European Congress on Computational Methods
in Applied Sciences and Engineering (ECCOMAS 2012)
J. Eberhardsteiner et.al. (eds.)
Vienna, Austria, September 10-14, 2012

ACCELERATING CONVERGENCE OF THE CFD LINEAR FREQUENCY DOMAIN METHOD BY A PRECONDITIONED LINEAR SOLVER

Andrew J. McCracken[1], **Sebastian Timme**[1], **and Kenneth J. Badcock**[1]

[1]School of Engineering, University of Liverpool,
Liverpool, L69 3GH
e-mail: A.J.McCracken@Liv.ac.uk

Keywords: Unsteady Aerodynamics, Linear Frequency Domain, Preconditioning, Implicit Methods

Abstract. *The Linear Frequency Domain (LFD) solver has found wide-spread use for flutter and flight dynamics studies. The LFD solver was available in the DLR (German Aerospace Center) TAU-code using an explicit Runge-Kutta approach or a semi-implicit LU-SGS approach for solving the linear system arising in the formulation. This work presents an alternative using a Krylov subspace solver with Incomplete Lower-Upper (ILU) preconditioning. The alternative option is tested for an inviscid NACA 0012 and a viscous NACA 64A010 two-dimensional aerofoil case, along with an inviscid and a viscous Goland wing case. It is shown that the new approach provides an order of magnitude improvement in solution time over the current methods. Preconditioners based on approximate Jacobians are also considered using a combination of Jacobian matrix terms from both the first and second-order spatial schemes. This is shown to improve the convergence of the linear solver by up to a factor of five compared to the preconditioner based on the Jacobian of the first-order spatial scheme. The optimum weighting of the preconditioner terms is case independent.*

1 INTRODUCTION

Unsteady aerodynamic problems often involve a periodic oscillation which, using time-accurate computational fluid dynamics (CFD) solvers, requires iterating through an initial transient before the periodic response is obtained. Frequency domain methods have become more prominent as they allow direct calculation of the periodic state. One such method is the Linear Frequency Domain (LFD) method. LFD was originally developed from the linearised Euler method in [6] for use in turbomachinery problems and was implemented in the DLR-TAU code [5] by Widhalm et al. in [11] which is the version used in this work.

Fully-implicit solution methods can offer improved convergence properties compared to explicit or semi-implicit methods. The motivation for this work was to accelerate the time to solution of the LFD solver in TAU. The problem arising from the fully-implicit formulation is the need to have an efficient linear solver and, demanding an effective preconditioner if Krylov methods are used. The Jacobian matrices encountered in an exact second-order spatial discretisation scheme along with unstructured meshes are usually very poorly conditioned and are very stiff. This increases the importance of the preconditioner in the solve. A preconditioner is presented that fulfills this need, improving the conditioning of the system whilst remaining robust across a vast number of test cases and conditions.

This paper continues with a description of the LFD method and the preconditioning used. Results are then presented for a number of cases showing the speed up achieved with the fully-implicit solver and a brief analysis of the preconditioner in both serial and parallel.

2 FORMULATION

2.1 Linear Frequency Domain

The LFD solver is based upon the assumptions of periodicity and small amplitude oscillations in order to linearise an unsteady motion about a steady mean state. Firstly, the governing flow equations are written in semi-discrete form as

$$\frac{\partial \mathbf{w}}{\partial t} + \mathbf{R}(\mathbf{w}, \mathbf{u}, \dot{\mathbf{u}}) = 0. \tag{1}$$

The conservative flow variables $\mathbf{w}$ and grid positions $\mathbf{u}$, are then modelled as a steady mean component plus a small perturbation based on the small amplitude assumption as

$$\mathbf{w}(t) = \bar{\mathbf{w}} + \tilde{\mathbf{w}}(t), \quad \mathbf{u}(t) = \bar{\mathbf{u}} + \tilde{\mathbf{u}}(t). \tag{2}$$

The perturbations are then assumed periodic in order to expand these as a Fourier series in terms of the circular frequency ω. Combining this with Eqs. 1 and 2, gives the complex valued system

$$\left\{ ik\omega I + \frac{\partial \mathbf{R}}{\partial \mathbf{w}} \right\} \hat{\mathbf{w}}_k = -\frac{\partial \mathbf{R}}{\partial \mathbf{u}} \hat{\mathbf{u}}_k - ik\omega \frac{\partial \mathbf{R}}{\partial \dot{\mathbf{u}}} \hat{\mathbf{u}}_k, \tag{3}$$

where the $\hat{}$ accent indicates a vector of Fourier coefficients. Limiting interest to the perturbations which are harmonic in the forced frequency, k is taken to be 1, and hence the non-linear Eq.1 has been reduced to a single linear equation. The right hand side is obtained from a finite difference calculation of the residuals at the extremes of the small amplitude oscillation.

For solution with an implicit linear solver, Eq. 3 must be written as a linear system of the form $A\mathbf{x} = \mathbf{b}$. To allow this, the real and imaginary parts in Eq. 3 are taken to form two coupled

real systems:

$$\begin{aligned} -\omega\hat{\mathbf{w}}_{Im} + \frac{\partial \mathbf{R}}{\partial \mathbf{w}}\hat{\mathbf{w}}_{Re} &= -\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\hat{\mathbf{u}}_{Re} + \omega\frac{\partial \mathbf{R}}{\partial \dot{\mathbf{u}}}\hat{\mathbf{u}}_{Im}, \\ \omega\hat{\mathbf{w}}_{Re} + \frac{\partial \mathbf{R}}{\partial \mathbf{w}}\hat{\mathbf{w}}_{Im} &= -\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\hat{\mathbf{u}}_{Im} - \omega\frac{\partial \mathbf{R}}{\partial \dot{\mathbf{u}}}\hat{\mathbf{u}}_{Re}. \end{aligned} \tag{4}$$

The linear system components are then written as

$$A = \begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \mathbf{w}} & -\omega I \\ \omega I & \frac{\partial \mathbf{R}}{\partial \mathbf{w}} \end{bmatrix}, \mathbf{x} = \begin{pmatrix} \hat{\mathbf{w}}_{Re} \\ \hat{\mathbf{w}}_{Im} \end{pmatrix}, \text{and } \mathbf{b} = \begin{bmatrix} -\frac{\partial \mathbf{R}}{\partial \mathbf{u}} & \omega\frac{\partial \mathbf{R}}{\partial \dot{\mathbf{u}}} \\ -\omega\frac{\partial \mathbf{R}}{\partial \dot{\mathbf{u}}} & -\frac{\partial \mathbf{R}}{\partial \mathbf{u}} \end{bmatrix} \begin{pmatrix} \hat{\mathbf{u}}_{Re} \\ \hat{\mathbf{u}}_{Im} \end{pmatrix}. \tag{5}$$

2.2 Solver Options

There are a few options available in TAU for the solution of the linear system in LFD. The first is termed Facemat and is the default option. This is a face-based approach for the formation of the matrices where the flow variables are stored at each face of the grid rather than at the vertices. The linear system is recast as

$$\tilde{A}\mathbf{x} = A\mathbf{x} - \mathbf{b}, \tag{6}$$

where $\tilde{A}$ is an approximation of A and the matrix-vector product $\tilde{A}\mathbf{x}$ is driven to an L2 norm of zero using either the semi-implicit LU-SGS iterative solver [3] or an explicit Runge-Kutta scheme with Multigrid [7] to accelerate the convergence. The Facemat option can also use a GMRes [10] Krylov solver.

This method only operates on the matrix-vector product and never stores the full Jacobian matrix explicitly in memory. This minimises the memory requirement to enable very large grids to be run on relatively inexpensive machines and gives this approach a competitive edge over other linear solvers in this sense.

The solver introduced in this work makes use of a Generalised Conjugate Residual (GCR) Krylov subspace solver [4]. The implementation works with a block matrix structure rather than element-wise so that the memory required for matrix storage is minimised. The solver uses a blocked version of the Incomplete Lower-Upper (ILU) preconditioner [9].

The GCR solver is a Krylov subspace method whereby the system is projected onto a subspace

$$\mathcal{K}_m\left(A, \mathbf{r}_0\right) = span\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, ..., A^{m-1}\mathbf{r}_0\}, \tag{7}$$

where m is the number of subspace vectors allocated in advance and the residual $\mathbf{r}_0$ is the term to be minimised. Increasing m leads to a better approximation of the problem in the subspace and as such, the better the convergence although this comes at a cost of memory.

2.3 Preconditioning

The most important consideration for an effective linear solver with respect to convergence is the preconditioner employed. Methods of preconditioning have been the focus of a number of papers summarised in [2]. The majority of the techniques when applied to CFD problems are primarily focussed on using properties of the given matrix to carry out the permutation of terms to improve the conditioning. Very few papers make use of properties of the underlying problem to improve the preconditioner performance. In [8, 12], the preconditioner is based upon an approximate Jacobian matrix and is shown to accelerate the convergence over using

the exact Jacobian. This work makes use of approximate Jacobians to significantly accelerate the solution of linear systems arising from the CFD derived LFD problem.

The purpose of preconditioning is to make a system easier to solve, thus improving the convergence properties of a solver. For left preconditioning, the linear system, $A\mathbf{x} = \mathbf{b}$, is recast as

$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}. \tag{8}$$

It can be seen that the ideal preconditioner matrix P would be equal to A and the inverse is found exactly to allow solution in one step of an iterative solver. However, finding the inverse of the large sparse matrices encountered in CFD can be very costly in terms of both time and memory. An ILU factorisation is used to form an approximation to the inverse by limiting the number of additional non-zero terms beyond the original sparsity pattern introduced during the factorisation process described in [9]. This is referred to as ILU(k) preconditioning where k indicates the level of fill-in. In the current work one level of fill-in is used, which generates around two additional non-zero terms in the matrix for each original non-zero term.

For CFD applications, the Jacobian matrix is usually based on a second order discretisation A_2 with a preconditioner matrix P. This however, is often found to lead to a very poor preconditioner in the sense of bad convergence of the iterative solver. A heuristic fix is sometimes used to base P on the Jacobian matrix of the first-order spatial scheme A_1, which seems to improve on this situation significantly in practice. A variation on this is proposed in this work whereby both first and second order Jacobian matrix terms are used to form the preconditioner as

$$A_\alpha = \alpha A_2 + (1 - \alpha)A_1 \tag{9}$$

The weighted preconditioner matrix P_α is then formed from the ILU factorisation of the matrix A_α where the subscript α indicates the percentage of second order Jacobian terms as a decimal number.

The above preconditioning has been implemented with a GCR iterative Krylov subspace solver. The effect of the Krylov subspace method chosen is small in comparison to that of the preconditioning and as such is not tested for other types.

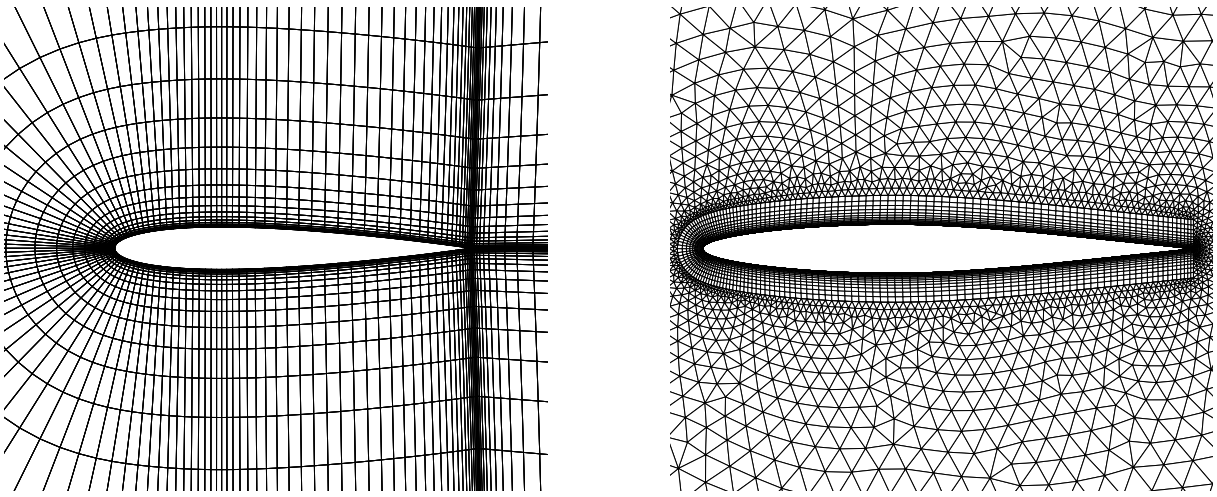
3 RESULTS

3.1 Test Cases

A number of test cases have been chosen to assess the performance of the fully implicit solver with the preconditioner. Cases chosen include both inviscid and viscous schemes in both two- and three-dimensions. Each of the cases requires a steady state solution at mean conditions which has previously been run to a residual of 1×10^{-8} which the LFD calculations were then started from. All cases are run with 20 Krylov vectors.

Two-Dimensions

The most simple of the cases presented here is the two-dimensional NACA 0012 aerofoil run for an inviscid test case at AGARD CT2 conditions. The computational grid is shown in Fig. 1(a) and has 12,672 points. The second two-dimensional case is the NACA 64A010 aerofoil run as a viscous case at AGARD CT8 conditions. The computational grid is shown in Fig. 1(b) which has 21,454 points, has an unstructured farfield with a regular boundary layer. The turbulence was modelled using a Spalart-Allmaras one equation model with Edwards' correction.



(a) NACA 0012 grid

(b) NACA 64A010 grid

Figure 1: Aerofoil computational grids

AGARD	CT2	CT8
Mach number, M	0.6	0.8
Mean incidence, α_0	3.16°	0.0°
Pitch amplitude, α_A	4.59°	0.5°
Reduced frequency, k	0.0811	0.1
Reynolds Number Re	-	12.5×10^{-6}

Table 1: AGARD cases

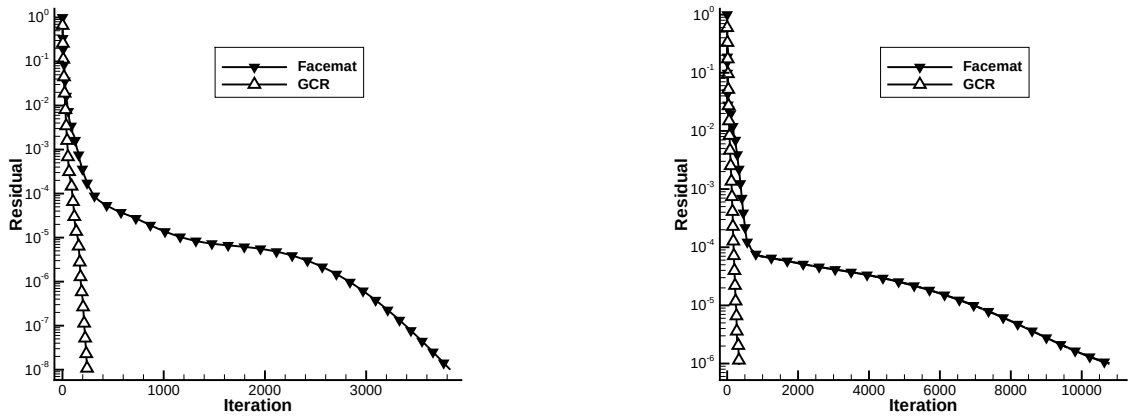
The conditions for the AGARD test cases can be found in [1] and also in Table 1.

The two aerofoils have been run with both the original Facemat formulation for solving the LFD system and with the GCR solver including the ILU preconditioning. The preconditioner uses one level of fill-in and for the NACA 0012 case is converged 8 orders of magnitude. For the NACA 64A010 the residual is converged 6 orders. The convergence for the two cases is shown in Figs. 2(a) and 2(b).

As expected, the fully-implicit implementation is considerably quicker than the Facemat formulation with a speed up in iterations of around 20 which corresponds to a speed up in time of around 15 due to the longer time per iteration for the implicit approach. The fully-implicit solver makes use of the ILU$_\alpha$ preconditioner at a value of $\alpha = 0.90$ which has been set as the default value within the solver.

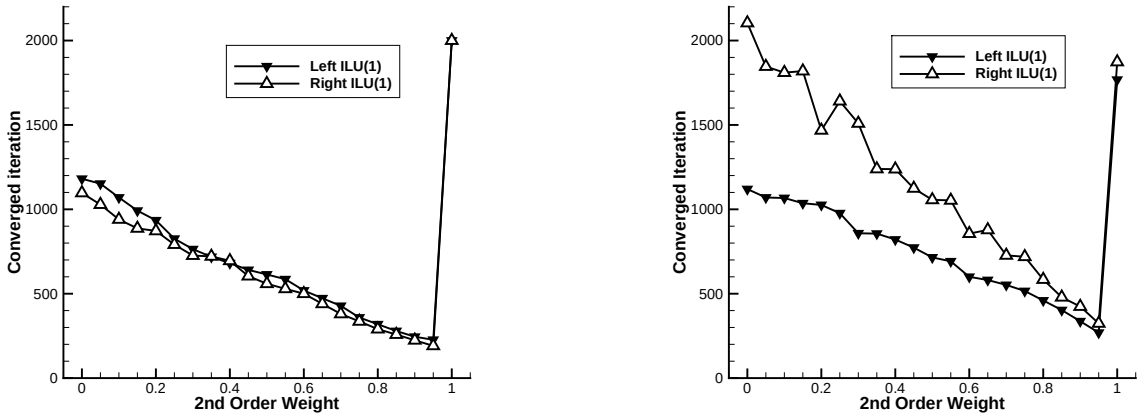
In order to analyse the effect of the weighting on the performance of the solver, both cases were run for various weightings in the preconditioner with one level of fill-in, along with left and right preconditioning. The results of this are shown in Figs. 3(a) and 3(b).

The use of a weighted preconditioner clearly shows a substantial benefit in the performance of the preconditioned Krylov solver over using the first-order preconditioner (far left) and the failure of the second-order preconditioner is seen with the maximum number of iterations reached for this case (far right). The shape of the graph is consistent for the left and right preconditioning options and for both of the aerofoil test cases. This suggests some degree of generality in the weighting. There is up to a factor of 5 improvement in the required number of iterations for convergence, as each iteration requires the same amount of time, this corresponds



(a) NACA 0012 (b) NACA 64A010

Figure 2: Solver convergence



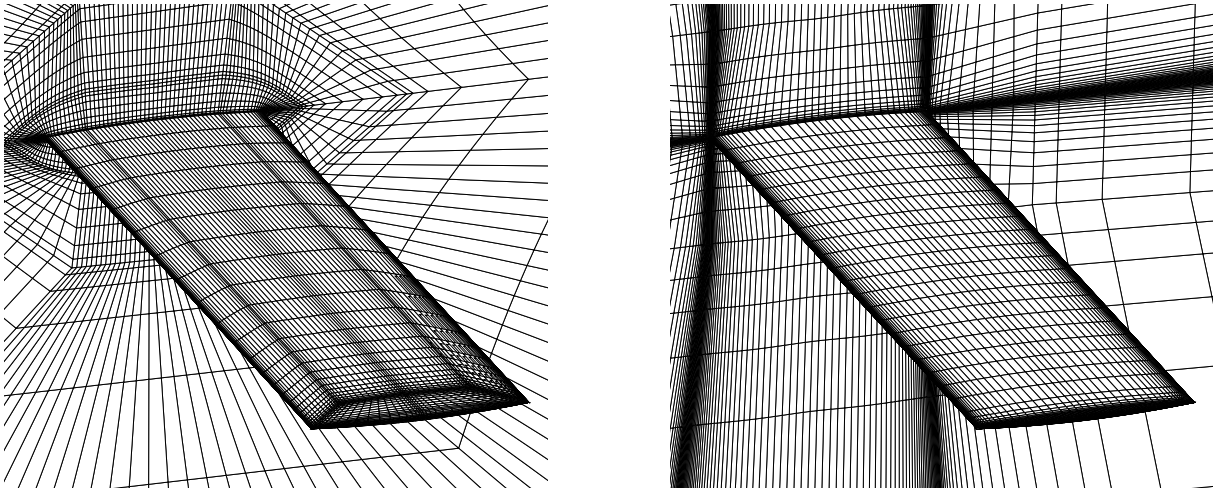
(a) NACA 0012 (b) NACA 64A010

Figure 3: Preconditioner weight convergence

to a speed up of up to 5 in time.

Three-Dimensions

The three-dimensional cases increase the complexity for the linear solver through the increased bandwidth in the Jacobian matrix from the larger stencils. The simplest used here is the Goland wing inviscid test case. This is an academic case typically used for aeroelastics analysis. The computational grid is shown in Fig. 4(a) and has 201,909 points. The Goland wing RANS case is the most complex reviewed here. The three-dimensional nature along with being a viscous computation (i.e. larger block size in the Jacobian matrix) leads to a Jacobian matrix which is very poorly conditioned and will test the effectiveness of the preconditioner. The computational grid is shown in Fig. 4(b) and has 991,075 points. As with the NACA 64A010 case, the turbulence is modelled with a Spalart-Allmaras one equation model with Edwards' correction.



(a) Goland wing Euler grid (b) Goland wing RANS grid

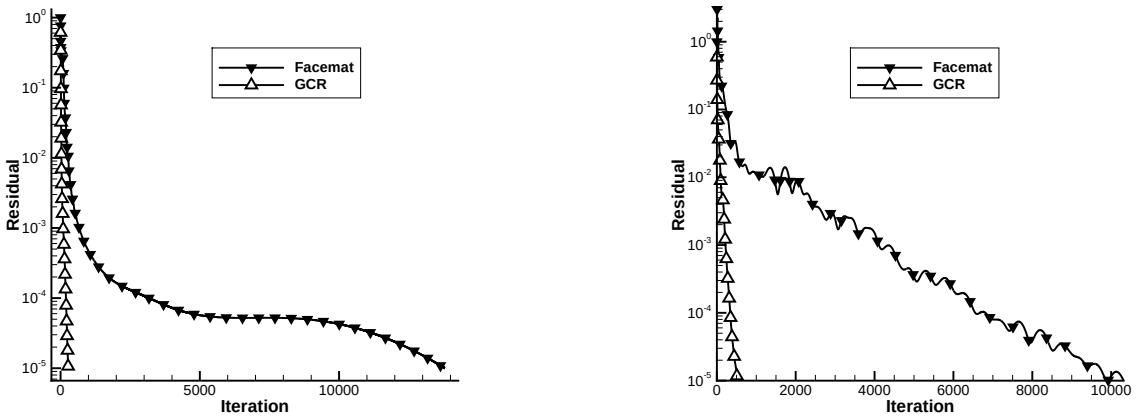
Figure 4: Three-dimensional computational grids

The conditions run for the two three-dimensional cases are given in Table 2.

	Goland (Euler)	Goland (RANS)
Mach number, M	0.8	0.925
Mean incidence, α_0	0.0°	0.0°
Pitch amplitude, α_A	1.0°	1.0°
Reduced frequency, k	0.025	0.025
Reynolds Number Re	-	15×10^{-6}

Table 2: 3D test case conditions

The two cases have been run with both the Facemat and GCR solver options and are shown in Fig. 5.

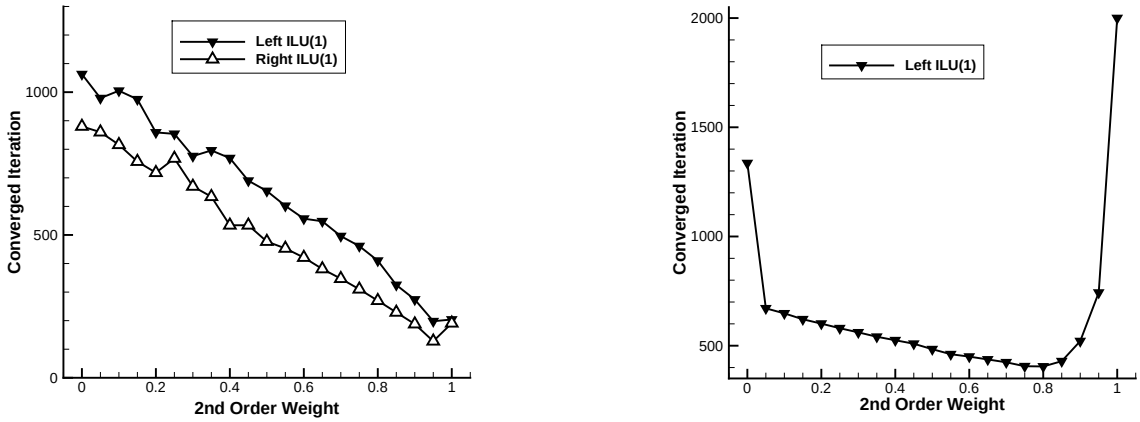


(a) Goland wing Euler (b) Goland wing RANS

Figure 5: Solver convergence

The GCR solver requires far fewer iterations to reach a converged state compared to the Facemat option. This is shown for both cases where a speed up of 50 is achieved for the Euler problem and a speed up of 20 is achieved for the RANS problem with respect to number of iterations. This corresponds to a speed up in time of around 28 and 14 respectively.

The next consideration is the effect of the preconditioner weighting on the number of iterations required for convergence. Fig. 6 shows this for the two cases.



(a) Goland wing Euler

(b) Goland wing RANS

Figure 6: Preconditioner weight convergence

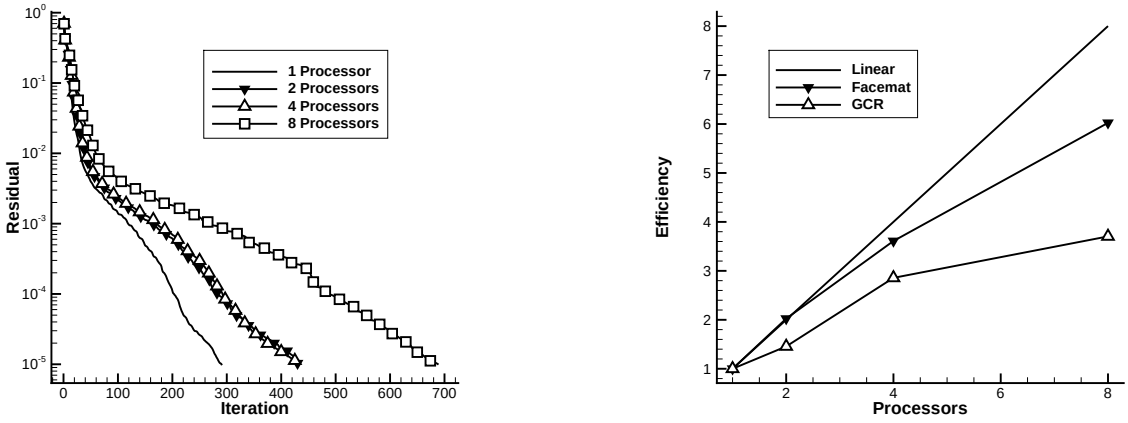
As with the two-dimensional cases, the three-dimensional Euler problem has the same graph shape indicating that increasing the amount of second-order terms in the preconditioner, improves the solver performance up to a value of $\alpha = 0.90$. On this occasion, the pure second-order preconditioner does converge but it does require more iterations than the optimum weight. For the RANS problem, this was run on a single processor, and it is seen that changing the weight improves the solver performance in line with previous results but the optimum lies at a slightly lower value of $\alpha = 0.80$.

3.2 Parallel Performance

The implementation of the ILU preconditioner has been such that it only works on the matrix local to the processor and does not communicate in the factorisation process due to complexity and time lost due to excessive communication. This will degrade the quality of the approximation of the preconditioner and is something which must be considered when running on a large test case across many processors.

The parallel performance of the preconditioner is tested on the Goland Euler case described previously with the iterations to convergence shown in Fig. 7(a) and the efficiency being shown in Fig. 7(b). The figure also contains the parallel efficiency of the Facemat option and a line indicating a 100% efficient process. It can be seen that the fully implicit method does not have the same efficiency when running in parallel compared to the Facemat option although the run times are still favourable for the fully implicit approach when running across a large number of processors.

The Goland RANS case has been tested for the effect of the number of processors on the optimum weight. This is shown in Fig. 8



(a) Parallel convergence

(b) Parallel efficiency

Figure 7: Effect of parallelisation

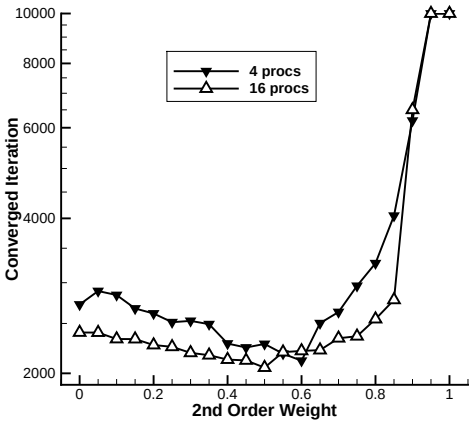


Figure 8: Effect of parallelisation on optimum weight

It is seen that increasing the number of processors used appears to move the optimum weight to the left and closer to the pure first-order preconditioner. This is possibly as a result of there being very few points on each processor with respect to the number of global points and could introduce extra effects for which the single processor weighting is not optimal.

This preconditioner has been used for a realistic production transport aircraft running very effectively across 816 cores using 1.5TB of RAM.

4 CONCLUSIONS

An alternative method for solving the CFD derived Linear Frequency Domain problem has been presented making use of an implicit linear solver. This has shown to have a speed up of more than one order of magnitude over the currently implemented methods within the TAU solver.

An alternative approach to preconditioning a GCR Krylov solver with ILU has also been presented. Mixing the first and second-order Jacobian matrix terms can have a beneficial effect on the rate of convergence of the linear solver. This has shown for serial calculations to prove

robust across several test cases in both two and three dimensions for both inviscid and viscous flows, where a speed up of up to 5 has been seen over the traditionally first-order preconditioner.

The performance of the preconditioner in parallel has also been reviewed where it has been shown that the linear solver is not perfectly scalable as expected. However, it has also been seen that the optimum weight in the preconditioner changes with the number of processors used, tending toward a pure first-order based preconditioner.

ACKNOWLEDGEMENTS

This work was supported by AIRBUS UK Ltd and the EPSRC. The author would also like to thank Marcus Widhalm and Anna Naumovich from DLR-AS Braunschweig for their input and criticism during the development of this work.

REFERENCES

[1] AGARD. R-702 compendium of unsteady aerodynamic measurements. Technical report, Advisory Group for Aerospace Research and Development - NATO, 1982.

[2] Michele Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[3] Richard Dwight. *Efficiency Improvements of RANS-Based Analysis and Optimization using Implicit and Adjoint Methods on Unstructured Grids*. PhD thesis, School of Mathematics, University of Manchester, 2006.

[4] S.C. Eisenstat, S.C. Elman, and M. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM, Journal of Numerical Analysis*, 20(2):345–357, April 1983.

[5] Thomas Gerhold. Calculation of complex three-dimensional configurations employing the dlr tau-code. In *35th AIAA Aerospace Sciences Meeting and Exhibit*, Reno,NV, January 1997.

[6] Kenneth Hall and Edward Crawley. Calculation of unsteady flows in turbomachinery using the linearized euler equations. *AIAA Journal*, 27(6):777–787, June 1989.

[7] A. Jameson. Solution of the euler equations by a multigrid method. *Applied Mathematics and Computation*, 13:327–356, 1983.

[8] Alberto Pueyo and David Zingg. Progress in newton-krylov methods for aerodynamic calculations. In *AIAA 35th Aerospace Sciences Meeting and Exhibit*, number AIAA-97-0877, Reno, NV, 6-9 January 1997.

[9] Youcef Saad. *Iterative Methods for Sparse Linear Systems*. Self Published, 2 edition, 2000.

[10] Youcef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, July 1986.

[11] M. Widhalm, R.P. Dwight, R. Thormann, and A. Hübner. Efficient computation of dynamic stability data with a linearized frequency domain solver. In *ECCOMAS CFD 2010*, Lisbon, Portugal, 14-17 June 2010.

[12] Peter Wong and David Zingg. Three-dimensional aerodynamic computations on unstructured grids using a newton-krylov approach. In *17th AIAA Computational Fluid Dynamics Conference*, number AIAA-05-5231, Toronto, Canada, 6-9 June 2005.