Reduced order modeling of steady flows subject to aerodynamic constraints

Zimmermann, Ralf; Vendl, Alexander; Goertz, Stefan

Published in: AIAA Journal

DOI: 10.2514/1.J052208

Publication date: 2014

Document version
Version created as part of publication process; publisher's layout; not normally made publicly available

Citation for published version (APA):
Reduced-Order Modeling of Steady Flows Subject to Aerodynamic Constraints

Ralf Zimmermann* and Alexander Vendl†
Technische Universität Braunschweig, 38106 Braunschweig, Germany
and
Stefan Görtz‡
DLR, German Aerospace Center, 38108 Braunschweig, Germany

DOI: 10.2514/1.J052208

A novel reduced-order modeling method based on proper orthogonal decomposition for predicting steady, turbulent flows subject to aerodynamic constraints is introduced. Model-order reduction is achieved by replacing the governing equations of computational fluid dynamics with a nonlinear weighted least-squares optimization problem, which aims at finding the flow solution restricted to the low-order proper orthogonal decomposition subspace that features the smallest possible computational fluid dynamics residual. As a second and new ingredient, aerodynamic constraints are added to the nonlinear least-squares problem. It is demonstrated that the constrained nonlinear least-squares problem can be solved almost as efficiently as its unconstrained counterpart and outperforms all alternative approaches known to the authors. The method is applied to data fusion, seeking to combine the use of computational fluid dynamics with wind-tunnel or flight testing to improve the prediction of aerodynamic loads. It is also demonstrated that it can be used to compute aerodynamic loads for a given aerodynamic configuration subject to aerodynamic design or performance targets. Exemplary results considering both applications are computed for the NACA 64A010 airfoil and the DLR-F15 high-lift configuration.

Nomenclature

- \( A \) = mean-flow state vector of given \( m \) flow solution snapshots; element of \( \mathbb{R}^{m} \)
- \( a \) = vector of proper orthogonal decomposition coefficients; element of \( \mathbb{R}^{m} \)
- \( C_L, C_D, C_{My}, C_{Mz}, C_{My} \) = lift, drag, moment, and pressure coefficients
- \( d \) = number of model parameters; element of \( \mathbb{N} \)
- \( d^k \) = search direction vector in Gauss–Newton method; element of \( \mathbb{R}^m \)
- \( E \) = total energy, \( \text{kJ} \cdot \text{m}^2 \)
- \( I \) = identity matrix; element of \( \mathbb{R}^{m \times m} \)
- \( m \) = number of flow solution snapshots; element of \( \mathbb{N} \)
- \( m \) = dimension of proper orthogonal decomposition subspace; element of \( \mathbb{N} \)
- \( M_{\infty} \) = freestream Mach number; element of \( \mathbb{R} \)
- \( n_e \) = number of constraints imposed on reduced-order modeling problem; element of \( \mathbb{N} \)
- \( n_g \) = number of grid cells; element of \( \mathbb{N} \)
- \( n_t \) = total dimension of discretized flow problem
- \( n_v \) = number of flow cells; element of \( \mathbb{N} \)
- \( p \) = number of flow variables; element of \( \mathbb{N} \)
- \( \text{RIC} (\tilde{m}) \) = relative information content of first \( \tilde{m} \) proper orthogonal decomposition eigenmodes
- \( \text{res} \) = discretized flux residual; \( \mathbb{R}^{n_t} \to \mathbb{R}^{m} \)
- \( r_j \) = relative information content of \( j \)th proper orthogonal decomposition mode; element of \( \mathbb{R} \)
- \( s^k \) = step length in Gauss–Newton method; element of \( \mathbb{R} \)
- \( u_x, u_y, u_z \) = flow velocity vectors, \( \text{ms}^{-1} \)
- \( U \) = \( j \)th proper orthogonal decomposition mode; element of \( \mathbb{R}^{m} \)
- \( Y \) = \( j \)th eigenvector of \( Y^T \Omega Y \); element of \( \mathbb{R}^{m} \)
- \( W \) = flow solution state vector; element of \( \mathbb{R}^{m} \)
- \( x, y, z \) = Cartesian spatial coordinates; element of \( \mathbb{R} \)
- \( \alpha \) = angle of attack, \( \text{deg} \)
- \( \beta \) = \( j \)th eigenvalue of \( Y^T \Omega Y \); element of \( \mathbb{R} \)
- \( \mu \) = eddy viscosity, \( \text{kg} \cdot \text{ms}^{-1} \)
- \( \nu \) = density, \( \text{kg} \cdot \text{ms}^{-3} \)
- \( \Omega \) = diagonal matrix of grid cell volumes; element of \( \mathbb{R}^{n_e \times n_e} \)
- \( \langle \cdot, \cdot \rangle_{L_2} \) = \( L_2 \) scalar product
- \( \| \cdot \|_{L_2} \) = \( L_2 \) norm; \( \sqrt{\langle \cdot, \cdot \rangle_{L_2}} \)

I. Introduction

VEN with today’s impressive high-performance computing resources at hand, the computational costs associated with high-fidelity computational fluid dynamics (CFD) often render it infeasible for applications such as design, optimization, or aeroloads prediction over the entire flight envelope. Also, CFD is not yet able to deliver the required degree of accuracy and throughput necessary to replace the wind tunnel. The challenge today is how to best combine the use of CFD with wind-tunnel or flight-test data to improve the prediction of aerodynamic loads while reducing the development cycle time [1]. Hence, efficient but nonetheless sufficiently accurate reduced-order models (ROMs) to be used to replace the full-order CFD model (in, e.g., coupled simulations) are sought after. Likewise, methods for merging CFD solutions with wind-tunnel or flight-test data are of high interest. A powerful tool currently considered state of the art (2) for order reduction of nonlinear systems is proper orthogonal decomposition (POD), a technique which has been demonstrated in many fields of application (see, e.g., [3–9]) and is subject to ongoing theoretical investigations (see, e.g., [10–13]). The
basic idea of POD is to replace solving the full-order governing equations of fluid dynamics by determining a suitable linear combination of POD basis vectors, which in turn rely on CFD flow solution snapshots. There exist various methods for computing the coefficients of such a linear combination: among them are interpolation [4,5] and solving a low-order partial-differential-equation (PDE) system [3,10,11,14]. The latter approach has been applied to steady-state ROM problems in [15,16].

Following the approach of [6], effective POD-based ROMs for steady aerodynamic problems can also be obtained by minimizing the POD approximation’s defect with respect to the governing equations of fluid dynamics. Although derived from high-fidelity CFD, the ROM approximation leads to error, which can be reduced by updating or calibrating, for example; a ROM derived from Euler simulations could potentially be updated with viscous data. In the paper at hand, the idea is to constrain the ROM to reproduce particular characteristics of higher-fidelity or experimental data. In particular, a POD-based ROM framework is used to tackle the following objective that arises in the context of aeroloads prediction.

Objective 1: Given an aerodynamic body (e.g., airfoil, wing, aircraft), some selected aerodynamic data (e.g., global lift or drag coefficient, pressure at a few selected pressure taps, etc.) and CFD solutions for a range of flow conditions estimate the flowfield around the body that precisely matches the prescribed data.

Practical applications that are subordinate to this general objective are the following:

1) The first application is data fusion. Assume that aerodynamic data have been obtained by a wind-tunnel or flight-test campaign. These data are to be combined or merged with CFD solutions to efficiently obtain computational solutions exactly in line with the measured data. This will permit small amounts of experimental data to be used to improve the accuracy of a ROM. Updated ROMs offer the potential to span more significant portions of the loads envelope of an aircraft at low cost.

2) The second application is target aerodynamics. Given the task to design a new aircraft featuring certain prescribed aerodynamic coefficients, the loads engineer requires early on in the design process an estimate of the corresponding aerodynamic loads.

The latter application aims at efficiently computing aerodynamic loads for a given aerodynamic configuration subject to aerodynamic design or performance targets, without having to perform full-fledged aerodynamic shape optimization. This may be required to meet industrial needs for a more stable and realistic loads basis throughout the design process, where a conflict between the need and the availability of the final design loads may exist. In this paper, a method is developed that achieves the above goals by reformulating the POD-based ROM approach as a constrained weighted nonlinear least-squares optimization problem restricted to the POD subspace. A method capable of efficiently computing (approximate) solutions to optimization problems of this kind is introduced. Note that, in the context of this paper, optimization refers to computing the best possible ROM approximation for a given aerodynamic shape and not to shape optimization; that is, the quality of the ROM solution is optimized rather than the aerodynamic shape itself.

The potential of the method is demonstrated by computing approximate flowfield solutions for the NACA 64A010 airfoil as well as for the DLR-F15 two-element high-lift configuration [17] subject to aerodynamic constraints, where both of the aforementioned applications are considered.

The paper is organized as follows. Section II briefly reviews the essentials of POD-based reduced-order modeling. Following [6], in Sec. III, it is outlined how an existing flow solver can be exploited to evaluate and minimize the residual associated with an approximate ROM flow solution. In Sec. IV, a method for considering constraints in the POD-based ROM formulation is introduced. Moreover, alternative approaches are discussed and similarities as well as differences to the gappy-POD method [5,18] are outlined. Section V deals with implementation issues, including a note on parallelization. Results for the two text cases are presented in Sec. VI. Section VII concludes the paper.

II. Theoretical Background

A. Governing Equations

Consider the Navier–Stokes equations, spatially discretized on a grid of size \( n_x \) for some aerodynamic configuration. Let \( n_x \) be the corresponding number of primitive mean-flow variables plus the number of primitive variables associated with the turbulence model. The primitive mean-flow variables are the density \( \rho \), the velocity components in all spatial directions \((u_x, u_y, u_z)\), and the total energy \( E \). The number of primitive turbulence variables depends on the chosen turbulence model. Let \( n_t = n_x n_x \) denote the total length of the discretized flow solution vectors. The corresponding system of semidiscrete ordinary differential equations can be written as

\[
\frac{d}{dt} W + \Omega^{-1} \text{res}(W) = 0 \in \mathbb{R}^n
\]

where \( W \in \mathbb{R}^n \) is the state vector of primitive variables, \( \text{res}(W) \in \mathbb{R}^n \) is the vector of flux residuals corresponding to the state solution \( W \), and \( \Omega \in \mathbb{R}^{n_x \times n_x} \) is a diagonal matrix with \( n_x \) subblocks, each containing the cell volumes \((v_{x_1}, \ldots, v_{x_{n_x}})\) of the corresponding computational grid on the diagonal. Denoting by \( W_{k,i} \) the value of flow variable \( k \) corresponding to grid cell \( i \) and by \( \text{res}_{k,i}(W) \) the flux residual of flow variable \( k \) in grid cell \( i \), system (1) may be written in cellwise form as

\[
\forall \ k = 1, \ldots, n_t, \ \forall \ i = 1, \ldots, n_x; \ \frac{d}{dt} W_{k,i} + \frac{1}{v_{i}} \text{res}_{k,i}(W) = 0
\]

See [19] for an equivalent formulation for conservative variables. The steady state is achieved if the time derivative drops out in Eqs. (1) or (2), or equivalently, if the CFD flux residual vanishes:

\[
0 = \Omega^{-1} \text{res}(W) \in \mathbb{R}^n
\]

The weights matrix \( \Omega^{-1} \) stems from the spatial discretization of the flow problem. When computing approximations to Eq. (3) restricted to a subspace of lower dimension, the weighting is significant. The ROM method presented in the following is not restricted to flow state vectors given in primitive variables, but it also applies to state vectors in conservative variables, or even different governing PDE problems.

B. Reduced-Order Modeling via Proper Orthogonal Decomposition

Brief reviews on POD-based ROM approaches in finite-dimensional vector spaces are given in [8,11,20]; for a comprehensive introduction, see [16,21] and references therein. For the reader’s convenience, we will review the essentials.

Let \( p_1, \ldots, p_d \) be the independent parameters of interest for building the reduced-order model. Suppose that \( m \) steady CFD flow solutions \( W := W(p) \in \mathbb{R}^n \), called snapshots, are given, where \( p' = (p_1', \ldots, p_d') \in \mathbb{R}^d \) denotes the \( r \)th combination of model parameters, \( i = 1, \ldots, m \). The parameters \( p \) specify the flow conditions at which one wants to compute the flowfield. For example, \( p' = (p_1', p_2') = (\alpha, M_{\infty}) \) gives the \( r \)th sample point in a space where \( d \) is two-dimensional (angle of attack, Mach).

The snapshots are used to assemble the centered snapshot matrix defined by

\[
Y := (\bar{W}^1, \ldots, \bar{W}^m) \in \mathbb{R}^{n_x \times m}
\]

where \( \bar{W}^i = W^i - A, i = 1, \ldots, m, \) are the centered snapshots, with

\[
A = \frac{1}{m} \sum_{i=1}^{m} W^i
\]

being the constant mean flow. Let the vectors \( V^i \in \mathbb{R}^m \) be the normalized eigenvectors of the \( m \times m \)-dimensional snapshot correlation eigenvalue problem:
where the correlation matrix \((Y^T \Omega Y)\) is computed with respect to the discrete \(L_2\) scalar product \((W_a, W_b)\) as \(W_a^T \Omega W_b\) associated with the computational domain. The ordering of the eigenvectors is such that the corresponding eigenvalues are \(\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m\). Note that

\[
\sum_i \hat{W}_i = 0
\]

Hence, the centered snapshots are linearly dependent. Therefore, \(\lambda_m = 0\) for the smallest eigenvalue. As a consequence, a maximum of \(m-1\) POD modes is sufficient for a perfect reconstruction of the snapshots and the orthonormal basis \(\{U^1, \ldots, U^{m-1}\}\) of POD modes is given by

\[
U^j = (\sqrt{\lambda_j}^{-1} V \in R^n, \quad j = 1, \ldots, m-1
\]

Thus, \(U^j\) corresponds to the largest eigenvalue \(\lambda_1\) and so forth.

The relative information content of the \(j\)th mode is defined as the ratio

\[
r_j = \lambda_j / \left( \sum_{i=1}^m \lambda_i \right)
\]

and the relative information content (RIC) of the first \(\tilde{m} \leq m-1\) basis modes is thus given by

\[
\text{RIC}(\tilde{m}) = \sum_{j=1}^{\tilde{m}} r_j
\]

With the POD modes at hand, the (possibly) reduced representation of the \(i\)th snapshot solution is given by

\[
W(a) = A + \sum_{j=1}^m a_j U^j
\]

with coefficients

\[
a_j = \langle \hat{W}_i, U^j \rangle_{L_2} = 1 / \sqrt{\lambda_j} \langle \hat{W}_i, U^j \rangle \Omega V^j = \sqrt{\lambda_j} V_i
\]

An approximate flow solution \(W\) can be constructed by solely depending on a coefficient vector \(a = (a_1, \ldots, a_m) \in R^m\) as follows:

\[
W(a) = A + (U^1, \ldots, U^m) a = A + \sum_{j=1}^m a_j U^j
\]

According to Eq. (7), the computation of an approximate flow solution at an untuned parameter combination \(p^*\) via POD is reduced to computing the unknown POD coefficient vector \(a(p^*) = a^* = (a_1^*, \ldots, a_m^*)\).

### III. Exploiting CFD information in Estimating POD Coefficients

In [6], LeGresley and Alonso proposed a method for computing the POD coefficients of an approximate flow solution by taking the CFD residual into account. Following this approach, the coefficients of the reduced-order POD solution are determined by minimizing the associated CFD flux residual, which is evaluated using a full-order CFD solver:

\[
\min_{a=(a_1, \ldots, a_m)} \| \Omega^{-1} \text{res}(W(a)) \|_{L_2}^2 = \sum_{i=1}^n \sum_{j=1}^m \frac{1}{\text{vol}_i} (\text{res}_{i,j}(W(a)))^2
\]

Since, in general, the POD subspace does not contain steady-state flow solutions that feature a zero residual, solutions to Eq. (8) are the best possible approximations to converged CFD flow solutions inside the POD subspace in a least-squares average sense. Thus, order-\(n_t\) equation system (3) is replaced by unconstrained nonlinear least-squares optimization problem (8) of order \(\tilde{m}\).

To allow for the same impact of all residual variables on optimization problem (8), regardless of their respective scale, a normalization is conducted. The normalization is based on the residual values of the starting solution used to initialize optimization problem (8). More precisely, if \(\text{res}(\rho) = (\text{res}(\rho)^1, \ldots, \text{res}(\rho)^n)\) is the starting vector of the density residuals in all grid points, then the normalized vector

\[
\text{res}^\ast(\rho) = \frac{1}{\| \text{res}(\rho)^0 \|} \text{res}(\rho)^0
\]

is actually used in Eq. (8). This normalization is performed in an analogous way for all residual variables.

### IV. Incorporating Constraints into the POD-ROM Approach

To address the data fusion and target aerodynamics application challenges stated in the Introduction (Sec. I), nonlinear least-squares optimization problem (8) is modified to incorporate aerodynamic constraints. Suppose that a number of \(n_c\) constraints are given in the form of a vector-valued function \(g(a) \in R^{n_c}\) depending on the POD coefficients. If \(n_c = 0\), then the function \(g\) drops out and the scheme automatically reduces to a standard unconstrained least-squares optimization algorithm. As will be explained in Sec. IV.D, there is a natural upper bound on the number of constraints. For example, if target lift and drag coefficients \(C_{L,\text{target}}\) and \(C_{D,\text{target}}\) are to be met, then \(g\) is two-dimensional and takes the form

\[
g(a) = \left( C_{L,\text{target}}(W(a)) - C_{L,\text{target}}, C_{D,\text{target}}(W(a)) - C_{D,\text{target}} \right)
\]

where \(W(a)\) is the approximate ROM solution corresponding to the vector of POD coefficients \(a\) according to Eq. (7).

Introducing the function \(f(a) = \text{res}(W(a))\), the analog to problem (8) can be written as a constrained weighted least-squares problem with regard to the Euclidean norm:

\[
\min_{a \in R^n} \frac{1}{2} \| \Omega^{-1/2} f(a) \|_{L_2}^2 \quad \text{subject to} \quad g(a) = 0
\]

Note that \(\| \Omega^{-1/2} f(a) \|_{L_2}^2 = \| \Omega^{-1} f(a) \|_{L_2}^2\). Hence, introducing the weights matrix \(\Omega^{-1/2}\) is mandatory to the end that the preceding Euclidean formulation actually corresponds to the minimum residual solution with regard to the \(L_2\) norm, which is the most natural choice of norm for the problem at hand. The iterative Gauss–Newton method for solving problems of the precise form of Eq. (9) is detailed in [22]. In short, given the approximate solution at iteration stage \(k\), the next iterate reads

\[
a^{k+1} = a^k + s^k d^k
\]

where \(s^k \in R\) is the step length, and \(d^k \in R^n\) is the search direction at stage \(k\).

According to [22] (Eq. 2.2), the search direction at iteration stage \(k\) is defined by
Here, $Dg \in \mathbb{R}^{n_v \times \hat{m}}$ is the Jacobian of the constraints function, $Df \in \mathbb{R}^{n_g \times \hat{n}}$ is the Jacobian of the least-squares objective function, and $(\mu_1, \mu_2)^T \in \mathbb{R}^{2 \times \hat{m}}$ is a vector of Lagrange multipliers at iteration stage $k$. Linear system (11) is referred to as the system equations, and it has a unique solution, if both Jacobians $Dg$ and $Df$ have full rank at stage $k$; see [22] (Lemma 2.1). Hence, the Gauss–Newton method applied to constrained optimization problem (9) requires, at each iteration stage, solving the $(n_v + n_g + \hat{m}) \times (n_v + n_g + \hat{m})$ equation system. Note that the system matrix scales in the grid size via $n_v = n_g$. Thus, it seems impossible to solve Eq. (11) in the reduced-order modeling context. A remedy, however, is introduced in the next section.

A. Efficient Solution of the System Equations

Going beyond [22], in this section, we derive a method for performing the constrained Gauss–Newton iteration almost as efficiently as its unconstrained counterpart. By setting

$$\begin{align*}
R &= (\Omega Df^T Df)^{-1} F = \begin{pmatrix} 0_{n_v \times \hat{m}} & Dg^T \\ 0_{\hat{m} \times \hat{m}} & 0_{\hat{m} \times \hat{m}} \end{pmatrix}
\end{align*}$$

the system matrix in Eq. (11) reads

$$\begin{pmatrix}
(0_{n_v \times n_v} & \mathbf{F}^T) \\
\mathbf{F} & R
\end{pmatrix}$$

and allows for a direct blockwise inversion ([23], p. 70) resulting in

$$\begin{pmatrix}
0 & \mathbf{F}^T \\
\mathbf{F} & R
\end{pmatrix}^{-1} = \begin{pmatrix} S & -SR^{-1}F \\
-R^{-1}SF & R^{-1}FSR^{-1} \end{pmatrix}$$

where $S = -(\mathbf{F}^T R^{-1} F)^{-1} \in \mathbb{R}^{n_v \times n_v}$ is the so-called Schur complement ([23], p. 68). By applying the same trick to the block matrix $R$ and exploiting this in Eq. (11), the following closed formula for the search direction is obtained:

$$\mathbf{d}^g = T Dg^T S g - (I + T Dg^T S Dg) T Dg^T \mathbf{1}$$

with $T = (Df^T \Omega^T Df)^{-1}$, $S = -(Df T Df^T)^{-1}$, and $I = I_{\hat{m} \times \hat{m}}$. Being the $\hat{m} \times \hat{m}$ identity matrix. In the Appendix, the reader may find more details on this derivation.

The aforementioned considerations yield Algorithm A for efficiently performing a constrained Gauss–Newton optimization. To keep the computational complexity at a feasible level, the line search for computing the optimal step size $\delta$ is omitted and $\delta$ is set to a constant value of $\delta = 1$ for all $k$. Convergence is detected if a Cauchy-type stopping criterion is hit: that is, if the iteration $k + 1$ does not differ significantly from the iteration $k$. More precisely, the algorithm stops if

$$\|\mathbf{d}^g\|_2 < \epsilon (\|a^g\|_2 + \epsilon)$$

where $\epsilon$ is a user-defined threshold.

Computing ROM solutions that meet specified aerodynamic constraints by the aforementioned method will be referred to as the constrained least-squares (LSQ)-ROM method.

B. Discussion of Alternative Methods for Solving the System Equations

1. Direct Solution of the System Equations

The system equations define a linear system of dimension $n_v + n_g + \hat{m} = n_v + n_g n_v + \hat{m}$. The most dominant factor is the grid size $n_v$, which may well take a value of $O(10^5)$ to $O(10^7)$ for industrial aircraft configurations. Therefore, a direct solution of the system equations requires an effort comparable to a full-order CFD solution, and thus is prohibitive in the context of reduced-order modeling.

2. Solution via Modified QR

In [22] (paragraph 2), it is proposed to compute a modified QR decomposition of the Jacobian appearing in Eq. (11). Note that this results in a dense square matrix $Q$, for which the dimensions scale in the grid size. While this is certainly beneficial from the numerical point of view, it is again simply unfeasible to even store the matrix $Q$ in the context of reduced-order modeling, especially in regard to industrial applications.

3. Solution via Sparse Linear Algebra

Note that the diagonal matrix of grid cell volumes $\Omega$ takes by far the largest part of the system matrix in Eq. (9). Hence, it is easy enough to directly write down a representation of the system matrix in, say, a compressed sparse row format ([24] paragraph 3.4) so that the system allows for applying sparse solvers. This is the only other (at the least) feasible alternative to the method developed in Sec. IV.A. The examples given in Sec. VI, however, show that this approach is noncompetitive.

Algorithm A Constrained least-squares Gauss–Newton method

Require iteration bound $k_{max}$, initial vector of POD coefficients $a^g$, finite difference step size $\delta > 0$, convergence threshold $\epsilon > 0$, POD data $U^j, \lambda_j, j = 1, \ldots, \hat{m}$.

1. for $k = 0 \rightarrow k_{max}$ do
2. take weights into account: compute $\tilde{f}_k = \Omega^{-1/2}(f(a^g))$
3. finite difference approximation of Jacobian of residual: compute $Df_{\tilde{f}} = \Omega^{-1/2}Df(a^g)$
4. finite difference approximation of Jacobian of constraints: compute $Dg_{\tilde{f}} = \Omega^{-1/2}Dg(a^g)$
5. compute $H = T^{-1} = Df_{\tilde{f}}^T Df_{\tilde{f}} \in \mathbb{R}^{\hat{m} \times \hat{m}}$, and $h = Df_{\tilde{f}} f_{\tilde{f}} \in \mathbb{R}^{\hat{m}}$
6. solve $HF = Df_{\tilde{f}}^T$ for $F \in \mathbb{R}^{\hat{m} \times \hat{m}}$
7. solve $\mathbf{Hx} = \mathbf{h}$ for $x \in \mathbb{R}^{n_v}$
8. compute $S = -(Dg_{\tilde{f}} P)^{-1} \in \mathbb{R}^{\hat{m} \times \hat{m}}$
9. compute $\mathbf{Q} = \mathbf{PS} \in \mathbb{R}^{\hat{m} \times \hat{m}}$
10. compute search direction $d^g = \mathbf{Q} \tilde{f}_k - (x + \mathbf{Q} Dg_{\tilde{f}} x) \in \mathbb{R}^{\hat{m}}$
11. if $\|d^g\|_2 < \epsilon (\|a^g\|_2 + \epsilon)$, break
12. $a^g + = a^g + d^g$
13. $k = k + 1$
14. end for

Remarks: For the results discussed in Sec. VI, a value of $\epsilon = 10^{-4}$ was chosen for the finite difference approximation in steps 3 and 4. The weighting of $\delta$ by the singular value $\lambda_j$ accounts for the different orders of magnitude of the POD coefficients. A value of $\epsilon = 10^{-8}$ was specified for the stopping criterion in step 11.
C. Alternative Optimization Methods

Instead of using the Gauss–Newton method for solving the constrained and weighted nonlinear least-squares problem, it is possible to apply more general optimization methods, e.g., the sequential quadratic programming (SQP) method ([25], paragraph 6.5). This method allows for tackling optimization problems of the more general form

\[
\min_a f(a) \quad \text{subject to } g(a) = 0
\]

where \( f: \mathbb{R}^n \to \mathbb{R} \) is an arbitrary (differentiable) function. By setting

\[
f(a) = \frac{1}{2} \| \Omega^{-(1/2)} f(a) \|^2
\]

this method applies to problem (9). Note, however, that the SQP approach obtains no information on the least-squares structure of Eq. (9), and therefore cannot exploit this structure efficiently. A comparison of the SQP approach to the method derived here is conducted in Sec. VI.

D. Comparison to Gappy POD

Gappy POD ([5,18]) is concerned with minimizing the least-squares error of the POD approximation with respect to some prescribed (gappy) data and has been previously used by the second author for data fusion purposes [26]. The coefficients of the gappy-POD representation are determined by solving a linear least-squares problem. The gappy-POD least-squares problem is well defined if there are at most as many POD modes as there are constraints. Otherwise, the gappy-POD least-squares system is underdetermined and cannot be solved uniquely. If it is overdetermined, then the side constraints are not fulfilled exactly but in a least-squares optimal sense. Note that, apart from the underlying snapshots, no additional CFD information enters the process of computing POD coefficients in the case of gappy POD.

The constrained LSQ-ROM method on the other hand is concerned with minimizing the CFD flux residual subject to equality constraints. This is a nonlinear least-squares problem. Each equality constraint reduces the degrees of freedom by one. Therefore, in order to minimize the residual in addition to satisfying the constraints, more degrees of freedom, i.e., more POD modes than side constraints, are required. In summary, the following holds:

1) The Gappy POD applies if the number of constraints is larger than the number of POD basis modes, \( n_c > m \).
2) The constrained LSQ-ROM method applies if the number of constraints is smaller than the number of POD basis modes, \( n_c < m \).
3) When the number of constraints is equal to the number of POD basis modes, \( n_c = m \), then the degrees of freedom perfectly match the number of constraints, leading to a uniquely determined vector of POD coefficients. In this case, both the gappy POD and the constrained LSQ-ROM produce the same result.

V. Implementation Issues

The constrained LSQ-ROM method as introduced in Secs. III and IV has been implemented within the Surrogate Modeling for Aerodynamic Design Optimization (SMART) of the DLR, German Aerospace Center (DLR); see the Acknowledgments. This code package combines efficient algorithms written and parallelized in the C programming language [27] with a layer of control scripts written in Python. It features an interface to the DLR flow solver TAU [28–30]. The TAU code itself features a direct interface to Python, allowing functions of the TAU code to be called from a user-created Python script and straightforward communication with external programs through file exchange or socket communication, eliminating file input-output during a complex (multidisciplinary) simulation. It is very important to emphasize that the constrained LSQ-ROM method presented here does not require intrinsic changes to the flow solver of choice. Any flow solver that is able to return the residual vector with respect to a given approximate flow state vector may be used as a black-box function.

With each flow residual evaluation, the DLR TAU code simultaneously returns the corresponding aerodynamic attributes, of which the lift, drag, and moment coefficients \( C_L, C_D, C_M_x, C_M_y, \) and \( C_M_z \) are the most important. Therefore, no additional computational effort is required when evaluating the constraint function, nor when computing its Jacobian.

The workflow of the constrained LSQ-ROM code realizing the Gauss–Newton method as described in detail in Algorithm A in Python is as follows:

1) Flow solution snapshots are computed with the DLR TAU code.
2) (Weighted) POD is conducted using DLR’s SMART.
3) An approximate ROM solution is passed on to the TAU-Python code in memory.
4) The flux residual is evaluated using the DLR TAU code and returned in memory. In this way, the objective function, the constraints function, and their respective Jacobians are obtained.
5) The large-scale matrix–matrix and matrix–vector products required for step 4 in Algorithm A are performed (possibly in parallel) by a C subfunction, which has been wrapped to Python.
6) All other data scales in the number of constraints \( n_c \) or in the number of POD basis modes \( m \). The associated matrix–matrix products and the solutions to the small-scale linear systems that accrue are performed using predefined standard algorithms from Scientific Python (SciPy) [31].

A. Computational Costs

Considering Algorithm A, at each Gauss–Newton step, the most dominant operations and the associated computational costs are as follows:

1) Step 2 evaluates the CFD flux residual vector and scales it by the square root of the cell volumes \([O(n_c) + 1 \text{ CFD residual evaluation}].
2) Step 3 computes the Jacobian of the objective function \([O(n_c m) + m \text{ CFD residual evaluation}].
3) Step 5 computes the symmetric product of the Jacobian and the matrix–vector product “Gaussian times objective function” \([O(n_c (m^2 + m))]\).

Note that these most dominant computations are exactly the same for both constrained and unconstrained least-squares optimizations. All other steps consist of computations that neither depend on the grid size nor require CFD residual evaluations. Hence, all operations scale either linearly in the grid size or do not depend on the grid size at all, and are thus negligible. As a consequence, the constrained LSQ-ROM approach performs asymptotically as fast as its unconstrained counterpart. A realistic estimate of the number of POD modes \( m \) is \( O(10^3) \), up to \( O(10^5) \). Even if \( m = 1000 \), then solving linear systems of dimensions \( 1000 \times 1000 \) is performed in virtually no time on today’s standard desktop computers.

As explained in Sec. V.D, the number of POD basis modes \( m \) is a natural upper bound on the number of constraints imposed on the LSQ-ROM problem. B. Note on Parallelization

If the flow solver of choice evaluates the residual in parallel, then obviously, the objective function in the constrained LSQ-ROM method is also evaluated in parallel. By storing the objective function vector and its Jacobian divided into \( l \) blocks, \( f^T = (f_1, \ldots, f_l) \) and \( Df^T = (Df_1^T, \ldots, Df_l^T) \), the large-scale matrix operations can be performed in parallel on \( p \) processes followed by a summation over all subblocks, e.g.,

\[
Df^T Df = \sum_{i=1}^l Df_i^T Df_i, \quad \quad Df^T f = \sum_{i=1}^l Df_i^T f_i
\]
As a consequence, the constrained LSQ-ROM approach can be performed in parallel; thus, the approach also applies to large-scale three-dimensional full aircraft configurations, just as demonstrated for its unconstrained counterpart in [9,32].

VI. Results

In the following, two test cases are considered to show the potential of the constrained ROM approach in solving problems of industrial interest, including data fusion and target aerodynamics. The Results section is organized as follows. Section VI.A features an application of the constrained LSQ-ROM to transonic aerodynamics, which shows that the constrained ROM is capable of treating strong shocks. This subsection also includes a comparison of all the optimization methods outlined in Sec. IV.A in terms of computational complexity only (Sec. VI.A.2). Subsequently, LSQ-ROM predictions are performed subject to a single constraint as well as to three constraints (Sec. VI.A.3). This test case exposes the performance differences of the competing approaches. Subsection VI.A.4 features an example that demonstrates the applicability of the constrained LSQ-ROM to the “target aerodynamics” problem stated in the Introduction (Sec. I).

To support the universal validity of the previous findings, the exercises performed in Sec. VI.A (excluding the basic performance test) are repeated in Sec. VI.B, but for a subsonic high-lift aerodynamics test case.

A. Reduced-Order Model for Transonic Aerodynamics Under Constraints

1. Test Case Setup

In this section, the constrained LSQ-ROM method is applied to the flow over the NACA 64A010 airfoil in the transonic flow regime. The underlying computational grid features 21,454 points and is displayed in Fig. 1. For computing snapshot flow solutions, the DLR TAU Reynolds-averaged Navier–Stokes solver [30] is applied using the Spalart–Allmaras (SA) one-equation turbulence model [33]. The Reynolds number is fixed at a value of \( Re = 7,500,000 \). The POD basis is constructed relying on 28 snapshots at pairs of \((\alpha, M_{\infty}) \in [0\ deg, 10\ deg] \times [0.73, 0.83]\). The precise snapshot sample locations can be read from Fig. 2. As prediction points, \((\alpha_1, M_{\infty,1}) = (7.0\ deg, 0.74)\) and \((\alpha_2, M_{\infty,2}) = (5.9\ deg, 0.815)\) were chosen (arbitrarily); see Fig. 2. Computing the CFD reference solutions at the prediction points until they converged to a residual smaller than 10\(^{-7}\) took 2378 iterations (or 940 s) and 2173 iterations (or 863 s), respectively. These computations, as well as all of the following, were conducted on the same standard desktop computer endowed with an AMD AthlonTM 64 X2 dual-core processor 3800+

A POD of the snapshot data with respect to the discrete \( L_2 \) scalar product results in an orthogonal representation, where the basis modes are represented by vectors of dimension \( n_t = n_p n_v = 6 \times 21,454 = 128,724 \). (The six variables for a two-dimensional airfoil considered here are the density, the two velocity components, the pressure, the SA viscosity, and the eddy viscosity.) Note that reducing the POD basis is a subtle task, when the relative information content is measured with respect to the \( L_2 \) inner product. This is because the \( L_2 \) measure corresponds to looking at the computational domain from a distance; see Fig. 1, left-hand side. From this perspective, the snapshots, which vary in Mach and angle of attack (AOA), appear to be numerically almost linearly dependent, because the distinctive nonlinear features (e.g., shocks) are essentially restricted to a very small region around the airfoil. Hence, reducing the POD basis according to the relative information content may result in a truncation of important higher-order modes necessary for

<table>
<thead>
<tr>
<th>Optimization method</th>
<th>CPU time</th>
<th>Objective function evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential quadratic program (fminlsqsp)</td>
<td>743 s</td>
<td>1599</td>
</tr>
<tr>
<td>Constrained Gauss–Newton (sparse)</td>
<td>&gt;5 h</td>
<td>532</td>
</tr>
<tr>
<td>Constrained Gauss–Newton (following Algorithm A)</td>
<td>92.3 s</td>
<td>532</td>
</tr>
<tr>
<td>Constrained Gauss–Newton (direct solution)</td>
<td>Prohibitive</td>
<td>–</td>
</tr>
<tr>
<td>Constrained Gauss–Newton (modified QR as in [22])</td>
<td>Prohibitive</td>
<td>–</td>
</tr>
</tbody>
</table>
of the computational time when reducing a POD basis from (for example) order $O(10^2)$ to order $O(10)$ is negligible. Therefore, to capture the complete information content, the maximum number of 27 POD modes is kept in the POD basis. Note that the mean-flow vector is a constant vector that is not subject to any estimation/optimization procedure. Hence, according to Eqs. (8) and (9), the initial flow problem of order $nt = 128, 724$ is reduced to a constrained order-$O(27)$ optimization problem. Least-squares minimization is performed simultaneously for the residuals of the four primitive mean-flow variables, i.e., the density residuals, the velocity residuals, and the energy residuals. Therefore, the residual vector of the starting solution is normalized to a value of 4.0 with regard to the discrete $L^2$ norm.

2. Computational Complexity of the System Equation Solvers

Before focusing on the aerodynamic details, a comparison of all the optimization methods discussed in Sec. IV is conducted based on the sole criterion of the inherent computational effort. For doing so, POD-based reduced-order flow solutions were computed for the NACA 64A010 airfoil at a flow condition of Mach $M_{\infty} = 0.78$ and angle $\alpha = 5.0$ deg subject to a single constraint of a prescribed lift value. The results are summarized in Table 1, where fminslsqp refers to the sequential least-squares quadratic programming minimization method in SciPy [31]. The constrained LSQ-ROM method developed in this paper clearly outperforms all the other approaches with respect to the computational effort.

3. Transonic Data-Fusion Example

In this section, the basic capability of the constrained LSQ-ROM method for the purpose of data fusion is demonstrated. In the absence of suitable experimental data, the aerodynamic coefficients of a full-order TAU CFD reference solution act as target constraints. More precisely, POD-based reduced-order flow solutions for the NACA 64A010 airfoil are computed at a transonic flow condition of $M_{\infty} = 0.74$, $\alpha = 7.0$ deg, subject to a constrained lift of $C_{L,\text{target}} = 0.7169$, which is the value of the TAU CFD reference solution.

Table 2 shows the performance of the constrained LSQ-ROM method compared to the quadratic programming minimization method fminslsqp. Note that both methods produce very similar results, but they differ significantly in the computational effort. To support the claim that the constrained LSQ-ROM performs almost as efficiently as its unconstrained counterpart, the performance results of the unconstrained LSQ-ROM are also included. As can be seen from this table, the quadratic programming method is slower than the constrained LSQ-ROM approach by a factor larger than 12. The optimized residual of the constrained LSQ-ROM solution is slightly larger than the optimized residual of unconstrained LSQ-ROM. This is perfectly in line with the theory, because when conducting constrained optimization, there remain less degrees of freedom to decrease the residual. Note that both of the constrained optimization methods, least-squares and quadratic programming, match the side constraint up to nearly machine precision (error below $10^{-13}$), while the unconstrained LSQ-ROM solution predicts a lift of $C_{L} = 0.74052$ (error = 3.29%). The corresponding surface $C_p$ distributions are displayed in Fig. 3. The solution corresponding to the POD coefficients used to initialize the optimization problem is also displayed in Fig. 3 and was obtained by POD-based interpolation using the thin-plate splines (TPS) radial basis function approach; see ([35], p. 46). The corresponding field pressure contours are displayed in Fig. 4. The quadratic programming solution virtually coincides

![Fig. 3 NACA 64A010: $C_p$ distributions at $M_{\infty} = 0.74$, $\alpha = 7.0$ deg. (Constrained $C_L$).](image1)

![Fig. 4 NACA 64A010: Pressure contour lines at $M_{\infty} = 0.74$, $\alpha = 7.0$ deg. (Constrained $C_L$).](image2)

Table 2 Computational effort and optimization performance of C-LSQ-ROM, quadratic programming minimization method (fminslsqp), and unconstrained LSQ-ROM method (Uc-LSQ-ROM)

<table>
<thead>
<tr>
<th>Optimization method</th>
<th>C-LSQ-ROM</th>
<th>fminslsqp</th>
<th>Uc-LSQ-ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time, s</td>
<td>67.8</td>
<td>1073</td>
<td>56.56</td>
</tr>
<tr>
<td>Objective function evaluations</td>
<td>197</td>
<td>2411</td>
<td>173</td>
</tr>
<tr>
<td>(Normalized) starting residual</td>
<td>4.0</td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td>Optimized residual</td>
<td>0.2454</td>
<td>0.7169</td>
<td>0.1352</td>
</tr>
<tr>
<td>$C_L$ (set as a constraint)</td>
<td>0.7169 (0.0%)</td>
<td>0.7169 (0.0%)</td>
<td>0.7405 (3.3%)</td>
</tr>
<tr>
<td>$C_L$ (relative error)</td>
<td>0.0769 (1.4%)</td>
<td>0.0777 (2.2%)</td>
<td>0.0792 (4.4%)</td>
</tr>
<tr>
<td>$C_M$ (relative error)</td>
<td>-0.0106(5.8%)</td>
<td>-0.0109(8.3%)</td>
<td>-0.0110(9.3%)</td>
</tr>
</tbody>
</table>

*aBoth the C-LSQ-ROM and fminslsqp methods are constrained with the lift coefficient of the CFD reference solution.

*b$M_{\infty} = 0.74$, $\alpha = 7.0$. 

<table>
<thead>
<tr>
<th>Constraints: $C_L$ of CFD reference</th>
<th>Contour: Pressure of CFD reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-LSQ-ROM, target $C_L$</td>
<td>Solid line: C-LSQ - ROM</td>
</tr>
<tr>
<td>fmin_slspq - ROM, target $C_L$</td>
<td>Dashed: Standard LSQ - ROM</td>
</tr>
</tbody>
</table>

![Contour: Pressure of CFD reference](image3)
with the constrained LSQ-ROM solution and is therefore not displayed.

As a second test case, we repeat the preceding exercise subject to an augmented number of constraints. To this end, the forces and moments of the TAU CFD reference solution, given by $C_{L, \text{target}} = 0.7169$ (as before) as well as $C_{D, \text{target}} = 0.0759$ and the pitching moment $C_{M, \text{target}} = -0.0100$, were specified as target values. The performance of the various methods is summarized in Table 3, which compares the computational effort and the performance of the constrained LSQ-ROM method (C-LSQ-ROM) and the quadratic programming method (fminslsqp), both constrained with the lift, drag, and moment coefficients of the CFD reference solution at $M_{\infty} = 0.74$, $\alpha = 7.0$ deg.

<table>
<thead>
<tr>
<th>Optimization method</th>
<th>C-LSQ-ROM</th>
<th>fminslsqp</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time, s</td>
<td>65.3</td>
<td>1002</td>
</tr>
<tr>
<td>Objective function evaluations</td>
<td>197</td>
<td>2250</td>
</tr>
<tr>
<td>(Normalized) starting residual</td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td>Optimized residual</td>
<td>0.2561</td>
<td>1.476</td>
</tr>
</tbody>
</table>

Table 3: Computational effort and optimization performance of constrained LSQ-ROM method and quadratic programming minimization method (fminslsqp), both constrained with the lift, drag, and moment coefficients of the CFD reference solution at $M_{\infty} = 0.74$, $\alpha = 7.0$.

Table 4: Computational effort and optimization performance of constrained LSQ-ROM method and quadratic programming minimization method (fminslsqp), and the unconstrained LSQ-ROM method.

<table>
<thead>
<tr>
<th>Optimization method</th>
<th>C-LSQ-ROM</th>
<th>fminslsqp</th>
<th>Uc-LSQ-ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time, s</td>
<td>92.3</td>
<td>782</td>
<td>88.2</td>
</tr>
<tr>
<td>Objective function evaluations</td>
<td>309</td>
<td>1717</td>
<td>318</td>
</tr>
<tr>
<td>(Normalized) starting residual</td>
<td>4.0</td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td>Optimized residual</td>
<td>1.758</td>
<td>1.758</td>
<td>1.071</td>
</tr>
</tbody>
</table>

Table 4: Computational effort and optimization performance of constrained LSQ-ROM method and quadratic programming minimization method (fminslsqp), and the unconstrained LSQ-ROM method.

As can be seen, subject to the given three side constraints, the constrained LSQ-ROM arrives at a solution very similar to the one obtained when constraining the lift exclusively. In contrast, the quadratic programming method is not only much slower but fails to reach a comparable local minimum, corresponding to a significantly higher optimized flux residual. In fact, it predicts a solution close to the thin-plate spline solution shown in Fig. 3, which was used to initialize the optimization procedure in both cases. The lower quality of the quadratic programming method’s prediction is confirmed by the comparison shown in Fig. 5, which shows the surface $C_p$ distributions of both methods compared to the reference CFD solution. Figure 6 shows the field solutions in terms of pressure contour lines of the reference CFD solution, compared to the constrained LSQ-ROM solution (solid lines) and the quadratic programming solution (dashed lines), both matching the lift, drag, and moment coefficients of the reference solution.

4. Transonic Target-Lift Example

In this section, reduced-order flow solutions for the NACA 64A010 airfoil are computed at a flow condition of $M_{\infty} = 0.815$, $\alpha = 5.9$ deg, subject to the constraint of producing 110% of the lift when compared to the reference solution; that is, the side constraint was specified as $C_{L, \text{target}} = 1.1 \cdot C_{L, \text{ref}} = 1.1 \cdot 0.6079 = 0.6687$. Such a target aerodynamics task might occur in the early stages of the wing/aircraft design, where the engineer assumes that the final design will feature a certain lift and wants to get a first idea of how the corresponding aerodynamic loads distribution might look.

Table 4 shows the performance of the constrained LSQ-ROM method compared to the quadratic programming minimization method (fminslsqp). Note that both methods produce virtually the same results but differ significantly in the computational effort. To support the claim that the constrained LSQ-ROM performs as efficiently as its unconstrained counterpart, the results of the unconstrained LSQ-ROM are also included. The corresponding surface $C_p$ distributions are displayed in Fig. 2. Note that the corresponding quadratic programming solution is omitted for better readability, since it virtually coincides with the constrained LSQ-ROM solution. This example shows that the constrained LSQ-ROM method is able to produce flowfield approximations that maintain a prescribed target lift, drag, and/or moment coefficient, which is higher than that obtained from CFD for the same aerodynamic shape and flow conditions, while featuring a reasonably physical pressure distribution (notice the sharp shock and the linear pressure level upstream of the shock). The corresponding flowfield solutions are displayed in Fig. 8. Note that the corresponding quadratic programming solution is omitted for better readability, since it virtually coincides with the constrained LSQ-ROM solution.

B. Reduced-Order Model for High-Lift Aerodynamics Under Constraints

1. Test Case Setup

In this section, the viscous flow past a DLR-F15 two-element high-lift wing section [17] at a subsonic freestream Mach number of
$M_\infty \in [0.22, 0.24]$ and a Reynolds number of 10.549 million is considered. The underlying computational grid is a hybrid one, which is structured near the surface of the configuration and unstructured otherwise. It consists of 101,618 grid points and is depicted in Fig. 9.

The POD basis for the ROM to be investigated in this section is constructed relying on six snapshots at pairs of $(\alpha, M_\infty) \in \{(1.0 \, \text{deg}, 0.22), (4.0 \, \text{deg}, 0.22), (7.0 \, \text{deg}, 0.22)\}$ and $(\alpha, M_\infty) \in \{(1.0 \, \text{deg}, 0.24), (4.0 \, \text{deg}, 0.24), (7.0 \, \text{deg}, 0.24)\}$; see Fig. 10.

The snapshot set at hand is an extract from a series of snapshots, which were computed with the TAU flow solver [28–30] by using the Spalart–Allmaras one-equation turbulence model [33]. The CFD computation of the first snapshot at the lowest angle of attack was initialized with freestream conditions. All the others were computed in steps of $\alpha = 1.0 \, \text{deg}$ starting from the previous flow solution snapshot. This procedure was followed in order to avoid multiple values of the aerodynamic coefficients at a particular angle of attack. In fact, it has been observed that the lift and drag coefficients differ if the CFD computations were started from freestream values or from a solution at a lower angle of attack [36]. This phenomenon is called aerodynamic hysteresis.

Convergence to steady state was detected based on the absolute change in the aerodynamic coefficients. More precisely, if the change in the aerodynamic coefficients dropped below a user-defined tolerance over some specified interval of iterations, the CFD solutions were considered as converged. For the snapshots at hand, the interval was chosen as 100 iterations and the tolerances for the change in the lift coefficient $C_L$, drag coefficient $C_D$ and pitching moment coefficient $C_M$ were set to $10^{-3}$, $10^{-5}$, and $10^{-7}$, respectively. Thus (e.g., for the lift coefficient), convergence is detected if $|C_L^k - C_L^{k-1}| < 10^{-3}$ for all $j = 1, \ldots, 100$, where $k$ is the current number of iterations.

Following this procedure, it took 1300 iterations of the CFD solver on average to obtain a converged snapshot solution. On a standard desktop computer endowed with an AMD Athlon™64 X2 dual-core processor 3800+, this corresponds to a CPU time effort of 2.200 s, or about 37 min.

A POD of the snapshot set with respect to the discrete $L_2$ scalar product results in an orthogonal representation, where the basis modes are represented by vectors of dimension $n_r = n_x n_y = 6 \cdot 101 \cdot 609 = 609,708$ [where $n_r = 6$ for the vector of flow variables $(\rho, u_x, u_y, p, \nu, \mu)$]. A maximum number of five POD modes is kept in the POD basis for the reasons discussed in Sec. VI.A.1. Note that the mean-flow vector is a constant vector that is not subject to any estimation/optimization procedure. Hence, according to Eqs. (8) and (9), the initial flow problem of order $n_r = 609,708$ is reduced to a constrained order-5 optimization problem. Least-squares minimization is performed simultaneously for the residuals of the four...
primitive mean-flow variables, i.e., the density residuals, the velocity residuals, and the energy residuals. Therefore, the residual vector of the starting solution is normalized to a value of 4.0 with regard to the discrete $L_2$ norm.

2. High-Lift Data-Fusion Example

In this section, ROM approximations to the DLR-F15 high-lift configuration at $(\alpha, M_\infty) = (8.0 \text{ deg}, 0.23)$ subject to exactly reproducing the lift, drag, and moment coefficients of the CFD reference solution at this flow condition, given by $C_L_{\text{target}} = 2.815$, $C_D_{\text{target}} = 0.0659$, and $C_M_{\text{target}} = -0.7387$, are computed. To make this subsonic test case more challenging, in addition to choosing a coarse snapshot sampling scheme, the flow condition for the DLR-F15 high-lift test case is more challenging, in addition to choosing a coarse snapshot sampling scheme, the flow condition for the DLR-F15 high-lift test case.

The performance of the constrained LSQ-ROM method compared to the quadratic programming solution and the unconstrained LSQ-ROM approach is summarized in Table 5. As can be seen from this table, the constrained LSQ-ROM method as well as the quadratic programming approach are able to find the same local residual minimum and both match the imposed constraints. Again, the constrained LSQ-ROM outperforms the quadratic programming method and requires a computational effort very comparable to that of the unconstrained standard LSQ-ROM. The latter arrives at a lower residual objective but exhibits considerable relative errors in the aerodynamic coefficients. Figure 11 shows the corresponding surface residual objective but exhibits considerable relative errors in the aerodynamic coefficients.

Table 5 Computational effort and optimization performance of constrained LSQ-ROM method, quadratic programming minimization method (fminslsqp), and unconstrained LSQ-ROM method.

<table>
<thead>
<tr>
<th>Optimization method</th>
<th>C-LSQ-ROM</th>
<th>fminslsqp</th>
<th>Uc-LSQ-ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time, s</td>
<td>49.1</td>
<td>277</td>
<td>44.4</td>
</tr>
<tr>
<td>Objective function evaluations (Normalized) starting residual</td>
<td>43</td>
<td>157</td>
<td>37</td>
</tr>
<tr>
<td>Optimized residual $C_L$ (relative error)</td>
<td>2.216</td>
<td>2.216</td>
<td>0.4001</td>
</tr>
<tr>
<td>Optimized residual $C_D$ (relative error)</td>
<td>2.815 (0.0%)</td>
<td>2.815 (0.0%)</td>
<td>2.882 (2.4%)</td>
</tr>
<tr>
<td>Optimized residual $C_M$ (relative error)</td>
<td>-0.7387 (0.0%)</td>
<td>-0.7567 (0.0%)</td>
<td>-0.7768 (18.0%)</td>
</tr>
</tbody>
</table>

Both the LSQ-ROM and fminslsqp methods are subject to reproducing the lift, drag, and moment coefficients of the reference CFD solution.

$C_p$ distributions, which (surprisingly) hardly differ. The disagreement between the constrained and the unconstrained LSQ-ROM flow solutions becomes more prominent when looking at the field plots in Fig. 12 and at the aerodynamic coefficients in Table 5. Note that the corresponding quadratic programming solution (not displayed) coincides with the constrained LSQ-ROM result, yet it comes at significantly higher computational costs. With respect to the computational time, a speedup by a factor larger than 26 is gained when compared to the full-order CFD solver.
Fig. 13 DLR-F15: $C_p$ distributions at $M_{\infty} = 0.23$, $\alpha = 6.0$ deg. (Constrained $C_L = 1.12 \cdot C_{L,ref}$).

3. Subsonic Target Lift Example

Next, the target lift problem is demonstrated for the DLR-F15 configuration. As a constraint, the requirement of producing 112% of the lift of the CFD reference solution is imposed on the reduced-order model. More precisely, a target lift coefficient of $C_{L,target} = 1.12 \cdot C_{L,ref} = 1.12 \cdot 2.589 = 2.900$ is added as a constraint to LSQ-ROM optimization problem (9). The ROM approximation is conducted for a flow condition of $(\alpha, M_{\infty}) = (6.0 \text{ deg}, 0.23)$; see Fig. 10. The performance of the constrained LSQ-ROM method compared to the quadratic programming solution is summarized in Table 5. Note that the constrained LSQ-ROM method as well as the quadratic programming approach are able to find the same local residual minimum. Both match the imposed constraint of producing 112% of the lift of the reference CFD solution. Yet again, the constrained LSQ-ROM outperforms the quadratic programming method in terms of computational effort. Figure 13 shows the corresponding surface $C_p$ distributions, again demonstrating that the constrained LSQ-ROM approach results in physical pressure distributions that feature the target aerodynamic coefficients, making this a valuable tool for the designer to efficiently estimate target aerodynamic loads for a given aircraft configuration at given flow conditions based on a set of CFD solutions at a range of flow conditions.

VII. Conclusions

In this paper, a method capable of efficiently computing flowfield approximations subject to aerodynamic constraints, called the constrained LSQ-ROM approach, was developed. The method subordinates to the context of proper-orthogonal-decomposition-based reduced-order modeling. It has been pointed out that the inherent computational complexity scales linearly in the grid size. For the NACA 64A010 airfoil, as well as for the DLR-F15 high-lift configuration considered in the Results section (Sec. VI), the constrained LSQ-ROM performed one order of magnitude faster than the full-order CFD-model. For three-dimensional configurations, an even larger speedup factor is expected.

Even though the reduced-order modeling aspect of the constrained LSQ-ROM method is very appealing, the major benefit is actually that the approach permits tackling aeroloads prediction problems. As examples in this regard, the data fusion and the target aerodynamics process have been discussed, and the principle capabilities of the constrained LSQ-ROM approach have been demonstrated on a transonic test case and a subsonic high-lift test case. A comparison of different methods for solving the optimization problem occurring in the formulation of the constrained LSQ-ROM yielded that the constrained Gauss–Newton algorithm proposed here is the most efficient.

Note that it has been confirmed both theoretically and practically that the constrained LSQ-ROM performs almost as fast as the well-established unconstrained LSQ-ROM, which has previously been applied to industrial aircraft configurations by the authors.

The constrained LSQ-ROM is also related to the gappy-POD method, which has previously been used by the authors for data fusion purposes. The approach suggested herein, however, applies under different conditions in terms of the number of available CFD snapshot solutions and the number of constraints, and hence does not replace the gappy POD but rather complements it.

Appendix A. Derivation for Search Direction in Constrained Gauss–Newton Scheme

According to Eq. (11), it is sufficient to compute the last $\tilde{m}$ rows of the inverse of the system matrix in order to compute the search direction $d^k$, which is given precisely by the last $\tilde{m}$ entries of the vector of unknowns in system (11). Let the matrices $R, F, S,$ and $T$ be as introduced in Sec. IV.A. According to Eq. (12), the required last $\tilde{m}$ rows are given by the last $\tilde{m}$ rows of the subblocks $R^{-1}F$ and $R^{-1} + R^{-1}FSF^TR^{-1}$.

Blockwise inversion ([23] p. 70, Eq. (4.4)) of the matrix $R$ leads to

$$R^{-1} = \left( \Omega^{-1} - \Omega^{-1}DTDF^T \Omega^{-1} \Omega^{-1}DT \right)^{-1}TDG^T = \begin{pmatrix} R_{11}^{-1} & R_{12}^{-1} \\ R_{21}^{-1} & R_{22}^{-1} \end{pmatrix} \begin{pmatrix} 0 & \tilde{S} \\ Dg^T & 0 \end{pmatrix}$$

As a consequence, the last $\tilde{m}$ rows of

$$R^{-1}FS = \begin{pmatrix} R_{11}^{-1} & R_{12}^{-1} \\ R_{21}^{-1} & R_{22}^{-1} \end{pmatrix} \begin{pmatrix} 0 \\ Dg^T \end{pmatrix}$$

are given by $-TDg^T$. Making use of Eq. (14) once more, the last $\tilde{m}$ rows of $R^{-1} + R^{-1}FSF^T R^{-1}$ are given by $(I + TDg^TSDg)TDg^T \Omega^{-1} - (I + TDg^TSDg)T)$. As a consequence, the search direction is obtained via multiplying the last $\tilde{m}$ rows of the inverse of the system matrix times the right-hand side; that is,

$$d^k = (-TDg^T S((I + TDg^TSDg)TDg^T \Omega^{-1}) - (I + TDg^TSDg)T) - g$$

which yields Eq. (13).

Acknowledgments

The work presented in this paper uses and extends the Surrogate Modeling for Aero-data Toolbox, a code package developed by the Aero-Loads Prediction Group of the DLR, German Aerospace Center, with contributions from the Institute ‘Computational Mathematics’ of the Technische Universität Braunschweig. The authors would like to thank their DLR colleague Michael Mifsud for providing the DLR-F15 high-lift test-case snapshots.

References


Queries

1. AU: Please review the revised proof carefully to ensure your corrections have been inserted properly and to your satisfaction.