Robust and efficient adjoint solver for complex flow conditions

Shenren Xu*, Sebastian Timme

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

Abstract

A key step in gradient-based aerodynamic shape optimisation using the Reynolds-averaged Navier–Stokes equations is to compute the adjoint solution. Adjoint equations inherit the linear stability and the stiffness of the nonlinear flow equations. Therefore for industrial cases with complex geometries at off-design flow conditions, solving the resulting stiff adjoint equation can be challenging. In this paper, Krylov subspace solvers enhanced by subspace recycling and preconditioned with incomplete lower-upper factorisation are used to solve the stiff adjoint equations arising from typical design and off-design conditions. Compared to the baseline matrix-forming adjoint solver based on the generalized minimal residual method, the proposed algorithm achieved memory reduction of up to a factor of two and convergence speedup of up to a factor of three, on industry-relevant cases. These test cases include the DLR-F6 and DLR-F11 configurations, a wing-body configuration in pre-shock buffet and a large civil aircraft with mesh sizes ranging from 3 to 30 million. The proposed method seems to be particularly effective

*Corresponding author

Email address: shenren.xu@liverpool.ac.uk (Shenren Xu)

for the more difficult flow conditions.

Keywords: adjoint method, Krylov solvers, subspace recycling, GCRO-DR, RANS equations

1. Introduction

Over the past few decades, adjoint-based aerodynamic shape optimisation using computational fluid dynamics (CFD) has been widely used for the design of automobiles [1], aeroplanes [2, 3, 4, 5] and turbomachines [6, 7, 8]. It was first proposed in [9] to use the adjoint equations to efficiently compute the design gradient for aerodynamic shape optimisation. The method was later extended to configurations of increasing complexity such as the redesign of the wing of a transonic business jet using Euler equations on multiblock structured meshes [2] as well as for Navier–Stokes equations on unstructured meshes to capture the viscous effect on complex shapes [10, 11]. A comprehensive strategy for developing and implementing discrete adjoint methods for aerodynamic shape optimisation problems is presented in [12] and demonstrated in a three-dimensional unstructured Reynolds-averaged Navier–Stokes (RANS) adjoint solver on several cases including a high-lift configuration and a modern transport configuration. The methodology was later extended in [13, 14] to include multigrid in the line-implicit adjoint solver for better convergence and applied to the drag-reduction optimisation of a wing body configuration.

With the maturing of the adjoint method, applications nowadays are more focused on realistic configurations under both design and off-design conditions. The increased complexity in both geometry and flow conditions can

pose significant computational challenges for the adjoint solver. Flow and adjoint solvers using well-established fixed-point iterations, either explicit or implicit, could have difficulty converging. One such example is reported in [15] for a transonic viscous case with a mesh consisting of 69,000 points with stretched cells in the boundary layer. Similar issues are reported for more realistic cases in [16], where the DLR-TAU adjoint solver is used to optimise the DLR-F6 wing body configuration and the DLR-F11 high-lift configuration. For the DLR-F6 case, side-of-body separation near the trailing edge destabilises fixed-point iteration and recursive projection method (RPM) [17] is applied to stabilise the adjoint. However, RPM fails to stabilise the adjoint for DLR-F11 [16] because the unstable fixed point iteration diverged too fast, and generalised minimal residual method (GMRES) [18] was used to successfully converge the case.

The numerical stiffness discussed above is mainly due to the ill-conditioned coefficient matrix in the adjoint equations. The issue could be alleviated to some extent by using an approximate instead of exact flow Jacobian matrix. Essentially, one is trading accuracy for solver efficiency and robustness. One typical remedy is to use the frozen turbulence assumption when solving the adjoint RANS equations as it is well known that coupling the turbulence equation with the mean flow equation significantly increases the numerical stiffness and sometimes it even destabilises the time marching scheme. The effect of various other approximations of the Jacobian matrix on the gradient accuracy and the optimisation results is investigated in detail in [19].

Alternatively, one could adhere to the exact Jacobian matrix and solve the stiff adjoint equations more efficiently so that the resulting adjoint solution,

and consequently the design gradient, remains accurate. To avoid the linear instability issue of any fixed-point iterative solver, Krylov solvers are usually preferred for solving the stiff and marginally stable adjoint equations. It is proposed that the Jacobian-free Newton–Krylov method is preferred for critical aerodynamic simulations and shape optimisation applications where numerical stiffness constantly causes convergence difficulties [20, 21]. A few key aspects on the efficient implementation of the method are also highlighted to show that once properly implemented, superior efficiency and reliability can be achieved, compared with other more well-established solution methods such as multi-stage explicit schemes, point or block implicit procedure and implicit factorisation methods. The method has been successfully applied to solve adjoint equations arising from aerodynamic shape optimisaion [22] and error estimation [23].

Krylov solvers are also affected by the conditioning of the system matrix. For example, restarted GMRES could suffer from convergence stagnation for challenging problems unless m is sufficiently large, which would then result in prohibitively high memory overhead. An obvious remedy to alleviate the memory bottleneck of the Krylov solver for difficult cases is to use a stronger preconditioner. For example, a clean wing geometry for the common research model at cruise condition is studied in [24] using a mesh with 28 million points. For this case, incomplete lower-upper (ILU) factorisation with fill-in level of two, i.e., ILU(2), is necessary to effectively precondition GMRES, which is then able to converge with m of 200. Had a weaker preconditioner such as ILU(0) been used, the Krylov solver would have required many more vectors to converge.

The fundamental reason for the convergence stagnation of GMRES(m) is that the restarted subspace is often close to the previous subspace. Generalised conjugate residual with optimal truncation (GCROT) [25], its simplified and flexible variant [26] and generalised conjugate residual with deflated restarting (GCRO-DR) [27] have been proposed to address this shortcoming by recycling a selected subspace from one cycle to the next. The subspace recycling technique allows the solvers to converge without stagnation with much lower memory requirement. GCRO-DR was shown to be effective in both lowering the stagnation memory threshold and accelerating the convergence for large scale linearised aerodynamics analysis [28].

In this paper, we replace the baseline GMRES solver within the DLR-TAU adjoint solver with GCRO-DR. The proposed method is applied to solve the adjoint equations to demonstrate its effectiveness in both reducing memory overhead and accelerating convergence for solving the adjoint equations arising from industry-relevant cases with complex geometries under both design and off-design flow conditions.

The remainder of the paper is organized as follows. The mathematical formulation of the flow and adjoint equations is explained in Sec. 2. The details of the Krylov solvers are given in Sec. 3 and the preconditioning technique is discussed in Sec. 4. The application of the proposed method to five test cases is presented in Sec. 5. A comprehensive comparison between GMRES, GCROT and GCRO-DR is first given for a small, yet stiff, two-dimensional aerofoil case for a parameter study. Both GCRO-DR and GMRES are then applied to several more realistic three-dimensional industry-relevant cases.

2. Nonlinear flow and adjoint solvers

2.1. Nonlinear flow solver

The DLR-TAU code is a CFD software package widely used as production code in the European aerospace industry as well as a research code for method development [29, 30]. The RANS equations are solved with a finite-volume discretisation on unstructured grids with various options of spatial and temporal discretisation schemes and turbulence models. In this paper, the mean flow is by default discretised with the Jameson–Schmidt–Turkel (JST) scheme [31] with matrix dissipation [32], unless stated otherwise. The Spalart–Allmaras model [33] is discretised using first-order accurate Roe scheme [34]. The nonlinear flow equations are pseudo time marched using the first-order backward Euler implicit scheme. At each pseudo time step, agglomeration multigrid is used to accelerate the convergence with lower-upper symmetric-Gauss–Seidel [35] as the multigrid smoother.

2.2. Adjoint solver

The cost function for optimisation $\mathbf{J} := (J_1, J_2, ..., J_N)^T$ is a function of the flow solution $\mathbf{U}$, the coordinates of the computational mesh points $\mathbf{X}$ and the design variable $\boldsymbol{\alpha} := (\alpha_1, \alpha_2, ..., \alpha_M)^T$. To evaluate the design gradient, the cost function is linearised as

$$\frac{\mathrm{d}\mathbf{J}}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\partial \mathbf{J}}{\partial \boldsymbol{\alpha}} + \mathbf{v}^T \mathbf{f}$$

where $\mathbf{v}$ is the solution to the adjoint equation

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right)^T \mathbf{v} = \mathbf{g} \tag{1}$$

with $\mathbf{f}$ and $\mathbf{g}$ defined as

$$\mathbf{f} := -\frac{\partial \mathbf{R}}{\partial \boldsymbol{\alpha}} \quad \text{and} \quad \mathbf{g}^T := \frac{\partial \mathbf{J}}{\partial \mathbf{U}}$$

and $\mathbf{R}$ is the nonlinear residual vector. Note that the design variables do not appear in the adjoint equation thus Eq. (1) needs to be solved only as many times as the number of cost functions. For aerodynamic applications, the cost functions are usually limited to a handful, such as lift, drag and moment, while the design variables could be many more. The adjoint approach is therefore very efficient.

The adjoint equation is solved using a Jacobian-forming Newton-Krylov approach. The exact flow Jacobian matrix corresponding to the second-order accurate spatial discretisation is computed using the hand-differentiated nonlinear residual subroutine. The Jacobian matrix is stored in block compressed sparse row format, with each block containing a 6-by-6 dense matrix. The Jacobian matrix is then transposed to obtain the coefficient matrix for the adjoint equation. Computing the Jacobian matrix and its transpose are done in parallel with negligible computational time compared to the adjoint solution time for all the cases considered in this work. The right-hand side for each cost function is computed using the linearised subroutine that computes the cost function. No simplification such as frozen turbulence is used in this work so that an exact dual adjoint solution is solved. Once the coefficient matrix and the right-hand side are formed, the resulting large sparse linear system of equations is then solved using ILU preconditioned Krylov solvers, which are explained in detail in the following two sections.

3. Krylov subspace solvers and subspace recycling

3.1. Basic Krylov solver GMRES

To solve the adjoint problem in Eq. (1), or to be more general, to solve a linear system of equations

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

with A denoting the coefficient matrix, $\mathbf{x}$ the solution vector and $\mathbf{b}$ the right-hand side vector, Krylov subspace solvers can be used. These solvers approximate the solution in the Krylov subspace

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

with the constraint that the resulting residual should be perpendicular to the subspace $A\mathcal{K}_m$.

In GMRES, one of the most popular Krylov solvers, Arnoldi procedure is used to generate a vector basis, $V_m := [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m]$, that spans the Krylov subspace. A by-product of the Arnoldi procedure is the upper Hessenberg matrix $\bar{H}_m$ satisfying the Arnoldi relation

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

where $V_{m+1} := [V_m, \mathbf{v}_{m+1}]$. Then the solution is approximated using the linear combination of the basis vectors

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

and the coefficient vector $\mathbf{d}_m$ is the minimiser of the resulting residual norm

$$\underset{\mathbf{d}_m}{\text{minimize}} \ \|\mathbf{r}(\mathbf{d_m})\| := \|\mathbf{b} - AV_m\mathbf{d}_m\| \tag{2}$$

An equivalent formulation for finding the minimiser $\mathbf{d}_m$ is to use the definition of the Krylov method, i.e., the resulting residual needs to be perpendicular to $A\mathcal{K}_m$, which is spanned by the column vectors of V_{m+1}. Therefore $\mathbf{d}_m$ can be solved from

$$V_{m+1}^H \mathbf{r}(\mathbf{d}_m) = 0$$

Using the Arnoldi relation, the left-hand side can be reduced to $V_{m+1}^H\mathbf{b} - \bar{H}_m\mathbf{d}_m$. Therefore, the coefficient vector $\mathbf{d}_m$ is simply the least square solution of

$$\bar{H}_m\mathbf{d}_m = V_{m+1}^H\mathbf{b}. \tag{3}$$

which is a system of much lower dimension compared to Eq. (2). In practice, Givens rotation is used to compute $\mathbf{d}_m$ in Eq. (3). The matrix $\bar{H}_m$ is an upper Hessenberg matrix of dimension $(m+1)$-by-m and it can be diagonalised via Givens rotation to an upper triangular matrix of dimension m-by-m with an additional row of all zeros at the bottom. Discard the last row, and the solution $\mathbf{d}_m$ is computed by back-substitution.

To limit the memory use, restarted GMRES, denoted by GMRES(m), is used. Once a maximum of m Krylov vectors are built, the solution is updated, and GMRES is restarted using the updated solution and residual vectors. Although GMRES(m) is very robust for many problems, it often encounters convergence stagnation unless m is large enough.

3.2. Nested Krylov solver GCRO

The deflated solver proposed in this work is a type of the nested Krylov solvers. To better illustrate it, we first introduce the GCRO solver as a framework for nested solvers that use generalised conjugate residual solver (GCR) for the outer loop and another Krylov solver such as GMRES for the inner loop. GCR is mathematically equivalent to GMRES but the numerical procedure is different and more flexible. In GCR, two vector bases

$$U_k = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k] \text{ and } C_k = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k]$$

satisfying

$$C_k = AU_k \text{ and } C_k^H C_k = I_k \tag{4}$$

are constructed. For the numerical procedure regarding the construction of the vector bases, refer to [36]. The solution is then approximated on the subspace spanned by the column vectors of U_k

$$\mathbf{x}_k = \mathbf{x}_0 + U_k \mathbf{d}_k$$

subject to the constraint that the resulting residual is perpendicular to the subspace spanned by the column vectors of AU_k, or equivalently, C_k, i.e.,

$$C_k^H \mathbf{r} = C_k^H (\mathbf{r}_0 - AU_k \mathbf{d}_k) = \mathbf{0}$$

which leads to

$$\mathbf{d}_k = C_k^H \mathbf{r}_0$$

Consequently the residual is updated as

$$\mathbf{r}_k = \mathbf{r}_0 - AU_k\mathbf{d}_k = (I - C_kC_k^H)\mathbf{r}_0$$

The nested solver GCRO wraps GCR around another Krylov solver such as GMRES. After U_k and C_k are constructed and the residual is updated as

$$\mathbf{r}_k \leftarrow (I - C_kC_k^H)\mathbf{r}_0,$$

instead of setting

$$\mathbf{u}_{k+1} \leftarrow \mathbf{r}_k,$$

as in GCR, we set

$$\mathbf{u}_{k+1} \leftarrow \tilde{\mathbf{x}},$$

where $\tilde{\mathbf{x}}$ is the approximate solution to

$$(I - C_kC_k^H)A\tilde{\mathbf{x}} = \mathbf{r}_k \tag{5}$$

solved with a few GMRES iterations. Intuitively, it can be seen that compared with $\mathbf{r}_k$, $\tilde{\mathbf{x}}$ is a better approximation for the final solution. In fact, if Eq. (5) is solved exactly, then the outer loop is also fully converged.

3.2.1. GCRO-DR

The GCRO solver offers a flexible framework for solving the linear system of equations in a nested approach. Different from GMRES where the vector basis V_m has to span a Krylov subspace, GCR is flexible in the sense that the vector basis U_k can be any combination as long as the relation in Eq. (4)

holds. As shown in [27], recycling the approximate interior eigenvectors from each inner GMRES cycle to form the U_k and C_k vector bases seems to be very effective in improving the performance. Because the recycled subspace is related to the eigenvectors, the resulting algorithm is a deflated solver, thus 'deflated restarting' in its name. Deflation itself is an established technique to reduce the condition number of a matrix by removing eigenvectors with extreme eigenvalues. Its use has been popular in improving the convergence of the conjugate gradient (CG) iteration when solving the pressure-Poisson equation [37] in incompressible flows. GCRO-DR uses the deflation technique by recycling a set of approximate interior eigenvectors from one cycle to the next. More specifically, GCRO-DR(m, k) extracts a subspace of dimension k (usually much smaller than m) from the Krylov subspace formed within the inner GMRES cycle, and uses the 'recycled' small subspace to construct the U_k and C_k vector bases in the outer loop and then continue with the next cycle with $(m - k)$ iterations of GMRES.

The algorithm of GCRO-DR begins with a start-up GMRES cycle with m Arnoldi iterations which produces the upper Hessenberg matrix $\bar{H}_m$ and the Krylov vectors V_m. The solution $\mathbf{x}$ and residual vectors $\mathbf{r}$ are first updated as in GMRES. After that, an additional step to extract the approximate interior eigenvectors $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$ of the matrix A is taken. To compute these approximate interior eigenvectors, the eigenvalue problem

$$(H_m + h_{m+1,m}^2 H_m^{-H} \mathbf{e}_m \mathbf{e}_m^H)\mathbf{p}_i = \theta_i \mathbf{p}_i, \ i = 1, \ldots, m \tag{6}$$

is first solved, where the square matrix H_m is $\bar{H}_m$ without the last row and

$h_{m+1,m}$ is the non-zero entry of $\bar{H}_m$ on its last row. Set

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

where $P_k = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k]$ are the eigenvectors corresponding to the k smallest eigenvalues θ. The matrices C_k and U_k are constructed from Y_k by setting

$$C_k \leftarrow V_{m+1} Q \quad \text{and} \quad U_k \leftarrow Y_k R^{-1} \tag{7}$$

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. (4). The start-up cycle is followed by a deflated GMRES cycle in which we perform $(m-k)$ Arnoldi iterations starting with $\mathbf{v}_1 = \mathbf{r}/\|\mathbf{r}\|$ using the linear operator $(I - C_k C_k^H)A$ such that the Krylov vectors to be formed are orthogonal to C_k. Matrices U_k and V_{m-k} are then combined to form a subspace to approximate the solution. 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 $D_k = \text{diag}(\|\mathbf{u}_1\|^{-1}, \|\mathbf{u}_2\|^{-1}, \ldots, \|\mathbf{u}_k\|^{-1})$ and $B_{m-k} = C_k^H A V_{m-k}$. The solution update $\delta\mathbf{x}$ is approximated over the subspace spanned by the columns of $\hat{V}_m$ and we solve for the coefficient vector $\mathbf{d}_m$ that minimises the norm of

the resulting 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 by solving a least square problem of very low dimension. According to [36], a more efficient alternative is to first solve for the last $(m-k)$ components of $\mathbf{d}_m$ using $\bar{H}_{m-k}$ only and then solve for the first k components of $\mathbf{d}_m$ that correspond to the basis vectors U_k. This alternative approach could be more accurate for some cases, although no notable difference is observed in our work. The solution and residual vectors are then updated with $\mathbf{d}_m$. In addition, we compute θ_i and $\mathbf{p}_i$ of the generalised eigenvalue problem

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

similar to Eq. (6). The approximate interior eigenvectors of the coefficient matrix are $Y_k = \hat{V}_m P_k$ with P_k containing the k interior eigenvectors as its columns. To form C_k and U_k, first perform QR-factorisation of $\bar{G}_m P_k$ and then set

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

similar to Eq. (7). The deflated GMRES cycle is repeated using the most recent solution and residual vectors until the stopping criterion is met.

Upgrading an existing GMRES solver to GCRO-DR is straightforward, involving only the solution of a few low-dimensional eigenvalue problems of size m. This task can be done using off-the-shelf linear algebra libraries, such as LAPACK [38]. Details of the GCRO-DR solver and its implementation in DLR-TAU code have previously been presented in [28] for solving a complex-valued forward problem.

3.2.2. GCROT

Similar to GCRO-DR, GCROT [25] is another Krylov solver that takes advantage of subspace recycling technique. It recycles a smaller subspace from the subspace generated by the inner GMRES cycle such that the loss of orthogonality with respect to the truncated space in minimised. The original version relies on singular-value-decomposition technique both to select the subspace to recycle and to determine which subspace to discard during truncation. A simplified and flexible variant denoted by GCROT(m, k) is proposed in [26], in which only the residual update vector is recycled to replace the oldest vector in C_k in the outer loop. It is also flexible in that a non-stationary preconditioner, such as approximate-Schur method, can be used which is found to outperform the additive-Schwarz method when solving the adjoint equations [39].

4. ILU preconditioner based on blended Jacobian

Incomplete lower-upper factorisation is used in this work to precondition the Krylov solvers. The ILU preconditioner is based on the blended Jacobian matrix. Denoting the transposed Jacobian matrices for spatial discretisation of first- and second-order accuracy as A^{2ndO} and A^{1stO}, the blended Jacobian matrix is a linear interpolation of the two as

$$A^{\mathrm{blend}}(\beta) = (1-\beta)A^{\mathrm{1stO}} + \beta A^{\mathrm{2ndO}} \tag{8}$$

The incomplete factorisation based purely on A^{2ndO} is not very effective unless a large fill-in level is used [40, 41]. However, the preconditioning effect of ILU significantly improves if β is sufficiently away from unity [42]. To

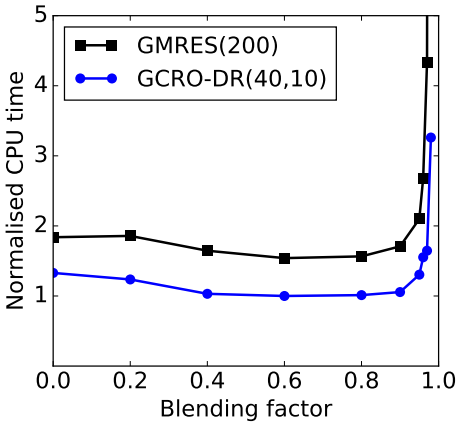

Figure 1: CPU time comparison of GMRES and GCRO-DR for different β values. The CPU time is normalised by that of GCRO-DR(40,10) for $\beta = 0.6$.

demonstrate this, Fig. 1 shows the CPU time variation of the Krylov solvers when different values of β are used. The results are based on case 2a which is investigated more thoroughly in the results section. The preconditioning effect drastically worsens when a value close to unity is chosen for β. Both solvers suffer from stagnation when a purely second-order Jacobian matrix based ILU is used. The exact β value is case-dependent. The curves are flat in the center, we thus chose $\beta = 0.5$ as the default value for all the cases studied in this paper.

The incomplete factorisation with p fill-in levels, denoted by ILU(p), means that the resulting ILU matrix has the sparsity pattern of the coefficient matrix to the power of $p + 1$. Intuitively, a larger value for p usually provides better preconditioning effect, but at the cost of both larger memory overhead due to the extra fill-ins and larger CPU time per iterations due to the extra floating-point operations. The optimal value of p to achieve the best CPU time vs. memory is highly case-dependent. We will explore this in de-

tail for a two-dimensional aerofoil case in Sec. 5, which indicates that ILU(0) seems to be the sweet spot and is thus used for all other three-dimensional test cases.

4.1. Parallel scalability of local ILU

For parallel implementation, the ILU preconditioning matrices are computed for each partition without considering the exchange of information between partitions. The decoupled approach allows a simple and essentially sequential ILU implementation for each partition at the cost of deteriorating convergence rate with increased number of partitions. The parallel scalability of the ILU(0)-preconditioned GCRO-DR is studied for case 5a (detail of this case can be found in the results section) using 20 to 200 cores on Westmere and Ivy Bridge CPUs. Shown in Fig. 2 is the parallel speedup on both architectures and the iteration numbers for converging ten orders of magnitude. The linear solver is shown to scale poorly for more than 100 cores. This is mainly due to the communication latency of the global sum operation required in each iteration for orthogonalisation and normalisation of the Krylov base vectors [43] as well as the suboptimal partitioning algorithm that is de facto chosen to achieve load balancing for computing the nonlinear flow solution. Contrary to conventional wisdom, the poor parallel scalability does not seem to be mainly caused by the local ILU preconditioner. As shown in Fig. 2, a maximum of $\pm 10\%$ fluctuation is observed for the number of iterations, indicating the adverse decoupling effect of the local ILU is relatively small. Therefore, replacing the current local ILU preconditioner with a more global variant would not improve the parallel scalability much, if at all, since it would require additional inter-core communication and would involve

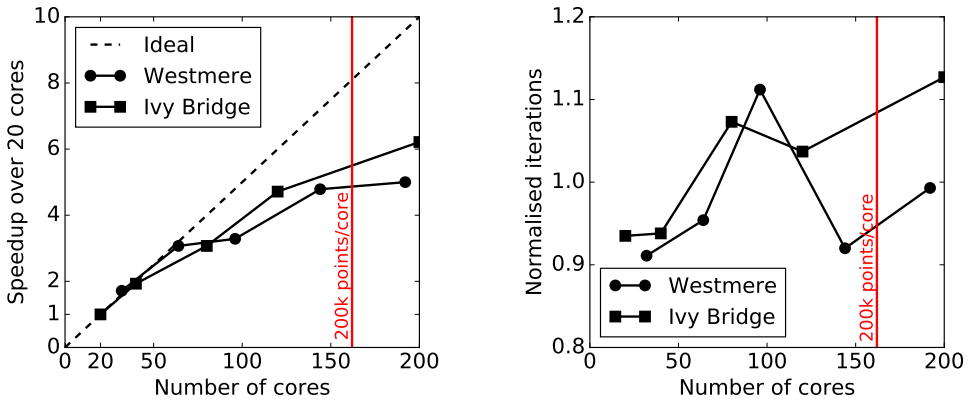

Figure 2: Parallel scalability and normalised iteration number variation of ILU(0)-preconditioned GCRO-DR for DLR-F11 (case 5a). The iteration numbers are normalised against the average value of all the 10 data points. Each Westmere node has 12 cores while each Ivy Bridge node has 16.

more floating-point operations for the matrix-vector product at the preconditioning step. For industry-relevant cases, the local ILU preconditioning seems to be a good compromise between solver performance and simplicity of implementation. To allow a more generalised comparison, the red vertical line indicates the core number for which approximately 200,000 grid points are assigned to each core, excluding the additional halo points at partition boundaries.

4.2. Additive-Schwarz vs. approxiamte-Schur preconditioning

Two popular paradigms of applying the preconditioner in the distributed manner are additive-Schwarz [44] and approximate Schur [45]. The former computes the preconditioning matrix based on the diagonal block of the system matrix that is local to each parallel partition and the off-diagonal block matrices are omitted. Due to this decoupling, the preconditioning effectiveness often degenerates as the number of partitions increases. The

approximate-Schur preconditioning is devised to maintain a strong coupling to allow better parallel scalability for massively parallel computations. The approximate-Schur approach combined with a flexible Krylov solver such as flexible GMRES (FGMRES) or flexible GCROT have been shown previously to outperform additive-Schwarz in previous studies [26, 39]. However, the approximate-Schur preconditioner is less straightforward to implement compared to additive-Schwarz.

The local ILU preconditioner described in the previous section belongs to the category of additive-Schwarz type preconditioner. We chose this based on the following considerations. First, it is easy to implement, involving only applying the sequential ILU factorisation algorithm to the diagonal block Jacobian matrix that is local to each partition. Secondly, as shown in Fig. 2 (right), the performance degeneration due to the decoupling among the parallel partitions is not critical for the cases investigated in this work. Thirdly, for massively parallel computations, for example, with over 1000 cores, the parallel efficiency is likely to be limited by the Krylov solver itself. In that case, using a communication-avoiding Krylov solver seems to be a more urgent task [46]. Lastly, although not completely independent of each other, the Krylov solver and the preconditioning method are relatively separately aspects of the linear solver, and in this work we concentrate on the improvement of the Krylov solver itself, assuming that if a better preconditioning approach is used, the beneficial effect could be multiplied.

5. Results

Different Krylov solvers, namely, GMRES, GCRO-DR and GCROT, are applied to the computation of the adjoint solutions for a two-dimensional aerofoil case and four three-dimensional industry-relevant test cases at both design and off-design flow conditions to demonstrate the improvement regarding both the CPU time and memory requirement. The off-design cases used to demonstrate the computational challenges include a half wing-body model at large angle of attack, a wing-body model near buffet-onset and a high-lift configuration. For all the cases considered in the paper, the adjoint convergence behaviour for either drag or lift coefficient as the cost function is very similar and thus the lift coefficient is used as the cost function throughout.

5.1. NACA 0012 aerofoil

The first test case is a two-dimensional NACA 0012 aerofoil in transonic flow. The freestream Mach number is 0.76 with a Reynolds number of 10 million. Fully turbulent flow is assumed. The angles of attack used are 0° and 3.5° which are referred to as cases 1a and 1b respectively. For case 1b, the flow condition is near buffet onset. The two-dimensional computational domain is meshed using mixed elements of quadrilateral and triangular type with a total of 30,000 points. To compute the steady-state nonlinear flow solution at both angles of attack, the solution is initialised using freestream condition and directly started with '4V' multigrid cycles with Courant-Friedrichs-Lewy (CFL) number of 100 on fine grid and 20 for all coarse grids. The steady-state flow solutions are found when the density residual has reduced ten orders of

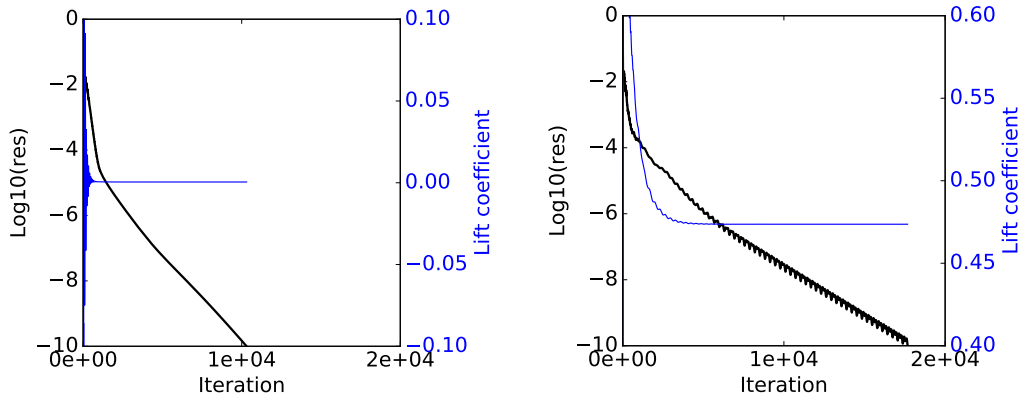

Figure 3: Convergence of density residual and lift coefficient at angle of attack of 0° (left) and 3.5° (right).

magnitude. Shown in Fig. 3 are the convergence histories for both cases. Case 1b requires nearly twice number of time steps to converge, presumably due to the stiffness near the buffet onset.

The adjoint equations are solved using GMRES, GCRO-DR and GCROT, preconditioned by ILU(0), ILU(1) and ILU(2). Convergence is reached when the inner residual of the Krylov solvers has dropped ten orders of magnitude. For this test case, the adjoint solver is run in sequential mode to exclude any effect due to the partition decoupling. We first run full GMRES to fully converge each case so that we get the upper bound of the number of Krylov vectors m_{max} for benchmarking GMRES. We then gradually decrease m and measure the number of matrix-vector multiplications and CPU time for different m. When m is sufficiently small, a sudden increase in the iteration number can be observed, indicating it is reaching the memory threshold for stagnation. Similar procedure is taken for GCRO-DR and GCROT, with some slight complication as two parameters are required. We exhaust all

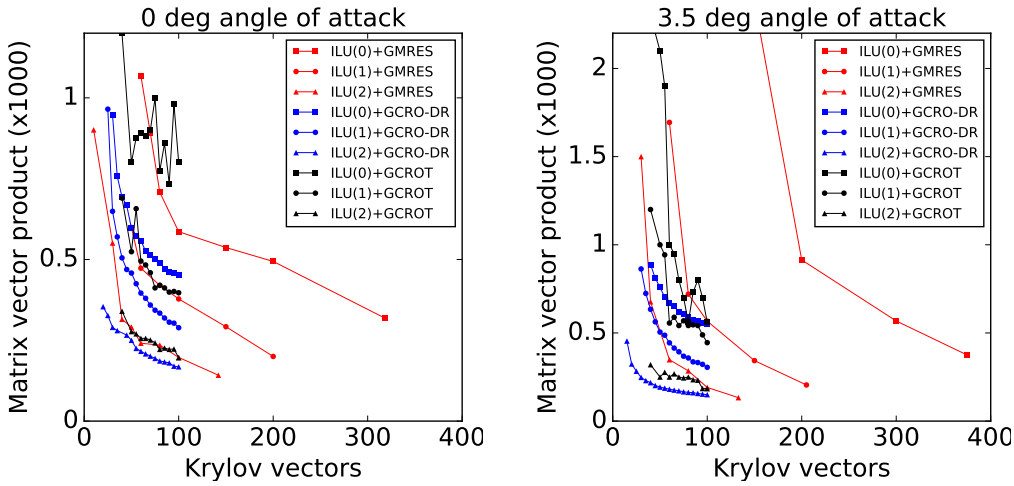

Figure 4: Iteration vs. Krylov vectors for GMRES, GCRO-DR and GCROT preconditioned by ILU(0), ILU(1) and ILU(2) at angle of attack of 0° (left) and 3.5° (right)

feasible combinations of m and k for a fixed $m+k$ value so that the number of vectors that need to be stored remains constant. We present the iteration number and CPU time corresponding to the optimal choice of m and k. The performance of GCRO-DR and GCROT for a fixed value of $m+k$ varies for different combinations of m and k. We found in our numerical experiment that the variation is much smaller for GCRO-DR than for GCROT, indicating the robustness of the former method. The number of iterations, or equivalently, the numbers of matrix vector product, for different Krylov solvers are plotted in Fig. 4. For all solvers, ILU with larger fill-in level always reduces the iteration number for the same number of Krylov vectors. Both solvers with recycling outperform GMRES with GCRO-DR consistently being the most efficient and robust one.

Besides the iteration numbers, we are more concerned with CPU time vs. memory. In Fig. 5 the CPU time is plotted against the memory requirement

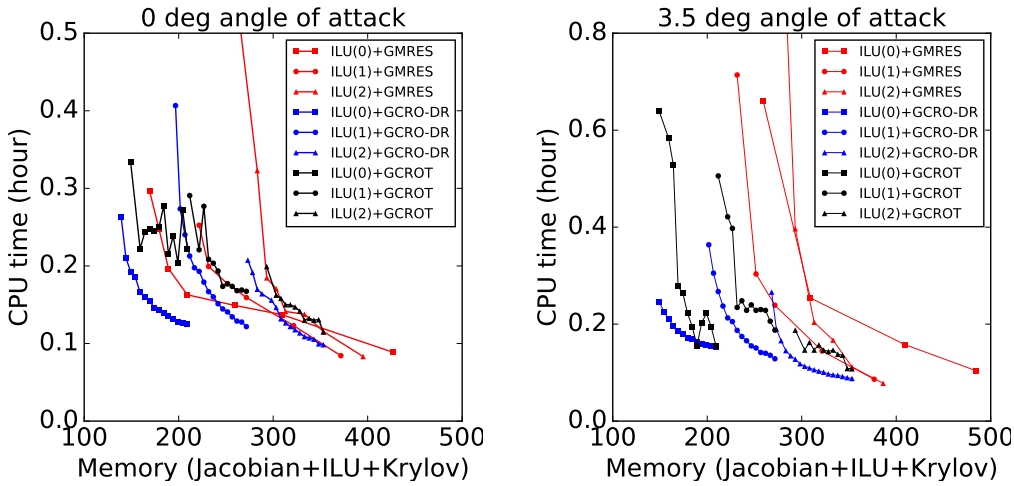

Figure 5: CPU time vs. memory required (bottom) for GMRES, GCRO-DR and GCROT preconditioned by ILU(0), ILU(1) and ILU(2) at angle of attack of 0° (left) and 3.5° (right). The memory required is normalised by the memory for storing one Krylov vector.

of each linear solver, including the Jacobian matrix, ILU preconditioning matrix and the Krylov vectors. The memory requirement is normalised by the memory for storing one flow solution or Krylov vector. The memory required by other auxiliary components of the solver such as the mesh metrics and the flow solution is negligible compared to the core linear solver and is thus not accounted for in the plot. This case has 180,000 degrees of freedom. Both the Jacobian matrix and ILU(0) have 9.8 million non-zero entries, while ILU(1) and ILU(2) have 21.0 and 35.7 million non-zero entries respectively. Therefore, the solvers preconditioned by ILU(0), ILU(1) and ILU(2), besides storing their respective Krylov vectors, require additional memory equivalent to 109, 171 and 253 Krylov vectors, to store both the Jacobian matrix and ILU preconditioning matrix. Therefore, compared to Fig. 4, the curves in Fig. 5 are shifted to the right accordingly. After taking into account the

additional memory of more fill-ins in ILU, the optimal solver option regarding CPU time vs. memory is then obviously ILU(0) preconditioned GCRO-DR. Furthermore, it also has the lowest memory threshold for convergence stagnation and is thus the most robust.

Although it is found in our numerical experiments that GCRO-DR seems to constantly outperform GCROT, it should be pointed out that there is no consensus in the research community regarding which is better [27]. In fact, Newton–Krylov type nonlinear flow and adjoint solvers using GCROT as its core Krylov solver have been successfully applied in [22, 47] with challenging applications of aircraft aerodynamic optimisation with realistic configurations. That the superior performance reported therein is not observed in our work is possibly due to the low fill in used in ILU. As shown in Fig. 4, the performance curve for GCROT does get less oscillatory (some indication of insensitivity to parameter change and thus increased robustness) when ILU fill-in level increases from 0 to 2, suggesting that GCROT be used in combination with ILU with high fill-in level.

5.2. DLR-F6 wing-body configuration

The DLR-F6 half wing-body model was used in the second AIAA CFD Drag Prediction Workshop to assess the state-of-the-art computational methods as practical aerodynamic tools for force and moment prediction on increasingly complex airframe geometries. The emphasis is on drag prediction accuracy [48]. The case has a mean aerodynamic chord of 141.2 mm and projected half-span of 585.647 mm. The wing is defined by four aerofoil sections. The DLR-F6 model is available both as wing-body-only and wing-body-nacelle-pylon configuration in order to assess the accuracy of the

computed interference drag due to installation. Nacelle and pylon are not included here for simplicity, although the inclusion of them does add to the complexity of the resulting flows which could potentially increase the stiffness of the equations. The computational mesh used can be found in the workshop's online resources. The mesh is unstructured with mixed-type elements and prisms in the boundary layers and consists of a total of 2.4 million points. The final wall-normal spacing is around 1, in wall units, and the cell growth is limited to be no larger than 1.25. The far field boundary is located 100 reference chord lengths away.

The flow conditions are a freestream Mach number of 0.75 and a Reynolds number based on mean aerodynamic chord of 3 million. Fully turbulent flow is assumed. Steady state flow solutions are calculated for angles of attack –4, –3.5, ..., 1.5, 2 and 4.5°, while the adjoint solution with lift coefficient as the cost function is computed only at angle of attack of 0 (case 2a) and 4.5° (case 2b). The zero angle of attack is considered as a representative design condition, while 4.5° describes a typical off-design condition.

To compute the steady-state nonlinear flow solutions, a total of 2000 iterations are first performed on the fine grid before switching to '4V' multigrid cycles. The CFL number is 100 for the fine and 20 for all coarse grid levels. Flow solutions are obtained by converging the density residual five orders of magnitude. Steady-state flow solutions at 0 and 4.5° angle of attack are shown in Figs. 6 and 7 along with the convergence histories of the nonlinear flow solver. Both cases reveal a separation zone at the wing-fuselage junction toward the trailing edge of the wing and the separation is significantly larger for 4.5° angle of attack. It also took 50% longer time to converge the 4.5°

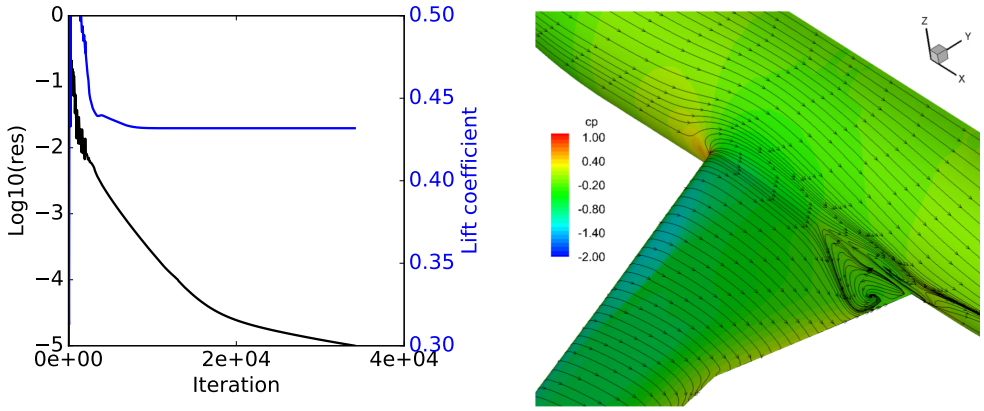

Figure 6: Left: convergence history of the nonlinear flow solver. Right: pressure coefficient and skin-friction lines based on surface shear force vector of DLR-F6 at 0° angle of attack.

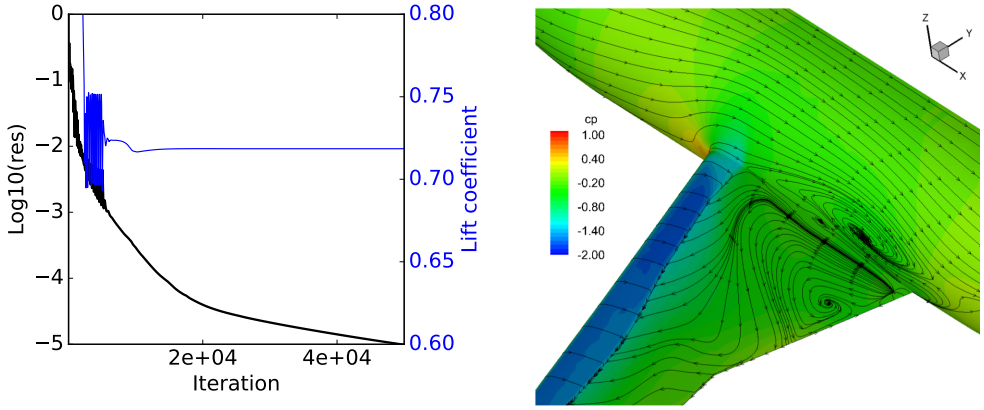

Figure 7: Left: convergence history of the nonlinear flow solver. Right: pressure coefficient and skin-friction lines based on surface shear force vector of DLR-F6 at 4.5° angle of attack.

angle of attack case, an indication of the numerical stiffness associated with the larger separation.

Lift and drag coefficients are compared with the experimental data in Fig. 8 with good agreement. Experimental data is available up to 1.82° angle of attack [49], thus only computational results for up to 2° angle of

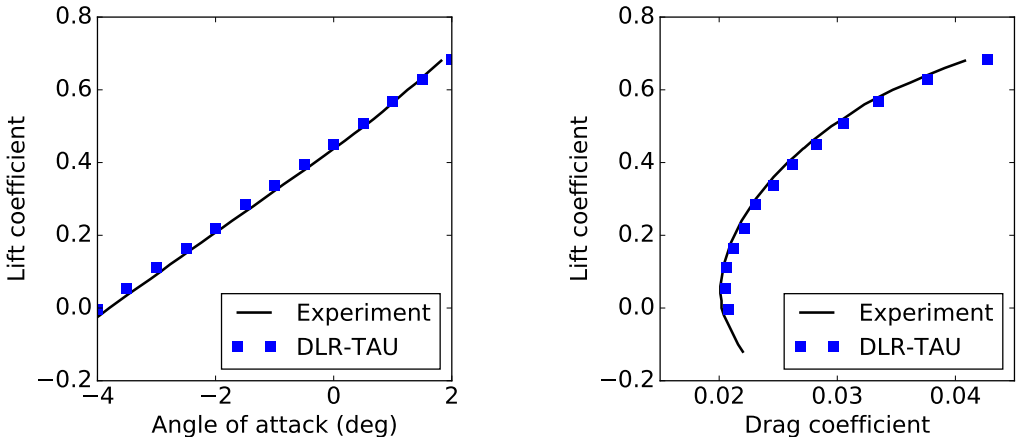

Figure 8: Lift curve and drag polar of DLR-F6 compared with experimental data.

attack are used for comparison. Since the focus of this work is on solving the stiff adjoint equation at off-design condition rather than the accuracy of the nonlinear flow solution. We assume the flow solution for 4.5° angle of attack is reasonably representative of the flows one would expect at such off-design conditions and compute its adjoint anyway.

The convergence performance for the adjoint solver to converge ten orders of magnitude is presented in Fig. 9 where the CPU time is plotted against the numbers of Krylov vectors. For 0° angle of attack GCRO-DR converged on average twice as fast as GMRES. The improvement is much more significant at 4.5° angle of attack. In contrast to GMRES which requires a minimum of 400 Krylov vectors to converge within five hours of CPU time, GCRO-DR converges in less than two hours with only 80 vectors.

All calculations, both flow and adjoint, are performed on 48 cores. The memory overhead for various solvers and CPU time information with their respective optimal options, regarding both CPU time and memory require-

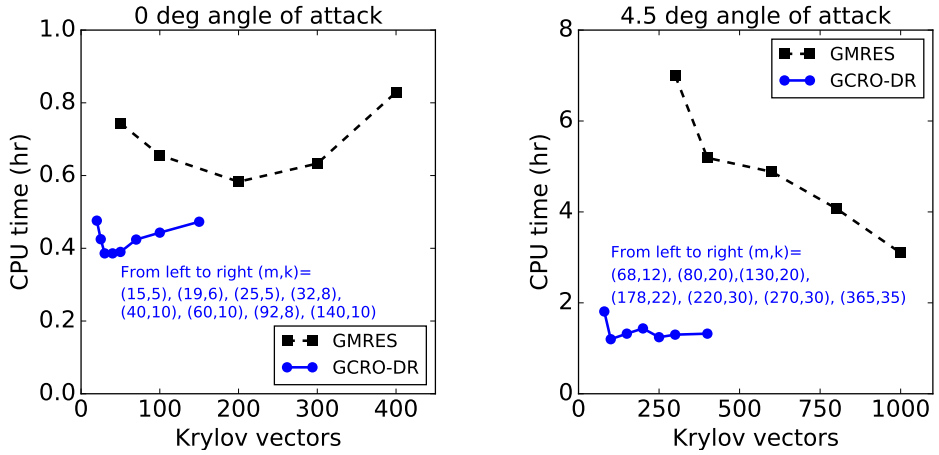

Figure 9: Adjoint convergence behaviour of DLR-F6 at 0° and 4.5° angle of attack.

Table 1: Solver performance for case 2a (2.4 million mesh points)

Solver		Memory (GB)				Cores	CPU time (hr)
		Other	JAC	ILU(0)	Krylov		
Flow		5	—	—	—	48	2.5
Adj	GMRES(200)	—	34.5	34.5	23.0	48	0.58
	GCRO-DR(40,10)	—	34.5	34.5	5.7	48	0.39

Table 2: Solver performance for case 2b (2.4 million mesh points)

Solver		Memory (GB)				Cores	CPU time(hr)
		Other	JAC	ILU(0)	Krylov		
Flow		5	—	—	—	48	4
Adj	GMRES(1000)	—	34.5	34.5	115	48	3.1
	GCRO-DR(80,20)	—	34.5	34.5	11.5	48	1.2

ment, are shown in Tables 1 and 2, illustrating the significant reduction of the memory overhead and convergence acceleration, especially for the high angle of attack condition.

5.3. Half wing-body configuration near transonic buffet

The third test case is another half wing-body model. The flow condition for this case is just below the onset of transonic shock buffet. The main flow parameters are a freestream Mach number of 0.8, a chord Reynolds number of 3.75 million and an angle of attack of 3°. This configuration has recently been investigated both numerically and experimentally to study the shock buffet phenomenon [50, 51, 52]. The model has a span of 1.10 m and a mean aerodynamic chord of about 0.279 m. The local chord lengths corresponding to the centre line and wing tip are 0.592 m and 0.099 m, respectively. The wing is twisted, tapered and has a constant sweep angle of 25°. The mesh with 2.7 million grid points is the same as the one used in a previous study [50].

The JST scheme with scalar artificial dissipation, instead of the default matrix dissipation option, was applied to evaluate the inviscid fluxes of the mean flow equations, due to the significant convergence difficulty when the latter was used. Multigrid is not used. To converge the density residual by ten orders of magnitude, 5000 iterations were taken at angle of attack of 1° and 2°, respectively, before another 71636 iterations were taken at 3° angle of attack, with a constant CFL number of 30. The flow solver converged in 1.39 hours on 144 cores. The converged flow solution is shown in Fig. 10 along with the convergence history of both the density residual and the lift coefficient.

The CPU time for converging the adjoint equation by ten orders of magnitude using different solvers is plotted against the number of Krylov vectors in Fig. 11. The trough-to-trough CPU time ratio is around two, while GCRO-

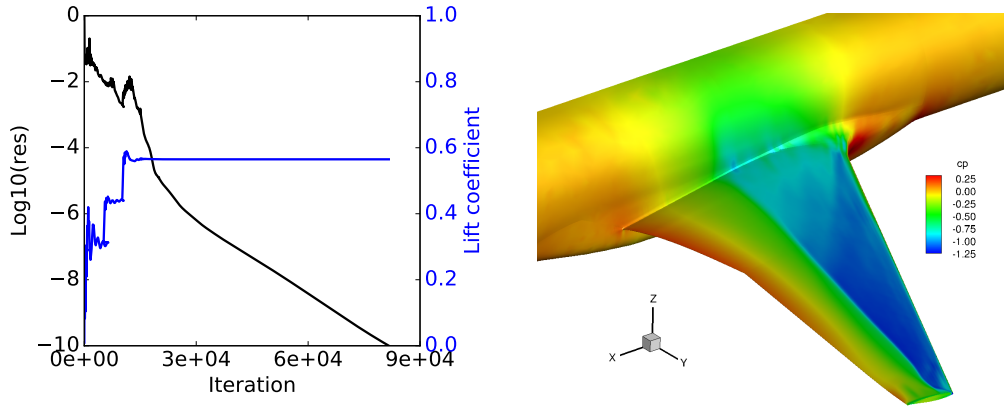

Figure 10: Convergence history of the flow solver and pressure coefficient at 3° angle of attack of half wing-body configuration.

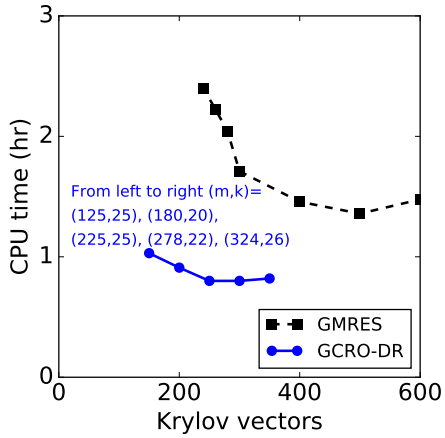

Figure 11: Adjoint convergence behaviour of pre-buffet half wing-body configuration at 3° angle of attack.

DR needs to store only half as many Krylov vectors compared to GMRES. The adjoint solution is computed on 144 cores. The memory required is 104 GB for the fastest GCRO-DR(180,20) compared with 143 GB for the best performing GMRES(500) as shown in Table 3.

Table 3: Solver performance for case 3 (2.7 million mesh points)

Solver		Memory (GB)				Cores	CPU time (hr)
		Other	JAC	ILU(0)	Krylov		
Flow		5.6	—	—	—	144	1.39
Adj	GMRES(500)	—	39	39	65	144	1.36
	GCRO-DR(180,20)	—	39	39	26	144	0.91

5.4. Civil aircraft configuration

The fourth test case is a civil aircraft wing-body-pylon-nacelle configuration with both horizontal and vertical tail planes. The engine uses a flow-through boundary condition. The mesh used has about 3 million grid points in total with 42 prism layers off the viscous walls. The flow is computed at a freestream Mach number and Reynolds number that are typical of a transonic cruise condition. The angle of attack is set to around 3° to achieve the target lift coefficient C_{L0}.

A '4V' multigrid scheme is used for convergence acceleration with CFL numbers of 5 for the fine grid and 1 for all three coarse grid levels. The density residual is reduced by five orders of magnitude and is believed to have entered the asymptotic convergence regime. The surface pressure coefficient of the civil aircraft configuration is shown in Fig. 12 along with the convergence history of the flow solver.

The adjoint solution is computed for the converged flow field. A few representative convergence curves are shown in Fig. 13 for both linear solvers with different numbers of Krylov vectors. At least 400 Krylov vectors are needed for GMRES to converge by ten orders of magnitude within two hours, while GCRO-DR with a total of only 50 vectors converges in a third of an hour. In addition, the memory saved due to the reduced number of vectors

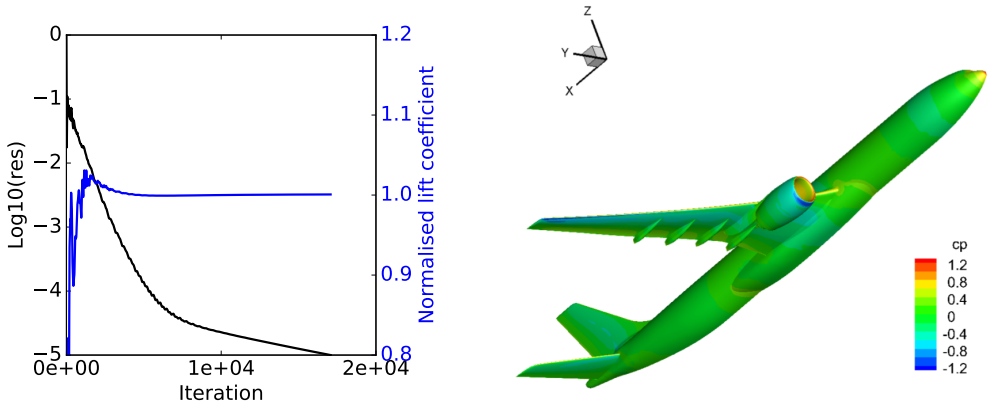

Figure 12: Left: convergence history of the flow solver and lift coefficient normalised against C_{L0}. Right: surface pressure coefficient of the civil aircraft configuration at 3° angle of attack.

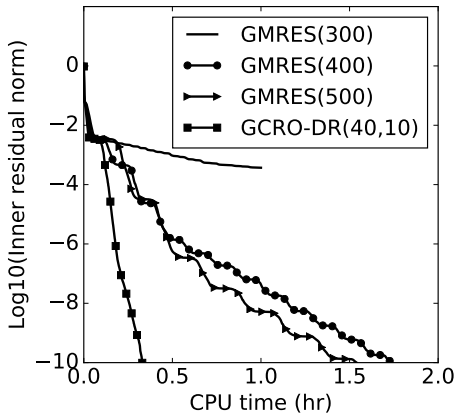

Figure 13: Adjoint convergence histories of civil aircraft configuration at 3° angle of attack.

is nearly one order of magnitude. All calculations, both flow and adjoint, are performed on 144 cores. Statistics regarding the memory breakdown and CPU time is shown in Table 4.

Table 4: Solver performance for case 4 (3.0 million mesh points)

Solver		Memory (GB)				Cores	CPU time (hr)
		Other	JAC	ILU(0)	Krylov		
Flow		6.25	—	—	—	144	0.52
Adj	GMRES(400)	—	43	43	57.5	144	1.74
	GCRO-DR(40,10)	—	43	43	7.2	144	0.33

5.5. DLR-F11 high-lift configuration

The DLR-F11 high-lift configuration is used in the second AIAA CFD High Lift Prediction Workshop. This model is a wide-body Airbus-type research configuration with a half span of 1.4 m. In this work configuration 4 is used featuring a slat at 26.5° and flap at 32.0° with both slat and flap track fairings included. The computational mesh has a total of 32.4 million grid points with 1.4 million surface elements. The boundary layer is captured using 23 layers of prismatic elements above the viscous walls. The Reynolds number used is 15.1 million and the freestream Mach number is 0.175. Three angles of attack of 0, 10 and 20° are considered for the nonlinear flow solutions, while the adjoint solution is only computed for angle of attack of 0° (case 5a) and 20° (case 5b).

At 0° angle of attack, the uniformly initialized flow is first iterated with CFL number of 1 for 2000 iterations before switching to '4V' multigrid with CFL numbers of 20 and 1.8 for fine and coarse grids, respectively. After about 30,000 iterations, the convergence seems to have entered limit-cycle oscillation (LCO) at residual level of about 10^{-6}. This turns out to be a transient behaviour. After additional 100,000 iterations with a CFL number of 10 on the fine grid and 1.8 for coarse grids, the tolerance of 10^{-8} is reached. Three-dimensional high-lift configurations often exhibit some transient con-

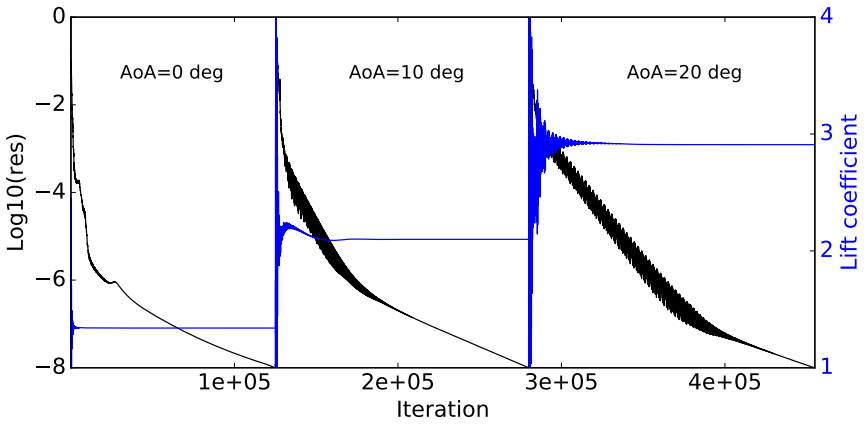

Figure 14: Convergence history of the three nonlinear flow solves and the evolution of the lift coefficient.

vergence stall at low residual level (in this case 10^{-6}) while global integral values such as lift and drag are far from fully converged. A tighter residual tolerance is thus needed for an accurate resolution of the flow physics. Similar findings on the necessity of converging the flow to very low residual level are repoted in [53]. Similar procedure is taken for finding the flow solutions for 10 and 20° angle of attack, which are both initialised using the converged solutions at lower angle of attack. The entire convergence history for the angle of attack sweep of 0°, 10° and 20° is shown in Fig. 14 for both density residual and lift coefficient. The convergence history becomes more and more oscillatory with increased angle of attack, presumably related to the more and more pronounced flow separation The skin-friction lines and pressure coefficient contour plots for 0° and 20° angle of attack are shown in Fig. 15. The lift and drag coefficients are compared with the wind tunnel data in Fig. 16 with reasonable agreement.

The adjoint solutions for both 0 and 20° angle of attack can be converged

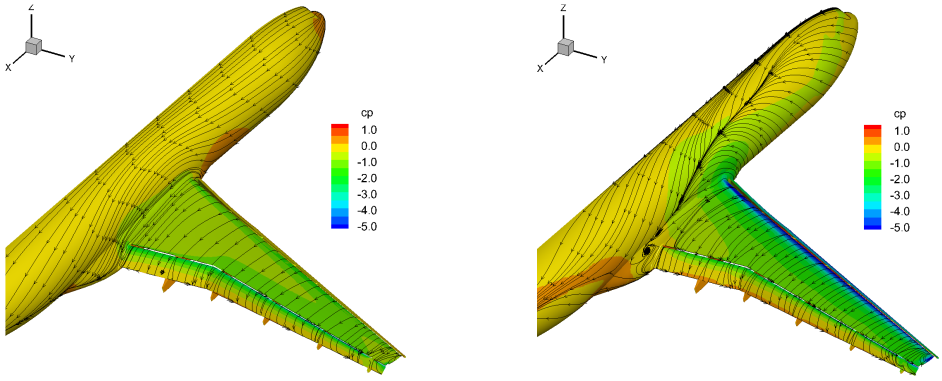

Figure 15: Surface pressure coefficient and skin-friction lines of DLR-F11 at 0° and 20° angle of attack.

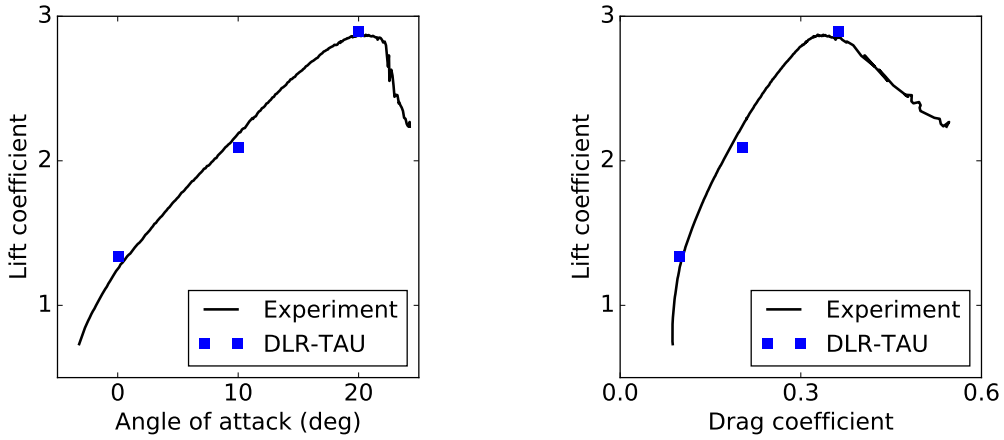

Figure 16: Lift and drag coefficient of DLR-F11 compared with experimental data.

using GCRO-DR while GMRES fails to converge with the settings imposed by available computing resources. The ILU(0)-preconditioned GMRES solver using 200 Krylov vectors stagnated after a few hundred iterations at a high residual level for both cases. As mentioned in the introduction, for a case with similar mesh size but under more benign flow condition in [24], GMRES(200) is able to converge efficiently only when it is preconditioned with ILU(2),

Table 5: Solver performance for case 5a (32.4 million mesh points), DNC=did not converge

Solver		Memory (GB)				Cores	CPU time (hr)
		Other	JAC	ILU(0)	Krylov		
Flow		67.5	—	—	—	144	20.5
Adj	GMRES(200)	—	466.5	466.5	311.0	192	DNC
	GCRO-DR(120,20)	—	466.5	466.5	217.0	192	16.0

Table 6: Solver performance for case 5b (32.4 million mesh points), DNC=did not converge

Solver		Memory (GB)				Cores	CPU time (hr)
		Other	JAC	ILU(0)	Krylov		
Flow		67.5	—	—	—	144	28.4
Adj	GMRES(200)	—	466.5	466.5	311.0	192	DNC
	GCRO-DR(140,20)	—	466.5	466.5	248.8	192	12.5

resulting in a total memory overhead of 2 TB (J.R.R.A. Martins, personal communication, September 19, 2016). Had ILU(2) been used in our DLR-F11 case, GMRES(200) might have been able to converge as well. However, in that case, 4 TB of memory would have been required as the hybrid mesh used in our case results in much denser Jacobian and ILU matrices. Yet, the computational resources available in this work are limited to 192 cores with 6 GB of memory each, with a total of 1152 GB of memory only. After storing the Jacobian and ILU(0) matrices and the auxiliary variables associated with the mesh metrics, a maximum number of 200 Krylov vectors can be used.

For 0° angle of attack, GCRO-DR(120,20) converges ten orders of magnitude in 16 hours, while the adjoint solution at 20° angle of attack is converged with GCRO-DR(140,20) in 12.5 hours. Statistics regarding the memory breakdown and CPU time are shown in Tables 5 and 6.

6. Conclusions

In this paper, a generalized conjugate residual solver with deflated restarting (GCRO-DR) is applied to the solution of the discrete adjoint equation for typical aeronautical applications under challenging flow conditions. The deflation technique recycles a few interior eigenvectors from the Arnoldi cycle to improve the convergence of the subsequent restarted cycle, which in turn accelerates the convergence of the linear system. Significant improvement over the baseline Krylov solver, generalized minimal residual (GMRES), regarding both the memory requirement and CPU time, is demonstrated, highlighting the potential of the proposed method for enabling aerodynamic design optimisation over the whole flight envelope.

The cases investigated include a two-dimensional turbulent transonic aerofoil, two turbulent transonic half wing-body configurations at both low and high angles of attack including a pre-shock-buffet point, a large civil aircraft with engine-pylon-nacelle and both horizontal and vertical tail planes at near cruise condition, and finally a high lift configuration with deployed slat/flap settings. These test cases present a wide variety of flow phenomena ranging from design to off-design conditions. Memory requirement for the entire adjoint solver is reduced by up to a factor of two with up to three times speedup in convergence. In addition, the improvement is particularly pronounced for the more difficult cases.

A second deflated Krylov solver, generalised conjugate residual with optimal truncation (GCROT), is presented to provide a comprehensive overview of the state-of-the-art sparse iterative solvers. However, the convergence improvement, although still better than GMRES, is less significant compared

to GCRO-DR for the cases tested in this work. It warrants further investigation to better assess the performance of GCROT vs. GCRO-DR, including the effect of more fill-in in the ILU preconditioner, different mesh types and flow conditions.

7. Acknowledgements

The research leading to these results is co-funded by Innovate UK, the UK's innovation agency, as part of the Enhanced Fidelity Transonic Wing project. The authors wish to thank Dr John Pattinson and Dr Stefan Melber-Wilkending for the meshes in cases 4 and 5. We also thank Professor Eric de Sturler, Professor David Zingg, Professor Jason Hicken, Professor Joaquim R.R.A. Martins and Dr Gaetan Kenway for the helpful discussions.

References

[1] C. Othmer. Adjoint methods for car aerodynamics. *Journal of Mathematics in Industry*, 4(1):1–23, 2014.

[2] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders. Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation. *AIAA paper*, 94, 1996.

[3] M.B. Giles and N.A. Pierce. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 65(3-4):393–415, 2000.

[4] J. Brezillon and N. Gauger. 2D and 3D aerodynamic shape optimisation using the adjoint approach. *Aerospace Science and Technology*, 8(8):715–727, 2004.

[5] N. Kroll, N.R. Gauger, J. Brezillon, R. Dwight, A. Fazzolari, D. Vollmer, K. Becker, H. Barnewitz, V. Schulz, and S. Hazra. Flow simulation and shape optimization for aircraft design. *Journal of Computational and Applied Mathematics*, 203(2):397–411, 2007.

[6] M.S. Campobasso, M.C. Duta, and M.B. Giles. Adjoint calculation of sensitivities of turbomachinery objective functions. *Journal of Propulsion and Power*, 19(4):693–703, 2003.

[7] D. Wang and L. He. Adjoint aerodynamic design optimization for blades in multistage turbomachinespart I: Methodology and verification. *Journal of Turbomachinery*, 132(2):021011, 2010.

[8] D. Wang, L. He, Y. Li, and R. Wells. Adjoint aerodynamic design optimization for blades in multistage turbomachines–part II: Validation and application. *Journal of Turbomachinery*, 132(2):021012, 2010.

[9] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260, 1988.

[10] A. Jameson, L. Martinelli, and N.A. Pierce. Optimum aerodynamic design using the Navier–Stokes equations. *Theoretical and Computational Fluid Dynamics*, 10(1-4):213–237, 1998.

[11] W.K. Anderson and V. Venkatakrishnan. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers & Fluids*, 28(4):443–480, 1999.

[12] E.J. Nielsen, J. Lu, M.A. Park, and D.L. Darmofal. An implicit, exact

dual adjoint solution method for turbulent flows on unstructured grids. *Computers & Fluids*, 33(9):1131–1155, 2004.

[13] D.J. Mavriplis. Multigrid solution of the discrete adjoint for optimization problems on unstructured meshes. *AIAA Journal*, 44(1):42–50, 2006.

[14] D.J. Mavriplis. Discrete adjoint-based approach for optimization problems on three-dimensional unstructured meshes. *AIAA Journal*, 45(4):741–750, 2007.

[15] T. Nambu, D.J. Mavriplis, and K. Mani. Adjoint-based shape optimization of high-lift airfoils using the NSU2D unstructured mesh solver. In *52nd Aerospace Sciences Meeting*, page 0554, 2014.

[16] J. Brezillon, R.P. Dwight, and M. Widhalm. Aerodynamic optimization for cruise and high-lift configurations. In *MEGADESIGN and MegaOpt-German Initiatives for Aerodynamic Simulation and Optimization in Aircraft Design*, pages 249–262. Springer, 2009.

[17] G. Shroff and H. Keller. Stabilization of unstable procedures: the recursive projection method. *SIAM Journal on Numerical Analysis*, 30(4):1099–1120, 1993.

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

[19] R.P. Dwight and J. Brezillon. Effect of approximations of the discrete adjoint on gradient-based optimization. *AIAA Journal*, 44(12):3022–3031, 2006.

[20] D.A. Knoll and D.E. Keyes. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[21] T.T. Chisholm and D.W. Zingg. A Jacobian-free Newton–Krylov algorithm for compressible turbulent fluid flows. *Journal of Computational Physics*, 228(9):3490–3507, 2009.

[22] L. Osusky, H. Buckley, T. Reist, and D.W. Zingg. Drag minimization based on the Navier–Stokes equations using a Newton–Krylov approach. *AIAA Journal*, 53(6):1555–1577, 2015.

[23] J.E. Hicken. Output error estimation for summation-by-parts finite-difference schemes. *Journal of Computational Physics*, 231(9):3828–3848, 2012.

[24] Z. Lyu, G.K. Kenway, and J.R.R.A. Martins. Aerodynamic shape optimization investigations of the common research model wing benchmark. *AIAA Journal*, 53(4):968–985, 2014.

[25] E. de Sturler. Truncation strategies for optimal Krylov subspace methods. *SIAM Journal on Numerical Analysis*, 36(3):864–889, 1999.

[26] J.E. Hicken and D.W. Zingg. A simplified and flexible variant of GCROT for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 32(3):1672–1694, 2010.

[27] M.L. Parks, E. de Sturler, G. Mackey, D. Johnson, and S. Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5):1651–1674, 2006.

[28] S. Xu, S. Timme, and K.J. Badcock. Enabling off-design linearised aerodynamics analysis using Krylov subspace recycling technique. *Computers & Fluids*, 140:385–396, 2016.

[29] D. Schwamborn, T. Gerhold, and R. Heinrich. The DLR TAU-code: Recent applications in research and industry. In *ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics, Egmond aan Zee, The Netherlands, September 5-8, 2006*.

[30] T. Gerhold. Overview of the hybrid RANS code TAU. In *MEGAFLOW-Numerical Flow Simulation for Aircraft Design*, pages 81–92. Springer, 2005.

[31] A. Jameson, W. Schmidt, and E. Turkel. Numerical solutions of the Euler equations by finite volume methods using Runge–Kutta time-stepping schemes. AIAA paper 1981-1259, 1981.

[32] R.C. Swanson and E. Turkel. On central-difference and upwind schemes. *Journal of Computational Physics*, 101(2):292–306, 1992.

[33] S.R. Allmaras, F.T. Johnson, and P.R. Spalart. Modifications and clarifications for the implementation of the Spalart–Allmaras turbulence model. In *Seventh International Conference on Computational Fluid Dynamics (ICCFD7)*, pages 1–11, 2012.

[34] P.L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372, 1981.

[35] S. Yoon and A. Jameson. Lower-upper symmetric-Gauss–Seidel method

for the Euler and Navier–Stokes equations. *AIAA Journal*, 26(9):1025–1026, 1988.

[36] E. de Sturler. Nested Krylov methods based on GCR. *Journal of Computational and Applied Mathematics*, 67(1):15–41, 1996.

[37] P.F. Fischer. An overlapping Schwarz method for spectral element solution of the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 133(1):84–101, 1997.

[38] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' guide*, volume 9. SIAM, 1999.

[39] M. Osusky. *A parallel Newton–Krylov–Schur algorithm for the Reynolds-averaged Navier–Stokes equations*. PhD thesis, University of Toronto, 2013.

[40] A. Pueyo and D.W. Zingg. Efficient Newton–Krylov solver for aerodynamic computations. *AIAA Journal*, 36(11):1991–1997, 1998.

[41] R. Dwight and J. Brezillon. Efficient and robust algorithms for solution of the adjoint compressible Navier–Stokes equations with applications. *International Journal for Numerical Methods in Fluids*, 60(4):365–389, 2009.

[42] A. McCracken, A. Da Ronch, S. Timme, and K.J. Badcock. Solution of linear systems in Fourier-based methods for aircraft applications. *International Journal of Computational Fluid Dynamics*, 27(2):79–87, 2013.

[43] P. Ghysels, T.J. Ashby, K. Meerbergen, and W. Vanroose. Hiding global communication latency in the GMRES algorithm on massively parallel machines. *SIAM Journal on Scientific Computing*, 35(1):C48–C71, 2013.

[44] D.E. Keyes. Aerodynamic applications of Newton–Krylov–Schwarz solvers. In *Fourteenth International Conference on Numerical Methods in Fluid Dynamics*, pages 1–20. Springer, 1995.

[45] Y. Saad and M. Sosonkina. Distributed Schur complement techniques for general sparse linear systems. *SIAM Journal on Scientific Computing*, 21(4):1337–1356, 1999.

[46] M. Mohiyuddin, M. Hoemmen, J. Demmel, and K. Yelick. Minimizing communication in sparse matrix solvers. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, page 36. ACM, 2009.

[47] T.A. Reist and D.W. Zingg. High-fidelity aerodynamic shape optimization of a lifting-fuselage concept for regional aircraft. *Journal of Aircraft*, 2016.

[48] K.R. Laflin, S.M. Klausmeyer, T. Zickuhr, J.C. Vassberg, R.A. Wahls, J.H. Morrison, O.P. Brodersen, M.E. Rakowitz, E.N. Tinoco, and J.-L. Godard. Data summary from second AIAA computational fluid dynamics drag prediction workshop. *Journal of Aircraft*, 42(5):1165–1178, 2005.

[49] C.-C. Rossow, J.-L. Godard, H. Hoheisel, and V. Schmitt. Investigations

of propulsion integration interference effects on a transport aircraft configuration. *Journal of Aircraft*, 31(5):1022–1030, 1994.

[50] F. Sartor and S. Timme. Reynolds-averaged Navier–Stokes simulations of shock buffet on half wing-body configuration. *AIAA Paper 2015–1939*, 2015.

[51] F. Sartor and S. Timme. Delayed detached-eddy simulation of shock buffet on half wing-body configuration. *accepted for publication in AIAA Journal*, 2016.

[52] S. Lawson, D. Greenwell, and M.K. Quinn. Characterisation of buffet on a civil aircraft wing. *AIAA paper 2016-1309*, 2016.

[53] S. Langer. Agglomeration multigrid methods with implicit Runge–Kutta smoothers applied to aerodynamic simulations on unstructured grids. *Journal of Computational Physics*, 277:72–100, 2014.