Bespoke Stability Analysis Tool in Next-Generation Computational Fluid Dynamics Solver

U S Vevek, Jelle Houtman and Sebastian Timme

School of Engineering
University of Liverpool
Liverpool, England L69 3GH, United Kingdom

ABSTRACT

This paper presents some of the first results of linearised stability analyses performed using a bespoke eigensolver that has been recently implemented in the next generation flow solver framework CODA. The eigensolver benefits from the automatic differentiation capability of CODA that allows computation of the exact product of the Jacobian matrix with an arbitrary complex vector. It implements the Krylov–Schur algorithm for solving the eigenvalue problem. The bespoke tool has been validated for the case of laminar flow past a circular cylinder with numerical results computed using the TAU code and those reported in the literature. It has been applied with both second-order finite volume and high-order discontinuous Galerkin schemes for the case of laminar flow past a square cylinder. It has been demonstrated that using high-order schemes on coarser grids leads to well-converged eigenmodes with a shorter computation time compared to using second-order schemes on finer grids.

1.0 Introduction

Stability analysis of numerical steady-state solutions plays an important role in our understanding of the onset and dynamics of self-sustained unsteadiness in aerodynamic and aeroelastic applications[1,2]. The large but sparse eigenvalue problem formulated on the discretised governing equations is typically solved for the leading and physically-relevant modes, i.e. the few eigenvalues with the largest real part. Knowledge of these modes is useful for identifying the physical mechanisms responsible for the amplification of small-amplitude perturbations and, consequently, for designing effective control strategies. At the same time, considering the high dimensional system involved when simulating unsteady non-linear aerodynamics, the extraction of dominant modal features can aid in constructing low dimensional models and/or can avoid the long time-integration of the original system.

While the stability analysis of flow, and modal analysis in general, has a long-standing history, it remains a very active topic in fluid mechanics. The focus herein is on the continued development of operator-based modal identification using computational fluid dynamics (CFD). On the one hand, high-performance computing has seen remarkable progress in efficiency and scalability with heterogeneous computing that combines distributed memory message passing interface (MPI) parallelisation and shared memory OpenMP/GPU parallelisation. On the other hand, advanced numerical schemes and physical models have improved the modelling accuracy of CFD simulations. These include high-order schemes, advanced turbulence models (such as Reynolds stress models) and transition models. Despite these profound advances, current generation CFD tools predominantly use simple models such as the one-equation Spalart–Allmaras turbulence model and, relating to the stability analysis, often rely on older-generation eigensolver libraries, such as ARPACK[3], that are limited to homogeneous MPI parallelisation only.

The next generation flow solver CODA[4,5] is being developed to take advantage of emerging computing capabilities to eliminate limitations faced by current generation codes. The newly incorporated automatic differentiation (AD) capability allows matrix-vector products with the Jacobian operator to be evaluated accurately (and matrix-free for reduced memory footprints) regardless of the complexity of the underlying discretisation schemes and physical models. This is an important step forward from computing the Jacobian matrix by hand-differentiation (or finite-differencing) which becomes cumbersome (and inaccurate) for complex models. The exactness of the matrix-vector product operation is crucial for the iterative Krylov subspace methods used in linearised stability analysis for large problems of engineering relevance. The linear systems in CODA are solved using preconditioned Krylov subspace solvers that are available in CODA's sparse linear algebra library SPLISS[6]. SPLISS operates on two levels of parallelism with partitioning across MPI processes as well as OpenMP/GPU threads for enhanced scalability. This allows for more effective preconditioning with full parallelisation at the shared memory level while maintaining a strict block-Jacobi type approach (i.e. no parallel communication) at the distributed memory level.

The focus of this work is the Krylov–Schur algorithm[7] for solving large-scale eigenvalue problems that has been implemented in CODA to benefit from the latest capabilities in both CODA and SPLISS. It extends upon the linear frequency domain solver that has been successfully implemented in CODA earlier[8]. The paper continues with a discussion of the relevant theory, including the Krylov–Schur method, in Sec. 2, before outlining the numerical methodology in Sec. 3. Results for two test cases, specifically the laminar flow at Reynolds number $Re = 50$ around the circular and square cylinders, are presented in Sec. 4.

2.0 Theory

2.1 Basics

To derive the eigenvalue problem for linearised stability analysis, we begin with the unsteady Navier–Stokes equations written in a semi-discrete form

$$\frac{\mathrm{d}\boldsymbol{W}}{\mathrm{d}t} + \boldsymbol{R}(\boldsymbol{W}) = \boldsymbol{0}, \tag{1}$$

where $\boldsymbol{W} = [\rho, \rho u, \rho E, \dots]$ denotes the vector of conservative variables (depending on the chosen flow model which can include additional equations for modelling turbulence and transition) and $\boldsymbol{R}(\boldsymbol{W})$ denotes the corresponding non-linear residual vector in discretised form. Linearising Eq. (1) about a steady-state solution $\overline{\boldsymbol{W}}$, while assuming small-amplitude perturbations of the form $\widehat{\boldsymbol{W}} \exp(\lambda_J t)$, yields the eigenvalue problem

$$J\widehat{\boldsymbol{W}} = \lambda_J \widehat{\boldsymbol{W}}, \tag{2}$$

where $J = -\partial \boldsymbol{R}/\partial \boldsymbol{W}$ is the Jacobian operator, $\widehat{\boldsymbol{W}}$ is the eigenvector and λ_J is the eigenvalue. The imaginary part of λ_J corresponds to the frequency of the oscillations, while the real part of λ_J indicates if the oscillation amplitude grows ($\mathfrak{Re}(\lambda_J) > 0$) or decays ($\mathfrak{Re}(\lambda_J) < 0$) with time. The adjoint eigenvalue problem is similarly defined as

$$J^{\dagger}\widehat{\boldsymbol{W}}^{+} = \lambda_J^{*}\widehat{\boldsymbol{W}}^{+}, \tag{3}$$

where the adjoint Jacobian operator $J^{\dagger}$ satisfies the duality relation $\langle \boldsymbol{a}, J\boldsymbol{b} \rangle = \langle J^{\dagger}\boldsymbol{a}, \boldsymbol{b} \rangle$ (with a suitable inner product defined as $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^{H} M \boldsymbol{b}$ for arbitrary vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ and a positive definite weight matrix M), and $\widehat{\boldsymbol{W}}^{+}$ is its corresponding adjoint eigenvector with the eigenvalue λ_J^{*} which is the complex-conjugate of λ_J. Both J and $J^{\dagger}$ have the same set of eigenvalues λ_J. The adjoint Jacobian operator can be explicitly given as $J^{\dagger} = M^{-1} J^{T} M$. The weight matrix M is the diagonal matrix of cell volumes for the finite volume (FV) scheme and it is simply the identity matrix for the discontinuous Galerkin (DG) schemes as the DG schemes in CODA use an orthonormal polynomial basis.

To compute the relevant eigenvalues of the Jacobian operator J (or its adjoint) close to a given complex shift σ, often available from engineering insight, we apply the shift-invert spectral transformation such that $A = (J - \sigma I)^{-1}$ and cast the problem into the form

$$A\widehat{\boldsymbol{W}} = \lambda_A \widehat{\boldsymbol{W}}, \tag{4}$$

where $\lambda_A = (\lambda_J - \sigma)^{-1}$ is the transformed eigenvalue. The eigenvector $\widehat{\boldsymbol{W}}$ is unchanged by the transformation. The closer an eigenvalue λ_J is to the shift σ, the greater the absolute value of the transformed eigenvalue λ_A, which is beneficial for many iterative solution schemes such as the Krylov–Schur algorithm because desired eigenvalues are amplified. The drawback is that each application of A is a costly operation that involves solving a large, sparse linear problem with the coefficient matrix $A^{-1} = (J - \sigma I)$.

2.2 Krylov–Schur algorithm

The Krylov–Schur algorithm[7,9] is a Krylov subspace method that can be used to determine a small number of eigenvalue-eigenvector pairs (eigenpairs) of a large sparse linear operator A.

Since the Krylov subspace cannot be extended indefinitely due to memory constraints and the cost of orthogonalisation over a large subspace, a suitable restart technique is needed to filter the existing subspace to preserve the relevant information. The Krylov–Schur algorithm improves upon the implicitly restarted Arnoldi algorithm[10] by proposing a simple but robust restarting technique.

The key to the restarting technique is the Schur form

$$AQ_k = Q_{k+1}\begin{bmatrix} S_k \\ \boldsymbol{b}_k^H \end{bmatrix} = Q_{k+1}\tilde{S}_k, \tag{5}$$

where $Q_{k+1} = [Q_k \ \boldsymbol{q}_{k+1}]$ is an orthogonal matrix with $Q_1 = [\boldsymbol{q}_1]$, S_k is a $k \times k$ upper-triangular matrix and $\boldsymbol{b}_k^H$ is a $1 \times k$ row vector. The eigenvalues of S_k, which are found along its diagonal, are good approximations to the exterior/largest eigenvalues of A, i.e. the wanted eigenvalues of J after shift-invert transformation.

It is possible to split the Schur form into two parts, one consisting of the first m columns and the other consisting of the remaining $(k - m)$ columns, such that

$$A\begin{bmatrix} Q_m & Q_{k-m} \end{bmatrix} = \begin{bmatrix} Q_m & Q_{k-m} & \boldsymbol{q}_{k+1} \end{bmatrix}\begin{bmatrix} S_m & S_{m,k-m} \\ 0 & S_{k-m} \\ \boldsymbol{b}_m^H & \boldsymbol{b}_{k-m}^H \end{bmatrix}. \tag{6}$$

The restart technique simply involves discarding the last $(k-m)$ columns of Q_k and S_k yielding a contracted Schur form

$$AQ_m = \begin{bmatrix} Q_m & \boldsymbol{q}_{m+1} \end{bmatrix}\begin{bmatrix} S_m \\ \boldsymbol{b}_m^H \end{bmatrix} = Q_{m+1}\tilde{S}_m. \tag{7}$$

Note that vector $\boldsymbol{q}_{k+1}$ is retained and re-labelled as $\boldsymbol{q}_{m+1}$ to be consistent with the Schur form notation. The contraction procedure from S_k to S_m is shown schematically in Fig. 1 for $k = 5$ and $m = 3$. It was previously shown[7] that this restart technique is equivalent to applying a polynomial filter, $p(t) = (t - s_{m+1}) \times \cdots \times (t - s_k)$ where s_j is the jth diagonal element in S_k. If the diagonal elements of S_k had been arranged such that the last $(k - m)$ elements belonged to the unwanted part of the spectrum, then the contraction procedure effectively purges the decomposition of the unwanted part of the spectrum.

It is evident from the preceding discussion that it is crucial to maintain (or, at least, to be able to return to) the Schur form to benefit from the simple restarting technique. In the present implementation, the Schur form is preserved at the end of each Krylov–Schur iteration. The dense matrix operations required to do so are relatively inexpensive compared to the application of the linear operator A which involves solving a large linear problem. Let us assume that we begin with the Schur form given in Eq. (5). Performing an Arnoldi iteration on vector $\boldsymbol{q}_{k+1}$ results in

$$AQ_{k+1} = Q_{k+2}\begin{bmatrix} S_k & \boldsymbol{f}_k \\ \boldsymbol{b}_k^H & g \\ & h \end{bmatrix} = Q_{k+1}T_{k+1} + h\boldsymbol{q}_{k+2}\boldsymbol{e}_{k+1}^T, \tag{8}$$

where the $k \times 1$ column vector $\boldsymbol{f}_k$ and the scalars g and h are obtained by orthogonalising $A\boldsymbol{q}_{k+1}$ over Q_{k+1}. Since the Arnoldi iteration has disrupted the Schur form, we must perform a series of orthogonal similarity transformations to bring T_{k+1} back to the Schur form. First, T_{k+1} is brought into the upper-Hessenberg form $H_{k+1} = V_1 T_{k+1} V_1^H$ using Householder

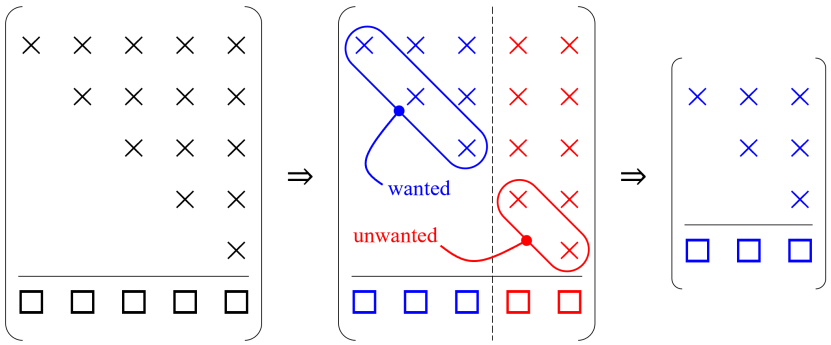

Figure 1. Schematic of $\tilde{S}_k$ during a restart for $k = 5$ and $m = 3$.

reflections. Then, H_{k+1} is brought into the complex Schur form $S'_{k+1} = V_2 H_{k+1} V_2^H$ using the shifted QR algorithm[11,12]. Finally, the diagonal elements of the upper-triangular matrix S'_{k+1} are reordered in descending order of their absolute values using special orthogonal matrices (see Appendix) which yields the ordered Schur form $S_{k+1} = V_3 S'_{k+1} V_3^H$. The reordering is necessary so that the restart discards information about the eigenvalues farthest away from σ. The combined effect of these three operations can be represented by a single similarity transformation $S_{k+1} = V T_{k+1} V^H$ where $V = V_3 V_2 V_1$ is an orthogonal matrix. Substituting $T_{k+1} = V^H S_{k+1} V$ into Eq. (8) results in

$$A Q_{k+1} = Q_{k+1} \left(V^H S_{k+1} V \right) + h \boldsymbol{q}_{k+2} \boldsymbol{e}_{k+1}^T. \tag{9}$$

Multiplying both sides of the latter equation by V^H from the right and setting $Q'_{k+1} = Q_{k+1} V^H$ and $\boldsymbol{b}_{k+1}^H = h \boldsymbol{e}_{k+1}^T V^H$ results in

$$A Q'_{k+1} = Q'_{k+1} S_{k+1} + \boldsymbol{q}_{k+2} \boldsymbol{b}_{k+1}^H \tag{10}$$

which is the sought-after Schur form. The expansion from S_k to S_{k+1} during the Krylov–Schur iteration for $k = 3$ is shown schematically in Fig. 2. Last but not least, we observe that the Arnoldi iteration is equivalent to a Krylov–Schur iteration for $k = 0$. Starting from a random unit vector $\boldsymbol{q}_1$, we perform an Arnoldi iteration to obtain

$$A Q_1 = \begin{bmatrix} Q_1 & \boldsymbol{q}_2 \end{bmatrix} \begin{bmatrix} S_1 \\ \boldsymbol{b}_1^H \end{bmatrix}, \tag{11}$$

where S_1 and $\boldsymbol{b}_1^H$ are just scalars and $Q_1 = [\boldsymbol{q}_1]$. Nonetheless, it can be seen that Eq. (11) is in the Schur form.

The sought eigenvectors of A can be approximated from the Schur form as

$$W_k = Q_k P_k, \tag{12}$$

where matrix $W_k = \begin{bmatrix} \widehat{\boldsymbol{W}}_1 & \dots & \widehat{\boldsymbol{W}}_k \end{bmatrix}$ contains the approximate eigenvectors (Ritz vectors) of A and $P_k = [\boldsymbol{p}_1 \ \dots \ \boldsymbol{p}_k]$ is a $k \times k$ matrix whose columns are the eigenvectors of S_k. The

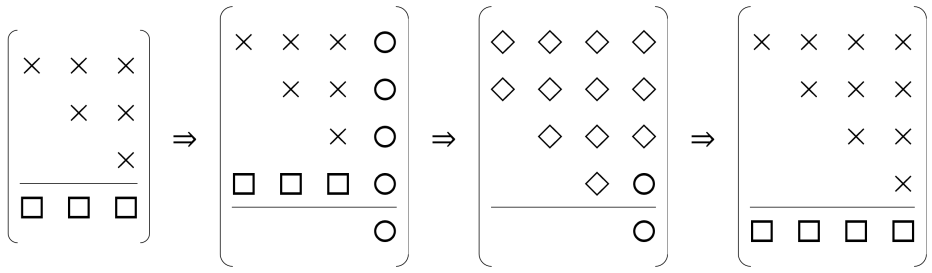

Figure 2. Schematic of $\bar{S}_k$ during a Krylov-Schur iteration for $k = 3$.

eigenvectors $\boldsymbol{p}_j$ for $j = 1 \ldots k$ can be computed from the relationship $S_k \boldsymbol{p}_j = s_j \boldsymbol{p}_j$ combined with the fact that both S_k and P_k are upper-triangular.

The error in the jth eigenvector approximation can be computed as

$$\varepsilon_j = \|A\widehat{\boldsymbol{W}}_j - s_j \widehat{\boldsymbol{W}}_j\|_2. \tag{13}$$

Substituting $\widehat{\boldsymbol{W}}_j = Q_k \boldsymbol{p}_j$ and $s_j \boldsymbol{p}_j = S_k \boldsymbol{p}_j$ into Eq. (13) leads to

$$\varepsilon_j = \|(AQ_k - Q_k S_k)\, \boldsymbol{p}_j\|_2 = \|\boldsymbol{q}_{k+1} \boldsymbol{b}_k^H \boldsymbol{p}_j\|_2 = \|\boldsymbol{b}_k^H \boldsymbol{p}_j\|_2. \tag{14}$$

The second equality arises from the Schur form itself and the last equality is due to $\boldsymbol{q}_{k+1}$ having unit norm. The approximation errors are computed at the end of each Krylov–Schur iteration. Once the jth error norm has dropped below a prescribed threshold value (e.g. 10^{-10}), the eigenpair is deemed to have converged and is locked by setting the jth component of $\boldsymbol{b}_k^H$ to zero. Subsequent transformations on S_k are only applied from the $(j+1)$th row and column leaving the top-left $j \times j$ part of S_j and the first j columns of Q_k unchanged. The computation is terminated when the last desired eigenpair has converged.

3.0 Methodology

3.1 Non-linear steady-state problem

The steady-state problem $\boldsymbol{R}(\overline{\boldsymbol{W}}) = \boldsymbol{0}$ was solved first since the Jacobian operator $J = -\partial \boldsymbol{R}/\partial \boldsymbol{W}$ for the eigenvalue problem must be computed about a suitable reference state $\overline{\boldsymbol{W}}$. The non-linear flow solutions were computed in CODA using the implicit backward Euler scheme with local time stepping until the density residual norm dropped by ten orders of magnitude. Courant-Friedrich-Levy (CFL) number ramping of the local time steps was employed to accelerate non-linear convergence. The linear system at each outer iteration was solved using a generalized minimum residual (GMRES) solver[13] until the linear residual norm dropped by one order of magnitude. The solver used a maximum of 100 Krylov vectors with a maximum of nine restarts and 50 iterations of a block-Jacobi solver as preconditioner. The GMRES solver utilises a matrix-free approach using AD to compute the matrix-vector products $\boldsymbol{Jx}$ exactly for a real or complex vector $\boldsymbol{x}$. The preconditioner, on the other hand, uses an explicit matrix whose block-diagonal is factorised using LU decomposition. The explicit matrix is formed with the help of AD as well using compact stencils. The compact stencil for a cell

consists of the cell and its immediate face-neighbours only. Since DG schemes use compact stencils, the explicit matrix is exact. For the FV scheme, which relies on extended stencils, the explicit matrix is approximate and essentially only first-order accurate. However, this is not a severe disadvantage. In fact, the approximate matrix is more diagonally dominant leading to improved stability of the preconditioner[14].

Spatial discretisation was performed herein using both FV and DG methods. In the FV method, the solution is computed as the cell averages of the conservative variables, i.e. each cell represents one degree of freedom (DOF) per variable. In the DG method, each variable is represented as an nth order polynomial in three dimensions where we limit our interest herein to $n \in [2, \ldots, 8]$. Therefore, each cell represents ${}^nC_3 = n(n+1)(n+2)/6$ DOF for each variable and ${}^nC_k$ is the binomial coefficient. For the two-dimensional cases considered in this study, the DOF per cell can be significantly reduced (only ${}^nC_2$) since the third dimension is redundant. However, as CODA does not currently support such a two-dimensional DG formulation, the full three-dimensional formulation was used despite the redundancy.

The inviscid fluxes were computed using the Roe scheme with no entropy fix based on the face values. In the FV method, the face values at the face centroids were first approximated using distance-weighted interpolation of the face-adjacent neighbours' cell averages. Then, cell average gradients were computed from the approximate face values using the Green–Gauss method. The final face values were computed as a linear function of the cell average values and gradients. In the DG method, the face values were computed by evaluating the polynomial at the face quadrature points. The flux over each face was integrated as a weighted average of the fluxes computed at the quadrature points. The viscous fluxes were computed based on face gradients. In the FV method, the face gradients were computed as the distance-weighted averages of the neighbouring cell average gradients with edge-normal augmentation[15]. In the DG method, the viscous fluxes were computed using the second approach proposed by Bassi and Rebay known as the BR2 scheme[16].

3.2 Eigenvalue problem

Once the steady-state solution has been computed, the Krylov–Schur algorithm was used to determine the rightmost (in the complex plane) eigenpair. Observe that the flow conditions in this study for the chosen test cases are such that $\Re\mathrm{e}(\lambda_J) > 0$ for the leading eigenmode and it describes unstable flow, specifically vortex shedding in the wake behind bluff bodies. In the Krylov–Schur algorithm, the Krylov subspace was allowed to expand to a maximum size of 20 before a restart. The restart truncated the size of the subspace to five. The linear system of the form $(J - \sigma I)\boldsymbol{x} = \boldsymbol{q}_{k+1}$ in each Krylov–Schur iteration was solved using the GMRES solver with the same settings as that used for the steady-state problem except that 100 block-Jacobi iterations were used as preconditioner. The shift σ was chosen close to the unstable eigenvalue based on the prior knowledge of the physics and numerical experiments to ensure fast convergence. During the numerical experiments, the approximation error of a converged eigenpair was observed to be directly dependent on the convergence of the linear solver in each Krylov–Schur iteration. To achieve an approximation error below 10^{-10}, the linear system in each Krylov–Schur iteration was solved until the residual norm dropped below 10^{-10}.

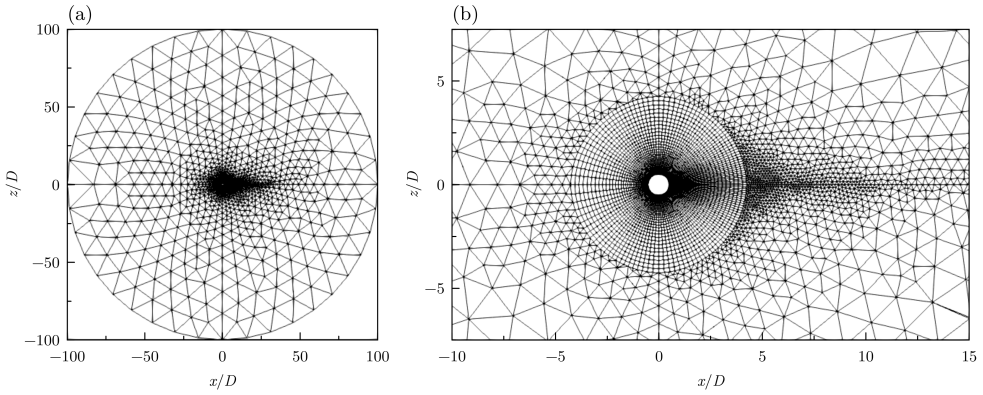

Figure 3. Grid used for laminar flow past circular cylinder case.

4.0 Results and Discussion

4.1 Laminar flow past circular cylinder

We first consider laminar $M = 0.2$ flow past a circular cylinder as a validation case, which has been widely documented in the literature. The unstructured grid shown in Fig. 3 was used, which includes a quasi-structured near-field region and a total of 11638 cells (corresponding to 9743 vertices in a two-dimensional set-up). The computational domain has a radial extent of $100D$, where $D = 1$ is the diameter of the circular cylinder, and a unit extent in the spanwise direction. The far-field view in Fig. 3(a) shows the wake refinement. The quasi-structured grid around the cylinder has a first wall-normal cell spacing of $D/1000$ and becomes unstructured starting at a radial distance of $4.25D$ from the origin as seen in Fig. 3(b).

The problem was computed at a slightly supercritical Reynolds number of $Re = 50$ to ensure the presence of an unstable mode linked to von Kármán vortex shedding. The complex shift σ was chosen such that its imaginary part was close to the critical vortex-shedding frequency and its real part was slightly positive so as to hone in on the unstable mode. No attempt was made in this study to assess the impact of the spectral distance between shift and wanted eigenvalue on the performance of the eigenmode solver. Both the direct eigenvalue problem in Eq. (2) and the adjoint eigenvalue problem in Eq. (3) were solved using CODA and TAU*. For CODA, only the FV method was used for this case. The non-dimensional eigenvalues λ_J (made non-dimensional using D and free-stream velocity U_∞) of the von Kármán wake mode computed by CODA and TAU are 4.2507e-03 + i7.2656e-01 and 1.5829e-02 + i7.3105e-01, respectively. The positive real parts of λ_J indicate that the modes are indeed unstable. Their imaginary parts are quite close to the numerical shedding frequency of 0.732 predicted by Crouch et al.[1] based on a similar stability analysis.

The real parts of the streamwise momentum component $\widehat{\rho u}$ of the eigenvectors are plotted in Fig. 4. The eigenvectors were scaled such that the maximum of the $\widehat{\rho u}$ component has a

* The TAU code of the German Aerospace Center is an industrial second-order code with a cell-vertex finite-volume formulation capable of dealing with complex geometries and is widely used in the European aerospace community[17]. Spatial discretisation herein uses the code's default formulation of a central scheme with matrix artificial dissipation for inviscid fluxes and Green–Gauss gradients for viscous terms. Its extension to solve eigenvalue problems has been detailed previously[2].

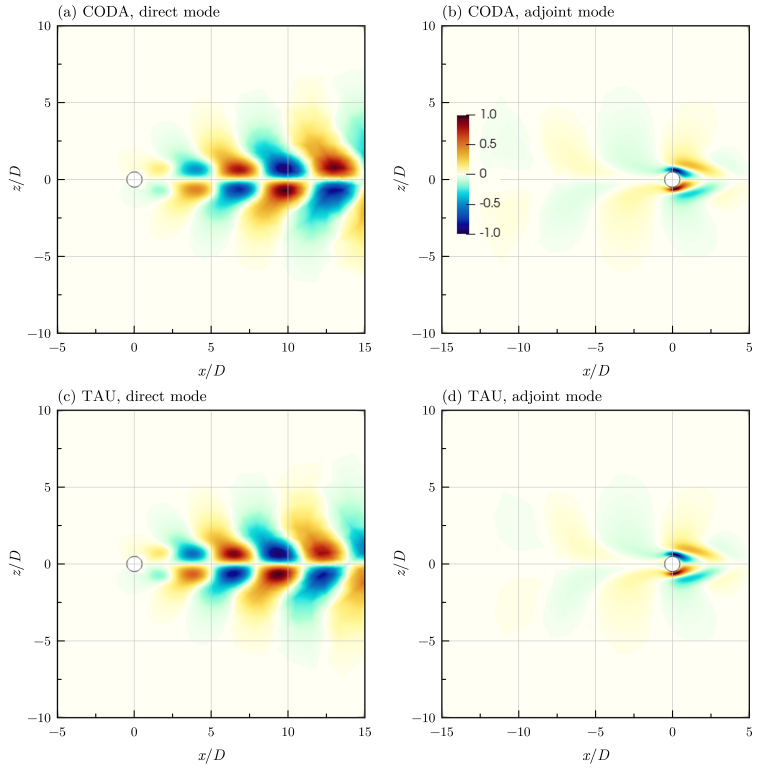

Figure 4. Streamwise momentum components of unstable direct and adjoint global modes for laminar flow past circular cylinder case.

value of $1 + i0$, i.e. a maximum amplitude of one with zero phase. The direct modes exhibit the well-known vortex shedding pattern downstream of the cylinder. The regions of high amplitudes of the adjoint modes highlight the locations where a harmonic forcing has the greatest effect on the global flowfield. It can be seen that the modal features agree qualitatively between the codes and also with existing literature in the field. Obviously, the coarseness of the chosen mesh to demonstrate the implementation of the methods would not allow a fully mesh-converged solution to be identified. This is exemplified by the streamwise position in the cylinder wake of the maximum cross-stream momentum component of the direct mode found at approximately $x = 14.6D$ for CODA and $x = 10.8D$ for TAU. Compare these with a maximum at approximately $25D$ for more mesh-converged results[18]. To address this point, the second test case of a square cylinder with a more detailed assessment is discussed next.

4.2 Laminar flow past square cylinder

We now consider the case of laminar $M = 0.2$ flow past a square cylinder at $Re = 50$, with a special focus on the high-order DG scheme in CODA. When using high-order DG schemes, it becomes necessary to use high-order grids to represent the curved geometries to achieve optimal accuracy[19]. However, since a square cylinder has no curves, it can be represented

Table 1
Degrees of freedom per equation for grids used for laminar flow past square cylinder case. Gray colour for cases not discussed herein.

Grid	Vertices (TAU)	Cells	DG2	DG3	DG4	DG5	DG6
L0	608	570	2280	5700	11400	19950	31920
L1	2356	2280	9120	22800	45600	79800	127680
L2	9920	9760	39040	97600	195200	341600	546560
L3	40300	39975	159900	399750	799500	1399125	2238600
L4	162440	161785	647140	1617850	3235700	5662475	9059960
L5	652240	650925	2603700	6509250	3018500	22782375	36451800

exactly (and easily) with linear grids. Hence, the added complexity of generating high-order grids was avoided for this case. All CODA cases were computed on a single compute node that comprises 384 GB of memory and two Intel Xeon Gold 6230 (Cascade Lake) CPUs each consisting of 20 hardware cores. The cases were run using two MPI processes with 20 OpenMP threads per process.

The case was computed on six structured grids (labelled as L0 to L5) with levels of varying refinement. A systematic global refinement approach with halving the mesh spacing and doubling the cell count in each spatial dimension was chosen. Thus, each new level has approximately four times as many cells as the previous level. The number of DOF per equation is listed in Table 1 for the FV schemes in TAU (vertex-centered) and CODA (cell-centered) and for the DG schemes in CODA until sixth-order. The number of DOF for DG schemes are computed for the three-dimensional formulation including the spanwise direction which is redundant for the two-dimensional square cylinder case. Far-field and near-field views of grids L0 and L2 are shown in Fig. 5. The computational domain extends to a circular far-field distance of approximately $200D$ where $D = 1$ is the length of the side of the square cylinder. DG schemes were employed in this study on grids L0 to L2 only, which is instructive when comparing with standard FV results. On each grid from L0 to L2, DG computations were performed from second order up to the minimum order that is needed for the lift coefficient C_L to become smaller than 10^{-8} in magnitude. Note that the theoretical lift coefficient for this symmetrical case is zero. Hence, deviation from the theoretical value is indicative of the adequacy of the spatial resolution for a given scheme.

The steady-state lift and drag coefficients for all cases computed are plotted in Fig. 6. Since lift coefficients were sometimes negative, their magnitudes are plotted instead for visualisation purposes. The FV scheme in CODA is slightly less accurate than that in TAU for the coarse grids L0 to L2, but it improves from grid L3 onward until eventually it surpasses TAU on the fine grids L4 and L5. This should be expected considering the very established inviscid flux discretisation in TAU. The steeper slope of CODA's FV scheme indicates that grid refinement has a more significant effect on accuracy for CODA than TAU. Nonetheless, the FV scheme in neither solver is able to achieve the specified tolerance of 10^{-8} for the lift coefficient on the available grids, although the CODA result comes close. In contrast, a sufficiently high-order DG scheme in CODA was able to achieve the expected accuracy requirement for the lift coefficient on all three grids tested; a sixth-order DG scheme was required on grid L0 while a fifth-order scheme was adequate on grids L1 and L2. The slopes for the DG curves are

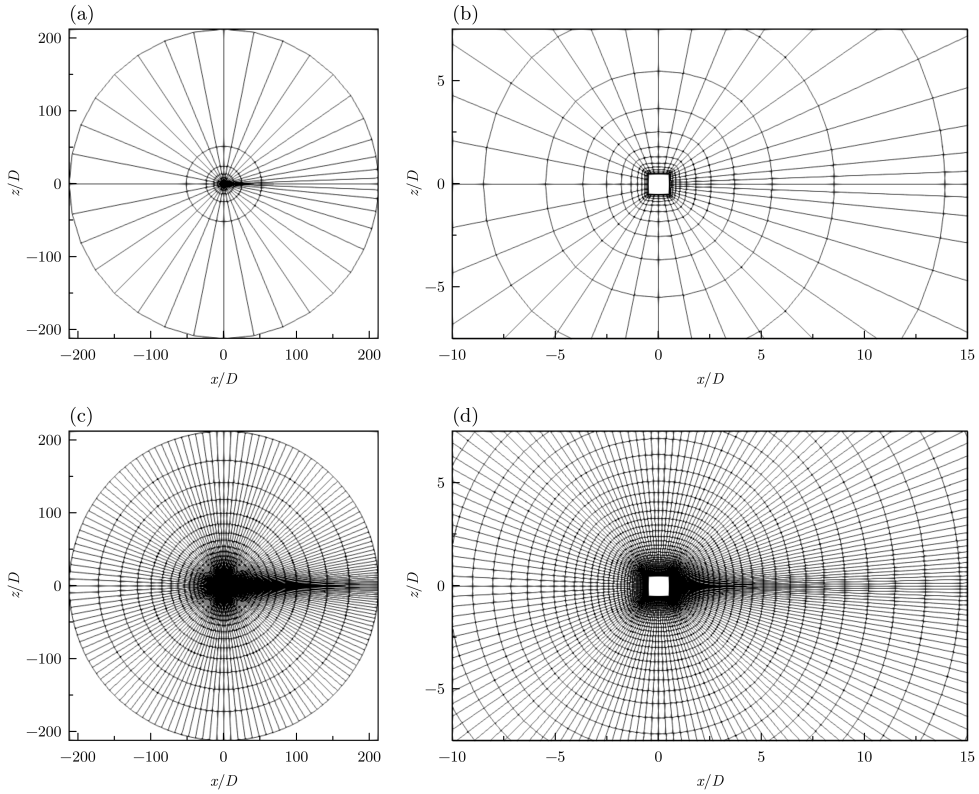

Figure 5. Grids L0 (top) and L2 (bottom) used for laminar flow past square cylinder case.

significantly steeper than those for the FV schemes indicating that, for a given problem size, increasing the order of the DG scheme (p-refinement) results in better accuracy than using a finer mesh (h-refinement) with FV scheme. It must be emphasised that the DG results are plotted against the number of DOF for the three-dimensional formulation. Using the number of DOF actually required for this purely two-dimensional problem would lead to even better estimates for the accuracy of DG schemes as this would not only shift the curves to the right but also increase their slopes as seen from the faint lines in Fig. 6(a). Figure 6(b) indicates that the drag coefficient converges to a value of approximately 1.52735, judging from the results of the FV computation in TAU on grid L5 and the fifth-order DG computation in CODA on grid L2. Using this approximate value as a reference, it can be seen that the FV scheme in TAU is more accurate than that in CODA on a given grid. For a given number of DOF, third- and higher-order DG schemes are more accurate than the FV scheme in CODA.

Similar to the circular cylinder case earlier, this case also possesses an unstable mode at the chosen flow conditions. The complex shift σ for the Krylov–Schur computations was chosen based on numerical experiments on the coarse L0 and L1 grids. Only the direct eigenvalue problem in Eq. (2) was considered for this case. Figure 7 shows the real parts of the streamwise momentum component $\widehat{\rho u}$ of the unstable mode computed on grid L0 with CODA. The DG results were mapped on grid L4 to visualize the sub-cell variation. The results were scaled

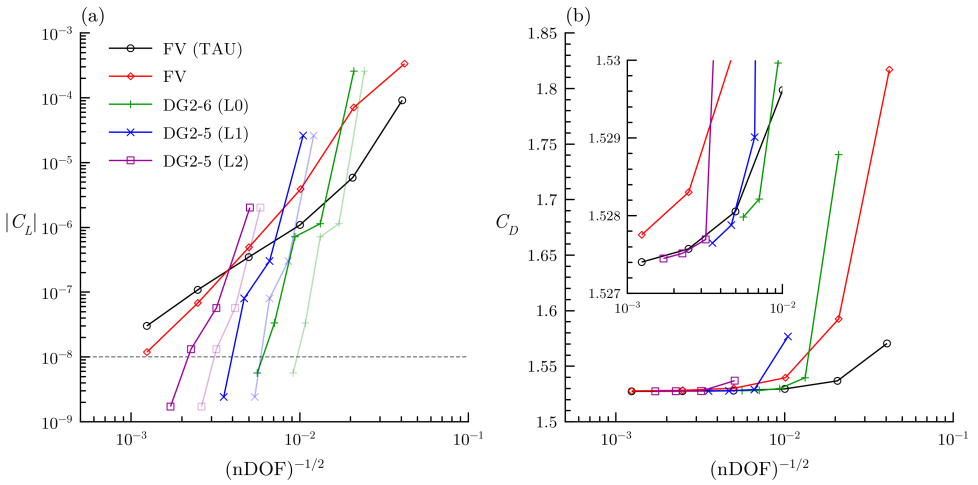

Figure 6. Steady-state lift and drag coefficients for laminar flow past square cylinder case. Faint lines for lift coefficients are plotted based on a theoretical purely two-dimensional DG formulation.

as described for the circular cylinder. It is apparent that the vortex shedding pattern becomes better defined as the order of the DG scheme increases. Note that at nominal second order, the DG2 formulation gives improved results compared with the FV scheme, when judging visually from the wake structures.

The non-dimensional eigenvalues λ_J for the unstable mode are plotted in Figure 8. It can be noticed that the FV scheme in CODA does not capture the unstable mode on grids L0 to L2, as evident from the negative real parts of the eigenvalue, but the FV scheme in TAU does so on all grids. With the exception of the second-order DG scheme on grids L0 and L1, all the DG computations are able to capture the unstable mode. The eigenvalues appear to be converging towards the same value with both *h*- and *p*-refinements. The closeup view in Fig. 8(b), which shows the two (to three) most converged eigenvalues for each curve, supports the notion that there is one *true* eigenvalue which can be computed with sufficient refinement. Even though not shown herein, eigenvalues computed using seventh- and eighth-order DG schemes on grid L0 and the FV scheme in TAU on grid L6 (roughly four times finer than grid L5) converged towards this same *true* point. Since we do not know the exact eigenvalue, we resort to measuring the convergence of the eigenvalues using relative changes in their magnitudes instead. For the FV computations, the relative changes are computed over successive grid levels (*h*-refinement), whereas for the DG computations, they are computed over successive orders of the DG scheme (*p*-refinement). Given the eigenvalues on successive refinement levels upon *h*- or *p*-refinement, specifically, λ_J^j and λ_J^{j-1}, the relative change is defined as

$$\Delta\lambda_J = \frac{\left|\lambda_J^j - \lambda_J^{j-1}\right|}{\frac{1}{2}\left(\left|\lambda_J^j\right| + \left|\lambda_J^{j-1}\right|\right)}. \tag{15}$$

The non-dimensional eigenvalues λ_J and their relative changes $\Delta\lambda_J$ are given in Table 2 for FV computations and in Tables 3 through 5 for DG computations.

As expected, the value of the relative changes in the eigenvalues decreases with refinement in general. The eigenvalues computed using CODA's FV scheme undergo larger changes with

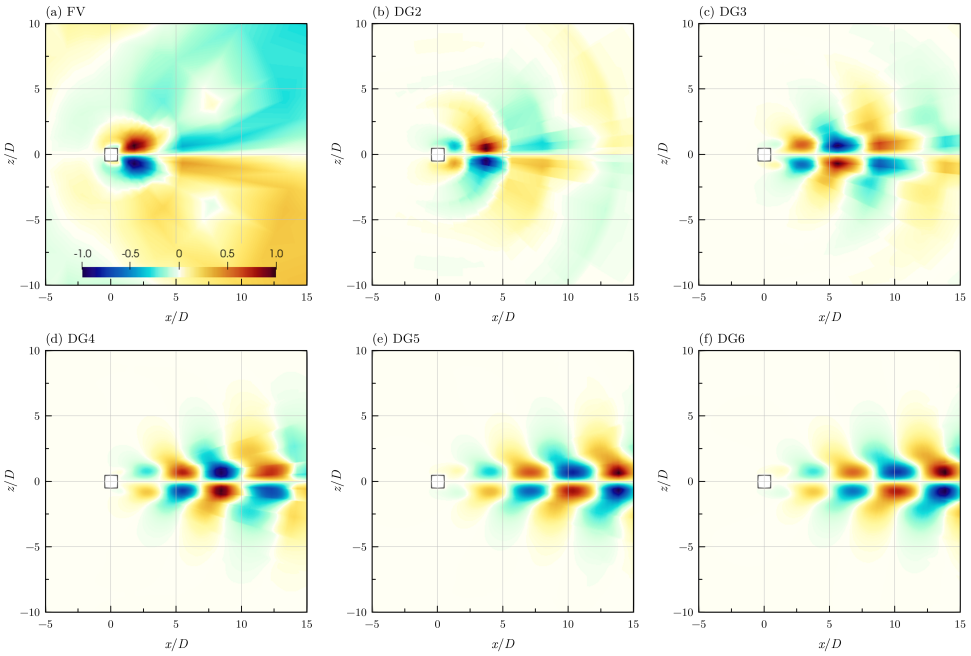

Figure 7. Streamwise momentum component of unstable direct modes computed on grid L0 with CODA.

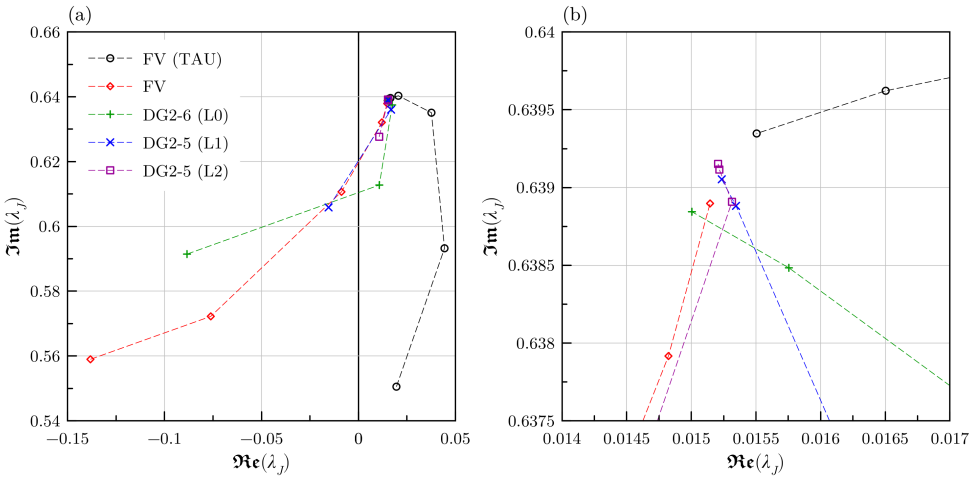

Figure 8. Non-dimensional eigenvalues of the unstable mode for laminar flow past square cylinder case.

grid refinement than those computed using TAU's FV scheme indicating that the eigenvalues computed on the coarser meshes using the former solver are further away from the exact value than those computed using the latter. Despite the initial slow convergence, FV computation on grid L5 using CODA shows similar convergence to that of TAU with about 0.16% change in the eigenvalue (cf. Table 2). This can also be confirmed in Fig. 8(b) in which the eigenvalues

Table 2
Eigenvalues for laminar flow past square cylinder case using FV schemes.

	TAU		CODA	
	λ_J	$\Delta\lambda_J$	λ_J	$\Delta\lambda_J$
L0	1.9583e-02 + *i*5.5053e-01	—	-1.3821e-01 + *i*5.5898e-01	—
L1	4.4385e-02 + *i*5.9324e-01	8.6e-02	-7.6152e-02 + *i*5.7223e-01	1.1e-01
L2	3.7634e-02 + *i*6.3509e-01	6.9e-02	-8.7558e-03 + *i*6.1066e-01	1.3e-01
L3	2.0590e-02 + *i*6.4032e-01	2.8e-02	1.2061e-02 + *i*6.3210e-01	4.8e-02
L4	1.6503e-02 + *i*6.3962e-01	6.5e-03	1.4823e-02 + *i*6.3792e-01	1.0e-02
L5	1.5505e-02 + *i*6.3935e-01	1.6e-03	1.5145e-02 + *i*6.3890e-01	1.6e-03

Table 3
Eigenvalues for laminar flow past square cylinder case using DG on grid L0.

	λ_J	$\Delta\lambda_J$
DG2	-8.8413e-02 + *i*5.9145e-01	—
DG3	1.0756e-02 + *i*6.1276e-01	1.7e-01
DG4	1.7415e-02 + *i*6.3747e-01	4.1e-02
DG5	1.5754e-02 + *i*6.3848e-01	3.0e-03
DG6	1.5004e-02 + *i*6.3884e-01	1.3e-03

Table 4
Eigenvalues for laminar flow past square cylinder case using DG on grid L1.

	λ_J	$\Delta\lambda_J$
DG2	-1.5398e-02 + *i*6.0585e-01	—
DG3	1.6835e-02 + *i*6.3604e-01	7.1e-02
DG4	1.5345e-02 + *i*6.3888e-01	5.0e-03
DG5	1.5235e-02 + *i*6.3905e-01	3.2e-04

Table 5
Eigenvalues for laminar flow past square cylinder case using DG on grid L2.

	λ_J	$\Delta\lambda_J$
DG2	1.0705e-02 + *i*6.2766e-01	—
DG3	1.5314e-02 + *i*6.3891e-01	1.9e-02
DG4	1.5217e-02 + *i*6.3912e-01	3.6e-04
DG5	1.5207e-02 + *i*6.3915e-01	5.9e-05

computed on grid L5 using the FV schemes in CODA and TAU are somewhat equidistant from the exact value which should be in the vicinity of the eigenvalue computed using fifth order DG scheme on grid L2. It can be seen from Table 3 that a sixth order DG scheme on grid L0 surpasses this with only 0.13% change in the eigenvalue. The convergence of the eigenvalues computed using DG schemes improves on grids L1 and L2 with 0.032% and 0.0059% changes observed for a fifth-order scheme, respectively.

While the advantages of DG schemes for the non-linear flow problem are often stated in the literature in terms of computational cost, for the herein presented linearised stability analysis

it can be said that high-order DG schemes were able to achieve a given level of convergence (measured using $\Delta\lambda_J$) much faster than the FV scheme. For instance, CODA FV computation on grid L5 and sixth order DG scheme on grid L0 both achieved the same level of convergence but the computation time for the latter was an order of magnitude lower than the former. However, this is a rather intricate discussion; for instance, convergence can be strongly affected by the condition number of the shift-invert matrix eigenvalue problem.

5.0 Summary

The Krylov–Schur algorithm for solving large eigenvalue problems has been implemented within the framework of the next-generation flow solver CODA. The implementation was validated using laminar flow past a circular cylinder case for which the eigenvalues and eigenvectors computed using the FV scheme in CODA were shown to agree qualitatively with those computed using the FV scheme in TAU for both the direct and adjoint eigenvalue problems. The case of laminar flow past a square cylinder was used to investigate the possible benefits of using high-order DG schemes over FV scheme for solving the eigenvalue problem. The results presented herein demonstrate that, unlike the FV schemes which require very fine meshes, high-order DG schemes lead to well-converged eigensolutions on coarser meshes at a lower computational cost.

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.

CODA is the computational fluid dynamics (CFD) software being developed as part of a collaboration between the French Aerospace Lab ONERA, the German Aerospace Center (DLR), Airbus, and their European research partners. CODA is jointly owned by ONERA, DLR and Airbus.

We thank the University of Liverpool for computing time on the high-performance computing system. The simulation data that support the findings of this study are available from the authors upon reasonable request.

Appendix

The diagonal elements of the upper-triangular matrix S_k need to be reordered using orthogonal similarity transformations prior to the restart. Without loss of generality, let us consider a 2×2 upper-triangular matrix

$$S_2 = \begin{bmatrix} a & b \\ & c \end{bmatrix}, \tag{16}$$

to which we apply an orthogonal similarity transformation $S_2' = US_2U^H$ which swaps the diagonal elements a and c while keeping S_2' upper-triangular. Let us assume that $a \neq c$ since

if $a = c$, no swapping would be needed in the first place. A 2×2 orthogonal matrix U can be written as

$$U = \begin{bmatrix} p & q \\ q^H & -p^H \end{bmatrix}, \tag{17}$$

where $pp^H + qq^H = 1$. Since we require the bottom-left element of S_2' to be zero, we arrive at the condition

$$q^H \left(dp^H - bq^H \right) = 0, \tag{18}$$

where $d = c - a$. Setting $q^H = 0$ satisfies the condition but it leads to the trivial case $U = I$ which does not swap the diagonal elements. Therefore, the term inside the brackets must vanish and Eq. (18) reduces to

$$\frac{p^H}{q^H} = \frac{b}{d}. \tag{19}$$

Since $a \neq c$, the denominator does not vanish. Multiplying Eq. (19) with its complex conjugate and substituting $qq^H = 1 - pp^H$ leads to

$$pp^H = \frac{bb^h}{bb^H + dd^H}. \tag{20}$$

We can choose p to be real and positive with a magnitude equal to the square root of the value on the right-hand-side of Eq. (20). The value of q can be obtained from Eq. (19) as $q = pd^H/b^H$.

To prove that the diagonal elements are indeed swapped, let us compute the top-left element of S_2' and show that it is equal to c;

$$\begin{aligned} app^H + bpq^H + cqq^H &= app^H + bpq^H + c\left(1 - pp^H\right) \\ &= c - dpp^H + bpq^H \\ &= c - p\cancel{\left(dp^H - bq^H\right)} \\ &= c \end{aligned}$$

The cancellation in the third line occurs due to Eq. (19). A similar computation of the bottom-right element of S_2' shows that it is equal to a. Alternatively, one can reason that since an orthogonal similarity transformation preserves the eigenvalues and since the eigenvalues of an upper-triangular matrix appear on its diagonal if the top-left element of S_2' is c, the bottom-right element of S_2' must be a.

In the current implementation, the diagonal elements of S_k are reordered by repeatedly sweeping down the diagonal and swapping adjacent diagonal elements as needed until no more swaps occur during a sweep. If we wish to swap diagonal elements s_j and s_{j+1}, the similarity transformation matrix V can be constructed by replacing the 2×2 diagonal block of a $k \times k$ identity matrix at location (j, j) with the 2×2 orthogonal matrix U, specifically

$$V = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & p & q & & \\ & & p^H & -q^H & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix} \tag{21}$$

References

1. J. D. Crouch, A. Garbaruk, and D. Magidov. Predicting the onset of flow unsteadiness based on global instability. *Journal of Computational Physics*, 224(2):924–940, 2007.
2. S. Timme. Global instability of wing shock-buffet onset. *Journal of Fluid Mechanics*, 885:A37, 2020.
3. R.B. Lehoucq, D.C. Sorensen, and C. Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.
4. M. Wagner, J. Jägersküpper, D. Molka, and T. Gerhold. *Performance analysis of complex engineering frameworks*, pages 123–138. Springer, Cham, 2021.
5. T. Leicht, J. Jägersküpper, D. Vollmer, A. Schwöppe, R. Hartmann, J. Fiedler, and T. Schlauch. DLR-Project Digital-X – Next Generation CFD Solver 'Flucs'. In *Deutscher Luft- und Raumfahrtkongress 2016*, 2016.
6. M. Wagner. The CFD Solver CODA and the Sparse Linear Systems Solver Spliss: Evaluation of Performance and Scalability. In *NHR CFD Workshop Day 2021*, 2021.
7. G.W. Stewart. A Krylov–Schur algorithm for large eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 23(3):601–614, 2002.
8. U S Vevek, S. Timme, J. Pattinson, B. Stickan, and A. Büchner. Next-generation computation fluids dynamics capabilities for aircraft aeroelasticity and loads. In *19th International Forum on Aeroelasticity and Structural Dynamics, IFASD 2022*, 2022.
9. G.W. Stewart. Addendum to "A Krylov–Schur algorithm for large eigenproblems". *SIAM journal on matrix analysis and applications*, 24(2):599–601, 2002.
10. D.C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *Siam journal on matrix analysis and applications*, 13(1):357–385, 1992.
11. J.G.F. Francis. The QR transformation a unitary analogue to the LR transformation—Part 1. *The Computer Journal*, 4(3):265–271, 1961.
12. J.G.F. Francis. The QR transformation—part 2. *The Computer Journal*, 4(4):332–345, 1962.
13. Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.
14. A. McCracken, A. Da Ronch, S. Timme, and K.J. Badcock. Solution of linear systems in fourier-based methods for aircraft applications. *International Journal of Computational Fluid Dynamics*, 27(2):79–87, 2013.
15. A. Schwöppe and B. Diskin. Accuracy of the cell-centered grid metric in the DLR TAU-code. In *New Results in Numerical and Experimental Fluid Mechanics VIII*, pages 429–437. Springer, 2013.
16. F. Bassi, A. Crivellini, S. Rebay, and M. Savini. Discontinuous Galerkin solution of the Reynolds-averaged Navier–Stokes and k–ω turbulence model equations. *Computers & Fluids*, 34(4-5):507–540, 2005.
17. D. Schwamborn, T. Gerhold, and R. Heinrich. The DLR TAU-code: Recent applications in research and industry. In *ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics*, 01 2006.

18. D. Sipp and A. Lebedev. Global stability of base and mean flows: a general approach and its applications to cylinder and open cavity flows. *Journal of Fluid Mechanics*, 593: 333–358, 2007.

19. R. Hartmann and T. Leicht. Generation of unstructured curvilinear grids and high-order discontinuous Galerkin discretization applied to a 3D high-lift configuration. *International Journal for Numerical Methods in Fluids*, 82(6):316–333, 2016.