IFASD 2015
International Forum on Aeroelasticity and Structural Dynamics
June 28 — July 02, 2015 ▲ Saint Petersburg, Russia

KRYLOV SUBSPACE RECYCLING FOR LINEARISED AERODYNAMICS ANALYSIS USING DLR–TAU

S. Xu[1], S. Timme[1], and K. J. Badcock[1]

[1]School of Engineering, University of Liverpool
Liverpool L69 3GH, United Kingdom
shenren.xu@liverpool.ac.uk

Abstract: The major computational challenge, when using frequency domain linearised computational fluid dynamics in the analysis of aeroelastic problems such as aircraft flutter, gust response or shock buffet, are the excessive memory and CPU time requirements to solve the large sparse linear systems of equations. To address these issues found with the generalised minimal residual linear equation solver currently used in the DLR–TAU code, the generalised conjugate residual solver with deflated restarting is adopted. Here an invariant Krylov subspace is recycled both between restarts when solving a single linear frequency domain problem and for a sequence of equations when varying the system matrix and forcing terms. The proposed method is applied to three test cases including the forced excitation of a pitch-plunge aerofoil, the fast prediction of shock buffet onset for an aerofoil using global stability analysis and the computation of a reduced order model basis for a realistic passenger aircraft. The memory requirements for the problems investigated are reduced by up to an order of magnitude, while the CPU times are reduced by up to a factor of three.

Keywords: Krylov Subspace Recycling, GMRES, GCRO-DR, Linearised Computational Fluid Dynamics, DLR–TAU Code, Flutter, Gust Response, Shock Buffet

1 INTRODUCTION

This paper presents the details of an investigation to improve the generalised minimal residual (GMRES) linear equation solver currently implemented in the DLR–TAU code both in terms of memory requirements and convergence rates when using the generalised conjugate residual solver with deflated restarting (GCRO-DR). Linearised computational fluid dynamics (CFD) is chosen to model the transonic aerodynamics in the aeroelastic test cases presented. The range of potential applications include flutter and gust response analysis as well as the prediction of transonic shock buffet onset using eigenvalue calculation methods.

For modern aircraft design, the accurate and efficient calculation of both the flutter onset and gust response at an early design stage is of critical importance as it, to large extent, determines the flight envelope and structural sizing. Despite significant advances in computational algorithms for CFD and structural dynamics, using time-domain simulation for such routine analyses is still not a viable option. Linear frequency domain functionality, implemented in the DLR–TAU code [1], can be used to efficiently compute the aerodynamic response due to structural excitation, the information of which is then needed in a flutter analysis [2] similar to industrial approaches using the classical doublet lattice method (DLM). In addition, linear and nonlinear reduced order models can be constructed for gust response analysis when using eigenmodes of the aeroelastic system as a projection basis [3, 4].

More specifically, to predict the flutter onset using CFD aerodynamics in a DLM-like approach, the aerodynamic response due to excitation in the structural modes at various frequencies is first pre-computed, and then various interpolation techniques can be used to reconstruct a response surface to quickly provide aerodynamic data when solving the small flutter eigenvalue problem. Typically, for a complete passenger aircraft, the part of the analysis, requiring access to the CFD solver functionality with millions of degrees of freedom, involves up to 100 structural modes and seven to ten excitation frequencies resulting in several hundred solves per configuration.

An even more computationally challenging situation arises when the fluid part exhibits an instability, which further complicates the fluid-structure interaction. An example of such flow instability with aeronautical relevance is transonic shock buffet. For small-scale problems, the onset of the flow instability can be found by solving for a few eigenvalues of the system, using a direct sparse equation solver, at each flow condition and tracking the eigenvalue which first crosses the imaginary axis [5–7]. For realistic wing cases, a direct method is not a viable option and an iterative sparse equation solver combined with a shifted inverse eigenvalue method could be used instead to calculate the relevant eigenvalue. Different from a flutter analysis, where the frequency range of interest is dictated by the frequencies of the structural dynamics in vacuum, self-induced flow instability is independent of the structural motion. A good initial guess for the shift is thus not easy to find. In [8, 9] it was discussed however that the aerodynamic response at pre-buffet conditions exhibits resonance when excited near the frequency of the flow instability. This information can then serve as a good initial guess. Another computational challenge associated with tracking the destabilising eigenvalue is that the shifted fluid Jacobian matrix is nearly singular. For instance, in [10] the preconditioned restarted GMRES solver was used in DLR–TAU to compute the frequency response of an aerofoil at pre-buffet conditions and the linear solver failed to converge in many cases, presumably due to the worsened stiffness at those conditions.

The theoretical formulation of the linearised CFD aerodynamics is introduced in Section 2, while the numerical methods used for solving the resulting large sparse linear systems of equations are discussed in Section 3. Results demonstrating the memory and runtime efficiency of the GCRO-DR solver compared to the baseline GMRES solver are shown in Section 4 for three test cases.

2 LINEARISED FREQUENCY DOMAIN AERODYNAMICS

The transient nonlinear equation describing the unsteady aerodynamics is written in semi-discrete form as

$$\dot{\mathbf{w}}_f = \mathbf{R}_f(\mathbf{w}_f, \mathbf{x}, \dot{\mathbf{x}}, \boldsymbol{\theta}) \tag{1}$$

where $\mathbf{w}_f$ and $\mathbf{R}_f$ denote the fluid unknowns and corresponding residual vector, respectively, and $\boldsymbol{\theta}$ are the system parameters. The vectors $\mathbf{x}$ and $\dot{\mathbf{x}}$ are location and velocity of the fluid mesh points which are functions of the structural mode shapes and the modal amplitudes denoted $\boldsymbol{\eta}$. Linearising about an equilibrium point and assuming small amplitude harmonic motion, the latter equation can be re-formulated as

$$\left(\frac{\partial \mathbf{R}_f}{\partial \mathbf{w}_f} - i\omega^{(k)} I\right)\boldsymbol{\phi}_f^{(j,k)} = -\left(\frac{\partial \mathbf{R}_f}{\partial \boldsymbol{\eta}} + i\omega^{(k)}\frac{\partial \mathbf{R}_f}{\partial \dot{\boldsymbol{\eta}}}\right)\boldsymbol{\phi}_\eta^{(j)} \tag{2}$$

where $\boldsymbol{\phi}_f$ and $\boldsymbol{\phi}_\eta$ are complex-valued amplitudes of fluid and structure, respectively. The equation gives the aerodynamic response $\boldsymbol{\phi}_f$ following a disturbance of the structure $\boldsymbol{\phi}_\eta$. To find aerodynamic data for further analysis, this equation usually has to be pre-computed for each structural mode shape (denoted by superscript j) and for a range of forced sinusoidal excitations in the modal amplitudes at different frequencies ω (denoted by superscript k). As mentioned earlier, several hundreds of this equation need to be solved for industrial problems with the number of fluid unknowns easily exceeding several tens of millions.

Adding equations to describe the unsteady motion of the structure in terms of the modal amplitudes $\boldsymbol{\eta}$ gives a coupled problem to be solved for investigation in flutter stability and also gust response behaviour. The basis of a reduced order model for such aeroelastic analyses can be calculated from the eigenvectors of the coupled system. The fluid part of the direct (i.e. right) eigenvalue problem is equivalent to Eq. (2), except that the structural part of the eigenvector $\boldsymbol{\phi}_\eta$ is now part of the solution rather than a pre-defined user input and that the eigenvalue (i.e. frequency) corresponds to a particular eigenvector (with superscript $k = j$). The corresponding adjoint (i.e. left) eigenvalue problem for the fluid part of the eigenvector $\boldsymbol{\psi}_f$ can be derived as

$$\left(\left(\frac{\partial \mathbf{R}_f}{\partial \mathbf{w}_f}\right)^T + i\omega^{(j)} I\right)\boldsymbol{\psi}_f^{(j)} = -\left(\frac{\partial \mathbf{R}_{\dot{\eta}}}{\partial \mathbf{w}_f}\right)^T \boldsymbol{\psi}_{\dot{\eta}}^{(j)} \tag{3}$$

where $\mathbf{R}_{\dot{\eta}}$ is the residual vector corresponding to the structural unknowns. Details of the mathematical formulation of linearised frequency domain aerodynamics and model reduction in the context of aeroelastic analysis can be found in [4].

Another type of problem arises when the fluid exhibits an instability without structural motion. One typical example is the shock-buffet problem where the shock wave interacts with the boundary layer and destabilises the steady flow beyond a critical parameter. To find the buffet onset, shifted inverse methods are an obvious choice to calculate few eigenvalues close to an initial guess. To choose such initial shift (i.e. a characteristic frequency of the instability), either engineering judgement is required or the resonant behaviour of the flow when excited at frequencies in the vicinity of the instability can be exploited [8, 9], which would be equivalent to solving Eq. (2). The closer the shift to the target eigenvalue, the faster the algorithm converges. This however leads to the second, even bigger challenge using shifted inverse methods. The linear system to be solved is nearly singular. Using a direct sparse linear equation solver quickly becomes infeasible for everything beyond two dimensional problems. Thus a preconditioned sparse iterative linear equation solver is a possible alternative.

We use the shifted inverse method from [11] referred to as inverse correction. The equation to be solved is

$$\left(\frac{\partial \mathbf{R}_f}{\partial \mathbf{w}_f} - \sigma I\right)\delta\boldsymbol{\phi}_f = -\left(\frac{\partial \mathbf{R}_f}{\partial \mathbf{w}_f} - \lambda I\right)\boldsymbol{\phi}_f \tag{4}$$

where σ is the constant complex-valued shift close to the target eigenvalue λ and the eigenvector is updated as

$$\boldsymbol{\phi}_f \leftarrow \boldsymbol{\phi}_f + \delta\boldsymbol{\phi}_f \tag{5}$$

until the norm of the right-hand side converges below a given tolerance. The right-hand side in Eq. (4) represents the residual vector of the eigenvalue problem based on the current approximation to the eigenvector $\boldsymbol{\phi}_f$ of unit length and eigenvalue based on the Rayleigh quotient

$$\lambda = \boldsymbol{\phi}_f^H \frac{\partial \mathbf{R}_f}{\partial \mathbf{w}_f} \boldsymbol{\phi}_f \tag{6}$$

The eigenvector update $\delta\boldsymbol{\phi}_f$ is always initialised to zero, which is convenient since for converging outer iterations the update will go to zero. The inner convergence is defined relative to the convergence of the outer iteration giving a nearly constant number of inner iterations. In this work, four orders of magnitude is chosen as stopping criterion of the inner linear system.

As can be seen from this brief introduction of linearised aerodynamics, the efficient solution of large sparse linear systems of equations is at the heart of it. For convenience in the following discussion, the coefficient matrix (i.e. fluid Jacobian matrix plus a complex-valued shift) is denoted A, while the various right-hand side terms are called $\mathbf{b}$.

3 SOLVING LARGE SPARSE LINEAR SYSTEMS OF EQUATIONS

The main challenge in using linearised CFD aerodynamics is solving large sparse linear systems of equations. Restarted generalised minimal residual (GMRES) solver [12] was implemented in the DLR–TAU code to solve the linearised equations. As mentioned in the introduction, for stiff problems, restarted GMRES often suffers from stagnation unless a large number of Krylov vectors is kept. It is not uncommon to keep several hundred Krylov vectors in order to converge. This large memory requirement could then become the bottleneck when solving large cases. To ease these difficulties, we implemented generalised conjugate residual solver with deflated restarting (GCRO-DR), which converges almost like full GMRES but has small memory requirement. In addition, recycling a certain Krylov subspace between different equations is possible for GCRO-DR and is thus favourable for solving a sequence of linear equations with similar coefficient matrices.

In this section, both GMRES and GCRO-DR will be explained. Furthermore, GCRO-DR combined with inter-equation recycling, dubbed GCRO-DR-R, is introduced.

3.1 GMRES

The baseline linear solver used is GMRES. The theory and implementation detail are well documented in [12] and only a brief introduction is given here. When solving the linear system

$$A\mathbf{x} = \mathbf{b}$$

one first forms the Krylov subspace

$$\mathcal{K}_m(A, \mathbf{b}) = \mathrm{span}(\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, \ldots, A^{m-1}\mathbf{b})$$

using the Arnoldi iteration. After m Arnoldi steps, a unit vector basis $V_m = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m]$ that spans the Krylov subspace is constructed satisfying the Arnoldi relation

$$AV_m = V_{m+1}\bar{H}_m$$

where $\bar{H}_m$ is an upper Hessenberg matrix. The solution $\mathbf{x}$ is approximated as

$$\mathbf{x} = \mathbf{x}_0 + V_m\mathbf{d}_m$$

where the coefficient vector $\mathbf{d}_m$ is solved through the least square problem minimising the resulting residual $\mathbf{r} = \mathbf{b} - A\mathbf{x}$. It can be shown that

$$\begin{aligned}\|\mathbf{r}\| &= \|\mathbf{r}_0 - AV_m\mathbf{d}_m\| = \|\beta\mathbf{v}_1 - V_{m+1}\bar{H}_m\mathbf{d}_m\| \\ &= \|V_{m+1}\|\|\beta\mathbf{e}_1 - \bar{H}_m\mathbf{d}_m\| = \|\beta\mathbf{e}_1 - \bar{H}_m\mathbf{d}_m\|\end{aligned}$$

where $\mathbf{e}_1$ is the first standard basis vector of $\mathbb{R}^{m+1}$ and $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ is the initial residual. The least square problem is then reduced to a very low dimension m. The restarted version simply forms the Krylov subspace and solves the least square problem again from the updated initial solution and residual vectors.

In the DLR–TAU solver, GMRES is preconditioned using incomplete lower-upper (ILU) factorisation due to its robustness compared to other options such as multigrid. Once the matrices L and U are computed, the only modification to GMRES without preconditioning is whenever a vector is multiplied by the coefficient matrix, the resulting vector is further left-multiplied first by L^{-1} and then by U^{-1}. The triangular matrices L and U are inverted using forward and backward substitutions. Alternatively, right or split preconditioning could also easily be achieved by plugging in these two additional matrix inversions in a slightly different manner [12]. There seems to be no obvious advantage for any type of preconditioning over the others, therefore we only used left preconditioning throughout this work as the least modification is required. In addition, due to memory considerations, a complex-valued version of ILU with low fill-in is used.

3.2 GCR

Generalised conjugate residual (GCR) solver is a Krylov subspace solver that is algorithmically identical to GMRES, but with different procedures. Again, for the theory and the detailed implementation, refer to [12]. Only a brief introduction is given here.

Standard GCR constructs two vector bases

$$U_m = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m] \text{ and } C_m = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_m]$$

satisfying

$$C_m = AU_m \text{ and } C_m^H C_m = I. \tag{7}$$

The solution is approximated on the subspace spanned by the column vectors of U_m

$$\mathbf{x} = \mathbf{x}_0 + U_m\mathbf{d}_m$$

subject to the constraint that the resulting residual is perpendicular to the subspace spanned by the column vectors of C_m

$$C_m^H\mathbf{r} = C_m^H(\mathbf{r}_0 - AU_m\mathbf{d}_m) = \mathbf{0}$$

which is equivalent to $\mathbf{d}_m$ being the minimiser of $\|\mathbf{r}\|$. Similar to the Arnoldi iteration generating the Krylov vectors in GMRES, GCR uses a recursive procedure to generate the column vectors of C_m and U_m. At the i-th iteration, $i \geq 1$, set

$$\mathbf{u}_i \leftarrow \mathbf{r}_{i-1} \quad \text{and} \quad \mathbf{c}_i \leftarrow A\mathbf{u}_i \tag{8}$$

which are first orthogonalised against C_{i-1} if $i \neq 1$,

$$\mathbf{u}_i \leftarrow \mathbf{u}_i - (C_{i-1}C_{i-1}^H)\mathbf{u}_i \quad \text{and} \quad \mathbf{c}_i \leftarrow \mathbf{c}_i - (C_{i-1}C_{i-1}^H)\mathbf{c}_i$$

and then normalised by $\|\mathbf{c}_i\|$. The solution and residual vectors are then updated as

$$\mathbf{x}_i \leftarrow \mathbf{x}_{i-1} + \mathbf{u}_i(\mathbf{c}_i^H\mathbf{r}_{i-1}) \quad \text{and} \quad \mathbf{r}_i \leftarrow \mathbf{r}_{i-1} - \mathbf{c}_i(\mathbf{c}_i^H\mathbf{r}_{i-1}). \tag{9}$$

A preconditioned version of GCR is easily obtained by adding the matrix-vector multiplication step for the preconditioning matrices as in GMRES.

3.3 GCRO

Nested Krylov subspace solvers wrap one Krylov solver outside another and use the inner solver to precondition the outer one. One of those nested solvers is GCRO, which uses GCR for the outer loop and any Krylov subspace solver, such as GMRES, for the inner loop. The motivation behind can best be explained by revisiting Eq. (8). Instead of assigning the latest residual vector to $\mathbf{u}_i$, we could set

$$\mathbf{u}_i \leftarrow A^{-1}\mathbf{r}_{i-1}. \tag{10}$$

Then the solution is found immediately following the update step in Eq. (9) where the residual vector is obviously zero. Although for Eq. (10) it is generally not possible to invert the matrix A, it does imply that we could assign to $\mathbf{u}_i$ an approximate solution to the equation $A\mathbf{x} = \mathbf{r}_{i-1}$ to accelerate the convergence. To get the approximate solution, GMRES(k) is used in place of Eq. (8) solving

$$(I - C_{i-1}C_{i-1}^H)A\mathbf{u}_i = \mathbf{r}_{i-1}$$

for a maximum k iterations with initial solution of zero. The term in the bracket preceding A ensures that the Krylov subspace formed in the inner loop is normal to the Krylov subspace in the outer GCR loop to guarantee monotonic residual reduction.

3.4 GCRO-DR and GCRO-DR-R

For nested Krylov subspace solvers such as GCRO, the inner loop generates a Krylov subspace to approximately solve the equation, and the subspace is discarded when exiting the inner loop. It would be advantageous to recycle some information from the discarded subspace to aid the outer convergence. One approach is to truncate based on the principle angle between the Krylov vectors constructed during the inner loop and select the important ones to augment the outer Krylov subspace. The criterion to select them is to check how much worse the inner loop convergence would have been if the inner loop has stopped before those vectors are formed. One solver based on this idea is generalised conjugate residual with optimal truncation [13]. Another approach is to select the interior eigenvectors (eigenvectors corresponding to the smallest-in-magnitude eigenvalues) that can be computed during the Arnoldi iteration from the inner loop and use them to augment the outer Krylov subspace. The two most important solvers following this idea are generalised minimal residual with deflated restarting [14] and generalised conjugate residual with deflated restarting (GCRO-DR) [15]. Both are nested Krylov subspace solvers with the main difference being that the former uses GMRES for the outer loop while the latter uses GCR. Although the latter requires more memory, it has the advantage of recycling eigenvectors both between restarts and between equations with different coefficient matrices. Due to this flexibility, GCRO-DR is used in this work.

The algorithm of GCRO-DR is now explained. Standard GCRO-DR starts with m Arnoldi iterations which produces the upper Hessenberg matrix $\bar{H}_m$ and the Krylov vectors V_m. The solution and residual vectors are first updated as in standard GMRES. To extract the approximate interior eigenvectors of A, we solve the generalised eigenvalue problem

$$(H_m + h_{m+1,m}^2 H_m^{-H}\mathbf{e}_m\mathbf{e}_m^H)\mathbf{p}_i = \theta_i \mathbf{p}_i$$

where the square matrix H_m is $\bar{H}_m$ without the last row and then set

$$[\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_k] =: Y_k \leftarrow V_m P_k$$

where $P_k = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_k]$ with $\mathbf{p}_i$ $(1 \le i \le k)$ corresponding to the k smallest θ_i. The matrices C_k and U_k are constructed from Y_k by setting

$$C_k \leftarrow V_{m+1}Q \;\text{ and }\; U_k \leftarrow Y_k R^{-1}.$$

where $[Q, R]$ is the QR-factorisation of $\bar{H}_m P_k$. It can be verified that the resulting C_k and U_k satisfy the condition in Eq. (7). Next, we set $\mathbf{v}_1 = \mathbf{r}/\|\mathbf{r}\|$ and perform $(m-k)$ Arnoldi iterations using the linear operator $(A - C_k C_k^H A)$ such that the Krylov vectors are orthogonal to C_k as in standard GCRO. The key step is to combine U_k from the outer GCR and V_{m-k} from the inner Arnoldi iterations to form a subspace to approximate the solution. Thus, define

$$\hat{V}_m = [U_k D_k,\ V_{m-k}],\ \hat{W}_{m+1} = [C_k,\ V_{m-k+1}],\ \bar{G}_m = \begin{bmatrix} D_k & B_{m-k} \\ 0 & \bar{H}_{m-k} \end{bmatrix}$$

which satisfy the generalised Arnoldi relation

$$A\hat{V}_m = \hat{W}_{m+1}\bar{G}_m$$

where $D_k = \mathrm{diag}(\|\mathbf{u}_1\|^{-1}, \|\mathbf{u}_2\|^{-1}, \dots, \|\mathbf{u}_k\|^{-1})$ and $B_k = C_k^H A V_{m-k}$.

The solution is approximated over the subspace spanned by the columns of $\hat{V}_m$, i.e., we solve for the coefficient vector $\mathbf{d}_m$ to minimise the residual $\|\mathbf{r} - A\hat{V}_m\mathbf{d}_m\|$ which, due to the Arnoldi relation, is equivalent to $\|\hat{W}_{m+1}^H\mathbf{r} - \bar{G}_m\mathbf{d}_m\|$. The minimiser can be found from solving a least square problem. Once $\mathbf{d}_m$ is found, the solution and residual vectors are updated. In addition, we compute θ_i and $\mathbf{p}_i$ of the generalised eigenvalue problem

$$\bar{G}_m^H \bar{G}_m \mathbf{p}_i = \theta_i \bar{G}_m^H \hat{W}_{m+1}^H \hat{V}_m \mathbf{p}_i$$

The approximate interior eigenvectors of the coefficient matrix are $Y_k = \hat{V}_m P_k$ with P_k containing the k interior eigenvectors to the reduced system as its columns. To form C_k and U_k, first perform QR-factorisation of $\bar{G}_m P_k$ and then set

$$C_k \leftarrow \hat{W}_{m+1}Q \;\text{ and }\; U_k \leftarrow Y_k R^{-1}.$$

The next cycle of Arnoldi iterations is then restarted with constraint C_k. Different from standard GCRO, after C_k and U_k are formed, the solution and residual vectors are not immediately updated in GCRO-DR. This is because the least square problem after the inner Arnoldi iteration has included the basis U_k in the search subspace. The resulting algorithm is denoted as GCRO-DR(m, k) where m is the dimension of the Krylov subspace retained for approximating the solution while k is the number of eigenvectors recycled. The total number of Krylov vectors that needs to be stored is then $(m + k)$.

In GCRO-DR, the eigenvectors computed from the inner loop, Y_k, are 'recycled' to improve the next restarted cycle. Due to the flexibility of GCRO, any set of vectors could be recycled and regularised to form U_k and C_k, not only for the next restarted cycle, but also for solving the next equation, when solving a sequence of equations. Instead of performing m GMRES iterations to get k approximate interior eigenvectors, those from a previous solve can be recycled using the QR-factorisation to generate U_k and C_k that satisfy the condition in Eq. (7). We call this variant version that allows inter-equation recycling GCRO-DR-R. The algorithm for GCRO-DR-R as shown in Appendix A is identical to GCRO-DR except for line 3 [15].

The effectiveness of recycling the eigenvectors between equations strongly depends on how good an approximation the eigenvectors from one equation are for another. For varying right-hand sides, there are mainly three scenarios: i) identical coefficient matrices, ii) diagonally shifted coefficient matrices and iii) similar but different coefficient matrices. For the first case, GCRO-DR-R should be very effective as the spectral information of the coefficient matrix does not change. For case 2, GCRO-DR-R is also expected to be quite effective. Although the eigenvectors are preserved despite the shifted spectrum, the smallest eigenvectors for $(A - \sigma_1 I)$ are not necessarily the smallest eigenvectors for $(A - \sigma_2 I)$. Thus the deflation may not be as effective as in the first case. The last case is the most general one and there is no theory guaranteeing that recycling should work. It is only assumed that if the difference between two coefficient matrices is small, the difference in the smallest eigenvectors should also be small and then recycling should have some effect.

The other factor determining the effectiveness of recycling is the right-hand side. However, a theory regarding the convergence due to the right-hand side does not seem to exist (for a survey of the existing theories regarding the convergence properties of various Krylov subspace methods, refer to [16]). Some attempts have recently been made to formulate an asymptotic convergence bound taking into account the right-hand sides [17], however the more useful transient convergence behaviour is still unclear.

4 RESULTS

The linear equation solvers outlined in the previous section are now applied to three test cases to demonstrate their effectiveness in reducing the memory requirements and in accelerating convergence. The governing equations of the flow are solved using the DLR–TAU code and all linear systems are obtained from this solver's discretisation scheme. For the two dimensional aerofoil test cases, the Reynolds-averaged Navier-Stokes equations are used together with the one equation turbulence model of Spalart–Allmaras. The inviscid fluxes of the mean flow equations are discretised using the second-order central scheme with scalar dissipation, while the first-order Roe scheme is used for the turbulence

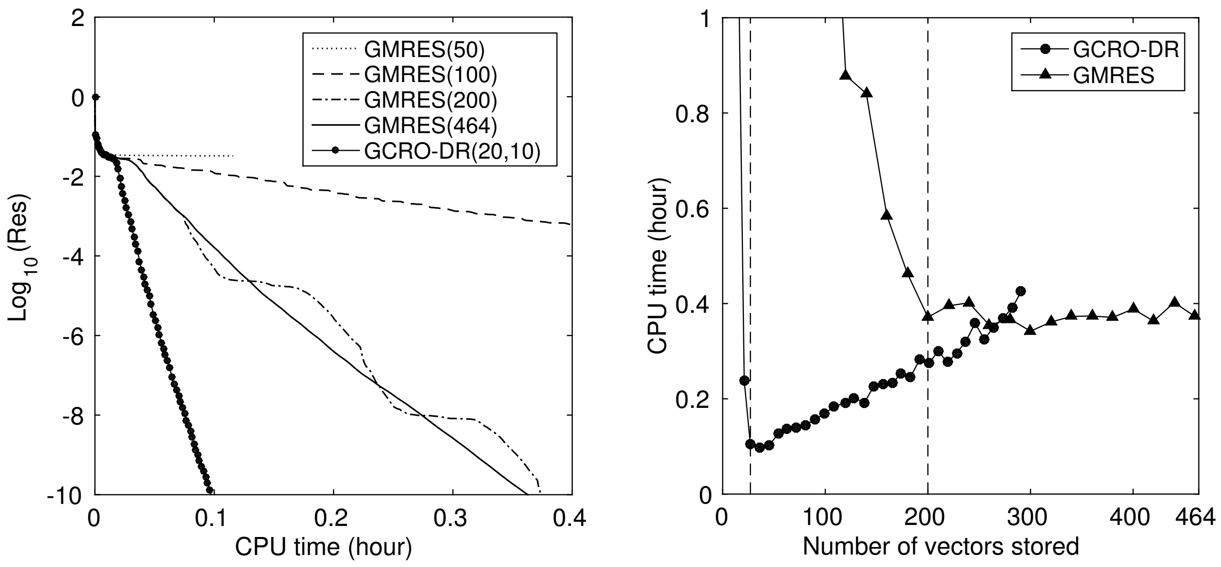


Figure 1: Case 1: Left: Convergence history using restarted GMRES and GCRO-DR; right: CPU time of GMRES and GCRO-DR using different number of vectors with tolerance 1e-10.

equation. Viscous fluxes follow the full gradient approach with the gradients reconstructed using the Green-Gauss theorem. The Euler equations are solved for the three dimensional test case using the same central scheme with scalar dissipation. The matrix, which the ILU factorisation is based, is formed by linearly blending the Jacobian matrices of first and second order spatial discretisation [18]. Further more, ILU with zero fill-in is used throughout this work, with one level of fill-in requiring appreciably more memory but resulting in limited speedup.

4.1 Calculating frequency response for an aerofoil

The first test case is a two dimensional aerofoil undergoing harmonic excitation in pitch and plunge modes at various frequencies in transonic flow. The freestream Mach number is 0.76 with a Reynolds number of 10 million. The angle of attack is 3.5 deg. The aerofoil case has 36k grid points, corresponding to 180k complex unknowns. The sparse Jacobian matrix has 16.7 million non-zero entries.

To investigate basic properties of the different linear solvers, we first solve the linear equation for a reduced frequency of 0.35 excited by the pitch mode. As will be explained below, the resulting linear system at this frequency is most difficult to solve. The convergence history for ten orders of magnitude residual drop using different solvers is plotted in Fig. 1. The residual here and in the following paragraphs is the normalised L_2 norm of the preconditioned residual, defined as $Res = \|U^{-1}L^{-1}(\mathbf{b} - A\mathbf{x})\|/\|U^{-1}L^{-1}\mathbf{b}\|$. As reference, full GMRES is first used and it converges with 464 iterations. Restarted GMRES is then run using 50, 100, 200 Krylov vectors respectively. At least 100 vectors are needed to avoid convergence stagnation and GMRES restarted with 200 vectors is found to be optimal. Using more than 200 vectors slows down the convergence due to the increased cost in orthogonalisation. On the other hand, GCRO-DR(20,10), requiring only 30 vectors to be stored, converges significantly faster, reducing the CPU time of the best performing GMRES solver by over a factor of three.

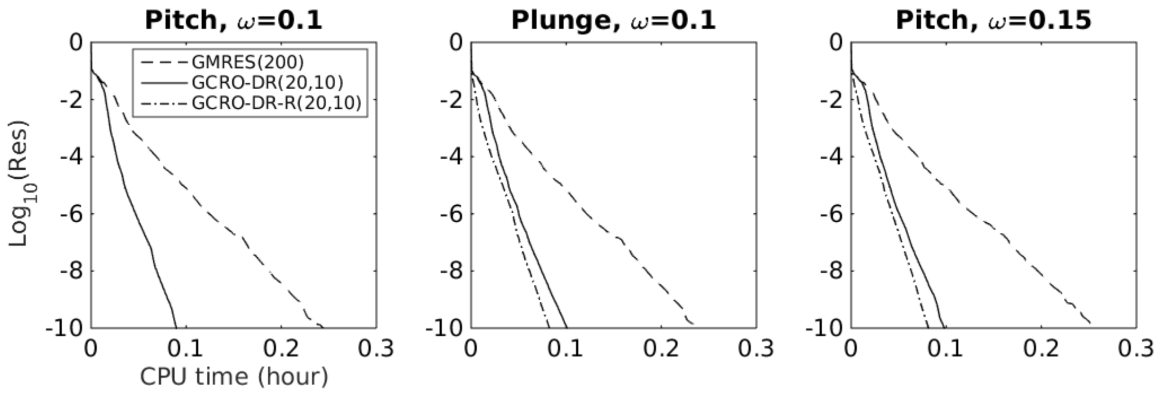


Figure 2: Case 1: Convergence history of GMRES(200), GCRO-DR(20,10) and GCRO-DR-R(20,10) for the first three equation solves.

To examine the memory requirements and CPU time for both solvers for the same linear system, different numbers of Krylov vectors are tested. All solvers are required to converge by ten orders of magnitude. Since there are two parameters m and k that can vary independently in GCRO-DR, we simplify the parameter study by setting $k = m/2$. Convergence stalls if less than 27 vectors are stored for GCRO-DR, above which the CPU time almost linearly increases with the fastest convergence achieved when storing 30 vectors. Therefore, GCRO-DR(20,10) is used for the remaining computation for this case. For GMRES, the restart number is decreased from $m = 464$ until the convergence severely slows down and eventually stalls below $m = 200$. The two dashed lines in Fig. 1 show the respective smallest number of vectors needed by the two solvers. Note that the reduced memory requirement is equivalent to storing six flow Jacobian matrices for this two dimensional case.

Next, inter-equation recycling is investigated in Figs. 2 and 3. The frequency response due to harmonic excitation in pitch and plunge modes is computed with three solvers: GMRES(200), GCRO-DR(20,10) and GCRO-DR-R(20,10). The reduced frequency varies from 0.1 to 0.6 with an increment of 0.05. For each frequency, the aerofoil is first excited using the pitch and then the plunge mode. The order of solving the sequence of equations is such that the coefficient matrices and their complex-valued ILU(0) preconditioner are formed as few times as possible and the coefficient matrices vary monotonically to allow effective recycling between equations. A total of 22 linear solves are performed. The convergence criterion is again a residual drop of ten orders of magnitude.

As can be seen in Fig. 2, compared with GCRO-DR, GCRO-DR-R completely avoids the initial phase of the slow convergence from the second linear equation solve due to recycling. The effect however slows down and the asymptotic convergence of the nested solvers is comparable, both outperforming GMRES. The CPU time breakdown of each linear solve for all three solvers for different modes excited at different frequencies is shown in Fig. 3. Compared to GMRES(200), GCRO-DR(20,10) reduces the overall CPU time by 64% and using recycling between equations speeds up another 15%.

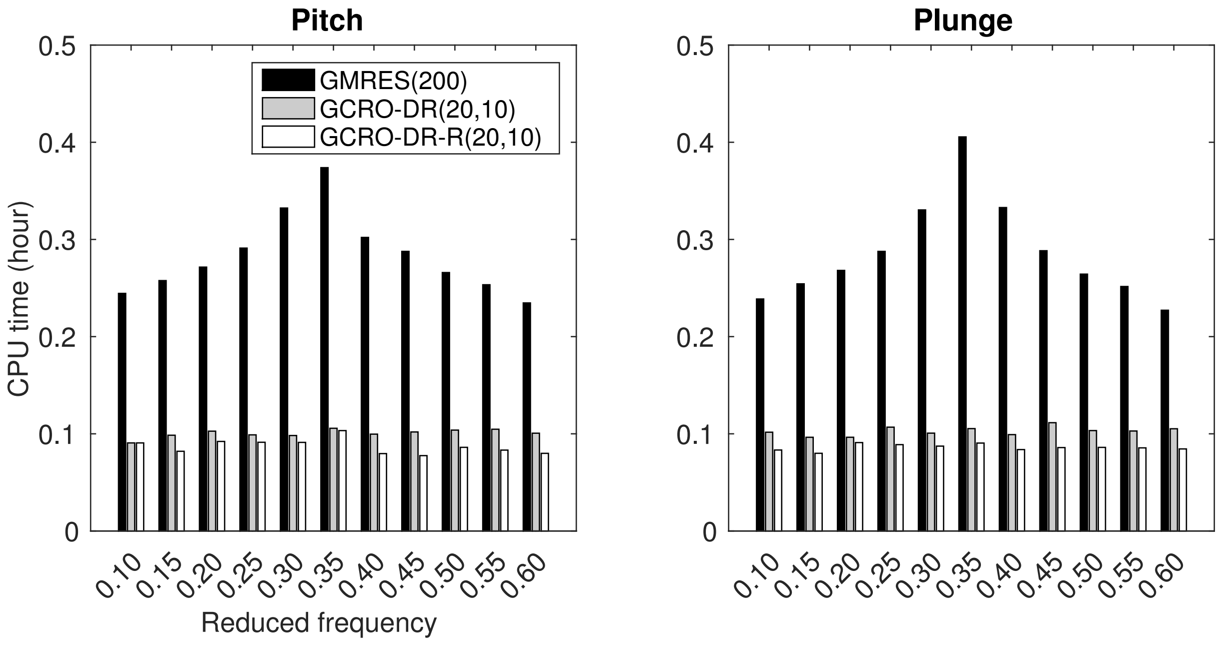


Figure 3: Case 1: CPU time breakdown for various solvers for computing the frequency response.

4.2 Calculating shock-buffet onset for an aerofoil

The CPU time of GMRES for the frequency sweep in the previous case shows a peak at the reduced frequency of 0.35 for both pitch and plunge modes, which is related to the near resonance fluid motion. This resonance behaviour is more evident from the plot on the left of Fig. 4 showing the magnitude of the unsteady lift derivative of the aerofoil excited by both modes at different reduced frequencies. This motivates the computation of the responsible fluid eigenvalue and eigenvector. To compute the eigen pair, inverse correction method [11] is used with shift $\sigma = 0.35i$ and a random initial guess for the eigenvector.

The first outer iteration is solved using GMRES with different restarts and GCRO-DR, for which the convergence history is shown on the right of Fig. 4. For GMRES, restarting after every 300 vectors seems to be optimal. For GCRO-DR with $m = 20$ using 10 recycled eigenvectors, the memory required is only one tenth of that of GMRES, while the convergence is accelerated by over 75%. Note that the convergence level used here is ten orders of magnitude. In practice, since only the outer iteration convergence is related to the actual convergence of the eigenvalue problem, the inner iteration convergence could be relaxed to achieve better overall performance. It is found that converging four orders of magnitude during inner iterations is sufficiently efficient, thus is chosen for the entire calculation.

For the eigenvalue problem, GMRES(300), GCRO-DR(20,10) and GCRO-DR-R(20,10) are used. Since the coefficient matrix does not change for constant shift, and only the right-hand side is updated every time the inner iteration has converged, eigenvector recycling is expected to be very effective. Shown in Fig. 5 is the convergence of all three solvers for the first three outer iterations, converging the residual by four orders of magnitude. Similar to case 1, the deflation technique significantly improves the convergence rate compared to GMRES. Further more, by recycling eigenvectors between equations, the convergence

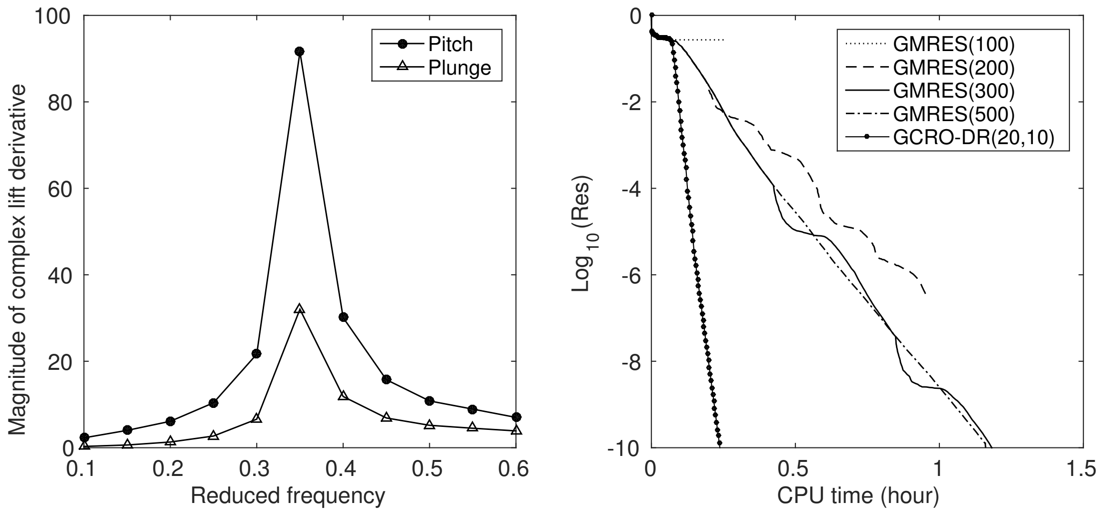


Figure 4: Case 2: Left: the complex lift derivative for 3.5 deg angle of attack excited by pitch and plunge modes over the range of reduced frequencies; right: convergence history of different linear solvers for the first outer iteration of the inverse correction solver.

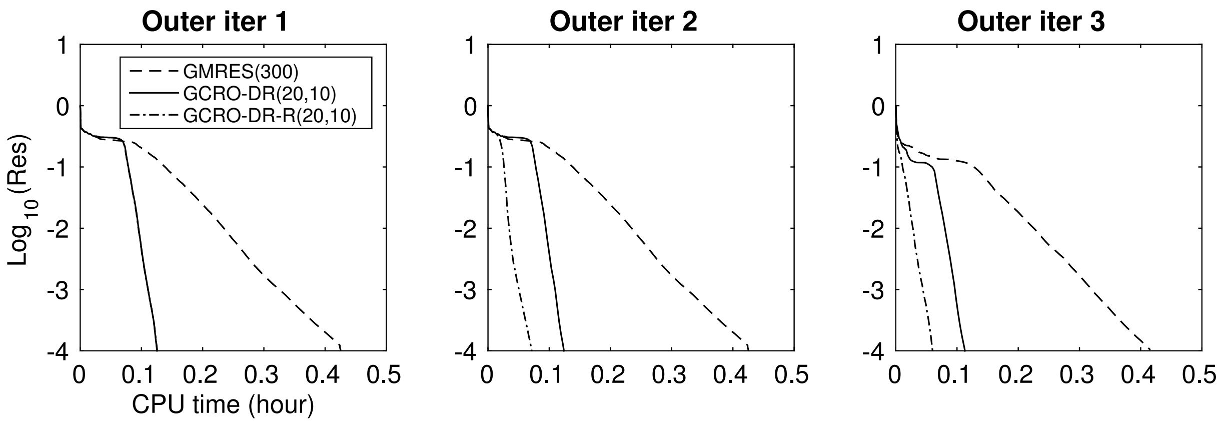


Figure 5: Case 2: Converge history for the first three outer iterations.

stagnation at the initial stage when using GCRO-DR without inter-equation recycling is completely removed from the second equation. Since the inner convergence is more relaxed compared to case 1, the relative speedup of GCRO-DR-R over GCRO-DR is even more pronounced. The CPU time of all three solvers for all nine outer iterations is plotted on the left in Fig. 6. The convergence of the outer eigenvalue problem is also shown in Fig. 6.

The evolution of the particular eigenvalue over the outer iterations is shown in Fig. 7 with the eigenvalues computed using Arnoldi method implemented in MATLAB. The numerical values of the eigenvalue is also compared with the one from MATLAB on the right of Fig. 7.

4.3 Calculating basis of reduced order model for an aircraft model

The XRF1 model is a wide-body passenger aircraft research configuration with a semi-span of about 30 m and an overall length of about 65 m. Flow conditions are a freestream Mach number of 0.85 at 1 deg angle of attack. The computational mesh for the Euler CFD

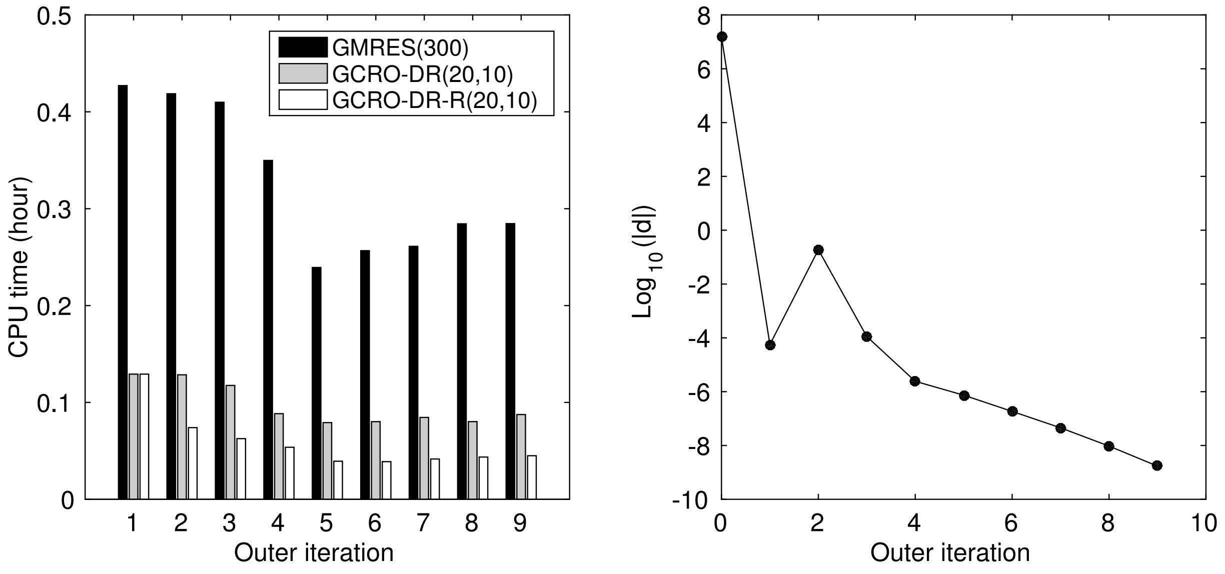
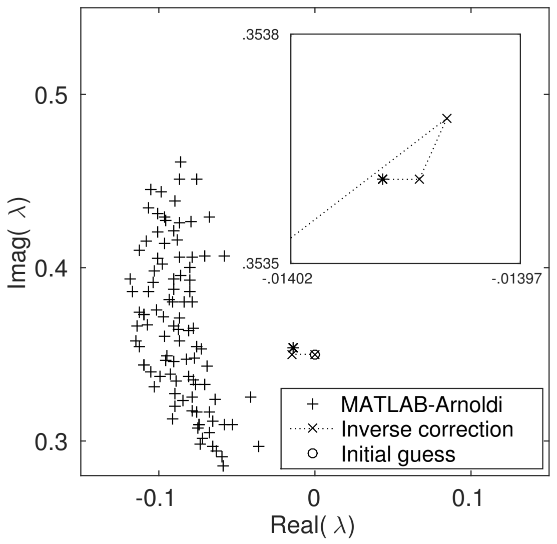


Figure 6: Case 2: Left: CPU time comparison of different solvers for all nine outer iterations; right: converge of the eigenvalue problem.



iter	real	imaginary
1	+4.1858e-7	+3.4999e-1
2	- 1.4787e-2	+3.5294e-1
3	- 1.3986e-2	+3.5369e-1
4	- 1.3992e-2	+3.5361e-1
5	- 1.4000e-2	+3.5361e-1
6	- 1.4001e-2	+3.5361e-1
7	- 1.4001e-2	+3.5361e-1
8	- 1.4001e-2	+3.5361e-1
9	- 1.4001e-2	+3.5361e-1
---------	-----------------	----------------
Arnoldi	- 1.4001e-2	+3.5361e-1

Figure 7: Case 2: Left: Evolution of the tracked eigenvalue during outer iterations plotted along with the eigenvalues computed using Arnoldi iterations implemented in MATLAB; right: the eigenvalue during outer iterations compared to the one from Arnoldi iterations.

calculation has 0.74 millions nodes, equivalent to 3.7 million complex-valued unknowns for the linear system. In terms of storage, the flow Jacobian matrix for the second order spatial discretisation has around 451 million non-zero real-valued entries requiring around 11.3 GB memory to be stored with double precision.

Using ten mode shapes of the structure (the first mode, dominant in wing bending, as mapped to the CFD surfaces is illustrated on the left of Fig. 8), ten eigenvalues are found using the Schur complement method. The Schur complement method can efficiently compute eigenvalues and the associated structural part of the right and left eigenvectors.

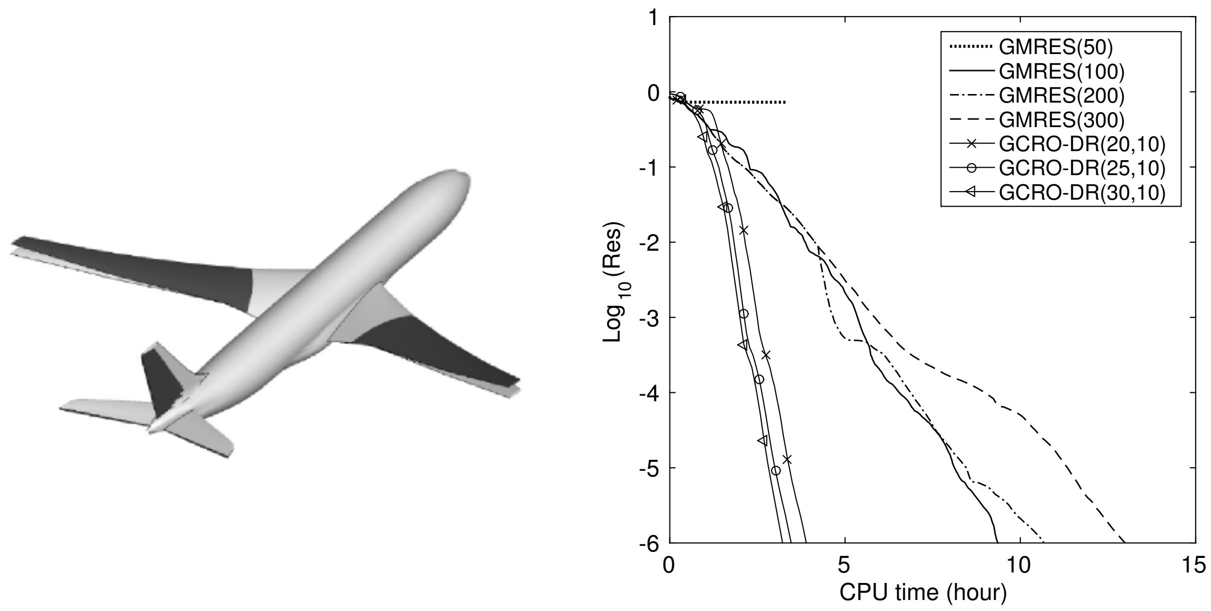


Figure 8: Case 3. Left: aerodynamic surfaces of XRF1 for projected mode 1; right: convergence for computing the first left eigenvector.

The remaining task is to find the fluid part of the eigenvectors using Eqs. (2) and (3). Once both the right and left eigenvectors of the coupled problems are found, a reduced order model can be constructed [4].

From previous experience, it is known that the adjoint (left) eigenvector problem is more challenging. Thus, GMRES and GCRO-DR using different parameters are first used to solve for the first left eigenvector with the convergence history shown on the right in Fig. 8. The convergence criterion has been set to six orders of magnitude, which is found to be sufficient to converge the unsteady lift derivative to within 1% accuracy. GMRES with 100 vectors is found to be optimal, i.e., using 50 vectors leads to convergence stagnation while using more slows down the convergence due to the increased cost in orthogonalisation. For GCRO-DR, there is only marginal variation in CPU time for the three combinations of parameters used thus GCRO-DR(30,10) is chosen for all the remaining eigenvector solves. Replacing GMRES(100) with GCRO-DR(30,10) reduces the total memory for the linear solve from 18.2 GB to 11.4 GB, with the difference being the 60 less vectors needed to be stored, while the CPU time is reduced by a factor of three.

The CPU time breakdown of the 20 eigenvector solves using GMRES(100) and GCRO-DR(30,10) is shown in Fig. 9. Using deflated restarting speeds up the whole calculation by a factor of two. However, the recycling technique does not seems to be effective for this case. Using eigenvector recycling even slows down the convergence for left eigenvector problems, while the speedup for the right eigenvector solves are marginal at around 6%. This is presumably due to the fact that the approximate eigenvectors recycled from solving the first eigenvector is no longer a good approximation for the eigenvectors of the second linear system.

It was reported in [15] that GCRO-DR with recycling may not necessarily accelerate the convergence even when recycling is applied to the same equation with both identical coefficient matrices and right-hand sides. This same behaviour was found in this case

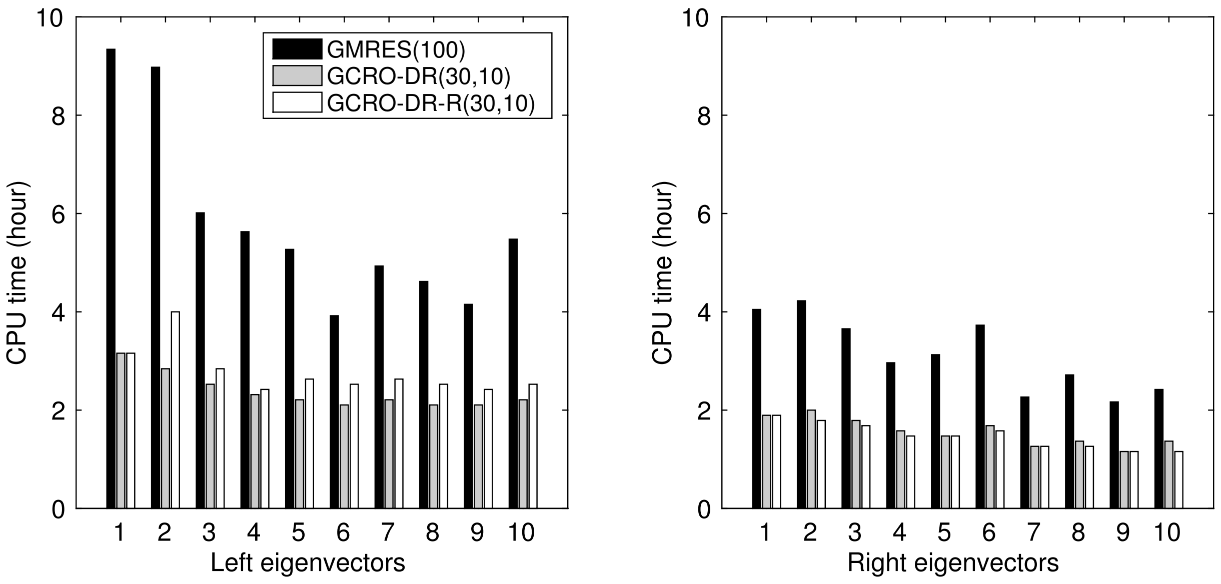


Figure 9: Case 3: CPU time for solving all eigenvectors using the three different solvers.

when repeatedly solving the first left eigenvector problem, i.e., solving the equation again with recycled eigenvectors from the previous solve of the same equation does not speed up convergence. Another feature of the convergence is that the left eigenvectors are in general more difficult to solve compared to the right eigenvectors, even though their spectra are identical. This implies that the right-hand side plays an important role in the convergence in these cases [17]. Further investigation into this problem is underway.

5 CONCLUSIONS

In this paper, generalised conjugate residual solver with deflated restarting is applied to a few typical problems of linearised aerodynamic analysis and the improvement over the baseline generalised minimal residual method, regarding both the memory requirement and CPU time, is demonstrated. The cases investigated are i) frequency response computation of an aerofoil undergoing pitch or plunge in transonic flow at near buffet condition, ii) eigenvalue solve via the inverse correction method for the same aerofoil, and iii) computing the left and right eigenvectors for an aircraft model. All the test cases involve solving large sparse linear systems of equations arising from linear frequency domain Navier-Stokes and Euler equations.

The deflation technique significantly reduces both the CPU time and the number of Krylov vectors that need to be stored for all cases. Although recycling eigenvectors between equation does not improve the asymptotic convergence rate, it does significantly improve the transient convergence by overcoming the initial stagnation, making it ideal for solving a sequence of linear systems of equations with relaxed convergence criteria, such as computing the least stable eigenvalue of the near-buffet aerofoil case where the convergence is accelerated by an additional factor of two. However, the convergence acceleration is not effective for computing the eigenvectors for the aircraft case. This is probably related to the right-hand sides and that the problem is not sufficiently stiff since the flow is assumed to be inviscid and the mesh is quite coarse.

Future work includes implementing the recycling algorithm in DLR–TAU code, taking advantage of the existing parallel infrastructure. A more realistic aircraft case in transonic flow with strong shock–boundary layer interaction will be investigated, where the linear system is expected to be much stiffer, exploiting the superior property of the deflation/recycling techniques.

6 ACKNOWLEDGEMENTS

The first author would like to thank Dr. Jens-Dominik Müller from Queen Mary University of London for the discussion on Krylov subspace recycling at the beginning of this work. The research leading to these results was co-funded by Innovate UK, the UK's innovation agency, as part of the Enhanced Fidelity Transonic Wing project.

7 REFERENCES

[1] Thormann, R. and Widhalm, M. (2013). Linear-frequency-domain predictions of dynamic-response data for viscous transonic flows. *AIAA Journal*, 51(11), 2540–2557.

[2] Timme, S., Marques, S., and Badcock, K. (2011). Transonic aeroelastic stability analysis using a kriging-based Schur complement formulation. *AIAA Journal*, 49(6), 1202–1213.

[3] Da Ronch, A., Badcock, K. J., Wang, Y., et al. (2012). Nonlinear model reduction for flexible aircraft control design. *AIAA Paper 2012–4404*.

[4] Timme, S., Badcock, K., and Da Ronch, A. (2013). Linear reduced order modelling for gust response analysis using the DLR-TAU code. *Proc. IFASD*.

[5] Crouch, J., Garbaruk, A., and Magidov, D. (2007). Predicting the onset of flow unsteadiness based on global instability. *Journal of Computational Physics*, 224(2), 924–940.

[6] Iorio, M., González, L., and Ferrer, E. (2014). Direct and adjoint global stability analysis of turbulent transonic flows over a NACA0012 profile. *International Journal for Numerical Methods in Fluids*, 76(3), 147–168.

[7] Sartor, F., Mettot, C., and Sipp, D. (2014). Stability, receptivity, and sensitivity analyses of buffeting transonic flow over a profile. *AIAA Journal*.

[8] Nitzsche, J. (2009). A numerical study on aerodynamic resonance in transonic separated flow. *Proc. IFASD*.

[9] Iovnovich, M. and Raveh, D. E. (2012). Reynolds-averaged Navier-Stokes study of the shock-buffet instability mechanism. *AIAA Journal*, 50(4), 880–890.

[10] Thormann, R., Nitzsche, J., and Widhalm, M. (2012). Time-linearized simulation of unsteady transonic flows with shock-induced separation. In *6th European Congress on Computational Methods in Applied Sciences and Engineering*.

[11] Rüde, U. and Schmid, W. (1995). *Inverse multigrid correction for generalized eigenvalue computations*. Technical Report, Universität Augsburg.

[12] Saad, Y. and Schultz, M. H. (1986). GMRes: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3), 856–869.

[13] De Sturler, E. (1999). Truncation strategies for optimal krylov subspace methods. *SIAM Journal on Numerical Analysis*, 36(3), 864–889.

[14] Morgan, R. B. (2002). GMRES with deflated restarting. *SIAM Journal on Scientific Computing*, 24(1), 20–37.

[15] Parks, M. L., De Sturler, E., Mackey, G., et al. (2006). Recycling krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5), 1651–1674.

[16] Liesen, J. and Tichỳ, P. (2004). Convergence analysis of krylov subspace methods. *GAMM-Mitteilungen*, 27(2), 153–173.

[17] Titley-Peloquin, D., Pestana, J., and Wathen, A. J. (2014). GMRES convergence bounds that depend on the right-hand-side vector. *IMA Journal of Numerical Analysis*, 34(2), 462–479.

[18] McCracken, A. J., Timme, S., and Badcock, K. J. (2012). Accelerating convergence of the CFD linear frequency domain method by a preconditioned linear solver. In *6th European Congress on Computational Methods in Applied Sciences and Engineering.*

COPYRIGHT STATEMENT

APPENDIX A

Algorithm 1 Preconditioned GCRO-DR-R($A, \mathbf{x}_0, \mathbf{b}, m, k, \text{iter\_max}, \text{tol}, U, L, \text{Recycle}$)

```
 1: $\mathbf{r} \leftarrow U^{-1}(L^{-1}(\mathbf{b} - A\mathbf{x}_0))$; iter $\leftarrow 0$
 2: **if** (Recycle=True) **then** % *QR factorisation of recycled* $Y_k$
 3:     $[Q, R] \leftarrow \text{QR}(Y_k)$; $C_k \leftarrow Q$; $U_k \leftarrow Y_k R^{-1}$
 4: **else** % *$m$ Arnoldi iterations to calculate* $Y_k$
 5:     $\beta \leftarrow \|\mathbf{r}\|$; $\mathbf{v}_1 \leftarrow \mathbf{r}/\beta$
 6:     **for** $i = 1, 2, \ldots, m$ **do**
 7:         $\mathbf{v}_{i+1} \leftarrow U^{-1}(L^{-1}(A\mathbf{v}_i))$; iter $\leftarrow$ iter $+ 1$
 8:         **for** $j = 1, 2, \ldots, i$ **do** % *Orthogonalisation*
 9:             $h_{j,i} \leftarrow \mathbf{v}_j^H \mathbf{v}_{i+1}$; $\mathbf{v}_{i+1} \leftarrow \mathbf{v}_{i+1} - h_{j,i}\mathbf{v}_j$
10:         **end for**
11:         $h_{i+1,i} \leftarrow \|\mathbf{v}_{i+1}\|$; $\mathbf{v}_{i+1} \leftarrow \mathbf{v}_{i+1}/h_{i+1,i}$
12:     **end for**
13:     Solve for $\mathbf{d}_m$ that minimises $\|\beta\mathbf{e}_1 - \bar{H}_m\mathbf{d}_m\|$
14:     $\mathbf{x} \leftarrow \mathbf{x}_0 + V_m\mathbf{d}_m$; $\mathbf{r} \leftarrow V_{m+1}(\beta\mathbf{e}_1 - \bar{H}_m\mathbf{d}_m)$
15:     Compute the $k$ interior eigenvectors $\mathbf{p}_i$ of $(H_m + h^2_{m+1,m}H_m^{-H}\mathbf{e}_m\mathbf{e}_m^H)\mathbf{p}_i = \theta_i\mathbf{p}_i$
16:     $[Q, R] \leftarrow \text{QR}(\bar{H}_m P_k)$; $C_k \leftarrow V_{m+1}Q$; $Y_k \leftarrow V_m P_k$; $U_k \leftarrow Y_k R^{-1}$
17: **end if**
18: $\mathbf{v}_1 \leftarrow \mathbf{r}/\|\mathbf{r}\|$
19: **for** $i = 1,\ 2,\ \ldots,\ m - k$ **do**
20:     $\mathbf{v}_{i+1} \leftarrow (I - C_k C_k^H)(U^{-1}(L^{-1}(A\mathbf{v}_i)))$; iter $\leftarrow$ iter $+ 1$
21:     **for** $j = 1,\ 2,\ \ldots,\ i$ **do** % *Orthogonalisation*
22:         $h_{j,i} \leftarrow \mathbf{v}_j^H \mathbf{v}_{i+1}$; $\mathbf{v}_{i+1} \leftarrow \mathbf{v}_{i+1} - h_{j,i}\mathbf{v}_j$
23:     **end for**
24:     $h_{i+1,i} \leftarrow \|\mathbf{v}_{i+1}\|$; $\mathbf{v}_{i+1} \leftarrow \mathbf{v}_{i+1}/h_{i+1,i}$
25: **end for**
26: $D_k \leftarrow \text{Diag}(\|\mathbf{u}_1\|^{-1}, \|\mathbf{u}_2\|^{-1}, \ldots, \|\mathbf{u}_k\|^{-1})$
27: $B_k \leftarrow C_k^H(U^{-1}(L^{-1}(AV_{m-k})))$; $\bar{G}_m \leftarrow \left[D_k, B_k; 0, \bar{H}_{m-k}\right]$
28: $\hat{V}_m \leftarrow [U_k D_k, V_{m-k}]$; $\hat{W}_{m+1} \leftarrow [C_k, V_{m-k+1}]$
29: Solve for $\mathbf{d}_m$ that minimises $J(\mathbf{d}_m) := \|\hat{W}_{m+1}^H\mathbf{r} - \bar{G}_m\mathbf{d}_m\|$
30: $\mathbf{x} \leftarrow \mathbf{x} + \hat{V}_m\mathbf{d}_m$; $\mathbf{r} \leftarrow \mathbf{r} - \hat{W}_{m+1}\bar{G}_m\mathbf{d}_m$
31: Solve for the $k$ interior eigenvectors $\mathbf{p}_i$ in $\bar{G}_m^H\bar{G}_m\mathbf{p}_i = \theta\bar{G}_m^H\hat{W}_{m+1}^H\hat{V}_m\mathbf{p}_i$
32: $[Q, R] \leftarrow \text{QR}(\bar{G}_m P_k)$; $C_k \leftarrow \hat{W}_{m+1}Q$; $Y_k \leftarrow \hat{V}_m P_k$; $U_k \leftarrow Y_k R^{-1}$
33: **if** $(J(\mathbf{d}_m) <$ tol or iter $>$ iter_max$)$ **then**
34:     $Y_k = U_k$; Terminate programme.
35: **end if**
36: Goto line 18
```