Resolvent Analysis using Subspace Algorithms for Aerodynamic Applications

U S Vevek* and Sebastian Timme†
University of Liverpool, Liverpool, England L69 3GH, United Kingdom

Resolvent analysis has become ubiquitous in aerodynamic studies to identify strong modal behaviour through optimal forcing/response pairs and their associated gains. Such dominant modes can be used, for instance, for the understanding of flow physics and the construction of reduced order models. When interest is in non-canonical three-dimensional test cases of practical interest, earlier computational limitations, when solving the underlying truncated singular value decomposition for a small number of the most dominant modes using matrix forming methods with direct matrix factorisation, have been overcome with either efficient time-stepping methods or sparse iterative solver technology within linear harmonic incarnations of production computational fluid dynamics codes. For the latter iterative methods, besides the so-called randomised singular value decomposition, either single-vector iterations with deflation to identify next-in-line optimal modes or subspace methods with random starting vectors have shown great potential. When using subspace methods, the optimal modes are often computed by adapting algorithms designed for solving eigenvalue problems, but these algorithms yield either the forcing or response modes leaving the other to be determined separately incurring additional computational cost. Herein we will investigate two specialised algorithms for singular value decomposition that allows computation of both sets of modes simultaneously. One is a Krylov subspace method referred to as the Krylov–Golub–Kahan algorithm that benefits from a simpler and more robust deflation procedure compared to the widely used implicitly restarted Arnoldi method. The other is an inverse subspace iteration method referred to as the inexact subspace iteration algorithm that uses cheap, low-accuracy linear solutions. The subspace methods were applied for a two-dimensional circular cylinder and the NASA Common Research Model. Both subspace methods were able to compute the three most dominant forcing-gain-response singular triplets for the circular cylinder problem to the desired tolerance while the single-vector iteration algorithm stalled for the third triplet. The inexact subspace iteration algorithm was found to be more efficient than the Krylov–Golub–Kahan algorithm. As for the NASA Common Research Model problem, only the inexact subspace iteration algorithm was able to achieve the desired convergence for the three most dominant singular triplets.

I. Introduction

The resolvent matrix is a linear operator that maps a harmonic input forcing to a harmonic output response [1]. The purpose of resolvent analysis in aerodynamic applications is to determine the optimal forcings that result in the most amplified responses. The extraction of such dominant modes can be useful, for instance, in the understanding of the leading flow physics and subsequently in the construction of low-rank reduced order models. Due to the non-normality of the linearised (Reynolds-averaged) Navier–Stokes equations, a resolvent analysis might be more revealing than a global stability analysis [2] as non-normal systems may exhibit significant pseudo-resonances even when forced at frequencies that are not necessarily close to the leading eigenvalues. Previously, time-stepping algorithms have been introduced in [3] and refined in [4] to solve the underlying truncated singular value decomposition (TSVD) for the optimal solution. Instead of integrating the direct governing equations forward and the adjoint equations backward in time until periodic states (subject to appropriate harmonic forcing) are reached to extract said optimal forcing/response modes upon convergence, linear harmonic solvers, now well established in many production computational fluid dynamics codes [5–7], can solve for the harmonic content immediately, avoiding the time-stepping through multiple cycles until a periodic state is converged. Key are efficient iterative linear solvers that have been exercised extensively e.g. when solving the related triglobal eigenvalue problem for stability [8].

*Postdoctoral Research Associate, School of Engineering.
†Reader, School of Engineering, sebastian.timme@liverpool.ac.uk. Member AIAA.

While single-vector iterative methods [9] are often sufficient when interest is in the first optimal solution, there can be scenarios where sub-optimal solutions are also sought, e.g. competing optimal modes on full-span aircraft problems, sub-optimal modes related to additional important physics, cases of mode switching, etc. Then, subspace methods are very attractive. In [10, 11] the TSVD problem was recast into a Hermitian eigenvalue problem and solved using well-established Krylov subspace methods, often encountered when dealing with eigenvalue problems, to obtain the optimal forcing mode which is then the leading eigenvector. However, determining the corresponding optimal response required solving an additional linear system. Another approach is that in [12] that addresses the TSVD problem directly using a randomised subspace method but, although not strictly necessary, they suggested solving the additional linear system to obtain the optimal response mode from the optimal forcing mode for improved accuracy. In this work, we consider two different subspace methods specialised for the character of the TSVD problem in that they do not require the additional linear solution. The first method is a slight variation of the Krylov–Schur SVD algorithm that was introduced in [13] and the second method is an inexact (inverse) subspace iteration algorithm inspired by the inexact inverse iteration algorithm proposed in [14] for generalised eigenvalue problems that has been tailored for computing the TSVD. Both iterative methods involve the application of matrix inverses on vectors in every iteration. For problems of moderate size (e.g. those arising from two-dimensional computational fluid dynamics cases), matrices can be factorised directly, but this becomes inefficient, impractical or outright prohibitive for sparse matrices arising from spatial discretisations of large scale three-dimensional flow problems. Therefore, one must resort to iterative linear solvers to effect the application of the matrix inverses. This leads to the concept of inexact inner-outer iterative methods whereby the inner part pertains to the iterative linear solver and the outer part to the iterative method for computing the TSVD problem itself. In the context of inner-outer iterative methods [15, 16], the term *inexact* is used to describe the approximate nature of the iterative linear solver which only solves the linear problems to a certain (inner) tolerance. Hence, both subspace methods discussed herein can be classified as inexact inner-outer iterative methods. However, we reserve the term *inexact* to refer to a low accuracy linear solution with a large inner tolerance (e.g., 10^{-1}) and distinguish it from a high accuracy linear solution with a small (or *deep*) inner tolerance (e.g., 10^{-8}).

We will continue with the pertinent theory in Section II wherein both algorithms will be described in detail, followed by the specific numerical methods in Section III. Results can be found in Section IV and conclusions in Section V. We will discuss the ubiquitous circular cylinder in laminar flow as a verification test case and a practical large aircraft case in high Reynolds number, turbulent and transonic flow to demonstrate the potential of the methods.

II. Theory

A. The Resolvent Problem

To derive the resolvent problem, we begin with the spatially-discretised (Reynolds-averaged) Navier–Stokes equations

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} + \boldsymbol{R}(\boldsymbol{w}) = \boldsymbol{f}_{\text{ext}}, \tag{1}$$

where $\boldsymbol{w}$ represents the vector of conservative variables, specifically fluid density ρ, Cartesian momentum components $\rho\boldsymbol{u}$, total energy ρE and additional variables resulting from turbulence modelling and similar, and $\boldsymbol{R}(\boldsymbol{w})$ represents the corresponding non-linear residual functions in discretised form. The vector $\boldsymbol{f}_{\text{ext}}$ represents any external forcing that may be present. Expanding the residual vector in Eq. (1) using a Taylor series about a steady-state (or mean-flow) solution $\bar{\boldsymbol{w}}$ results in

$$\frac{\mathrm{d}\tilde{\boldsymbol{w}}}{\mathrm{d}t} = J\tilde{\boldsymbol{w}} + \tilde{\boldsymbol{f}}, \tag{2}$$

where $\tilde{\boldsymbol{w}}$ denotes the time-varying perturbations in the fluid state (i.e., $\boldsymbol{w}(t) = \bar{\boldsymbol{w}} + \tilde{\boldsymbol{w}}(t)$) and $J = -\partial\boldsymbol{R}/\partial\boldsymbol{w}$ is the Jacobian operator. The higher order terms and external forcing are collected into a single vector, $\tilde{\boldsymbol{f}} = \tilde{\boldsymbol{f}}(t)$, on the right-hand side. The higher order terms can be viewed as the forcing due to non-linearities [17].

For the case of harmonic forcing at a single frequency ω causing the fluid state to respond with harmonic oscillations at (and only at) the same frequency, we can express the time variation of the forcing and response as $\tilde{\boldsymbol{f}}(t) = \hat{\boldsymbol{f}}e^{i\omega t}$ and $\tilde{\boldsymbol{w}}(t) = \hat{\boldsymbol{w}}e^{i\omega t}$, respectively. Substituting these two expressions into Eq. (2) and simplifying lead to

$$\hat{\boldsymbol{w}} = \mathcal{R}\hat{\boldsymbol{f}}, \tag{3}$$

where $\mathcal{R} = (i\omega I - J)^{-1}$ is the resolvent operator. Equation (3) represents an input-output relationship between the forcing (input) and the fluid response (output) at frequency ω. The objective now is to determine the optimal forcing(s)

that results in the most amplified response(s). This is achieved using a TSVD of the resolvent operator $\mathcal{R}F_k = W_k\Sigma_k$ for the k most dominant modes with the largest singular values. The k column vectors of F_k and W_k correspond to the forcing and response modes, respectively, and elements of the k-by-k diagonal matrix Σ_k represent the gains for the forcing-response pairs. The forcing and response modes are normalised herein with respect to the diagonal matrix of discrete cell volumes, Q. Defining the inner product (for arbitrary vectors $\boldsymbol{a}$ and $\boldsymbol{b}$) with respect to the positive definite matrix Q as $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^H Q \boldsymbol{b}$, the normalisation can be written as $\langle F_k, F_k \rangle = \langle W_k, W_k \rangle = I_k$ where I_k is the k-by-k identity matrix. We also introduce here the notation for the Q-norm $\|\boldsymbol{a}\| = \sqrt{\langle \boldsymbol{a}, \boldsymbol{a} \rangle}$ for later use. The adjoint of the resolvent operator $\mathcal{R}^\dagger$ is defined by the relationship $\langle \boldsymbol{a}, \mathcal{R}\boldsymbol{b} \rangle = \langle \mathcal{R}^\dagger \boldsymbol{a}, \boldsymbol{b} \rangle$ and it can be explicitly written as $\mathcal{R}^\dagger = (-i\omega I - J^\dagger)^{-1}$ where $J^\dagger = Q^{-1} J^T Q$. For convenience, let us denote $(-i\omega I - J^\dagger) = (i\omega I - J)^\dagger$. The TSVD with respect to the adjoint can be written as $\mathcal{R}^\dagger W_k = F_k \Sigma_k$.

B. Proposed Algorithms

Convergence and Locking

Prior to discussing the algorithms, we introduce the outer error norm for the jth forcing-gain-response triplet

$$\epsilon_j = \max(\|\boldsymbol{f}_j - \sigma_j (i\omega I - J)\boldsymbol{w}_j\|, \|\boldsymbol{w}_j - \sigma_j (i\omega I - J)^\dagger \boldsymbol{f}_j\|), \tag{4}$$

used to assess convergence. The same error norm definition, although expensive to compute, is used for all algorithms for a fair comparison. A singular triplet is deemed to have converged sufficiently when its error norm drops below a certain user-defined (outer) convergence tolerance. This check is performed in the `HasConverged()` function that will appear in all the three algorithms to be discussed next. The converged vectors are then locked and further manipulations are applied only from the next vector onward. The locked vectors participate in the orthogonalisations only.

Single Vector Iteration Algorithm

The single vector iteration (SVI) algorithm introduced in [9, 18] and implemented in the current code, is summarised in Algorithm 1. It is a simple inverse iteration algorithm with deflation that solves for multiple singular triplets of the resolvent operator $\mathcal{R}$. The triplets converge sequentially in descending order of the singular values. In other words, the triplet with the largest singular value converges first, followed by the triplet with the second largest singular value and so on. The achievable outer accuracy is determined by the inner convergence tolerance ϵ_{inner} for the linear systems. For each singular triplet, ϵ_{inner} is initially set to a large value (e.g., 10^{-3}) to save computational cost. It is then gradually reduced, synchronised with the outer error norm, until the linear systems are solved deeply close to convergence to achieve the desired outer accuracy.

Krylov–Golub–Kahan Algorithm

The Krylov–Golub–Kahan (KGK) factorisation [13] for the resolvent problem can be expressed as

$$\begin{aligned} \mathcal{R}F_m &= W_m \Sigma_m, \\ \mathcal{R}^\dagger W_m &= F_m \Sigma_m + \boldsymbol{f}_{m+1} \boldsymbol{\alpha}_m^T, \end{aligned} \tag{5}$$

where $F_{m+1} = [F_m \quad \boldsymbol{f}_{m+1}]$ and W_m are Q-orthogonal matrices, Σ_m is an m-by-m diagonal matrix with its positive and real elements arranged in descending order, and $\boldsymbol{\alpha}_m^T$ is a 1-by-m real row vector. It can be readily seen that when the first k elements of $\boldsymbol{\alpha}_m^T$ vanish, we obtain the sought-after TSVD of the resolvent operator for the k most dominant singular triplets. The presented algorithm, henceforth referred to as the KGK algorithm, is a Krylov subspace method that relies on the KGK factorisation. It consists of the two main steps of expansion and truncation.

The expansion proceeds as follows. Given the factorisation in Eq. (5), we compute new Krylov vectors $\boldsymbol{w}_{m+1}$ and $\boldsymbol{f}_{m+2}$ by ortho-normalising the vectors $\mathcal{R}\boldsymbol{f}_{m+1}$ and $\mathcal{R}^\dagger \boldsymbol{w}_{m+1}$ over the existing Krylov vectors in W_m and F_{m+1}, respectively. The operations $\mathcal{R}\boldsymbol{f}_{m+1}$ and $\mathcal{R}^\dagger \boldsymbol{w}_{m+1}$ are performed by solving the large, sparse linear problems $(i\omega I - J)\boldsymbol{y} = \boldsymbol{f}_{m+1}$ and $(i\omega I - J)^\dagger \boldsymbol{y} = \boldsymbol{w}_{m+1}$, respectively. These two operations are the costliest in the entire algorithm, particularly when dealing with large three-dimensional spatial discretisations, since the linear problems have to be solved deeply to ensure that the matrix inverses have been applied with sufficient accuracy. The special relationship between W_m and F_{m+1}

Algorithm 1 Single vector iteration algorithm

Inputs:
number of desired singular triplets k, maximum number of iterations n
Initialise:
$F = [], W = [], \boldsymbol{\sigma} = []$
for $j = 1$ to k **do**
  Initialise $\boldsymbol{f}_j$ to a random vector
  Deflate $\boldsymbol{f}_j := \boldsymbol{f}_j - F_{1:j-1}\langle F_{1:j-1}, \boldsymbol{f}_j\rangle$
  Normalise $\boldsymbol{f}_j := \boldsymbol{f}_j / \|\boldsymbol{f}_j\|$
  for $i = 1$ to n **do**
    Solve $(i\omega I - J)\boldsymbol{w}_j = \boldsymbol{f}_j$ inexactly based on ϵ_j
    Deflate $\boldsymbol{w}_j := \boldsymbol{w}_j - W_{1:j-1}\langle W_{1:j-1}, \boldsymbol{w}_j\rangle$
    Normalise $\boldsymbol{w}_j := \boldsymbol{w}_j / \|\boldsymbol{w}_j\|$

    Solve $(i\omega I - J)^\dagger \boldsymbol{f}_j = \boldsymbol{w}_j$ inexactly based on ϵ_j
    Deflate $\boldsymbol{f}_j := \boldsymbol{f}_j - F_{1:j-1}\langle F_{1:j-1}, \boldsymbol{f}_j\rangle$
    Compute singular value $\sigma_j = \|\boldsymbol{f}_j\|$
    Normalise $\boldsymbol{f}_{j+1} := \boldsymbol{f}_{j+1} / \sigma_j$

    if HasConverged($\boldsymbol{f}_j$, $\boldsymbol{w}_j$, σ_j) **then**
      Store $W := [W \quad \boldsymbol{w}_j]$, $F := [F \quad \boldsymbol{f}_j]$, $\boldsymbol{\sigma} = [\boldsymbol{\sigma}; \quad \sigma_j]$
      break for
    end if
  end for
end for

established in Eq. (5) leads to the following expressions to compute the new Krylov vectors

$$\begin{aligned} \beta \boldsymbol{w}_{m+1} &= \mathcal{R}\boldsymbol{f}_{m+1} - W_m \boldsymbol{\alpha}_m, \\ \gamma \boldsymbol{f}_{m+2} &= \mathcal{R}^\dagger \boldsymbol{w}_{m+1} - \beta \boldsymbol{f}_{m+1}, \end{aligned} \tag{6}$$

where the constants β and γ are determined from the normalisation conditions, $\|\boldsymbol{w}_{m+1}\| = \|\boldsymbol{f}_{m+2}\| = 1$. In Eq. 6, the coefficients $\boldsymbol{\alpha}_m$ and β computed earlier are used to calculate the new Krylov vectors but in finite precision arithmetic this leads to a loss of orthogonality. Therefore, in practice, the vectors $\mathcal{R}\boldsymbol{f}_{m+1}$ and $\mathcal{R}^\dagger \boldsymbol{w}_{m+1}$ are orthogonalised against *all* previous Krylov vectors to eliminate any round-off errors that might have been introduced and the resulting vectord sre normalised to obtain the constants β and γ. At this point, the Krylov vectors are related to each other as follows.

$$\begin{aligned} \mathcal{R}F_{m+1} &= W_{m+1}\begin{bmatrix} \Sigma_m & \boldsymbol{\alpha}_m \\ & \beta \end{bmatrix} = W_{m+1}\Gamma_{m+1}, \\ \mathcal{R}^\dagger W_{m+1} &= F_{m+1}\begin{bmatrix} \Sigma_m & \\ \boldsymbol{\alpha}_m^T & \beta \end{bmatrix} + \gamma \boldsymbol{f}_{m+2}\boldsymbol{e}_{m+1}^T = F_{m+1}\Gamma_{m+1}^T + \gamma \boldsymbol{f}_{m+2}\boldsymbol{e}_{m+1}^T. \end{aligned} \tag{7}$$

The final part of the expansion step involves returning the above relationships to the same form as those in Eq. (5). To this end, we compute the SVD of the small matrix Γ_{m+1} as $\Gamma_{m+1} = \Phi_{m+1}\Sigma_{m+1}\Psi_{m+1}^T$. Substituting this SVD into Eq. (7) and multiplying (from the right) the first expression with Ψ_{m+1} and the second expression with Φ_{m+1} lead to

$$\begin{aligned} \mathcal{R}(F_{m+1}\Psi_{m+1}) &= (W_{m+1}\Phi_{m+1})\Sigma_m, \\ \mathcal{R}^\dagger(W_{m+1}\Phi_{m+1}) &= (F_{m+1}\Psi_{m+1})\Sigma_m + \boldsymbol{f}_{m+2}(\gamma \boldsymbol{e}_{m+1}^T \Phi_{m+1}). \end{aligned} \tag{8}$$

Setting $F_{m+1} := F_{m+1}\Psi_{m+1}$, $W_{m+1} := W_{m+1}\Phi_{m+1}$ and $\boldsymbol{\alpha}_{m+1} = \gamma \boldsymbol{e}_{m+1}^T \Phi_{m+1}$ in Eq. (8) yields the KGK factorisation and concludes the expansion step. The starting vector $\boldsymbol{f}_1$ is initialised as a random vector such that $\|\boldsymbol{f}_1\| = 1$.

Since the Krylov subspaces cannot be extended indefinitely due to memory constraints, we need to suitably truncate the subspaces and the KGK factorisation naturally lends itself to such truncations. Taking inspiration from the

Algorithm 2 Krylov–Golub–Kahan algorithm

Inputs:
number of desired singular triplets k, size to truncate Krylov subspace to for restart $t \geq k$,
maximum size of Krylov subspace m
Initialise:
random vector $\boldsymbol{f}_1$ normalised such that $\|\boldsymbol{f}_1\| = 1$
$F = [\boldsymbol{f}_1]$, $W = []$, $\boldsymbol{\sigma} = []$, $\boldsymbol{\alpha} = []$, $j = 1$, $l = 0$
while $l < k$ **do**
  Solve $(i\omega I - J)\boldsymbol{w} = \boldsymbol{f}_j$ deeply
  Orthogonalise $\boldsymbol{w} := \boldsymbol{w} - W_{1:j-1}\langle W_{1:j-1}, \boldsymbol{w}\rangle$
  Compute norm $\beta = \|\boldsymbol{w}\|$
  Normalise $\boldsymbol{w} := \boldsymbol{w}/\beta$
  Store $W := [W \quad \boldsymbol{w}]$

  Solve $(i\omega I - J)^\dagger \boldsymbol{f} = \boldsymbol{w}_j$ deeply
  Orthogonalise $\boldsymbol{f} := \boldsymbol{f} - F_{1:j}\langle F_{1:j}, \boldsymbol{f}\rangle$
  Compute norm $\gamma = \|\boldsymbol{f}\|$
  Normalise $\boldsymbol{f} := \boldsymbol{f}/\gamma$
  Store $F := [F \quad \boldsymbol{f}]$

  Set $\Gamma = \begin{bmatrix} \mathrm{diag}(\boldsymbol{\sigma}_{l+1:j-1}) & \boldsymbol{\alpha}_{l+1:j-1} \\ & \beta \end{bmatrix}$ and compute its SVD $\Gamma = \Phi\Sigma\Psi^T$
  Update $F_{l+1:j} := F_{l+1:j}\Psi$, $W_{l+1:j} := W_{l+1:j}\Phi$, $\boldsymbol{\sigma}_{l+1:j} = \mathrm{diag}(\Sigma)$ and $\boldsymbol{\alpha}_{l+1:j} = \gamma\Phi^T\boldsymbol{e}_{j-l}$

  for $i = l$ to k **do**
    if HasConverged($\boldsymbol{f}_i$, $\boldsymbol{w}_i$, σ_i) **then**
      Increment number of converged singular triplets $l := l + 1$
    end if
  end for

  $j := j + 1$
  if $j > m$ **then**
    Copy $\boldsymbol{f}_{m+1}$ to $\boldsymbol{f}_{t+1}$
    Set $j = t + 1$
  end if
end while

Krylov–Schur algorithm devised in [19] for solving large eigenvalue problems, it was proposed in [13] to split the KGK factorisation in Eq. (5) into two parts as

$$\begin{aligned} \mathcal{R}[F_t \quad F_{m-t}] &= [W_t \quad W_{m-t}] \begin{pmatrix} \Sigma_t & \\ & \Sigma_{m-t} \end{pmatrix}, \\ \mathcal{R}^\dagger[W_t \quad W_{m-t}] &= [F_t \quad F_{m-t} \quad \boldsymbol{f}_{m+1}] \begin{pmatrix} \Sigma_t & \\ & \Sigma_{m-t} \\ \boldsymbol{\alpha}_t^T & \boldsymbol{\alpha}_{m-t}^T \end{pmatrix}, \end{aligned} \tag{9}$$

and discard the second part (terms with subscript $m - t$) to obtain a smaller KGK factorisation. Since the entries of the diagonal matrix Σ_m have been arranged in descending order, discarding the second part eliminates the subspace vectors corresponding to the smallest $m - t$ singular values in Σ_m while retaining those corresponding to the largest t singular values in Σ_m which we are interested in. This concludes the truncation step and the expansion can now be restarted with $\boldsymbol{f}_{t+1} = \boldsymbol{f}_{m+1}$. The simplicity of the restart procedure is the main advantage of working with the KGK (and Krylov–Schur) factorisation compared with implicitly restarted methods [20] that use the implicitly

Algorithm 3 Inexact subspace iteration algorithm

1: **Inputs:**
number of desired singular triplets k, size of subspace $m \geq k$
2: **Initialise:**
random vectors $F_m = [\boldsymbol{f}_1 \quad \dots \quad \boldsymbol{f}_m]$ such that $\langle F_m, F_m \rangle = I_m$
$W_m = 0$, $\boldsymbol{\sigma} = 0$, $l = 0$
3: **while** $l < k$ **do**
4: **for** $j = l + 1$ to m **do**
5: Set $\boldsymbol{w}_j := \sigma_j \boldsymbol{w}_j$
6: Solve $(i\omega I - J)\Delta\boldsymbol{w} = \boldsymbol{f}_j - (i\omega I - J)\boldsymbol{w}_j$ inexactly
7: Update $\boldsymbol{w}_j := \boldsymbol{w}_j + \Delta\boldsymbol{w}$
8: Orthogonalise $\boldsymbol{w}_j := \boldsymbol{w}_j - W_{1:j-1}\langle W_{1:j-1}, \boldsymbol{w}_j \rangle$
9: Normalise $\boldsymbol{w}_j := \boldsymbol{w}_j / \|\boldsymbol{w}_j\|$
10: **end for**
11: Compute $\Xi = \langle F_{l+1:m}, (i\omega I - J)W_{l+1:m} \rangle$
12: Compute SVD of $\Xi = \Psi \Sigma^{\Xi} \Phi^H$ with elements of Σ^{Ξ} arranged in ascending order
13: Update $F_{l+1:m} := F_{l+1:m}\Psi$, $W_{l+1:m} := W_{l+1:m}\Phi$, $\boldsymbol{\sigma}_{l+1:m} = \mathrm{diag}((\Sigma^{\Xi})^{-1})$

14: **for** $j = l + 1$ to m **do**
15: Set $\boldsymbol{f}_j := \sigma_j \boldsymbol{f}_j$
16: Solve $(i\omega I - J)^{\dagger}\Delta\boldsymbol{f} = \boldsymbol{w}_j - (i\omega I - J)^{\dagger}\boldsymbol{f}_j$ inexactly
17: Update $\boldsymbol{f}_j := \boldsymbol{f}_j + \Delta\boldsymbol{f}$
18: Orthogonalise $\boldsymbol{f}_j := \boldsymbol{f}_j - F_{1:j-1}\langle F_{1:j-1}, \boldsymbol{f}_j \rangle$
19: Normalise $\boldsymbol{f}_j := \boldsymbol{f}_j / \|\boldsymbol{f}_j\|$
20: **end for**
21: Repeat steps 11-13.

22: **for** $i = l$ to k **do**
23: **if** HasConverged($\boldsymbol{f}_i$, $\boldsymbol{w}_i$, σ_i) **then**
24: Increment number of converged singular triplets $l := l + 1$
25: **end if**
26: **end for**
27: **end while**

shifted QR algorithm [21] to retain the relevant subspace information. Similar to the SVI algorithm earlier, the outer convergence accuracy of the KGK algorithm (and in fact, any Krylov subspace method) is limited by the inner convergence tolerance. While it is possible to use relaxation strategies [22, 23] to reduce the inner tolerance as the iterations progress, a small inner tolerance is nonetheless required in the initial iterations. The inexact subspace iteration algorithm, which will be derived in the next section, was developed expressly to avoid this limitation.

The complete algorithm is summarised in Algorithm 2 where we have used MATLAB notation with the indexing starting at 1. The main difference between the KGK algorithm proposed herein and the Krylov–Schur SVD algorithm proposed in [13] is that the former does not alternate between the bidiagonal and KGK factorisations during restarts. The bidiagonal factorisation allows a three-term recurrence relationship for computing the new Krylov vectors but round-off errors eventually result in loss of orthogonality requiring additional orthogonalisation [24]. Since we are concerned with rather small subspace sizes, the cost of orthogonalisation is insignificant compared to the cost of the two deep linear solutions in every iteration. Hence, the bidiagonal factorisation is omitted for simplicity and a full orthogonalisation is adopted for robustness.

Inexact Subspace Iteration Algorithm

An inexact inverse iteration algorithm for solving for generalised eigenvalue problems was introduced in [14] whereby *corrections* to current approximations are sought. The main advantage of this approach is that the inner tolerance dictates only the rate of convergence and not the final achievable accuracy. Consequently, low accuracy, inexact

linear solutions were sufficient to achieve any desired outer convergence tolerance. The inexact subspace iteration (ISI) algorithm to be derived next can be viewed as an extension of their inexact inverse iteration algorithm specialised for the TSVD problem. Rewriting the TSVD of the resolvent operator as $(i\omega I - J)W_m^\Sigma = F_m$ with $W_m^\Sigma \equiv W_m\Sigma_m$, we use the current approximations to the singular solutions, W_m^Σ and F_m, to derive the correction equation for W_m^Σ as

$$(i\omega I - J)\Delta W_m^\Sigma = F_m - (i\omega I - J)W_m^\Sigma, \tag{10}$$

which is a set of m independent linear systems to be solved inexactly. As the approximations converge towards the exact solutions, the residual term on the right-hand-side of Eq. (10) and, consequently, the correction, ΔW_m^Σ, both approach zero. Whereas the KGK algorithm requires two deep linear solutions per iteration, the ISI algorithm requires $2m$ inexact linear solutions per iteration. The size of the subspace m must be chosen to be greater than (or equal) to the number of desired singular triplets k. The choice of $m = 2k$ was found to be adequate for the cases considered in this work.

After solving the linear systems in Eq. (10), the current approximation to W_m^Σ is updated as $W_m^\Sigma := W_m^\Sigma + \Delta W_m^\Sigma$. Then, W_m^Σ is orthogonalised using the QR decomposition [25] to obtain the updated approximation for W_m as $W_m^\Sigma = W_m\Theta_m$ where Θ_m is a m-by-m upper-triangular matrix that eventually converges to Σ_m. The orthogonalisation is performed such that the normalisation condition $\langle W_m, W_m\rangle = I_m$ is satisfied. With the updated approximation to W_m, we can accelerate convergence by using the Rayleigh–Ritz procedure to form optimal linear combinations $W_m\Phi_m$ and $F_m\Psi_m$ that satisfy the expression

$$(i\omega I - J)(W_m\Phi_m)\Sigma_m = (F_m\Psi_m) + E_m, \tag{11}$$

with the Galerkin condition imposed on the error matrix E_m such that $\langle F_m, E_m\rangle = 0$. Taking the inner product with F_m on both sides and applying the Galerkin condition, we arrive at

$$\langle F_m, (i\omega I - J)W_m\rangle = \Psi_m\Sigma_m^{-1}\Phi_m^H, \tag{12}$$

where we have used the normalisation condition $\langle F_m, F_m\rangle = I_m$. Denoting $\Xi_m = \langle F_m, (i\omega I - J)W_m\rangle$, we compute its SVD as $\Xi_m = \Psi_m\Sigma_m^\Xi\Phi_m^H$ which yields the orthogonal matrices Ψ_m and Φ_m used to compute the optimal linear combinations. The approximate singular values of the resolvent operator are obtained as $\Sigma_m = (\Sigma_m^\Xi)^{-1}$. The elements of Σ_m must be rearranged in descending order and the columns of Ψ_m and Φ_m must be reversed to match the reordering before updating the current approximations as $W_m := W_m\Phi_m$ and $F_m := F_m\Psi_m$.

A similar procedure can be derived from the TSVD for the adjoint operator $(i\omega I - J)^\dagger F_m^\Sigma = W_m$ where $F_m^\Sigma = F_m\Sigma_m$. First, we solve the set of m independent linear systems

$$(i\omega I - J)^\dagger\Delta F_m^\Sigma = W_m - (i\omega I - J)^\dagger F_m^\Sigma, \tag{13}$$

inexactly to obtain the corrections for F_m^Σ. Next, we update the current approximation to F_m^Σ as $F_m^\Sigma := F_m^\Sigma + \Delta F_m^\Sigma$ and orthogonalise it as $F_m^\Sigma = F_m\Theta_m$ with $\langle F_m, F_m\rangle = I_m$. Lastly, we compute optimal linear combinations of the columns of F_m and W_m from the SVD of Ξ_m as before. This also produces an updated approximation Σ_m.

The procedure described thus far is performed in every outer iteration of the algorithm. At the beginning of the algorithm, the initial approximation F_m is set to random vectors orthogonalised $\langle F_m, F_m\rangle = I_m$ and the initial approximation $W_m = 0$. The algorithm is summarised in Algorithm 3.

III. Methodology

A. Non-linear Solver

The industrial DLR-TAU code solves the Reynolds-averaged Navier–Stokes equations (plus a suitable turbulence closure model) using a second-order, finite-volume, vertex-centred discretisation [26]. The turbulence closure is provided herein through the negative Spalart–Allmaras model using the Boussinesq eddy-viscosity hypothesis. The inviscid fluxes are computed using a central scheme with matrix artificial dissipation. The gradients of flow variables for viscous fluxes (and source terms of the turbulence model) are evaluated following the Green–Gauss theorem. The far-field boundary is described as free-stream flow through a characteristic boundary condition while the no-slip condition on viscous walls is enforced strongly. A steady-state solution is calculated via the implicit backward Euler method with lower-upper symmetric Gauss–Seidel iterations and local time-stepping. A geometric multi-grid method, typically on three multi-grid levels, is also used to improve convergence rates.

B. Linear Harmonic Solver

The resolvent algorithms require solving linear problems of the general form $(i\omega I - J)\boldsymbol{x} = \boldsymbol{b}$ and $(i\omega I - J)^\dagger \boldsymbol{x} = \boldsymbol{b}$. Even though the Jacobian operator is purely real-valued, we are dealing with complex-valued linear systems of equations. The generalised conjugate residual solver with inner orthogonalisation and deflated restarting (GCRO-DR) [27] with preconditioning was used to solve the linear problems. The GCRO-DR solver is a Krylov subspace method that only needs the matrix-vector products $(i\omega I - J)\boldsymbol{a}$ and $(i\omega I - J)^\dagger \boldsymbol{a}$ for an arbitrary complex vector $\boldsymbol{a}$. The Jacobian matrix J and its adjoint $J^\dagger$ are explicitly formed from a hand-derived, analytical formulation, whereas, generally speaking, matrix-free matrix-vector products, enabled through finite-difference schemes or automatic differentiation, would fulfill the same purpose. An incomplete lower-upper factorisation of $(i\omega I - J)$ or $(i\omega I - J)^\dagger$ is used for preconditioning. In parallel computations, only the (MPI-) process-local blocks are factorised. Unlike a standard restarted generalised minimal residual (GMRES) solver, the GCRO-DR solver recycles subspace vectors between restarts and, in principle, can also recycle when solving varying right-hand-side sequences of linear problems. The recycled vectors (deflation vectors) are constructed from the existing Krylov subspace to approximate the interior eigenvectors of the underlying coefficient matrix, i.e. eigenvectors associated with the smallest eigenvalues by magnitude. This enables the GCRO-DR solver to avoid the stalling often observed when using the GMRES solver for particularly stiff cases [28]. The recycling usually speeds up convergence, too. A maximum of 100 Krylov vectors and 20 deflation vectors were used throughout for the cases considered in this paper.

IV. Results

All results are given in non-dimensional form unless explicitly stated otherwise.

A. Circular cylinder

The widely documented case of laminar flow past a circular cylinder is used for verification of the two subspace algorithms introduced in this work. The flow condition is for a subcritical Reynolds number of $Re = 40$ and a subsonic Mach number of $M = 0.2$. The case is computed on a relatively coarse grid of about 10 000 control volumes. The cylinder has a diameter of $D = 1$ and is located at the origin. The computational domain extends $100D$ from the origin in the radial direction. The steady-state computation was first performed with a convergence tolerance of 10^{-12}. Then, linearised stability analysis was performed with the steady-state solution as base flow to determine a suitable frequency for resolvent analysis. In the limiting case that the complex frequency $i\omega$ approaches an eigenvalue λ of the Jacobian matrix J, the resolvent operator $(i\omega I - J)^{-1}$ becomes singular and the optimal forcing and response modes, respectively, approach the adjoint and direct eigenmodes of J corresponding to λ with the maximum gain σ_1 approaching infinity. At the chosen flow conditions, J is known to have a nearly unstable mode, with eigenvalue close to the imaginary axis, that would eventually lead to the well-known vortex shedding instability in the cylinder wake at the critical Reynolds number of $Re \approx 47$. The direct eigenproblem, $J\hat{\boldsymbol{w}} = \lambda\hat{\boldsymbol{w}}$, and adjoint eigenproblem, $J^\dagger\hat{\boldsymbol{w}}^+ = \lambda^*\hat{\boldsymbol{w}}^+$, were solved for the least stable eigenmode using the triglobal stability tool described previously [8]. The eigenvalue with the largest real part was computed to be $\lambda = -0.0275 + i0.7145$, from which a frequency of $\omega = 0.7$ was chosen for resolvent analysis. The resolvent problem was solved for $k = 3$ dominant singular triplets using both algorithms. For the KGK algorithm, the maximum size of the Krylov subspace was set to $m = 4k = 12$ and the truncation size was set to $t = 2k = 6$. For the ISI algorithm, the subspace size was set to $m = 2k = 6$ only as the number of linear systems per iteration doubles with the subspace size. The singular triplets were deemed to have converged when the error norm (defined in Eq. (4)) dropped below an outer convergence tolerance of 10^{-6}. The KGK algorithm required an inner convergence tolerance (of the preconditioned problem) of $\epsilon_{\text{inner}} = 10^{-9}$ for the linear problems to achieve the prescribed outer tolerance, whereas $\epsilon_{\text{inner}} = 0.1$ was sufficient for the ISI algorithm to guarantee outer convergence. For comparison, the resolvent problem was also solved using the SVI algorithm [9, 18]. The inner convergence tolerance for the SVI algorithm was gradually reduced, following the relation $\epsilon_{\text{inner}} = 10^{-3}\min(1, \epsilon_j)$ where ϵ_j refers to the outer error norm of the jth singular triplet, as defined in Eq. 4, so that the linear solution converges progressively deeper as each triplet converges.

The steady-state solutions for the streamwise and cross-stream momentum components are shown in Fig. 1(a) and (b), respectively. The real parts of the streamwise momentum components of the direct and adjoint modes are shown in Fig. 1(c) and (d), respectively. The streamwise momentum components were scaled so that they reached a maximum amplitude of unity with zero phase. Similarly scaled real parts of the streamwise momentum components of the optimal forcing and response modes are given in Fig. 1(e) and (f), respectively. There is good qualitative agreement between the mode shapes as expected. The direct eigenmode and optimal response mode both exhibit vortex shedding downstream of the cylinder. The first three gains were computed to be $\sigma_1 = 2158.98$, $\sigma_2 = 162.66$ and $\sigma_3 = 153.86$.

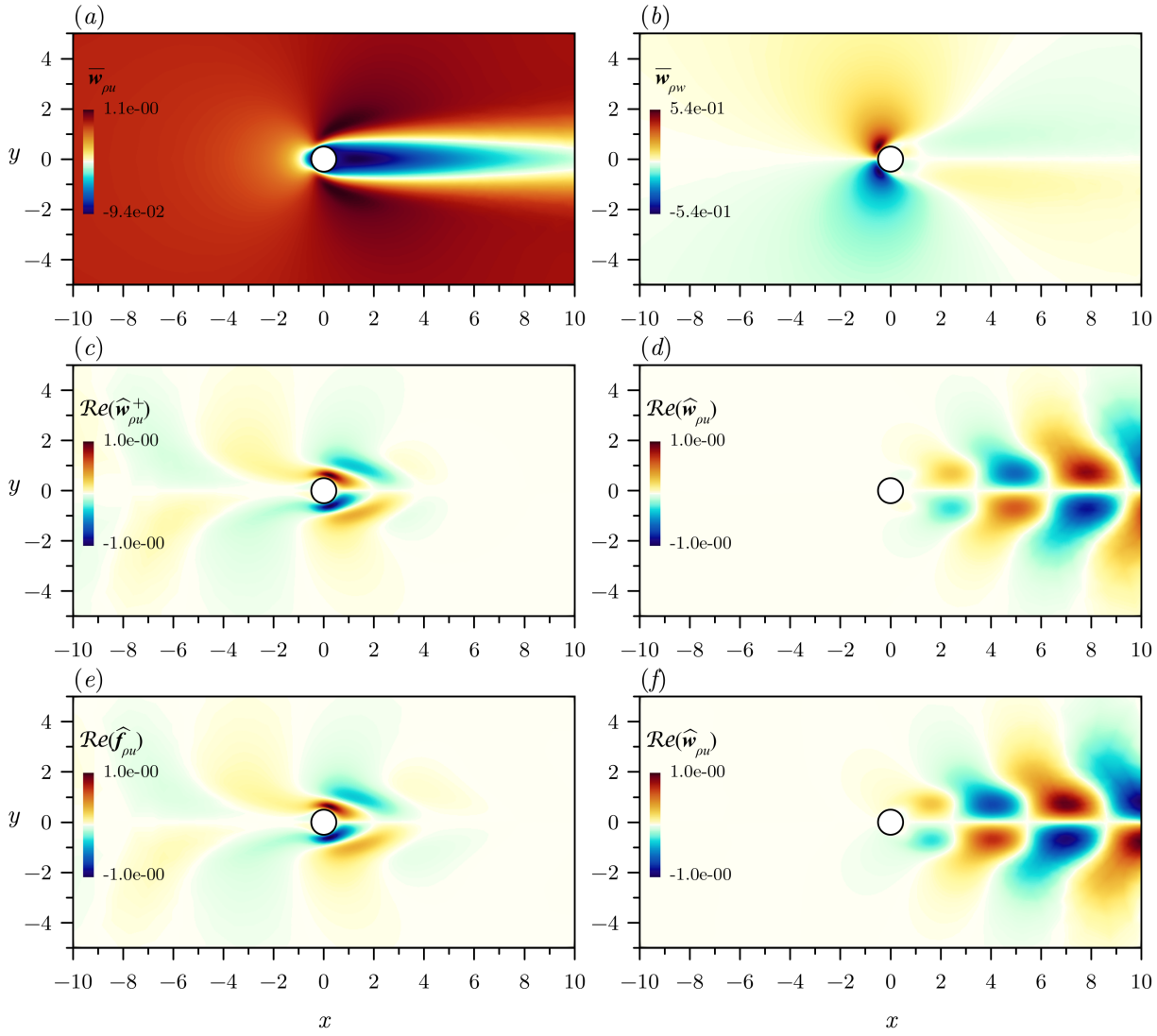

Fig. 1 Circular cylinder test case at $Re = 40$ showing (*a*) streamwise and (*b*) cross-stream momentum components of base flow, the scaled real parts of streamwise momentum component of the leading (*c*) adjoint and (*d*) direct eigenmodes, and the optimal (*e*) forcing and (*f*) response modes.

The convergence of (outer) error norms for the SVI, KGK and ISI algorithms are plotted in Fig. 2(*a*), (*b*) and (*c*), respectively. These computations were performed using four parallel processes on a Intel® Core™ i9-10900K CPU processor but the results and convergence trends were independent of the number of parallel processes used. The SVI algorithm converged the fastest for the first singular triplet which is well-separated from the second singular triplet with $\sigma_1/\sigma_2 \approx 13$. However, its convergence rate suffered for the second singular triplet whose singular value is close to that of the third with $\sigma_2/\sigma_3 \gtrsim 1$ and worsened for the third singular triplet. In fact, the SVI algorithm did not converge to the correct singular value for σ_3 even after 200 outer iterations since σ_3 and $\sigma_4 = 153.73$ are very close to each other. In contrast, both subspace methods were able to achieve convergence for all three singular triplets. The ISI algorithm required 558 inexact linear solutions while the KGK algorithm required 86 deep linear solutions. Despite needing far greater linear solutions, the ISI algorithm completed in 60% the time taken by the KGK algorithm.

Increasing the maximum Krylov subspace improves the performance of the KGK algorithm as more subspace information is retained and the frequency of discarding (potentially relevant) information is reduced. A maximum Krylov subspace size of $m = 33$ was sufficient for the KGK algorithm to complete without truncations. Under this

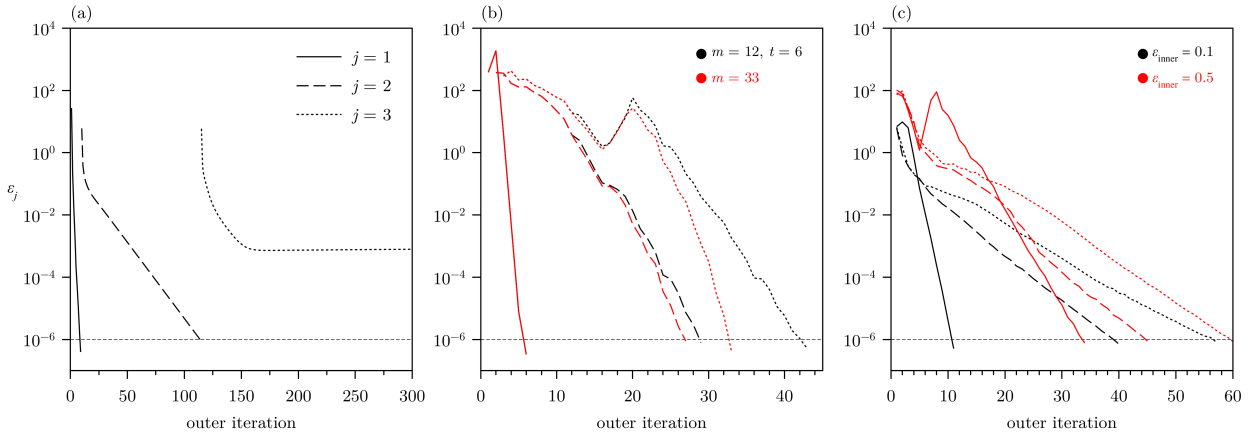

Fig. 2 Convergence of outer error norms for the first three singular triplets for the circular cylinder case computed using (*a*) SVI, (*b*) KGK and (*c*) ISI algorithms.

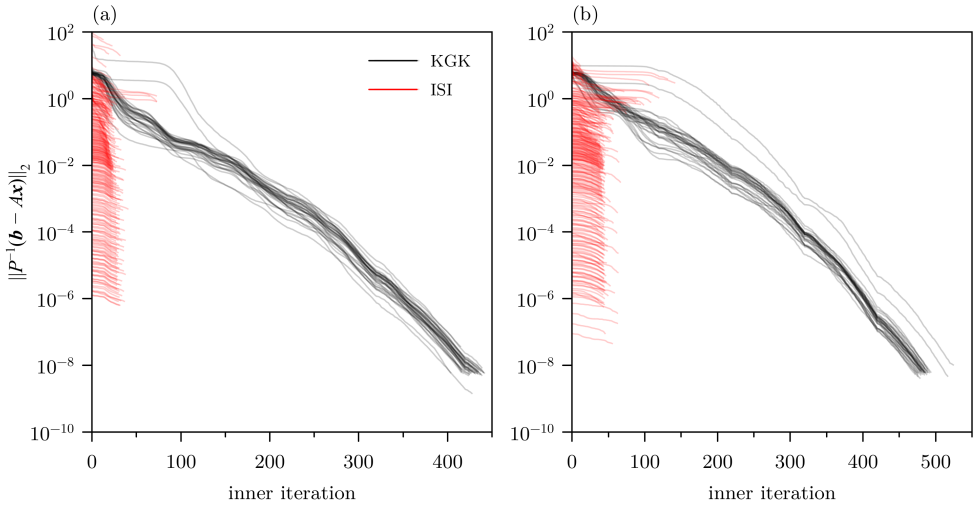

Fig. 3 Convergence of the linear residual norms for all preconditioned (*a*) forward and (*b*) adjoint systems for the circular cylinder case computed using the KGK algorithm with a maximum Krylov subspace of $m = 33$ and the ISI algorithm with $\epsilon_{\text{inner}} = 0.5$.

best possible scenario, it required $2m = 66$ deep linear solutions and completed 20% faster than with a maximum Krylov subspace of $m = 12$. Surprisingly, increasing the inner convergence tolerance for the ISI algorithm led to better performance as well. For example, with $\epsilon_{\text{inner}} = 0.5$, the ISI algorithm required 638 inexact linear solutions but completed 50% than with $\epsilon_{\text{inner}} = 0.1$. Between the KGK algorithm with a maximum Krylov subspace size of $m = 33$ and the ISI algorithm with $\epsilon_{\text{inner}} = 0.5$, the latter computation completed 65% faster. The superior performance of the ISI algorithm is due to the lower cost of an inexact linear solution which, for $\epsilon_{\text{inner}} = 0.5$, was, on average, 27 times faster than one deep linear solution with $\epsilon_{\text{inner}} = 10^{-9}$. Figure 3 shows the convergence of the linear residual 2-norms $\|P^{-1}(\boldsymbol{b} - A\boldsymbol{x})\|_2$ for all (left-preconditioned) forward and adjoint linear systems in the two computations. Multiplying from the left with P^{-1} denotes application of the preconditioner. Notice for the ISI algorithm that the initial linear residual norms gradually decrease with the outer convergence of the singular triplets. For an average forward/adjoint system, a deep linear solution required 429/471 inner iterations, whereas an inexact linear solution required only 27/40 inner iterations. Since an inexact solution almost never required more than 120 Krylov vectors, no restart was triggered and, therefore, no recycling (enabled through the GCRO-DR linear equation solver and then effectively operating as standard GMRES) was used for this cylinder test case.

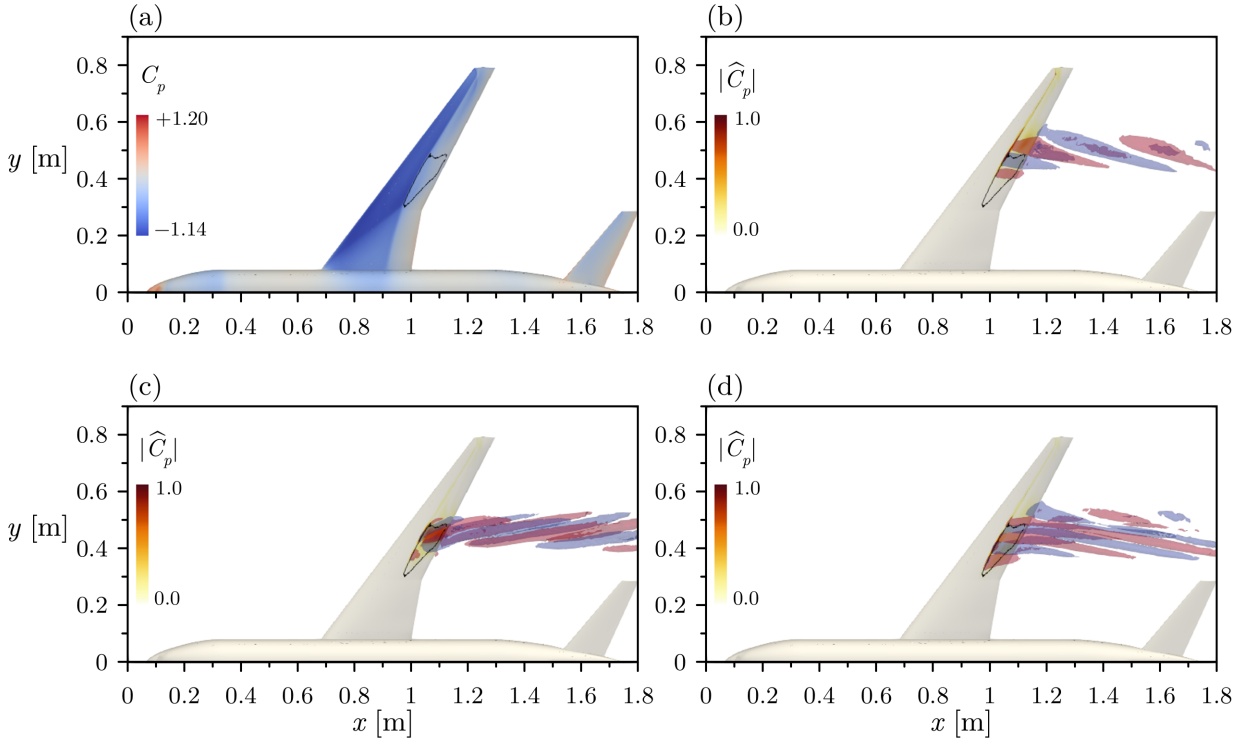

Fig. 4 NASA CRM test case at $Re = 5 \times 10^{-6}$ and $\alpha = 3.5$ deg showing (*a*) the steady-state surface pressure coefficient and the ±0.025 iso-surfaces for the scaled real parts of the streamwise momentum components of (*b*) first, (*c*) second and (*d*) third dominant response modes. The steady-state zero-skin-friction line is shown on all plots to indicate flow separation.

B. NASA Common Research Model

The NASA Common Research Model (CRM) is a generic commercial transport aircraft with open-source geometry to serve as a test case for numerical studies and available experimental data from various wind-tunnel campaigns [29]. The configuration has a design lift coefficient of 0.5 for a wing with an aspect ratio of 9, a taper ratio of 0.275 and a 35 deg quarter-chord sweep angle. The scaled-down wind-tunnel version is discussed herein featuring a mean aerodynamic chord of 0.189 m with a semi-span of 0.793 m and reference area (for the half model) of 0.140 m^2. For the wing/body/horizontal-tail configuration, the pylons and nacelles were discarded and the horizontal tail-setting angle was chosen as 0 deg. The computational mesh has approximately 6.2×10^6 points including 170 000 points on solid walls. A viscous wall normal spacing of less than one is ensured to avoid the need for wall functions in the turbulence modelling. The hemispherical far-field boundary is located at a distance of 100 semi-span lengths. The flow conditions were defined according to a test point of the experimental campaign in the European Transonic Windtunnel [30], specifically a Reynolds number (based on mean aerodynamic chord) of $Re = 5.0 \times 10^6$ and a reference free-stream Mach number of $M = 0.85$. A subcritical angle of attack of $\alpha = 3.5$ deg was chosen. To increase physical realism, the static deformation due to aerodynamic loading was interpolated for the chosen angle of attack from data measured in run 182 of the campaign [31]. An angular frequency of $\omega = 2.317$ close to the frequency of the global wing shock-buffet instability [8] was chosen for the resolvent analysis. Both subspace methods were used to compute the $k = 3$ most dominant singular triplets on two Intel® Xeon® Gold 6138 (Skylake) CPU processors with 40 parallel processes each. For the KGK algorithm, the maximum Krylov subspace size was set to a rather large value of $m = 50$ to minimize, and possibly avoid, truncations and the truncation size was set to $t = 25$. For the ISI algorithm, the subspace size was chosen as $m = 2k = 6$. The inner convergence tolerance was 10^{-8} for the KGK algorithm while inner convergence tolerances of 0.1 and 0.5 were used for the ISI algorithm. The outer convergence tolerance was set to 10^{-5} for both algorithms.

The steady-state surface pressure coefficient C_p plotted in Fig. 4(*a*) shows the characteristic shock-wave pattern along the wing span for a transonic configuration. The zero-skin-friction $C_f = 0$ line indicates the shock-induced flow

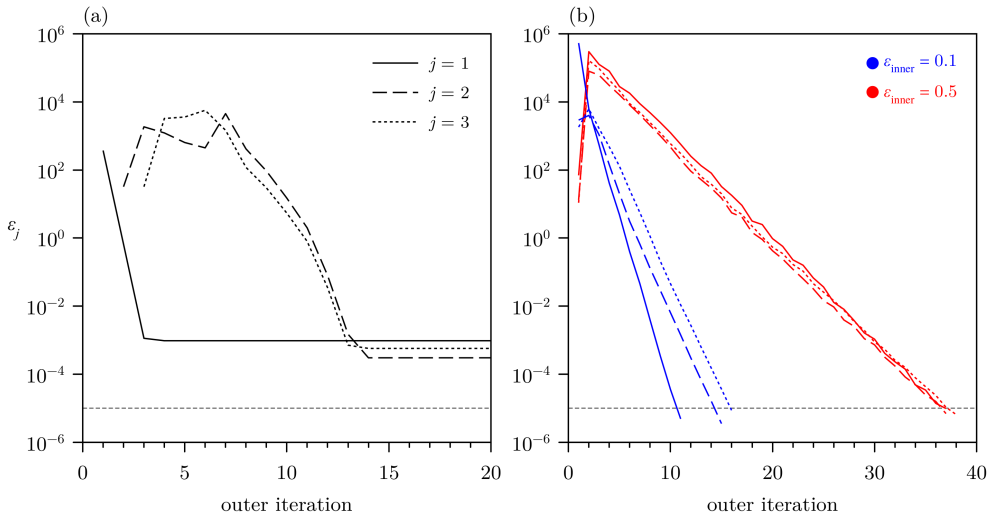

Fig. 5 Convergence of outer error norms for first three singular triplets for NASA CRM case computed using (*a*) KGK and (*b*) ISI algorithms.

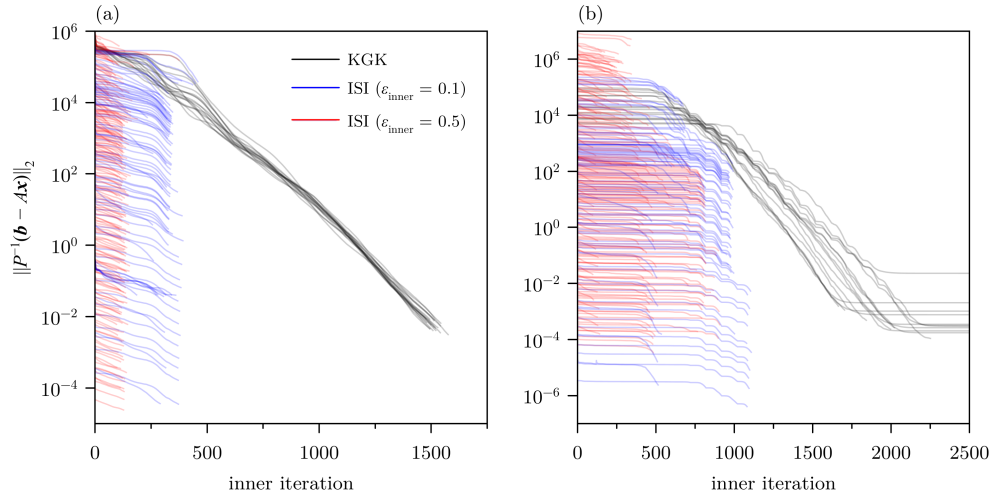

Fig. 6 Convergence of linear residual norms for all preconditioned (*a*) forward and (*b*) adjoint systems for NASA CRM case.

separation between approximately 38% (where the wing-inboard shock fronts merge) and 62% semi-span location. The first three gains were computed to be $\sigma_1 = 1.67 \times 10^6$, $\sigma_2 = 8.25 \times 10^4$ and $\sigma_3 = 7.93 \times 10^4$ by both KGK and ISI algorithms. Figures 4(*b*), (*c*) and (*d*) show the ±0.025 iso-surfaces of the scaled real parts of the streamwise momentum components of the first, second and third dominant response modes, respectively. The results were scaled such that the streamwise momentum component reached a maximum value of $1 + 0i$. The surface plots indicate the absolute value of perturbation in the surface pressure coefficient $\hat{C}_p$ scaled to a maximum value of unity. The dominant response in Fig. 4(*b*) bears strong resemblance to the unstable shock-buffet mode reported in detail in [8]. Iso-surfaces for the second and third responses also emanate from the separation region similar to that of the optimal response but, interestingly, the coherent structures of the second response mode are aligned in the opposite direction to those of the first and third.

The outer convergence of the two resolvent algorithms is plotted in Fig. 5. The KGK algorithm stalled above the required outer convergence tolerance of 10^{-5} for all three singular triplets. The ISI algorithm, however, converged to the desired accuracy for all three singular triplets with both inner convergence tolerances. The computation with $\epsilon_{\text{inner}} = 0.1$

completed about 10% faster than that with $\epsilon_{\text{inner}} = 0.5$. The outer convergence of the two resolvent algorithms is plotted in Fig. 5. The KGK algorithm stalled above the required outer convergence tolerance of 10^{-5} for all three singular triplets. The ISI algorithm, however, converged to the desired accuracy for all three singular triplets with both inner convergence tolerances. The computation with $\epsilon_{\text{inner}} = 0.1$ completed about 10% faster than that with $\epsilon_{\text{inner}} = 0.5$. For the KGK algorithm, one observes the expected convergence trend whereby the first singular triplet converges the fastest followed by the second and third singular triplets which converge together more slowly. For the ISI algorithm with $\epsilon_{\text{inner}} = 0.1$, the first singular triplet converges only slightly faster than the second and third whereas all three singular triplets converge almost simultaneously with $\epsilon_{\text{inner}} = 0.5$. The convergence of the linear residual norms for the preconditioned linear systems are plotted in Fig. 6. The convergence for the adjoint systems stalls in the initial iterations before recovering causing the linear solver to take significantly more iterations to converge for the adjoint systems compared to the forward systems. Consequently, an inexact adjoint linear solution with a convergence tolerance of 0.1 took only about twice as long as with 0.5. However, since the ISI algorithm with $\epsilon_{\text{inner}} = 0.1$ converged in less than half the number of outer iterations as with $\epsilon_{\text{inner}} = 0.5$, the former computation was slightly faster. The linear solver stalls altogether a second time before reaching the specified inner convergence tolerance of $\epsilon_{\text{inner}} = 10^{-8}$ for many of the adjoint problems preventing the KGK algorithm from reaching the desired outer accuracy. The inexact linear solutions have no trouble with convergence presenting the ISI algorithm with a clear advantage over the KGK algorithm for large problems where a deep linear solution is challenging.

V. Conclusions

Single-vector iteration algorithms for resolvent analysis are effective only when one needs to determine the optimal forcing-gain-response triplet that is well-separated from the sub-optimal ones. When there are competing optimal modes with clustered singular values or when knowledge of sub-optimal modes is required, subspace methods are more potent compared to single-vector iteration algorithms. In this work, two subspace methods, the Krylov–Golub–Kahan and inexact subspace iteration algorithms, have been introduced and explored for the resolvent problem to compute both optimal and sub-optimal singular triplets. The Krylov–Golub–Kahan algorithm is a Krylov subspace method with a simple restart strategy that requires deep linear solutions to ensure accurate applications of the matrix inverses. The inexact subspace iteration algorithm relies on inexact linear solutions to obtain only the corrections to the current approximations and uses Rayleigh–Ritz acceleration. The circular cylinder case at a sub-critical Reynolds number of $Re = 40$ was used to benchmark the subspace algorithms against a single-vector iteration algorithm. When computing the three most dominant singular triplets, the single-vector iteration algorithm converged the fastest for the first, well-separated singular triplet but struggled to converge for the second and stalled for the third. In contrast, both subspace algorithms managed to converge for all three singular triplets with the inexact subspace iteration algorithm being superior to the Krylov–Golub–Kahan algorithm thanks to the computationally cheap, inexact linear solutions. The NASA Common Research Model test case at a Reynolds number of $Re = 5 \times 10^6$, a transonic Mach number of $M = 0.85$ and a sub-critical (related to transonic buffet onset) angle-of-attack of $\alpha = 3.5$ deg served as a large-scale, industrially relevant configuration for resolvent analysis. The two subspace algorithms were used with their respective optimal settings to once again compute the three most dominant singular triplets but only the inexact subspace iteration algorithm achieved the specified level of convergence.

Acknowledgements

The work leading to these results received funding through the UK project Development of Advanced Wing Solutions (DAWS). The DAWS project is supported by the Aerospace Technology Institute (ATI) Programme, a joint government and industry investment to maintain and grow the UK's competitive position in civil aerospace design and manufacture. The programme, delivered through a partnership between ATI, Department for Business, Energy & Industrial Strategy (BEIS) and Innovate UK, addresses technology, capability and supply chain challenges.

References

[1] Trefethen, L. N., Trefethen, A. E., Reddy, S. C., and Driscoll, T. A., "Hydrodynamic stability without eigenvalues," *Science*, Vol. 261, No. 5121, 1993, pp. 578–584. https://doi.org/10.1126/science.261.5121.578.

[2] Theofilis, V., "Global linear instability," *Annual Review of Fluid Mechanics*, Vol. 43, 2011, pp. 319–352. https://doi.org/10.1146/annurev-fluid-122109-160705.

[3] Monokrousos, A., Åkervik, E., Brandt, L., and Henningson, D. S., "Global three-dimensional optimal disturbances in the Blasius boundary-layer flow using time-steppers," *Journal of Fluid Mechanics*, Vol. 650, 2010, pp. 181—-214. https://doi.org/10.1017/S0022112009993703.

[4] Gómez, F., Sharma, A. S., and Blackburn, H. M., "Estimation of unsteady aerodynamic forces using pointwise velocity data," *Journal of Fluid Mechanics*, Vol. 804, 2016, p. R4. https://doi.org/10.1017/jfm.2016.546.

[5] Thormann, R., and Widhalm, M., "Linear-frequency-domain predictions of dynamic-response data for viscous transonic flows," *AIAA Journal*, Vol. 51, No. 11, 2013, pp. 2540–2557. https://doi.org/10.2514/1.J051896.

[6] Stanford, B. K., and Jacobson, K. E., "Transonic Aeroelastic Modeling of the NACA 0012 Benchmark Wing," *AIAA Journal*, Vol. 59, No. 10, 2021, pp. 4134–4143. https://doi.org/10.2514/1.J060328.

[7] U S Vevek, Timme, S., Pattinson, J., Stickan, B., and Büchner, A., "Next-generation computational fluid dynamics capability for aircraft aeroelasticity and loads," *International Forum on Aeroelasticity and Structural Dynamics*, 2022.

[8] Timme, S., "Global instability of wing shock-buffet onset," *Journal of Fluid Mechanics*, Vol. 885, 2020, p. A37. https://doi.org/10.1017/jfm.2019.1001.

[9] Houtman, J., Timme, S., and Sharma, A., "Resolvent analysis of large aircraft wings in edge-of-the-envelope transonic flow," *AIAA SCITECH 2022 Forum*, 2022. https://doi.org/10.2514/6.2022-1329, AIAA 2022-1329.

[10] Qadri, U. A., and Schmid, P. J., "Frequency selection mechanisms in the flow of a laminar boundary layer over a shallow cavity," *Physical Review Fluids*, Vol. 2, No. 1, 2017, p. 013902. https://doi.org/10.1103/PhysRevFluids.2.013902.

[11] Bugeat, B., Chassaing, J.-C., Robinet, J.-C., and Sagaut, P., "3D global optimal forcing and response of the supersonic boundary layer," *Journal of Computational Physics*, Vol. 398, 2019, p. 108888. https://doi.org/10.1016/j.jcp.2019.108888.

[12] Ribeiro, J. H. M., Yeh, C.-A., and Taira, K., "Randomized resolvent analysis," *Physical Review Fluids*, Vol. 5, No. 3, 2020, p. 033902. https://doi.org/10.1103/PhysRevFluids.5.033902.

[13] Stoll, M., "A Krylov–Schur approach to the truncated SVD," *Linear Algebra and its Applications*, Vol. 436, No. 8, 2012, pp. 2795–2806. https://doi.org/10.1016/j.laa.2011.07.022.

[14] Golub, G. H., and Ye, Q., "Inexact inverse iteration for generalized eigenvalue problems," *BIT Numerical Mathematics*, Vol. 40, 2000, pp. 671–684. https://doi.org/10.1023/A:1022388317839.

[15] Berns-Müller, J., Graham, I. G., and Spence, A., "Inexact inverse iteration for symmetric matrices," *Linear Algebra and its Applications*, Vol. 416, No. 2-3, 2006, pp. 389–413. https://doi.org/10.1016/j.laa.2005.11.019.

[16] Freitag, M., and Spence, A., "Convergence of inexact inverse iteration with application to preconditioned iterative solves," *BIT Numerical Mathematics*, Vol. 47, 2007, pp. 27–44. https://doi.org/10.1007/s10543-006-0100-1.

[17] McKeon, B. J., and Sharma, A. S., "A critical-layer framework for turbulent pipe flow," *Journal of Fluid Mechanics*, Vol. 658, 2010, pp. 336–382. https://doi.org/10.1017/S002211201000176X.

[18] Houtman, J., Timme, S., and Sharma, A., "Resolvent analysis of a finite wing in transonic flow," *Flow*, 2023. https://doi.org/10.1017/flo.2023.8, accepted for publication.

[19] Stewart, G. W., "A Krylov–Schur algorithm for large eigenproblems," *SIAM Journal on Matrix Analysis and Applications*, Vol. 23, No. 3, 2002, pp. 601–614. https://doi.org/10.1137/S0895479800371529.

[20] Sorensen, D. C., "Implicit application of polynomial filters in a k-step Arnoldi method," *Siam journal on matrix analysis and applications*, Vol. 13, No. 1, 1992, pp. 357–385. https://doi.org/10.1137/0613025.

[21] Francis, J. G., "The QR transformation a unitary analogue to the LR transformation—Part 1," *The Computer Journal*, Vol. 4, No. 3, 1961, pp. 265–271. https://doi.org/10.1093/comjnl/4.3.265.

[22] Lai, Y.-L., Lin, K.-Y., and Lin, W.-W., "An inexact inverse iteration for large sparse eigenvalue problems," *Numerical Linear Algebra with Applications*, Vol. 4, No. 5, 1997, pp. 425–437. https://doi.org/10.1002/(SICI)1099-1506(199709/10)4:5<425::AID-NLA117>3.0.CO;2-G.

[23] Freitag, M. A., and Spence, A., "Shift-invert Arnoldi's method with preconditioned iterative solves," *SIAM Journal on Matrix Analysis and Applications*, Vol. 31, No. 3, 2010, pp. 942–969. https://doi.org/10.1137/080716281.

[24] Parlett, B. N., and Scott, D. S., "The Lanczos algorithm with selective orthogonalization," *Mathematics of computation*, Vol. 33, No. 145, 1979, pp. 217–238. https://doi.org/10.2307/2006037.

[25] Golub, G. H., and Van Loan, C. F., *Matrix computations*, JHU press, 2013.

[26] Schwamborn, D., Gerhold, T., and Heinrich, R., "The DLR TAU-code: Recent applications in research and industry," *ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics*, 2006. https://elib.dlr.de/22421/.

[27] Parks, M., de Sturler, E., Mackey, G., Johnson, D., and Maiti, S., "Recycling Krylov Subspaces for Sequences of Linear Systems," *SIAM Journal on Scientific Computing*, Vol. 28, No. 5, 2006, pp. 1651–1674. https://doi.org/10.1137/040607277.

[28] Xu, S., Timme, S., and Badcock, K. J., "Enabling off-design linearised aerodynamics analysis using Krylov subspace recycling technique," *Computers & Fluids*, Vol. 140, 2016, pp. 385–396. https://doi.org/10.1016/j.compfluid.2016.10.018.

[29] Vassberg, J., Dehaan, M., Rivers, M., and Wahls, R., "Development of a Common Research Model for Applied CFD Validation Studies," *26th AIAA Applied Aerodynamics Conference*, 2008. https://doi.org/10.2514/6.2008-6919.

[30] Lutz, T., Gansel, P. P., Waldmann, A., Zimmermann, D.-M., and Schulte am Hülse, S., "Prediction and Measurement of the Common Research Model Wake at Stall Conditions," *Journal of Aircraft*, Vol. 53, No. 2, 2016, pp. 501–514. https://doi.org/10.2514/1.C033351.

[31] Keye, S., and Gammon, M., "Development of Deformed Computer-Aided Design Geometries for the Sixth Drag Prediction Workshop," *Journal of Aircraft*, Vol. 55, No. 4, 2018, pp. 1401–1405. https://doi.org/10.2514/1.C034428.