The Aeronautical Journal (2024), **128**, pp. 1164–1182
doi:10.1017/aer.2023.108

RESEARCH ARTICLE

Bespoke stability analysis tool in next-generation computational fluid dynamics solver

U S Vevek, J. Houtman and S. Timme

School of Engineering, University of Liverpool, Liverpool, England L69 3GH, UK
Corresponding author: S. Timme; Email: sebastian.timme@liverpool.ac.uk

Received: 20 April 2023; **Revised:** 9 October 2023; **Accepted:** 16 October 2023

Keywords: computational fluid dynamics; discontinuous Galerkin; stability analysis; Krylov-Schur algorithm

Abstract
This paper presents some of the first results of global linear stability analyses performed using a bespoke eigensolver that has recently been 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.

Nomenclature

A	Shift-invert Jacobian operator
b	Residual vector of Schur form
C_L, C_D	Coefficients of lift and drag
D	Diameter of circular cylinder or length of side of square cylinder
H	Upper-Hessenberg matrix
$J, J^\dagger$	Jacobian operator and its adjoint
M	Weight matrix
q	Orthogonal basis (Arnoldi) vectors
Q	Matrix of orthogonal basis vectors
p	Eigenvector of matrix in Schur form
R	Residual vector
s	Diagonal element of matrix in Schur form
S	Matrix in Schur form
t	Time coordinate
T	Matrix before similarity transformation to return to Schur form
U_∞	Freestream velocity
V	Orthogonal matrix for similarity transformation
W	State vector of conservative variables
$\widehat{W}, \widehat{W}^+$	Eigenvectors of Jacobian operator and its adjoint
W	Matrix of eigenvectors of the Jacobian operator
x, z	Cartesian spatial coordinates

Greek Symbol

ϵ	Error norm
λ	Eigenvalue
σ	Complex shift

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 [1–3]. 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 nonlinear 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 [4], that are limited to MPI parallelisation only.

The next generation flow solver CODA (CFD for ONERA, DLR and Airbus) [5, 6] 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 linear stability analysis of large problems of engineering relevance. The linear systems in CODA are solved using preconditioned Krylov subspace methods that are available in CODA's sparse linear algebra library Spliss [7]. 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 [8] 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 [9]. The paper continues with a discussion of the relevant theory, including the Krylov–Schur method, in Section 2.0, before outlining the numerical methodology in Section 3.0. Results for two test cases, specifically the laminar flow at Reynolds number $Re = 50$ around the circular and square cylinders, are presented in Section 4.0.

2.0 Theory

2.1 Basics

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

$$\frac{d\boldsymbol{W}}{dt} + \boldsymbol{R}\left(\boldsymbol{W}\right) = \boldsymbol{0}, \tag{1}$$

where $\boldsymbol{W} = [\rho, \rho u, \rho E, \ldots]$ denotes the vector of conservative variables (depending on the chosen flow model which can include additional equations for modelling turbulence and transition) and $\boldsymbol{R}\left(\boldsymbol{W}\right)$ denotes the corresponding non-linear residual vector in discretised form. Linearising Equation (1) about a steady-state solution $\overline{\boldsymbol{W}}$, while assuming small-amplitude perturbations of the form $\widehat{\boldsymbol{W}}\exp\left(\lambda_J t\right)$, 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, and $\widehat{\boldsymbol{W}}$ and λ_J are the complex eigenvector and eigenvalue, respectively. 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 ($\Re\mathfrak{e}\left(\lambda_J\right) > 0$) or decays ($\Re\mathfrak{e}\left(\lambda_J\right) < 0$) 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 a and 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 (or element) 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 $\lambda_A = \left(\lambda_J - \sigma\right)^{-1}$ 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 larger 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 [8, 10] is a Krylov subspace method that can be used to determine a small number of eigenvalue-eigenvector pairs (eigenpairs) of a large sparse complex 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 while preserving the relevant information. The Krylov–Schur algorithm improves upon the implicitly restarted Arnoldi algorithm [11] by proposing a simple but robust restarting technique.

The key to the restarting technique is the Schur form

$$AQ_j = Q_{j+1}\begin{bmatrix} S_j \\ \boldsymbol{b}_j^H \end{bmatrix} = Q_{j+1}\tilde{S}_j, \tag{5}$$

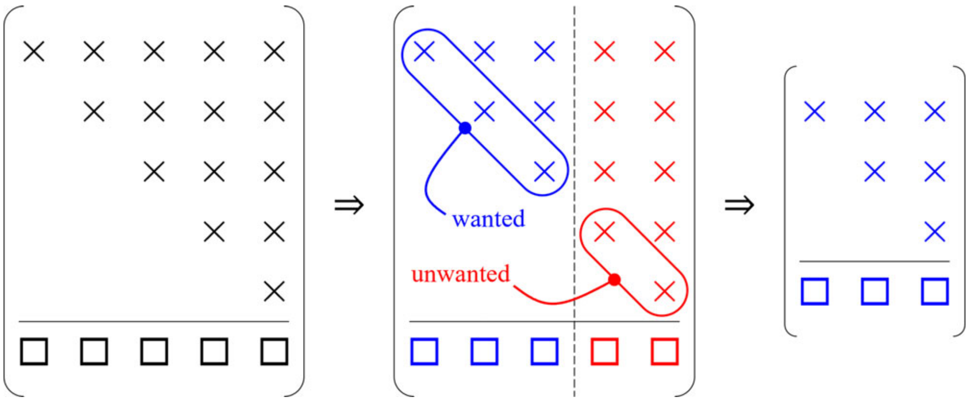


Figure 1. *Schematic of $\tilde{S}_m$ during a restart for $m = 5$ and $t = 3$.*

where $Q_{j+1} = \left[Q_j \; \boldsymbol{q}_{j+1}\right]$ is an orthogonal matrix with $Q_1 = \left[\boldsymbol{q}_1\right]$, S_j is a $j \times j$ upper-triangular matrix and $\boldsymbol{b}_j^H$ is a $1 \times j$ row vector. The eigenvalues of S_j, which are found along its diagonal, are good approximations to the largest eigenvalues of A, i.e. the wanted eigenvalues of J after shift-invert transformation, once the elements of $\boldsymbol{b}_j^H$ fall below a user-defined tolerance.

Upon reaching some maximum specified subspace size $j = m$, the Schur form is split into two parts, one consisting of the first t columns and the other consisting of the remaining $(m - t)$ columns, such that

$$A\left[Q_t \quad Q_{m-t}\right] = \left[Q_t \quad Q_{m-t} \quad \boldsymbol{q}_{m+1}\right] \begin{bmatrix} S_t & S_{t,m-t} \\ 0 & S_{m-t} \\ \boldsymbol{b}_t^H & \boldsymbol{b}_{m-t}^H \end{bmatrix}. \tag{6}$$

The restart technique simply involves discarding the last $(m - t)$ columns of Q_m and S_m yielding a contracted Schur form

$$AQ_t = \left[Q_t \quad \boldsymbol{q}_{m+1}\right] \begin{bmatrix} S_t \\ \boldsymbol{b}_t^H \end{bmatrix} = Q_{t+1}\tilde{S}_t. \tag{7}$$

Note that vector $\boldsymbol{q}_{m+1}$ is retained and re-labelled as $\boldsymbol{q}_{t+1}$ to be consistent with the Schur form notation. The contraction procedure from S_m to S_t is shown schematically in Fig. 1 for $m = 5$ and $t = 3$. It was previously shown [8] that this restart technique is equivalent to applying a polynomial filter, $p(r) = (r - s_{t+1}) \times \cdots \times (r - s_m)$ where s_j is the jth diagonal element in S_m. If the diagonal elements of S_m are arranged such that the last $(m - t)$ 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 Equation (5). Performing an Arnoldi iteration on vector $\boldsymbol{q}_{j+1}$ results in

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

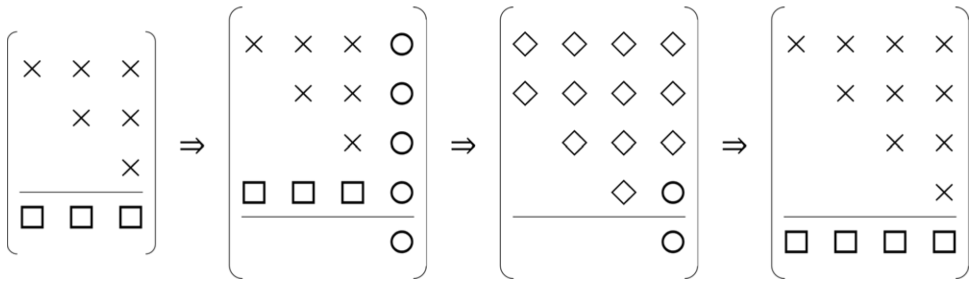


Figure 2. *Schematic of $\tilde{S}_k$ during a Krylov-Schur iteration for $j = 3$.*

where the $j \times 1$ column vector $\boldsymbol{f}_j$ and the scalars g and h are obtained by orthogonalising $A\boldsymbol{q}_{j+1}$ over Q_{j+1}. Since the Arnoldi iteration has disrupted the Schur form, we must perform a series of orthogonal similarity transformations to bring T_{j+1} back to the Schur form [12]. First, T_{j+1} is brought into the upper-Hessenberg form $H_{j+1} = V_1^H T_{j+1} V_1$ using Householder reflections. Then, H_{j+1} is brought into the complex Schur form $S'_{j+1} = V_2^H H_{j+1} V_2$ using the shifted QR algorithm [13, 14]. Finally, the diagonal elements of the upper-triangular matrix S'_{j+1} are reordered in descending order of their absolute values using special orthogonal matrices (see Appendix B) which yields the ordered Schur form $S_{j+1} = V_3^H S'_{j+1} V_3$. 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 $S_{j+1} = V^H T_{j+1} V$ where $V = V_1 V_2 V_3$ is an orthogonal matrix. Substituting $T_{j+1} = V S_{j+1} V^H$ into Equation (8) results in

$$AQ_{j+1} = Q_{j+1}\left(VS_{j+1}V^H\right) + h\,\boldsymbol{q}_{j+2}\,\boldsymbol{e}_{j+1}^T. \tag{9}$$

Multiplying both sides of the latter equation by V from the right and setting $Q_{j+1} = Q_{j+1}V$ and $b_{j+1}^H = he_{j+1}^T V$ results in

$$AQ_{j+1} = Q_{j+1}S_{j+1} + \boldsymbol{q}_{j+2}\,\boldsymbol{b}_{j+1}^H \tag{10}$$

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

$$AQ_1 = \begin{bmatrix} Q_1 & \boldsymbol{q}_2 \end{bmatrix} \begin{bmatrix} S_1 \\ \boldsymbol{b}_1^H \end{bmatrix}, \tag{11}$$

where S_1 and $\boldsymbol{b}_1^H$ are just scalars and $Q_1 = \left[\boldsymbol{q}_1\right]$. Nonetheless, it can be seen that Equation (11) is in the Schur form and requires no further transformation.

The desired $k \le j$ eigenvectors of A can be approximated from the Schur form as

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

where matrix $W_k = \left[\widehat{\boldsymbol{W}}_1 \ \ldots \ \widehat{\boldsymbol{W}}_k\right]$ contains the approximate eigenvectors (Ritz vectors) of A and $P_k = \left[\boldsymbol{p}_1 \ \ldots \ \boldsymbol{p}_k\right]$ is a $k \times k$ matrix whose columns are the eigenvectors of S_k which is the $k \times k$ matrix formed from the top-left corner of S_j. The eigenvectors $\boldsymbol{p}_i$ for $i = 1 \ldots k$ can be computed from the relationship $S_k\boldsymbol{p}_i = s_i\boldsymbol{p}_i$ combined with the fact that both S_k and P_k are upper-triangular.

The error in the ith eigenvector approximation can be computed as

$$\varepsilon_i = \| A\widehat{\boldsymbol{W}}_i - s_i\widehat{\boldsymbol{W}}_i \|_2. \tag{13}$$

Substituting $\widehat{\boldsymbol{W}}_i = Q_k\boldsymbol{p}_i$ and $s_i\boldsymbol{p}_i = S_k\boldsymbol{p}_i$ into Equation (13) leads to

$$\varepsilon_i = \| (AQ_k - Q_kS_k)\,\boldsymbol{p}_i \|_2 = \| \boldsymbol{q}_{k+1}\boldsymbol{b}_k^H\boldsymbol{p}_i \|_2 = \| \boldsymbol{b}_k^H\boldsymbol{p}_i \|_2 \tag{14}$$

where $\boldsymbol{b}_k$ is formed from the first k elements of $\boldsymbol{b}_j$. The second equality in Equation (14) 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 ith 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 ith component of $\boldsymbol{b}_j$ to zero. Subsequent transformations on S_j are only applied from the $(i+1)$th row and column leaving the top-left $i \times i$ part of S_j and the first i columns of Q_j unchanged. The computation is terminated when the last desired eigenpair has converged. The entire algorithm is summarised in Appendix A. Note that we have used Fortran indexing with the indices starting from 1.

3.0 Methodology

3.1 Non-linear steady-state problem

The steady-state problem $\boldsymbol{R}\left(\overline{\boldsymbol{W}}\right) = \boldsymbol{0}$ 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 10 orders of magnitude. No special solution stabilisation techniques (such as selective frequency damping) were required to achieve convergence herein. 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 restarted generalised minimal residual (GMRES) solver [15] until the linear residual norm dropped by one order of magnitude. The solver used a maximum of 100 Krylov vectors before restart with 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 $J\boldsymbol{x}$ exactly for an arbitrary real (or complex) vector $\boldsymbol{x}$. The preconditioner, on the other hand, uses an explicitly formed matrix whose block-diagonal (i.e. the coupled governing equations for all spatial degrees of freedom within a cell) 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 [16].

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 ${}^{n+2}C_3 = (n+2)\,(n+1)\,n/6$ DOF for each variable where ${}^{n}C_k$ is the binomial coefficient. For the two-dimensional cases considered in this study, the DOF per cell can be significantly reduced (only ${}^{n+1}C_2$) 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 [17]. In the DG method, the viscous fluxes were computed using the second approach proposed by Bassi and Rebay known as the BR2 scheme [18].

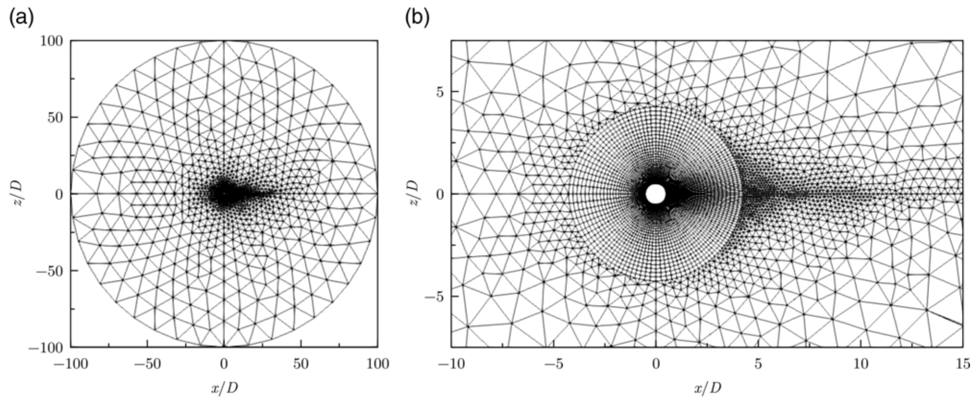


***Figure 3.** Grid used for laminar flow past circular cylinder case.*

3.2 Eigenvalue problem

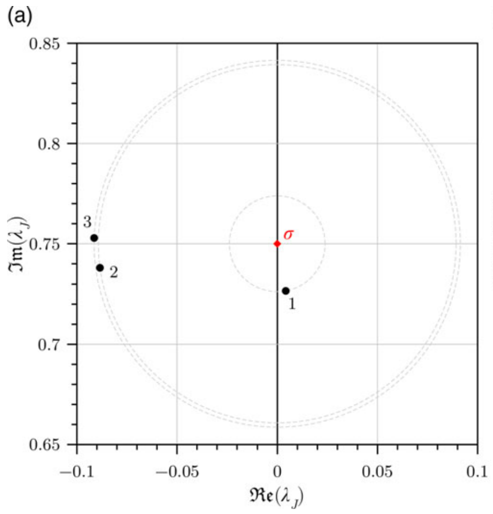
With the steady-state solution computed, the Krylov–Schur algorithm, implemented as part of this work, was used to determine the rightmost (in the complex plane) eigenpair. Observe that the chosen flow conditions are such that $\Re\mathfrak{e}\,(\lambda_J) > 0$ for the leading eigenmode and it describes unstable flow, specifically vortex shedding in the wake behind bluff bodies. In the Krylov–Schur algorithm, unless otherwise specified, the subspace was allowed to expand to a maximum size $m = 20$ before a restart, when it was truncated to $t = 5$. The linear system of the form $(J - \sigma I)\,\boldsymbol{x} = \boldsymbol{q}$ 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. CODA's linear algebra library Spliss can deal with the necessary complex algebra and further details on solving the complex linear problems can be found in previous work [9]. 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 a relative approximation error below 10^{-10}, the linear system in each Krylov–Schur iteration was solved until the residual norm dropped below 10^{-10}.

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 11,638 cells (corresponding to 9,743 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 test case was computed at a slightly supercritical Reynolds number $Re = 50$ to ensure the presence of an unstable mode linked to von Kármán vortex shedding. The complex shift, $\sigma = i0.75$, was chosen such that its imaginary part was close to the critical vortex-shedding frequency, which relies on engineering insight gained from the past study of bluff body aerodynamics. While 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, it is clear that there is a compromise between the rate of convergence



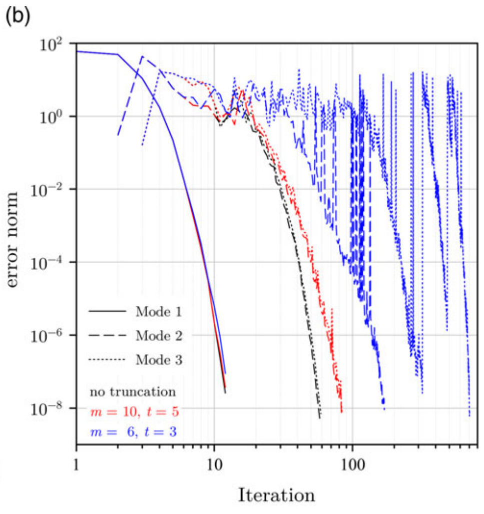


Figure 4. *Convergence characteristics of three direct modes closest to chosen shift for laminar flow past circular cylinder case using CODA.*

of the shift-inverted Krylov–Schur algorithm, which improves with a reduced spectral distance, and the numerical properties of the shifted linear system, which becomes increasingly singular and hence more difficult to solve with reduced spectral distance. Both the direct eigenvalue problem in Equation (2) and the adjoint eigenvalue problem in Equation (3) were solved using CODA and TAU.[1] For CODA, only the FV method was used for this case.

But first, to demonstrate the implementation of the Krylov–Schur algorithm in CODA and the convergence characteristics of the cylinder validation case, Fig. 4 shows details of the computation of the three direct modes closest to the chosen shift. As expected, the modes converge consecutively with increasing spectral distance, $|\lambda_J - \sigma|$. Three parameter settings of the Krylov–Schur algorithm were specified (in terms of maximum and truncated subspace size denoted m and t, respectively); no truncation, $[m = 10,\ t = 5]$ and $[m = 6,\ t = 3]$. It is evident that the leading, well-isolated wake mode converges in 12 iterations regardless of the subspace size after restart. Modes 2 and 3 are more difficult to converge; these modes are close to a large number of nondescript, spurious modes [21]. Without subspace truncation, these modes both converge in just under 60 iterations. For the setting with the bigger limited subspace, convergence is still straightforward in just over 80 iterations. Convergence is slow using the final setting, with the algorithm locking onto the incorrect modes repeatedly. Nevertheless, eventually the sought modes are converged in 170 and 700 iterations, respectively. Essentially, too small a subspace is insufficient for non-isolated modes.

The leading, physically meaningful, 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 $0.0042507 + i0.72656$ and $0.015829 + i0.73105$, respectively. The positive real parts of λ_J indicate that the modes are indeed unstable. Overall, the results for λ_J are quite close to the numerical predictions by

[1]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 [19]. Spatial discretisation herein used 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]. Specifically, the implicitly restarted Arnoldi method from the ARPACK library was used herein, in contrast to the bespoke Krylov–Schur algorithm in CODA. Resulting linear systems for the stability analysis were solved using the generalised conjugate residual solver with inner orthogonalisation and deflated restarting with incomplete lower-upper factorisation (zero fill) preconditioning [20].

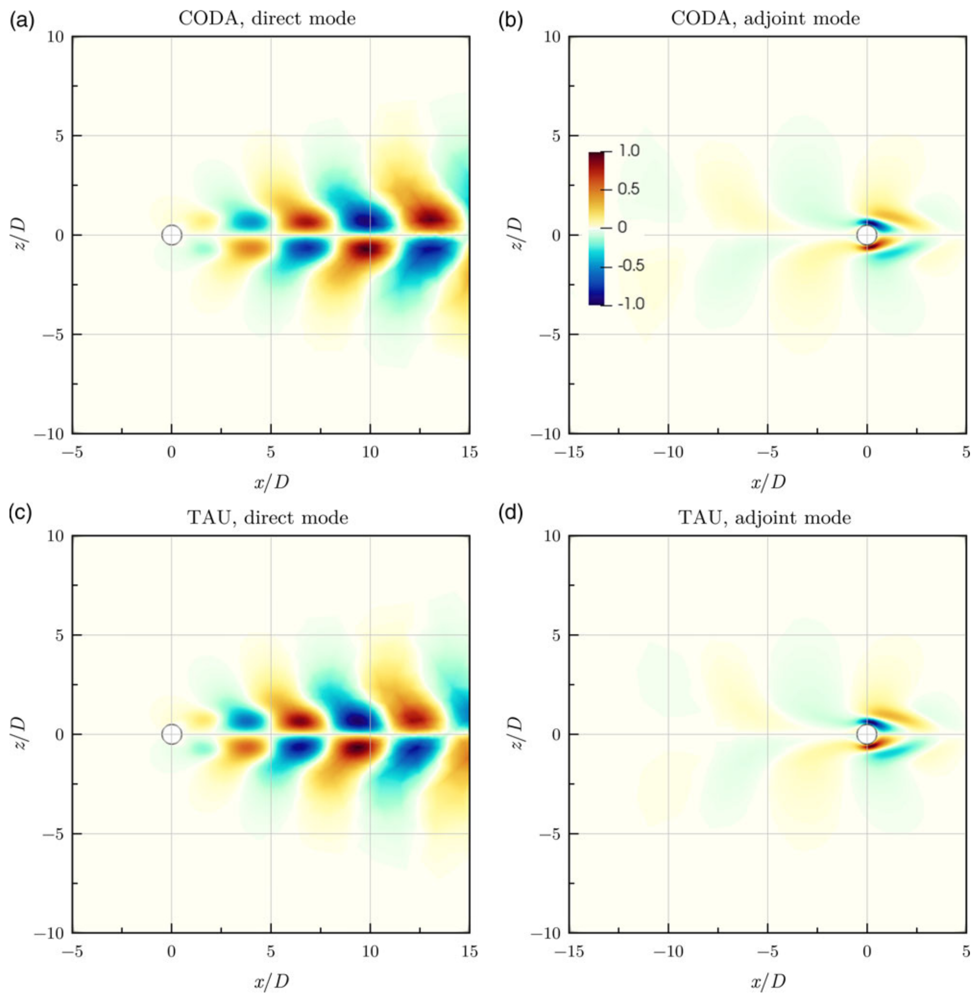


Figure 5. *Streamwise momentum components of unstable direct and adjoint global modes for laminar flow past circular cylinder case.*

e.g. Crouch et al. [1] and Fabre et al. [22] based on similar stability analyses. The real parts of the streamwise momentum component $\widehat{\rho u}$ of the eigenvectors are plotted in Fig. 5. The eigenvectors were scaled such that the maximum of the $\widehat{\rho u}$ component has a 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 [23]. To address this point, the second test case of a square cylinder in laminar flow with a more detailed assessment is discussed next.

Table 1. *Degrees of freedom per equation for laminar flow past square cylinder cases discussed herein; underlined cases are not further visualised for clarity*

Grid	Vertices (TAU)	Cells	DG2	DG3	DG4	DG5	DG6	DG7	DG8
L0	608	570	2,280	5,700	11,400	19,950	31,920	47,880	68,400
L1	2,356	2,280	9,120	22,800	45,600	79,800	127,680	191,520	
L2	9,920	9,760	39,040	97,600	195,200	341,600			
L3	40,300	39,975	<u>159,900</u>	<u>399,750</u>					
L4	162,440	161,785							
L5	652,240	650,925							
L6	2,613,920	2,611,285							

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 [24]. However, since a square cylinder has no curves, it can be represented 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) central processing units (CPUs) each consisting of 20 hardware cores. The cases were run using two MPI processes with 20 OpenMP threads per process.

The square cylinder was computed using seven structured grids (labelled as L0 to L6) 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-centred) and CODA (cell-centred) and for the DG schemes in CODA for the cases discussed herein. The number of DOF for DG schemes is 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. 6. 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 L3 only, which is instructive when comparing with standard FV results. On grid L0, DG computations were performed from second order up to the maximum (currently) available order of eight. On grids L1 through L3, DG computations were performed from second order up to (at least) the minimum order that is needed for the lift coefficient C_L to become smaller than (an arbitrarily chosen) 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 (except for DG on grid L3 to aid clarity of the presentation) are plotted in Fig. 7. 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. This is, of course, an intricate discussion considering inherent differences of cell-centred and cell-vertex schemes. Nonetheless, the FV schemes in both CODA and TAU achieve the specified tolerance of 10^{-8} for the lift coefficient on grid L6. In contrast, a sufficiently high-order DG scheme in CODA was able to achieve the expected accuracy requirement for the lift coefficient on all four 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 lift coefficient does not drop much further on grid L0 when moving to seventh- and eighth-order DG schemes which suggests that using

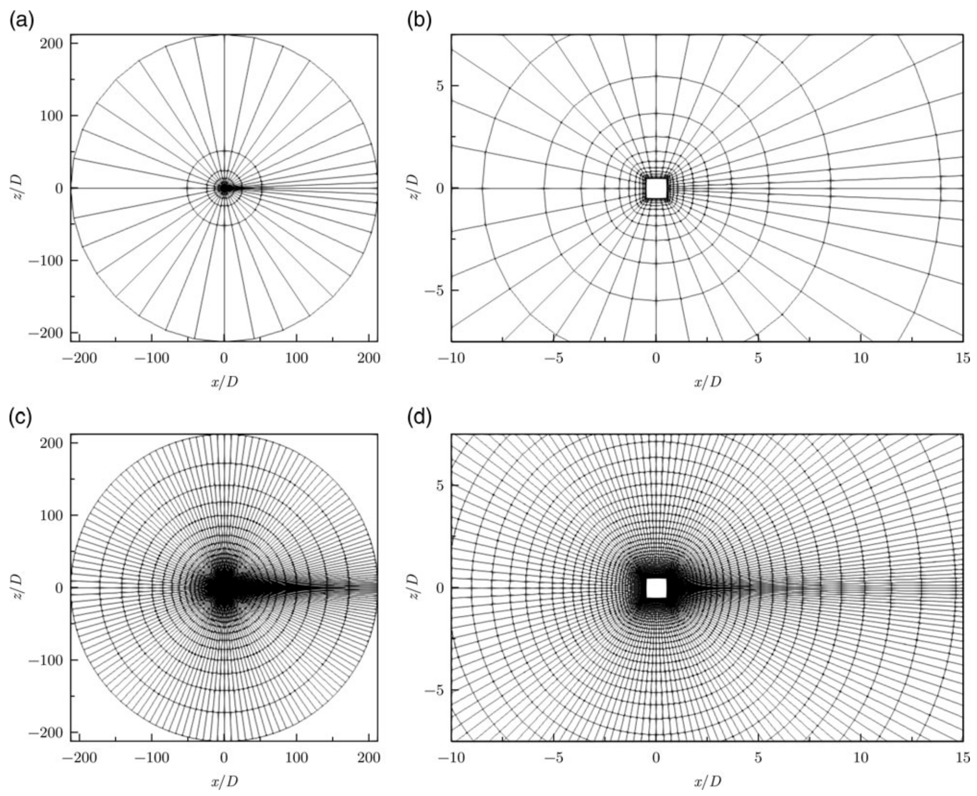


Figure 6. *Grids L0 (top) and L2 (bottom) used for laminar flow past square cylinder case.*

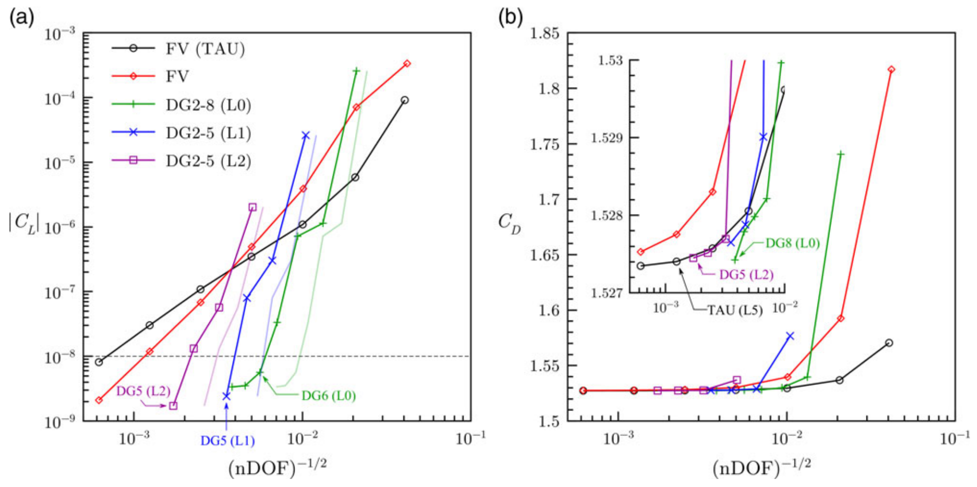


Figure 7. *Steady-state lift and drag coefficients for laminar flow past square cylinder case. Faint lines for lift coefficients are plotted based on a theoretical purely two-dimensional DG formulation.*

a finer mesh might be useful. For grid L3, our criterion on the lift coefficient was met with third-order DG. The slopes for the DG curves are 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 a FV scheme. It must be emphasised that the DG

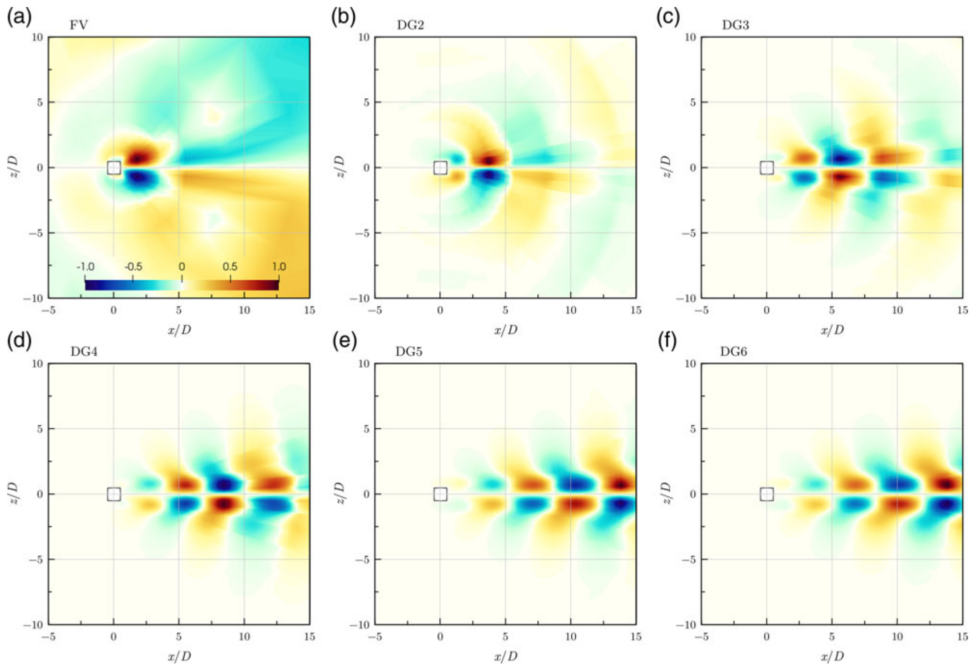


Figure 8. *Streamwise momentum component of unstable direct modes computed on grid L0 with CODA.*

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. 7(a). Figure 7(b) indicates that the drag coefficient converges to a value of approximately $C_D = 1.5274$, judging from the results of the FV computation in TAU and the DG computations in CODA. 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 for the Krylov–Schur computations was chosen based on numerical experiments on the coarse L0 and L1 grids, specifically $\sigma = 0.1 + i0.63$. Only the direct eigenvalue problem in Equation (2) was considered for this case. Figure 8 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 onto grid L4 to visualise the sub-cell variation. The results were scaled 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 Fig. 9. It can be noticed that the FV scheme in CODA does not predict the leading mode as unstable 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 converge with both *h*- and *p*-refinements. The closeup view in Fig. 9(b), which shows the most converged eigenvalues for each case, suggests that the remaining error, with respect to the *true* eigenvalue of the continuous equations, for the accomplished refinement levels is less than 0.03% (taking eighth-order DG on grid L0 and fifth-order DG on grid L2 as the worst and best solution, respectively). Since we do not know the *true* eigenvalue of the continuous equations, we resort to measuring the convergence of the eigenvalues using relative changes in their

***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
L6	1.5255e-02 + *i*6.3927e-01	4.1e-04	1.5205e-02 + *i*6.3912e-01	3.6e-04

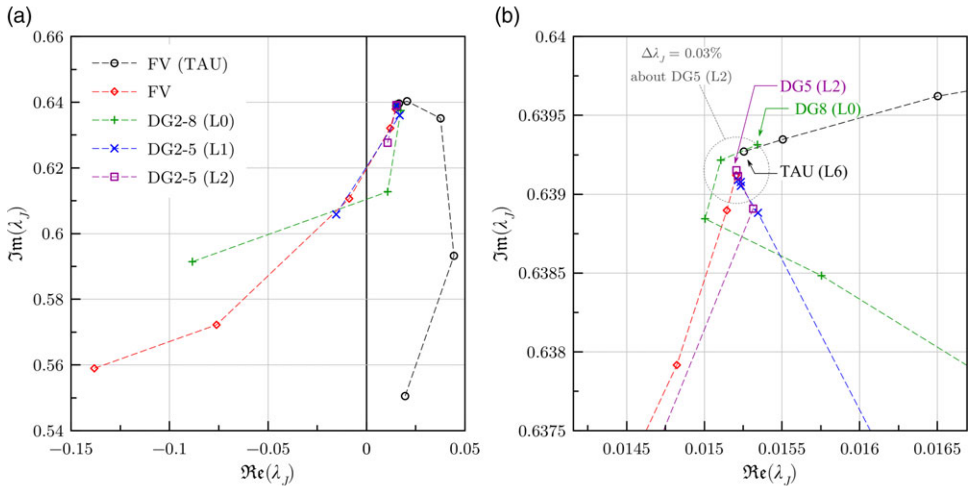


***Figure 9.** Non-dimensional eigenvalues of the unstable mode for laminar flow past square cylinder case.*

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) for a fixed grid level. 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 3–5 for DG computations.

As expected, the value of the relative changes in the eigenvalues decreases with refinement in general. Initially, the eigenvalues computed using CODA's FV scheme undergo larger changes with 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 computations on the finest grids L5 and L6 show similar convergence trends (cf. Table 2). This can also be confirmed in Fig. 9(b) in which the eigenvalues computed on grid L5 using the FV schemes in CODA and TAU are somewhat equidistant from the *true* value of the continuous equations, 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

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

	λ_J	$\Delta\lambda_J$
DG2	$-8.8413\text{e-}02 + i5.9145\text{e-}01$	—
DG3	$1.0756\text{e-}02 + i6.1276\text{e-}01$	1.7e-01
DG4	$1.7415\text{e-}02 + i6.3747\text{e-}01$	4.1e-02
DG5	$1.5754\text{e-}02 + i6.3848\text{e-}01$	3.0e-03
DG6	$1.5004\text{e-}02 + i6.3884\text{e-}01$	1.3e-03
DG7	$1.5106\text{e-}02 + i6.3922\text{e-}01$	6.0e-04
DG8	$1.5342\text{e-}02 + i6.3931\text{e-}01$	4.0e-04

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

	λ_J	$\Delta\lambda_J$
DG2	$-1.5398\text{e-}02 + i6.0585\text{e-}01$	—
DG3	$1.6835\text{e-}02 + i6.3604\text{e-}01$	7.1e-02
DG4	$1.5345\text{e-}02 + i6.3888\text{e-}01$	5.0e-03
DG5	$1.5235\text{e-}02 + i6.3905\text{e-}01$	3.2e-04
DG6	$1.5235\text{e-}02 + i6.3908\text{e-}01$	4.1e-05
DG7	$1.5217\text{e-}02 + i6.3909\text{e-}01$	3.4e-05

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

	λ_J	$\Delta\lambda_J$
DG2	$1.0705\text{e-}02 + i6.2766\text{e-}01$	—
DG3	$1.5314\text{e-}02 + i6.3891\text{e-}01$	1.9e-02
DG4	$1.5217\text{e-}02 + i6.3912\text{e-}01$	3.6e-04
DG5	$1.5207\text{e-}02 + i6.3915\text{e-}01$	5.9e-05

L0 surpasses this with only 0.13% change in the eigenvalue, while an eighth-order DG scheme on grid L0 achieves similar accuracy as the FV scheme in TAU on grid L6. The convergence of the eigenvalues computed using DG schemes improves on grid L1 with 0.0034% change for a seventh-order scheme and grid L2 with 0.0059% change for a fifth-order scheme. Finally, the mode computed on grid L3 using third-order DG is almost identical (with a 0.002% difference) to that of the fifth-order DG scheme on grid L2.

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.

Competing interests. The author(s) declare none.

References

[1] Crouch J.D., Garbaruk A. and Magidov D. Predicting the onset of flow unsteadiness based on global instability. *J. Comput. Phys.*, 2007, **224**, (2), pp 924–940.

[2] Timme S. Global instability of wing shock-buffet onset. *J. Fluid Mech.*, 2020, **885**, p A37.

[3] Houtman J. and Timme S. Global stability analysis of elastic aircraft in edge-of-the-envelope flow. *J. Fluid Mech.*, 2023, **967**, p A4.

[4] Lehoucq R.B., Sorensen D.C. and Yang C. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, 1998. https://epubs.siam.org/doi/book/10.1137/1.9780898719628

[5] Wagner M., Jägersküpper J., Molka D. and Gerhold T. *Performance Analysis of Complex Engineering Frameworks*. Cham: Springer, 2021, pp 123–138.

[6] Leicht T., Jägersküpper J., Vollmer D., Schwöppe A., Hartmann R., Fiedler J. and Schlauch T. DLR-Project Digital-X – Next Generation CFD Solver 'Flucs'. In *Deutscher Luft- und Raumfahrtkongress 2016*. Deutsche Gesellschaft für Luft- und Raumfahrt (DGLR), 2016.

[7] Krzikalla O., Rempke A., Bleh A., Wagner M. and Gerhold T. Spliss: a sparse linear system solver for transparent integration of emerging HPC technologies into CFD solvers and applications. In *New Results in Numerical and Experimental Fluid Mechanics XIII: Contributions to the 22nd STAB/DGLR Symposium*. Cham: Springer, 2021, pp 635–645. https://link.springer.com/chapter/10.1007/978-3-030-79561-0_60#author-information

[8] Stewart G.W. A Krylov–Schur algorithm for large eigenproblems. *SIAM J. Matrix Anal. Appl.*, 2002a, **23**, (3), pp 601–614.

[9] Vevek U.S., Timme S., Pattinson J., Stickan B. and Büchner A. Next-generation computation fluids dynamics capabilities for aircraft aeroelasticity and loads. In *19th International Forum on Aeroelasticity and Structural Dynamics, IFASD 2022*, 2022.

[10] Stewart G.W. Addendum to "A Krylov–Schur algorithm for large eigenproblems". *SIAM J. Matrix Anal. Appl.*, 2002b, **24**, (2), pp 599–601.

[11] Sorensen D.C. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 1992, **13**, (1), pp 357–385.

[12] Golub G.H. and Van Loan C.F. *Matrix Computations*, 4th edition. Baltimore, MD: Johns Hopkins University Press, 2013.

[13] Francis J.G.F. The QR transformation a unitary analogue to the LR transformation—part 1. *Compu. J.*, 1961, **4**, (3), pp 265–271.

[14] Francis J.G.F. The QR transformation—part 2. *Comput. J.*, 1962, **4** (4), pp 332–345.

[15] Saad Y. and Schultz M.H. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 1986, **7,** (3), pp 856–869.

[16] McCracken A., Da Ronch A., Timme S. and Badcock K.J. Solution of linear systems in fourier-based methods for aircraft applications. *Int. J. Comput. Fluid Dyn.*, 2013, **27**, (2), pp 79–87.

[17] Schwöppe A. and Diskin B. Accuracy of the cell-centered grid metric in the DLR TAU-code. In *New Results in Numerical and Experimental Fluid Mechanics VIII*. Berlin, Heidelberg: Springer, 2013, pp 429–437. https://link.springer.com/chapter/10.1007/978-3-642-35680-3_51#author-information

[18] Bassi F., Crivellini A., Rebay S. and Savini M. Discontinuous Galerkin solution of the Reynolds-averaged Navier–Stokes and k–ωturbulence model equations. *Comput. Fluids*, 2005, **34,** (4–5), pp 507–540.

[19] Schwamborn D., Gerhold T. and Heinrich R. The DLR TAU-code: Recent applications in research and industry. In *ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics*, 2006.

[20] Xu S., Timme S. and Badcock K.J. Enabling off-design linearised aerodynamics analysis using krylov subspace recycling technique. *Comput. Fluids*, 2016, **140**, pp 385–396.

[21] Timme S., Badcock K., Wu M. and Spence A. Lyapunov inverse iteration for stability analysis using computational fluid dynamics. In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, American Institute of Aeronautics and Astronautics (AIAA), 2012.

[22] Fabre D., Citro V., Ferreira Sabino D., Bonnefis P., Sierra J., Giannetti F. and Pigou M. A practical review on linear and nonlinear global approaches to flow instabilities. *Appl. Mech. Rev.*, 2019, **70,** (6), p 060802.

[23] Sipp D. and Lebedev A. Global stability of base and mean flows: a general approach and its applications to cylinder and open cavity flows. *J. Fluid Mech.*, 2007, **593**, pp 333–358.

[24] Hartmann R. and Leicht T. Generation of unstructured curvilinear grids and high-order discontinuous Galerkin discretization applied to a 3D high-lift configuration. *Int. J. Numer. Methods Fluids*, 2016, **82,** (6), 316–333.

Appendix A

Algorithm 1. Krylov–Schur algorithm.

```
1:  Inputs:
        complex shift $\sigma$, number of desired eigenpairs $k$,
        size to truncate Krylov subspace to for restart $t \geq k$,
        maximum size of Krylov subspace $m$
2:  Initialise:
        random vector $\boldsymbol{q}$ normalised such that $||\boldsymbol{q}|| = 1$
        $Q = [\boldsymbol{q}], S = [], j = 1, l = 0$
3:  while $l < k$ do
4:      Solve $(J - \sigma I)\boldsymbol{q} = Q(:, j)$
5:      for $i = 1 : j$ do
6:          $S(i, j) = Q(:, i)^H \boldsymbol{q}$
7:          $\boldsymbol{q} = \boldsymbol{q} - S(i, j)\, Q(:, i)$
8:      end for
9:      $S(j+1, j) = ||\boldsymbol{q}||$
10:     $\boldsymbol{q} = \boldsymbol{q}/S(j+1, j)$
11:     $Q(:, j+1) = \boldsymbol{q}$

12:     if $j > 1$ then
13:         Compute complex Schur decomposition with re-ordering $S(l+1 : j, l+1 : j) = VRV^H$
            (see Appendix B for re-ordering)
14:         $Q(:, l+1 : j) = Q(:, l+1 : j)\, V$
15:         $S(l+1 : j, l+1 : j) = R$
16:         $\alpha = S(j+1, j)$
17:         $S(j+1, l+1 : j) = \alpha V(j - l, :)$
18:     end if

19:     Compute eigenvalue decomposition $S(l+1 : k, l+1 : k)\, P = P\, \Lambda$
20:     for $i = l+1 : k$ do
21:         if $|S(j+1, l+1 : k)\, P(:, i)| < \epsilon_{\text{tol}}$ then
22:             $l = l+1$
23:         else
24:             break
25:         end if
26:     end for

27:     $j = j+1$
28:     if $j > m$ then
29:         $S(t+1, 1 : t) = S(m+1, 1 : t)$
30:         $S(t+2 :, 1 : t) = 0$
31:         $S(:, t+1 :) = 0$
32:         $Q(:, t+1) = Q(:, m+1)$
33:         $j = t+1$
34:     end if
35: end while

36: Compute eigenvalue decomposition $S(1 : k, 1 : k)\, P = P\, \Lambda$
37: Return eigenvalues diag($\Lambda$) and eigenvectors $W = Q(:, 1 : k)\, P$
```

Appendix B

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 Equation (18) reduces to

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

Since $a \neq c$, the denominator on the right-hand side does not vanish. Similarly, since $q^H = 0$ is not considered, the denominator on the left-hand side also does not vanish. Multiplying Equation (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 Equation (20). The value of q can be obtained from Equation (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 Equation (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, specifically

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

Cite this article: U S Vevek, Houtman J. and Timme S. (2024). Bespoke stability analysis tool in next-generation computational fluid dynamics solver. *The Aeronautical Journal*, **128**, 1164–1182. https://doi.org/10.1017/aer.2023.108