bar{\boldsymbol{\mu}})}(u_{\text{est}}(\bar{\boldsymbol{\mu}})) = \left\| \left(I - \mathbf{P}_{\mathcal{A}}^{K(\bar{\boldsymbol{\mu}})} \right) \sum_{i=1}^{n_w} \gamma_i w_i \right\|_{K(\bar{\boldsymbol{\mu}})} \quad (14)$$

$$\leq n_w^{1/2} \sqrt{\sum_{i=1}^{n_w} \left\| \left(I - \mathbf{P}_{\mathcal{A}}^{K(\bar{\boldsymbol{\mu}})} \right) \gamma_i w_i \right\|_{K(\bar{\boldsymbol{\mu}})}^2}, \quad (15)$$

where the triangle inequality and the norm equivalence relation $\|x\|_1 \leq n^{1/2}\|x\|_2$ have been used, and $\mathcal{A} \subset \mathbb{R}^N$ is a subspace.

POD can be used to compute the subspace \mathcal{A} that minimizes the quantity in Eq.(15), which represents an upper bound for $e_{\mathcal{A}}^{K(\bar{\boldsymbol{\mu}})}(u_{\text{est}}(\bar{\boldsymbol{\mu}}))$. Namely, the POD subspace of dimension n_ϕ , $1 \leq n_\phi \leq n_w$ using weighted snapshots $\gamma_i w_i$, $1 \leq i \leq n_w$ and computed with the $K(\bar{\boldsymbol{\mu}})$ -inner product is $\mathcal{P}(n_\phi, [\gamma_1 w_1, \dots, \gamma_{n_w} w_{n_w}], K(\bar{\boldsymbol{\mu}}))$, which we define as

$$\mathcal{P}(n_\phi, [x_1, \dots, x_{n_w}], K) \equiv \arg \min_{\mathcal{A} \in \mathcal{G}(n_\phi, N)} \sqrt{\sum_{i=1}^{n_w} \left\| (I - \mathbf{P}_{\mathcal{A}}^K) x_i \right\|_K^2}, \quad (16)$$

where $\mathcal{G}(\alpha, N)$ is the set of α -dimensional linear subspaces of \mathbb{R}^N (the Grassman manifold [1]).

Fortunately, we can easily compute a basis $\Phi(n_\phi, W\Gamma, K(\bar{\boldsymbol{\mu}}))$ for the POD subspace $\mathcal{P}(n_\phi, W\Gamma, K(\bar{\boldsymbol{\mu}}))$ using the symmetric eigenvalue decomposition. Namely, we use the matrix $Y = W\Gamma = [w_1\gamma_1, \dots, w_{n_w}\gamma_{n_w}]$ in Algorithm 1, where we have defined $W \equiv [w_1, \dots, w_{n_w}]$ and $\Gamma \equiv \text{diag}(\gamma_i)$.

Algorithm 1 Computation of $\Phi(n_\phi, Y, K)$ given snapshots $Y \in \mathbb{R}^{N \times n_w}$ and K SPD

- 1: $\hat{K} \leftarrow Y^T K Y$
 - 2: Solve symmetric eigenvalue problem $\hat{K} = \Psi \Lambda \Psi^T$
 - 3: Choose size of truncated basis $n_\phi \in \{1, 2, \dots, n_w\}$.
 - 4: $\Phi(n_\phi, Y, K) = Y \left[\frac{1}{\sqrt{\lambda_1}} \psi_1, \dots, \frac{1}{\sqrt{\lambda_{n_\phi}}} \psi_{n_\phi} \right]$, where $\Psi = [\psi_1, \dots, \psi_{n_w}]$ and $\Lambda \equiv \text{diag}(\lambda_i)$.
-

We can use the eigenvalues Λ computed in Step 2 of the algorithm to compute the upper bound Eq.(15) minimized by the POD subspace:

$$n_w^{1/2} \sqrt{\sum_{i=1}^{n_w} \left\| \left(I - \mathbf{P}_{\mathcal{P}(n_\phi, W\Gamma, K(\bar{\boldsymbol{\mu}}))}^{K(\bar{\boldsymbol{\mu}})} \right) \gamma_i w_i \right\|_{K(\bar{\boldsymbol{\mu}})}^2} = n_w^{1/2} \sqrt{\sum_{k=n_\phi+1}^{n_w} \lambda_k}. \quad (17)$$

This error is often considered to be the snapshot ‘‘energy’’ omitted by the POD basis. Often, we set $n_\phi = n_{\mathcal{E}}(\Lambda, v)$ in Step 3 of Algorithm 1 so the relative energy retained by the POD basis is greater than

some threshold $v \in [0, 1]$, where we have defined

$$n_{\mathcal{E}}(\Lambda, v) \equiv \min_{n \in \mathcal{V}(\Lambda, v)} n, \quad (18)$$

$$\mathcal{V}(\Lambda, v) \equiv \{n \in \{1, 2, \dots, n_w\} \mid \sum_{i=1}^n \lambda_i / \sum_{i=1}^{n_w} \lambda_i \geq v^2\}.$$

6.2. POD properties

We now highlight several important properties of the POD basis. In the following, $\Phi \equiv [\phi_1, \dots, \phi_{n_w}] \equiv \Phi(n_w, W\Gamma, K(\bar{\boldsymbol{\mu}}))$ for notational simplicity.

Property 1: The POD vectors ϕ_i , $1 \leq i \leq n_w$ are orthonormal in the $K(\bar{\boldsymbol{\mu}})$ -inner product

$$(\phi_i, \phi_j)_{K(\bar{\boldsymbol{\mu}})} = \delta_{ij}, \quad 1 \leq i \leq n_w, \quad 1 \leq j \leq n_w, \quad (19)$$

Proof: $\Phi^T K(\bar{\boldsymbol{\mu}}) \Phi = \Lambda^{-1/2} \Psi^T W^T K(\bar{\boldsymbol{\mu}}) W \Psi \Lambda^{-1/2}$. Since $W^T K(\bar{\boldsymbol{\mu}}) W = \hat{K} = \Psi \Lambda \Psi^T$, this becomes $\Phi^T K(\bar{\boldsymbol{\mu}}) \Phi = \Lambda^{-1/2} \Psi^T \Psi \Lambda \Psi^T \Psi \Lambda^{-1/2}$. However, since $\hat{K} = \hat{K}^T$, its eigenvectors are orthonormal, so we have $\Psi^T \Psi = I$. This gives $\Phi^T K(\bar{\boldsymbol{\mu}}) \Phi = \Lambda^{-1/2} \Lambda \Lambda^{-1/2} = I$. \square

Property 2: The POD vectors ϕ_i , $1 \leq i \leq n_w$ are optimally ordered in the following sense:

$$\text{span}\{\phi_i, 1 \leq i \leq n_\phi\} = \mathcal{P}(n_\phi, W\Gamma, K(\bar{\boldsymbol{\mu}})). \quad (20)$$

The optimal subspace of any dimension $n_\phi \leq n_w$ in terms of minimizing the right-hand side of Eq.(15), which is an upper bound for $e_{\mathcal{A}}^{K(\bar{\boldsymbol{\mu}})}(u_{\text{est}}(\bar{\boldsymbol{\mu}}))$, is the space spanned by the first n_ϕ POD vectors. \square

Property 2 shows that by employing the weighting scheme in Eq.(13) and the $K(\bar{\boldsymbol{\mu}})$ -inner product to compute the basis, the optimality of the POD subspace is aligned with its role in an augmented CG algorithm.

7. Adaptive POD-Krylov reduced-order model

We now present an adaptive POD-Krylov model reduction method. We employ a novel POD-augmented CG algorithm that exploits Properties 1 and 2 to efficiently compute the approximation.

We present the three stages of the algorithm in sections 7.1–7.3, where the approximate solution to a single equation $K(\bar{\boldsymbol{\mu}})u(\bar{\boldsymbol{\mu}}) = f(\bar{\boldsymbol{\mu}})$ is considered. We assume that we have previously computed a POD basis $\Phi(n_w, W\Gamma, K(\boldsymbol{\mu}^*))$ and corresponding eigenvalues Λ , where $\boldsymbol{\mu}^*$ is near $\bar{\boldsymbol{\mu}}$ in the parameter space.* Section 7.4 considers the application of the algorithm to solving multiple systems of equations with an invariant matrix such as those arising from executing a sensitivity analysis for structural optimization. Section 7.5 outlines the implementation of this ROM within an optimization framework.

In the following, we denote the POD basis and subspace by $\Phi(n_\phi) \equiv \Phi(n_\phi, W\Gamma, K(\boldsymbol{\mu}^*))$ and $\mathcal{P}(n_\phi) \equiv \mathcal{P}(n_\phi, W\Gamma, K(\boldsymbol{\mu}^*))$, respectively for notational simplicity.

7.1. Stage 1: Direct solution of reduced equations

The goal of the first stage is to “kick-start” the algorithm by computing a fairly accurate solution at a very low cost. Recall from Property 2 that the POD vectors are optimally ordered. We therefore expect $\tilde{u}_1 = \mathbf{P}_{\mathcal{P}(n_1)}^{K(\bar{\boldsymbol{\mu}})} u(\bar{\boldsymbol{\mu}})$ with n_1 small (e.g. $n_1 = n_{\mathcal{E}}(\Lambda, 0.9)$) to be accurate and inexpensive to compute.

Since $\tilde{u}_1 = \mathbf{P}_{\mathcal{P}(n_1)}^{K(\bar{\boldsymbol{\mu}})} u(\bar{\boldsymbol{\mu}})$ satisfies

$$\Phi(n_1)^T K(\bar{\boldsymbol{\mu}}) \Phi(n_1) \hat{u} = \Phi(n_1)^T f(\bar{\boldsymbol{\mu}}), \quad (21)$$

$$\tilde{u}_1 = \Phi(n_1) \hat{u}, \quad (22)$$

Stage 1 consists of solving the reduced equations Eqs.(21)–(22) by a direct method. To do this, we employ $X_1 = \Phi(n_1)$, $X_0 = \emptyset$, and $R_0 = \emptyset$ in Algorithm 2.

7.2. Stage 2: Augmented conjugate gradient with reduced equations

In the second stage, we compute the approximate solution over the POD subspace $\mathcal{P}(n_2)$ with $n_2 > n_1$. The corresponding reduced equations are

$$\Phi(n_2)^T K(\bar{\boldsymbol{\mu}}) \Phi(n_2) \hat{u} = \Phi(n_2)^T f(\bar{\boldsymbol{\mu}}), \quad (23)$$

$$\tilde{u}_2 = \Phi(n_2) \hat{u}. \quad (24)$$

* $K(\boldsymbol{\mu}^*)$ is used to compute the POD basis instead of $K(\bar{\boldsymbol{\mu}})$ for efficiency reasons (see 3–4 of Section 7.5).

If n_2 is large (e.g. $n_2 = n_{\mathcal{E}}(\Lambda, 0.999)$), it is expensive to compute the reduced matrix $\Phi(n_2)^T K(\bar{\boldsymbol{\mu}})\Phi(n_2)$, as it requires n_2 matrix-vector products and $(n_2^2 + n)/2$ dot products. Since N is large, operations that scale with N (especially matrix-vector products) should be avoided.

To decrease the number of these operations, we employ an augmented CG algorithm to solve Eqs.(23)–(24), since it requires only a single matrix-vector product at each iteration. However, the algorithm must converge in only a few iterations to achieve cost savings. This is typically accomplished by employing a preconditioner. In this case, however, computing a preconditioner is impractical, as it requires forming the reduced matrix $\Phi(n_2)^T K(\bar{\boldsymbol{\mu}})\Phi(n_2)$, which is exactly what we seek to avoid.

Recall from Property 1 that $\Phi(n_2)^T K(\boldsymbol{\mu}^*)\Phi(n_2) = I$. If $\boldsymbol{\mu}^*$ is close to the target configuration $\bar{\boldsymbol{\mu}}$ in the parameter space, we expect the matrices $K(\boldsymbol{\mu}^*)$ and $K(\bar{\boldsymbol{\mu}})$ to have a similar eigenstructure, so

$$\Phi(n_2)^T K(\bar{\boldsymbol{\mu}})\Phi(n_2) \approx I. \quad (25)$$

This implies that Eq.(23) is well-conditioned, so a preconditioner is unnecessary and CG will converge quickly. This should lead to significant cost savings over using a direct method to solve Eqs.(23)–(24).[†] Stage 2 consists of executing Algorithm 3 with quantities computed in Stage 1 as inputs: \tilde{u}_1 , X_1 , V_1 , R_1 , and $X_2 = [\phi_{n_1+1}, \dots, \phi_{n_2}]$.

Remark: Algorithm 3 orthogonalizes the search direction against all previous ones, so it is actually a full orthogonalization method [21]. This is necessary to ensure $K(\bar{\boldsymbol{\mu}})$ -conjugacy of the search directions. \square

7.3. Stage 3: Augmented preconditioned conjugate gradient with full equations

Stage 3 refines the approximation using an augmented PCG algorithm on the full equations $K(\bar{\boldsymbol{\mu}})u(\bar{\boldsymbol{\mu}}) = f(\bar{\boldsymbol{\mu}})$ (e.g. [16], Algorithm 3.6) with $\mathcal{Y} = \mathcal{P}(n_1) + \text{span}\{P_2\}$. Here $P_2 = [p^{(1)}, \dots, p^{(K_2)}]$ are the search directions generated in Stage 2. The algorithm should iterate until the approximation \tilde{u}_3 satisfies a prescribed tolerance ϵ , with $\epsilon \equiv \|f(\bar{\boldsymbol{\mu}}) - K(\bar{\boldsymbol{\mu}})\tilde{u}_3\|/\|f(\bar{\boldsymbol{\mu}})\|$. By searching in the Krylov subspace until convergence, this stage enables the approximation to satisfy any accuracy requirement.

7.4. Treatment of systems with multiple right-hand sides

We can extend this framework to solve multiple systems with an invariant matrix as in Eqs.(7). In structural optimization, each solution $u_i(\boldsymbol{\mu})$, $1 \leq i \leq n_{\text{RHS}}$ has a unique interpretation (i.e. state, sensitivity, or adjoint solution). We thus compute a *separate* POD basis $\Phi(n_w, W\Gamma_i, K(\boldsymbol{\mu}^*))$ with eigenvalues Λ_i for each of the n_{RHS} systems. We choose weights $\Gamma_i = \text{diag}(\gamma_{ij})$ to satisfy $u_i(\boldsymbol{\mu}^*) \approx u_{i,\text{est}}(\boldsymbol{\mu}^*) = \sum_{j=1}^{n_w} \gamma_{ij} w_j$.

Given this set of n_{RHS} POD bases, we can approximately solve Eqs.(7) using Algorithm 4. This is an augmented CG algorithm where \mathcal{Y} is the sum of the current POD subspace and the space spanned by all previous search directions. We denote $\Phi_i(n_w) \equiv \Phi(n_w, W\Gamma_i, K(\boldsymbol{\mu}^*))$ for simplicity.

Algorithm 2 Stage 1

- 1: Given X_1 , X_0 , R_0
 - 2: $V_1 = K(\bar{\boldsymbol{\mu}})X_1$, $\hat{K} = X_1^T V_1$
 - 3: $R_1 = \begin{bmatrix} R_0 & X_0^T V_1 \\ 0 & \hat{K} \end{bmatrix}$
 - 4: Use pivoted Cholesky ([22], p. 149) to make R_1 upper triangular.
Pivot (and remove if rank deficiency is detected) columns of X_1 and V_1 accordingly.
 - 5: $X_1 \leftarrow [X_0, X_1]$, $V_1 \leftarrow [V_0, V_1]$
 - 6: Solve $R_1^T R_1 \hat{u} = X_1^T f(\bar{\boldsymbol{\mu}})$, $\tilde{u}_1 = X_1 \hat{u}$
 - 7: **return** \tilde{u}_1 , X_1 , V_1 , R_1
-

7.5. Integration with optimization algorithms

To implement the adaptive ROM within an optimization setting, we propose using a standard rSQP algorithm, where Eq.(4) and Eq.(5) or Eq.(6) are approximately solved using Algorithm 4. The tolerance ϵ in Stage 3 should decrease as the optimum is approached according to Eq.(1).

As the optimization algorithm progresses, snapshots are gradually accumulated. This results in four natural phases (and algorithmic variations) of the framework that depend on the current optimization

[†]We note that a similar idea was explored in [14], where the $K(\boldsymbol{\mu}^*)$ -conjugacy of a set of search directions was exploited.

Algorithm 3 Stage 2

- 1: Given $\tilde{u}_1, X_1, V_1, R_1, X_2$
- 2: $\tilde{u}^{(0)} = \tilde{u}_1, \hat{r}_1^{(0)} = 0, \hat{r}_2^{(0)} = X_2^T (f(\bar{\mu}) - K(\bar{\mu})\tilde{u}^{(1)}), k = 0$
- 3: **while** $\|\hat{r}_2^{(k)}\|/\|\hat{r}_2^{(0)}\| < \delta$ **do**
- 4: $k \leftarrow k + 1$
- 5: $\hat{p}_2^{(k)} = \hat{r}_2^{(k-1)}$
- 6: Orthogonalize against X_1 by solving $R_1^T R_1 \hat{p}_1^{(k-1)} = - (V_1)^T (X_2 \hat{p}_2^{(k-1)})$
- 7: $\hat{p}^{(k)} = [\hat{p}_1^{(k)T}, \hat{p}_2^{(k)T}]^T$
- 8: **for** $j = 1, \dots, k$ **do**
- 9: $\hat{p}^{(k)} = \hat{p}^{(k)} - \hat{p}^{(j)} (\sigma^{(j)} z^{(j)T} \hat{p}_2^{(k)})$
- 10: **end for**
- 11: $p^{(k)} = [X_1, X_2] \hat{p}^{(k)}, v^{(k)} = K(\bar{\mu}) p^{(k)}$
- 12: $\sigma^{(k)} = 1 / (p^{(k)T} v^{(k)}), z^{(k)} = X_2^T v^{(k)}, \alpha^{(k)} = \sigma^{(k)} (\hat{r}_2^{(k-1)T} \hat{p}^{(k)})$
- 13: $\tilde{u}^{(k)} = \tilde{u}^{(k-1)} + \alpha^{(k)} p^{(k)}, \hat{r}_2^{(k)} = \hat{r}_2^{(k-1)} - \alpha^{(k)} z^{(k)}$
- 14: **end while**
- 15: **return** $K_2 = k, \tilde{u}_2 = \tilde{u}^{(K_2)}, P_2 = [p^{(1)}, \dots, p^{(K_2)}], V_2 = [v^{(1)}, \dots, v^{(K_2)}], \Sigma_2 = \text{diag}(\sigma^{(k)})$

Algorithm 4 Solving systems with multiple right-hand sides

- 1: Given $\Phi_i (n_w,)$ and $\Lambda_i, i = 1, \dots, n_{\text{RHS}}$
- 2: $X_0 = \emptyset, R_0 = \emptyset$
- 3: **for** $i = 1, \dots, n_{\text{RHS}}$ **do**
- 4: $f(\bar{\mu}) = f_i(\bar{\mu})$
- 5: $P_2 = \emptyset, V_2 = \emptyset, P_3 = \emptyset, V_3 = \emptyset$
- 6: Compute $n_1 = n_{\mathcal{E}}(\Lambda_i, 0.9), n_2 = n_{\mathcal{E}}(\Lambda_i, 0.999)$
- 7: $X_1 = \Phi_i (n_1), X_2 = [\phi_{i, n_1+1}, \dots, \phi_{i, n_2}]$
- 8: Stage 1: execute Algorithm 2
- 9: Stage 2: execute Algorithm 3
- 10: Stage 3: execute an augmented PCG algorithm with $\mathcal{Y} = \text{span}\{[X_1, P_2]\}$. P_3 are the search directions generated during this stage, with $V_3 = K(\bar{\mu}) P_3$ and $\Sigma_3 = \text{diag}(1/(p_{3,i}^T K(\bar{\mu}) p_{3,i}))$.
- 11: $X_0 \leftarrow [X_1, P_2, P_3], V_0 \leftarrow [V_1, V_2, V_3], R_0 \leftarrow \text{diag}(R_1, \Sigma_2^{-1/2}, \Sigma_3^{-1/2})$
- 12: **end for**

iteration n_{it} . $n_{\text{it,POD}}$ indicates the maximum number of optimization iterations before the snapshots should be truncated using POD.

1. $1 \leq n_{\text{it}} < n_{\text{it,POD}}$. POD bases are not required since the number of snapshots is small. Use $X_1 = W$ in line 7 of Algorithm 4 for $i = 1$.
2. $n_{\text{it}} = n_{\text{it,POD}}$. Same as above, but compute POD bases at the end. Reduced matrices \hat{K} for Algorithm 1 have been already computed in Stage 1, so this is inexpensive.
3. $n_{\text{it}} = n_{\text{it,POD}} + 1$. Employ the POD bases computed at the previous iteration as presented.
4. $n_{\text{it,POD}} + 1 < n_{\text{it}} < 2n_{\text{it,POD}}$. Augment the POD bases with recently captured snapshots W . Set $X_1 = [W, \Phi_1(n_1)]$ in line 2 of Algorithm 4 for $i = 1$. When $n_{\text{it}} = 2n_{\text{it,POD}}$, return to Step 2.

Since additional (linearly independent) snapshots are generated at each optimization iteration, we expect the approximation provided by the POD basis to continually improve. This should lead to cheaper evaluation of the adaptive ROM as the optimization algorithm progresses.

8. Numerical experiments

Here, we apply the adaptive POD-Krylov ROM to the analysis of a parameterized stiffened wing panel from the V-22 tiltrotor aircraft [23]. We use a finite element model of the panel consisting of 9,486 nodes and 18,272 triangular shell elements with 6 degrees of freedom per node, giving $N = 56,916$ total degrees of freedom. The panel is clamped at one end, and a uniform shortening of 0.225 inches is applied to the

other end. We parameterize the model with $n_\mu = 13$ design variables: 8 material properties and 5 shape parameters as shown in Figure 1. Table 1 lists the parameters and their bounds. Here, E_i are moduli of

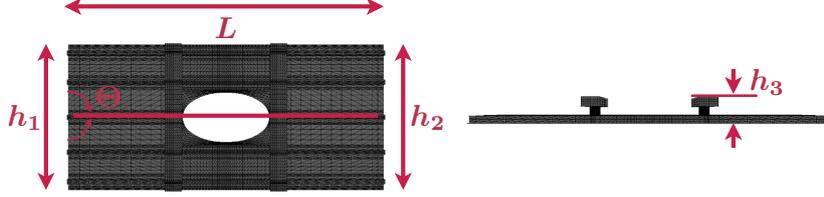


Figure 1: Shape parameters

i	1	2	3	4	5	6	7	8	9	10	11	12	13
Definition	h_1	h_2	h_3	L	$\tan \Theta$	E_1	E_2	E_3	E_4	t_1	t_2	t_3	t_4
$\mu_{i,lb}$	25	25	0.5	96	0	5.6	5.6	5.6	5.6	0.05	0.15	0.15	0.10
$\mu_{i,ub}$	45	45	10.5	56	0.4	15.6	15.6	15.6	15.6	0.15	0.35	0.35	0.40

Table 1: Design parameters μ_i and bounds. Distances given in inches, moduli of elasticity given in MSI.

elasticity and t_i are skin thicknesses. The material property subscripts in Table 1 indicate regions of the panel: region 1 is the basic skin, region 2 is the padded skin, region 3 is the skin around the access hole, and region 4 consists of the vertical stiffeners and attachment struts (see [23] for details). Note that h_1 and h_2 define the lengths of each side separately, allowing the panel to take on a wedged shape.

To benchmark the proposed method, we compare its performance with that of a variant of the augmented PCG algorithm proposed in [7] for systems with an invariant matrix. The variant considered here differs in the preconditioner (see below) and in the fact that it is a global rather than domain-decomposed approach. The test problem consists of computing approximate solutions to the state and direct sensitivity equations Eqs.(4)–(5). For the adaptive ROM, we first collect a set of snapshots corresponding to state vectors $u(\boldsymbol{\mu})$ and sensitivities $\frac{du}{d\mu_i}|_{\boldsymbol{\mu}}$, $1 \leq i \leq n_\mu$ for $n_{\text{ref}} = 10$ randomly-chosen reference configurations $\mathcal{D}_r = \{\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^{n_{\text{ref}}}\} \subset \mathcal{D}$. This results in $n_w = 140$ snapshots. Then, at two randomly-chosen target configurations $\bar{\boldsymbol{\mu}}^1$ and $\bar{\boldsymbol{\mu}}^2$, both the adaptive ROM and augmented PCG algorithm are used to solve the equations to a tolerance $\epsilon = 10^{-2}$.

For the adaptive ROM, a unique POD basis is computed for each right-hand side $\Phi(n_w, \text{WT}_i, K(\boldsymbol{\mu}^*))$, $i = 1, \dots, n_\mu + 1$, where $\boldsymbol{\mu}^* \in \mathcal{D}_r$. The weights Γ_i are chosen to estimate the i^{th} solution at the appropriate target $\bar{\boldsymbol{\mu}}$ according to Eq.(13). We use a Compact POD basis of order 1 [24] to approximately solve Eq.(4). Thus, Γ_1 corresponds to Taylor series and configuration radius weights. When solving the i^{th} of Eqs.(5), we use only snapshots corresponding to $\frac{du}{d\mu_i}|_{\boldsymbol{\mu}}$, $\boldsymbol{\mu} \in \mathcal{D}_r$ to build the POD basis. The weight given to the snapshot $\frac{du}{d\mu_i}|_{\boldsymbol{\mu}^k}$ corresponds to a Gaussian radial basis function

$$\gamma^{\text{RBF}}(\bar{\boldsymbol{\mu}}, \boldsymbol{\mu}^k) \equiv \exp(-\delta(\bar{\boldsymbol{\mu}}, \boldsymbol{\mu}^k)^2), \quad \delta(\bar{\boldsymbol{\mu}}, \boldsymbol{\mu}^k) \equiv \sqrt{\sum_{j=1}^{n_\mu} \left(\frac{\bar{\mu}_j - \mu_j^k}{\mu_{j,ub} - \mu_{j,lb}} \right)^2}. \quad (26)$$

This allows snapshots computed near the target to be better represented by the POD basis.

An incomplete Cholesky preconditioner with a drop tolerance of 2.5×10^{-5} is used for all PCG iterations.

Table 2 contains the results, where $\delta_{\min}(\bar{\boldsymbol{\mu}}, \mathcal{D}_r) \equiv \min_{\boldsymbol{\mu} \in \mathcal{D}_r} \delta(\bar{\boldsymbol{\mu}}, \boldsymbol{\mu})$ and $\bar{\delta}(\bar{\boldsymbol{\mu}}, \mathcal{D}_r) \equiv \frac{1}{n_{\text{ref}}} \sum_{k=1}^{n_{\text{ref}}} \delta(\bar{\boldsymbol{\mu}}, \boldsymbol{\mu}^k)$ indicate the proximity of the target configuration to the reference configurations. Figure 2 shows the convergence when solving the first system Eq.(4) to a tolerance $\epsilon = 10^{-2}$ with both methods.

The results indicate that the adaptive ROM computes approximate solutions at a significantly lower cost than the considered augmented PCG. Moreover, this is accomplished by exploiting data computed from only ten reference configurations in a 14-dimensional parameter space. Additionally, these reference configurations are not close to the new configurations, as indicated by the proximity metrics. When only the state equations are solved ($n_{\text{RHS}} = 1$), the relative cost of the proposed method is as low as 13.7% compared to PCG. When a direct sensitivity analysis is executed ($n_{\text{RHS}} = 14$), the cost of the new method as measured in flops is less than 60% that of the augmented PCG method. As shown in Figure

2, Stages 1 and 2 are effective at quickly reducing the error, while Stage 3 is required to decrease the error to the specified tolerance $\epsilon = 10^{-2}$.

Target configuration $\bar{\mu}$	$\delta_{\min}(\bar{\mu}, \mathcal{D}_r)$	$\bar{\delta}(\bar{\mu}, \mathcal{D}_r)$	$\delta(\bar{\mu}, \mu^*)$	Relative cost for $n_{\text{RHS}} = 1$		Relative cost for $n_{\text{RHS}} = 14$	
				flops	mat-vec	flops	mat-vec
$\bar{\mu}^1$	1.09	1.42	1.66	0.429	0.514	0.561	0.750
$\bar{\mu}^2$	1.19	1.46	1.19	0.137	0.217	0.584	0.752

Table 2: Results. δ_{\min} , $\bar{\delta}$, and δ are proximity metrics. Relative cost is the relative number of flops or matrix-vector products required by the adaptive ROM compared with standard PCG.

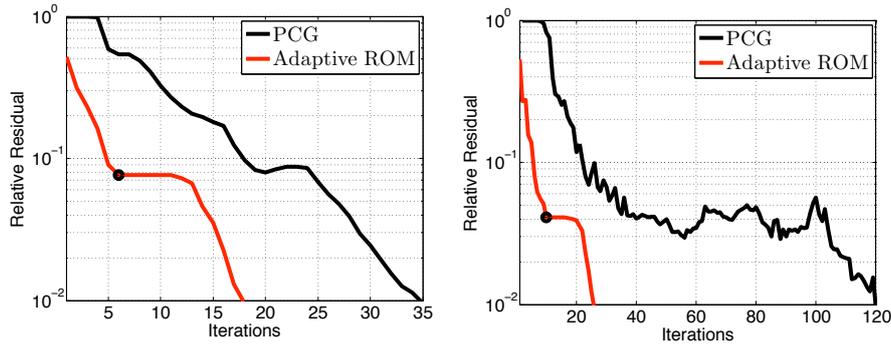


Figure 2: Convergence of the relative residual for the first RHS at $\bar{\mu}^1$ and $\bar{\mu}^2$, respectively. • end Stage 2

9. Conclusions and future work

We have introduced an adaptive POD-Krylov reduced-order model for efficiently computing approximate solutions at any prescribed accuracy. We presented a novel POD-augmented conjugate gradient algorithm to compute the approximate solution. This algorithm exploits a POD basis constructed using a snapshot weighting scheme and inner-product that align the optimality of the POD with its role in an augmented CG algorithm and result in well-conditioned reduced equations. The proposed reduced-order model can be integrated with a structural optimization algorithm as an “optimal surrogate,” since its accuracy can be increased as the optimum is approached. Numerical results indicate that the proposed method generates accurate solutions at a much lower cost than typical augmented PCG algorithms.

Future work includes implementing the reduced-order model within an optimization algorithm, combining the method with other Krylov subspace recycling techniques (e.g. deflation), and extending the methodology to non-SPD systems (e.g. by a POD-augmented CGNE/GMRES algorithm).

10. Acknowledgments

The authors thank Kurt Maute for providing SDESIGN to define $\bar{\mu}$ shape parameters. Kevin Carlberg acknowledges the consultation of Julien Cortial and David Amsallem, as well as the financial support of the Department of Defense NDSEG fellowship and the NSF Graduate Research Fellowship.

11. References

- [1] D. Amsallem and C. Farhat, An Interpolation Method for Adapting Reduced-Order Models and Application to Aeroelasticity, *AIAA Journal*, 46 (7), 1803–1813.
- [2] G. Weickum, M. Eldred and K. Maute, Multi-point Extended Reduced Order Modeling For Design Optimization and Uncertainty Analysis, *AIAA Paper 2006-2145*, 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Newport, RI, May 1–4, 2006.
- [3] D. Amsallem, J. Cortial, K. Carlberg and C. Farhat, An on-line method for interpolating linear reduced-order structural dynamics models, *International Journal for Numerical Methods in Engineering*, 2009, *submitted*.

- [4] N. Queipo, R. Haftka, W. Skyy, T. Goel, R. Vaidyanathan and P.K. Tucker, Surrogate-based analysis and optimization, *Progress in Aerospace Sciences*, 41 (1), 1–28, 2005.
- [5] D. O’Leary, The block conjugate gradient algorithm and related methods, *Linear Algebra and its Applications*, 29, 1980, 293–322.
- [6] Y. Saad, On the Lanczos method for solving symmetric linear systems with several right-hand sides, *Mathematics of Computation*, 651–662, 1987.
- [7] C. Farhat, L. Crivelli and F.X. Roux, Extending substructure based iterative solvers to multiple load and repeated analyses, *Computer Methods in Applied Mechanics and Engineering*, 117, 195–209, 1994.
- [8] J. Erhel and F. Guyomarc’h, An augmented conjugate gradient method for solving consecutive symmetric positive definite linear systems, *SIAM Journal on Matrix Analysis and Applications*, 21 (4), 1279–1299, 2000.
- [9] Y. Saad, Analysis of augmented Krylov subspace methods, *SIAM Journal on Matrix Analysis and Applications*, 18 (2), 435–449, 1997.
- [10] C. Rey, Developpement d’algorithmes paralleles de resolution en calcul non-lineaire de structures heterogenes: application au cas d’une butee acier-elastomere, *Ph.D. thesis*, Laboratoire de Modelisation et Mecanique des Structures, University of Paris VI, December 1994.
- [11] F.X. Roux, Parallel implementation of a domain decomposition method for non-linear elasticity, *Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering*, D. Keyes, Y. Saad and D.G. Truhlar (Eds.), SIAM, 161–175, 1995.
- [12] C. Farhat, Optimizing Substructuring Methods for Repeated Right Hand Sides, Scalable Parallel Coarse Solvers, and Global/Local Analysis, *Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering*, D. Keyes, Y. Saad and D.G. Truhlar (Eds.), SIAM, 161–175, 1995.
- [13] C. Farhat, K. Pierson and M. Lesoinne, The Second Generation of FETI Methods and their Application to the Parallel Solution of Large-Scale Linear and Geometrically Nonlinear Structural Analysis Problems, *Computer Methods in Applied Mechanics and Engineering*, 184, 333–374, 2000.
- [14] F. Risler and C. Rey, Iterative accelerating algorithms with Krylov subspaces for the solution to large-scale nonlinear problems, *Numerical Algorithms*, 23 (1), 1–30, 2000.
- [15] A. Chapman and Y. Saad, Deflated and augmented Krylov subspace techniques, *Numerical Linear Algebra with Applications*, 4 (1), 43–66, 1997.
- [16] Y. Saad, M. Yeung, J. Erhel and F. Guyomarc’h, A deflated version of the conjugate gradient algorithm, *SIAM Journal on Scientific Computing*, 21 (5), 1909–1926, 2000.
- [17] C. Rey and F. Risler, A Rayleigh–Ritz preconditioner for the iterative solution to large scale nonlinear problems, *Numerical Algorithms*, 17 (3), 279–311, 1998.
- [18] M.L. Parks, E. De Sturler, G. Mackey, D.D. Johnson and S. Maiti, Recycling Krylov subspaces for sequences of linear systems, *SIAM Journal on Scientific Computing*, 28 (5), 1651–1674, 2007.
- [19] E. De Sturler, Truncation strategies for optimal Krylov subspace methods, *SIAM Journal on Numerical Analysis*, 36 (3), 864–889, 1999.
- [20] L.T. Biegler, O. Ghattas, M. Heinkenschloss and B. van Bloemen Waanders, Large-Scale PDE-Constrained Optimization: An Introduction, *Large-Scale PDE-Constrained Optimization*, L.T. Biegler, O. Ghattas, M. Heinkenschloss and B. van Bloemen Waanders (Eds.), 3–13, 2003.
- [21] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, Philadelphia, 2003.
- [22] G. Golub and C. Van Loan, *Matrix Computations*, 3rd Ed., Johns Hopkins University Press, Baltimore, 1996.

- [23] D. Davis, T. Krishnamurthy, W. Stroud and S. McCleary, An accurate nonlinear finite element analysis and test correlation of a stiffened composite wing panel, *American Helicopter Society National Technical Specialists' Meeting on Rotorcraft Structures*, Williamsburg, Virginia, October 29–31, 1991.
- [24] K. Carlberg and C. Farhat, A Compact Proper Orthogonal Decomposition Basis for Optimization-Oriented Reduced-Order Models, *AIAA Paper 2008-5964, 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, Canada, September 10–12, 2008.