International Forum on Aeroelasticity and Structural Dynamics
IFASD 2022
13-17 June 2022, Madrid, Spain

NEXT-GENERATION COMPUTATIONAL FLUID DYNAMICS CAPABILITY FOR AIRCRAFT AEROELASTICITY AND LOADS

U S Vevek[1], Sebastian Timme[1], John Pattinson[2], Bernd Stickan[3] and Adam Büchner[4]

[1] University of Liverpool
Liverpool, England L69 3GH, United Kingdom
vevek.u-s@liverpool.ac.uk
sebastian.timme@liverpool.ac.uk

[2] Airbus Operations Ltd
Aerospace Avenue, Filton, Bristol BS34 7PA, United Kingdom
john.pattinson@airbus.com

[3] Airbus Operations GmbH (Loads and Aeroelastics)
Airbus-Allee 1, 28199 Bremen, Germany
bernd.b.stickan@airbus.com

[4] German Aerospace Center (DLR) − Institute of Software Methods for Product Virtualization
Zwickauer Straße 46, 01069 Dresden, Germany
adam.buechner@dlr.de

Keywords: linear frequency domain, automatic differentiation, solver scalability.

Abstract: This paper presents some of the first results obtained from the recently implemented linear frequency domain solver in CODA, the next generation flow solver framework. It uses automatic differentiation capability to compute the exact product of the Jacobian matrix with an arbitrary vector. Perturbation results for subsonic and transonic LANN wing cases show good agreement with experiments and results computed using DLR-TAU code. Computations have been performed using both one- and two-equation turbulence models for the NASA Common Research Model. Scalability assessment of the frequency-domain solver demonstrates the advantage of hybrid MPI/OpenMP partitioning over pure MPI partitioning.

1 INTRODUCTION

Aeroelasticity and loads considerations play an important role during the design, development, and certification of aircraft. Accurate modelling of the non-linear aerodynamics in the transonic regime is vital for prediction of static and dynamic loads imposed on the aircraft. Classical models based on potential flow equations such as the doublet lattice method [1], though efficient, neglect the nonlinearities, e.g., shock-wave/boundary-layer interaction. In contrast, computational fluid dynamics (CFD) is able to model such nonlinearities well. Therefore, CFD has become indispensable for modelling the complex unsteady aerodynamics in an industrial setting. In the context of aeroelasticity and loads, the linearised aerodynamics around such non-linear states have become established [2], because aircraft application requires very large numbers of simulations in a vast parameter space. Hence, interest has narrowed down to the unsteady aerodynamic response due to harmonic forcing such as in the structural degrees-of-freedom or due to external excitation (e.g., gust).

In recent years, there have been numerous advancements in CFD that have improved the modelling accuracy tremendously. These include the use of high-order schemes and advanced turbulence and transition models. Homogeneous and heterogeneous high-performance computing have significantly improved computational efficiency and scalability. Despite such advancements, the current industrial practice for aeroelasticity and loads predominantly relies on some variant of the one-equation Spalart-Allmaras (SA) turbulence model [3] with homogenous message passing interface (MPI) parallelisation only.

The next generation flow solver CODA [4, 5] is being developed to take advantage of emerging computing capabilities to eliminate limitations faced by previous generation codes such as those mentioned earlier. The newly incorporated automatic differentiation (AD) capability allows matrix-vector products with the Jacobian operator to be evaluated accurately regardless of the complexity of the underlying discretization schemes and physical models. This is an important step forward from computing the Jacobian matrix by hand-differentiation which becomes cumbersome and error-prone for complex models. With AD, there is no need to construct and store the explicit Jacobian matrix which brings about significant memory gains, though performance needs scrutiny with respect to the matrix-forming approach. The exactness of the matrix-vector product operation is crucial for the preconditioned Krylov subspace linear solvers used in the linearised CFD approach. The performance of these linear solvers heavily hinges upon the effectiveness of the preconditioner. CODA uses the sparse linear algebra library SPLISS [6] for constructing and solving linear systems that arise from the conventional implicit and linearised CFD approaches. SPLISS operates on a two-level parallelism with partitioning across MPI processes (distributed memory) as well as OpenMP/GPU threads (shared memory) for enhanced scalability. This allows for more effective preconditioning with full parallelization at the shared memory level and a strict block-Jacobi type approach (i.e., no parallel communication) at the distributed memory level.

The objectives of this work are (a) to verify and validate the recently implemented linear frequency domain (LFD) solver which capitalizes on the AD capability in CODA, and (b) to assess the performance/scalability of the solver on a large-scale case. The paper is organised as follows. In Section 2, a brief derivation of the LFD equation as implemented in CODA is provided. In Section 3, the main steps in the LFD solution procedure are outlined with some implementation details. Next, the results are presented in Section 4 for LANN wing and NASA CRM cases, concluding with an overview in Section 5 of the future work to be carried out.

2 THEORETICAL BACKGROUND

Standard time integration methods are very inefficient for computing the unsteady aerodynamic response to periodic structural deformation because time integration has to be performed for several cycles to eliminate transient effects with a large number of time steps per cycle. A faster and cheaper alternative is the LFD method which computes the periodic response directly. To derive the LFD equation, we begin with the unsteady Reynolds-averaged Navier-Stokes (URANS) equations written in a semi-discrete form,

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

in which $\boldsymbol{W}(t) = [\rho, \rho\boldsymbol{u}, \rho E, \dots]$ refers to the vector of conserved variables including those from the turbulence/transition models where ρ is the density, $\boldsymbol{u}$ is the velocity vector in Cartesian coordinates and E is the specific total energy. The symbols $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ denote the grid

node positions and velocities, respectively. The term $\boldsymbol{R}(\boldsymbol{W}, \boldsymbol{x}, \dot{\boldsymbol{x}})$ represents the spatially discretised non-linear residual vector. Lastly, $\mathrm{M}(\boldsymbol{x})$ is the diagonal mass matrix consisting of the cell volumes in the finite volume context.

Linearising Eq. (1) about a steady-state solution $\overline{\boldsymbol{W}}$, where $\boldsymbol{R}(\overline{\boldsymbol{W}}, \overline{\boldsymbol{x}}, \boldsymbol{0}) = \boldsymbol{0}$, and assuming small amplitude, harmonic perturbations ($\widehat{\boldsymbol{W}} \exp(i\omega t)$ and $\hat{\boldsymbol{x}} \exp(i\omega t)$) yields the standard LFD equation [7],

$$\left(i\omega\overline{\mathrm{M}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{W}}\right) \widehat{\boldsymbol{W}} = -\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\hat{\boldsymbol{x}} + i\omega\left(\frac{\mathrm{dM}}{\mathrm{d}\boldsymbol{x}}\hat{\boldsymbol{x}}\overline{\boldsymbol{W}} + \frac{\partial \boldsymbol{R}}{\partial \dot{\boldsymbol{x}}}\hat{\boldsymbol{x}}\right)\right] \tag{2}$$

where $\overline{\mathrm{M}} \equiv \mathrm{M}(\overline{\boldsymbol{x}})$ refers to the mass matrix of the undeformed grid. Equation (2) is a linear system which, for a given $\hat{\boldsymbol{x}}$ and ω, must be solved for $\widehat{\boldsymbol{W}}$. The solution $\widehat{\boldsymbol{W}}$ is the linear aerodynamic response to the grid deformation $\hat{\boldsymbol{x}}$, both oscillating harmonically at frequency ω. The grid deformation $\hat{\boldsymbol{x}}$ is computed from the structural deformation mode obtained, for instance, from free-vibration analysis of a finite-element structural model. A similar approach can be followed for computing the perturbation response to sinusoidal gust excitation.

3 METHODOLOGY

The LFD equation as given in Eq. (2) is not in the most suitable form for implementation in CODA because the non-linear residual vector and, correspondingly, the Jacobian matrix-vector product come pre-multiplied with the inverse of the mass matrix. To recast the LFD equation in a more suitable form, let us pre-multiply $\overline{\mathrm{M}}^{-1}$ to Eq. (2) and substitute $\boldsymbol{R} = \mathrm{M}\widetilde{\boldsymbol{R}}$ where $\widetilde{\boldsymbol{R}}$ denotes the residual vector computed in CODA. Upon simplification, this leads to a modified LFD equation,

$$(i\omega\mathrm{I} + \mathrm{J})\widehat{\boldsymbol{W}} = -\left[\overline{\mathrm{M}}^{-1}\frac{\partial\left(\mathrm{M}\widetilde{\boldsymbol{R}}\right)}{\partial \boldsymbol{x}}\hat{\boldsymbol{x}} + i\omega\left(\overline{\mathrm{M}}^{-1}\frac{\mathrm{dM}}{\mathrm{d}\boldsymbol{x}}\hat{\boldsymbol{x}}\overline{\boldsymbol{W}} + \frac{\partial \widetilde{\boldsymbol{R}}}{\partial \dot{\boldsymbol{x}}}\hat{\boldsymbol{x}}\right)\right] \tag{3}$$

where $\mathrm{J} = \partial\widetilde{\boldsymbol{R}}/\partial\boldsymbol{W}$ denotes the Jacobian matrix as defined in CODA.

The LFD procedure begins with a static aeroelastic coupled steady state solution of Eq. (1). The steady state solutions were computed in CODA using the implicit backward Euler scheme with local time stepping until the density and turbulence-variable residual norms dropped by ten orders of magnitude. Courant-Friedrich-Levy (CFL) number ramping of the local time steps was employed to accelerate non-linear convergence. The linear system at each outer iteration was solved using a generalized minimum residual (GMRES) solver [8] until the linear residual norm dropped by one order of magnitude. The solver used a maximum of 50 Krylov vectors with no restarts and a Jacobi-type iterative solver as preconditioner. Given a linear system $A\boldsymbol{u} = \boldsymbol{b}$, matrix A can be decomposed into $A = D_G + F$ where D_G is a *generalized* block diagonal matrix that may include off-diagonal blocks while matrix F consists of the remaining off-diagonal blocks. A Jacobi-type iteration can be defined as $\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + D_G^{-1}(\boldsymbol{b} - A\boldsymbol{u}^n)$. The key criterion for D_G is that it must be relatively easy to invert. The preconditioner was constructed using an explicit but approximate Jacobian matrix with contributions from the local time steps added along the main diagonal. The block tri-diagonal part of this approximate matrix was taken to be D_G (also known as 'lines inversion') and 50 Jacobi-type iterations were performed to precondition each Krylov vector.

With the steady state solution obtained, the right-hand-side (RHS) vector of Eq. (3) is computed next. To this end, the grid deformation vector $\hat{\boldsymbol{x}}$ is first computed from a prescribed structural mode. For simple rigid-body motions such as those considered in this paper, the grid deformation can be prescribed analytically. For more complicated motions, they can be computed using radial-basis-function volume-interpolation methods [9, 10]. With $\hat{\boldsymbol{x}}$ available, the terms in the RHS are computed using central difference approximations as follows.

$$\overline{\mathrm{M}}^{-1}\frac{\partial\big(\mathrm{M}\widetilde{\boldsymbol{R}}\big)}{\partial\boldsymbol{x}}\hat{\boldsymbol{x}} \approx \frac{\overline{\mathrm{M}}^{-1}}{2\epsilon}\begin{bmatrix}\mathrm{M}(\overline{\boldsymbol{x}}+\epsilon\hat{\boldsymbol{x}})\;\widetilde{\boldsymbol{R}}(\overline{\boldsymbol{W}},\overline{\boldsymbol{x}}+\epsilon\hat{\boldsymbol{x}},\boldsymbol{0}) \\ -\mathrm{M}(\overline{\boldsymbol{x}}-\epsilon\hat{\boldsymbol{x}})\;\widetilde{\boldsymbol{R}}(\overline{\boldsymbol{W}},\overline{\boldsymbol{x}}-\epsilon\hat{\boldsymbol{x}},\boldsymbol{0})\end{bmatrix} \tag{4a}$$

$$\overline{\mathrm{M}}^{-1}\frac{d\mathrm{M}}{d\boldsymbol{x}}\hat{\boldsymbol{x}}\overline{\boldsymbol{W}} \approx \frac{\overline{\mathrm{M}}^{-1}}{2\epsilon}[\mathrm{M}(\overline{\boldsymbol{x}}+\epsilon\hat{\boldsymbol{x}})-\mathrm{M}(\overline{\boldsymbol{x}}-\epsilon\hat{\boldsymbol{x}})]\overline{\boldsymbol{W}} \tag{4b}$$

$$\frac{\partial\widetilde{\boldsymbol{R}}}{\partial\dot{\boldsymbol{x}}}\hat{\boldsymbol{x}} \approx \frac{1}{2\epsilon}\big[\widetilde{\boldsymbol{R}}(\overline{\boldsymbol{W}},\overline{\boldsymbol{x}},+\epsilon\hat{\boldsymbol{x}})-\widetilde{\boldsymbol{R}}(\overline{\boldsymbol{W}},\overline{\boldsymbol{x}},-\epsilon\hat{\boldsymbol{x}})\big] \tag{4c}$$

The symbol ϵ denotes a small number, typically 10^{-8}, used for the finite differences. The arguments $\overline{\boldsymbol{x}}\pm\epsilon\hat{\boldsymbol{x}}$ indicate that the quantities are computed on deformed grids. It must be mentioned that these matrix-vector products will also be computed using AD in future. After computation, the term in Eq. (4a) and the sum of the terms in Eqs. (4b) & (4c) are stored separately for ease of generating the RHS vector for any frequency ω provided at runtime.

The final step in the LFD procedure is to solve the linear system given in Eq. (3). The linear system is solved using a Krylov subspace method which requires a means to compute the matrix-vector product with the linear operator $(i\omega\mathrm{I}+\mathrm{J})$. For an arbitrary complex vector $\boldsymbol{u}$, the matrix-vector product $(i\omega\mathrm{I}+\mathrm{J})\boldsymbol{u}$ can be computed as,

$$(i\omega\mathrm{I}+\mathrm{J})\boldsymbol{u} = i\omega\boldsymbol{u} + \big(\mathrm{J}\cdot\mathcal{R}e(\boldsymbol{u})\big)_{\mathrm{AD}} + i\big(\mathrm{J}\cdot\mathcal{I}m(\boldsymbol{u})\big)_{\mathrm{AD}} \tag{5}$$

whereby the subscript 'AD' refers to the fact that the matrix-vector product is computed in a matrix-free manner using AD. The product $(\mathrm{J}\boldsymbol{u})$ is computed component-wise since the matrix-vector product using AD permits only real vectors in the implementation in CODA. The LFD computations were performed with a restarted GMRES solver until the linear residual dropped by ten orders of magnitude. The solver used 100 Krylov vectors with a maximum of nine restarts. Preconditioning was performed using 100 Jacobi-type iterations with D_G taken to be the block diagonal part of the approximate Jacobian matrix with $i\omega$ added along the main diagonal. The preconditioner was chosen after testing different numbers of Jacobi-type iterations with both block diagonal and block tri-diagonal decompositions (refer to Section 4.2.1).

4 RESULTS & DISCUSSION

4.1 LANN Wing

The AGARD LANN wing [11] is a high-aspect-ratio trapezoidal wing with a supercritical aerofoil section. The configurations for a subsonic case (CT1) and a transonic case (CT5), following the experimental set-up, are given in Table 1. The wing has a quarter-chord sweep angle of 25° and an aspect ratio of 7.92. It has a semi-span length of $s = 1$ m and a root chord length of $c_r = 0.3608$ m. The Reynolds number and the non-dimensional frequency are both defined based on the mean aerodynamic chord $c_{ac} = 0.268$ m. In both cases, the wing is assumed to be rigid as it undergoes sinusoidal pitching oscillations. The pitching axis location is located at $x = 0.224$ m and $z = 0$ m as shown in Figure 1. The two configurations, CT1 and

Table 1: LANN wing case configurations.

	CT1	CT5
Mach number	0.62	0.82
Mean angle of attack	0.6°	
Reynolds number $Re = \rho_\infty U_\infty l_{ref}/\mu_\infty$	4.82×10^6	5.43×10^6
Reference length ($l_{ref} = c_{ac}$)	0.268 m	
Non-dimensional frequency $\omega^* = \omega l_{ref}/U_\infty$	0.198	0.152

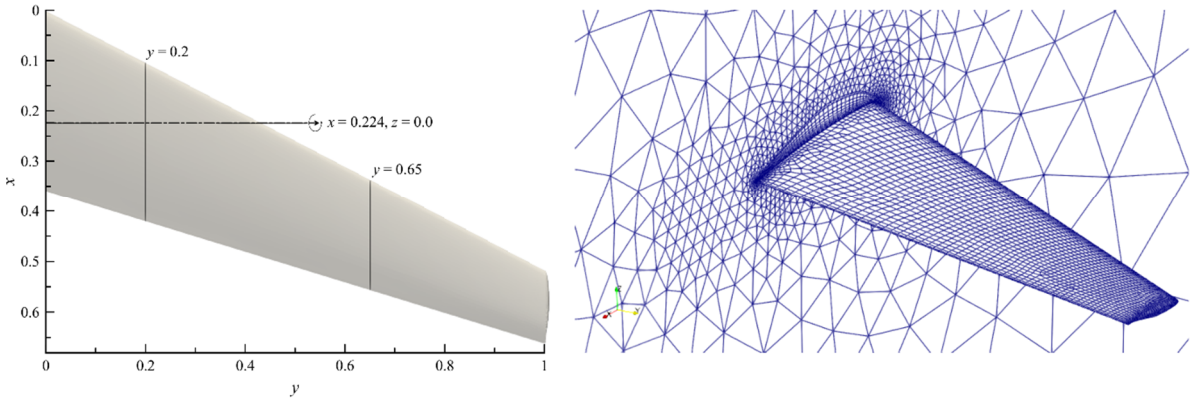


Figure 1: (left) LANN wing planform with the pitching axis & spanwise sampling locations and (right) wing surface and symmetry plane grids.

CT5, were computed in CODA using a hybrid, unstructured mesh consisting of about 0.328×10^6 cells. The hemispherical domain extends 50 m from the origin. The wing surface and symmetry plane grids are shown in Figure 1.

Inviscid fluxes were computed using the Roe scheme with an entropy-fix coefficient of 0.1. The SA-negative model was used to model turbulence. Spatial reconstruction was performed by representing the variables as cell-wise linear functions. The cell gradients were computed using Green-Gauss method with the face values initially approximated using distance-weighted averages of the neighbouring cell average values. The gradients were then bounded using a quintic spline limiter to prevent spurious oscillations before being used to reconstruct the final face values. Viscous fluxes were computed based on average face gradients obtained by augmenting the neighbouring cell gradients [12].

The steady and unsteady pressure coefficients sampled from two spanwise locations $y/s = 0.2$ and $y/s = 0.65$ are plotted in Figures 2 and 3 for cases CT1 and CT5, respectively. Corresponding results from DLR-TAU code as well as experimental data are plotted in the figures. The setup in TAU used the production code's default settings. In particular, Jameson-Schmidt-Turkel (JST) flux scheme [13] was used to compute the inviscid fluxes in TAU. The LFD linear system was solved using a GMRES solver with preconditioning done using an incomplete lower-upper (ILU) factorisation with zero fill-in. The uncertainties for the experimental unsteady pressure coefficients were computed from the LANN wing report as $\pm(0.02 + 0.05|q|)$, where q is $\mathcal{R}e(\Delta c_p)$ or $\mathcal{I}m(\Delta c_p)$.

It can be observed from Figure 2 that the results from CODA and TAU agree excellently with each other for the subsonic case. The agreement with experimental data is also reasonably good.

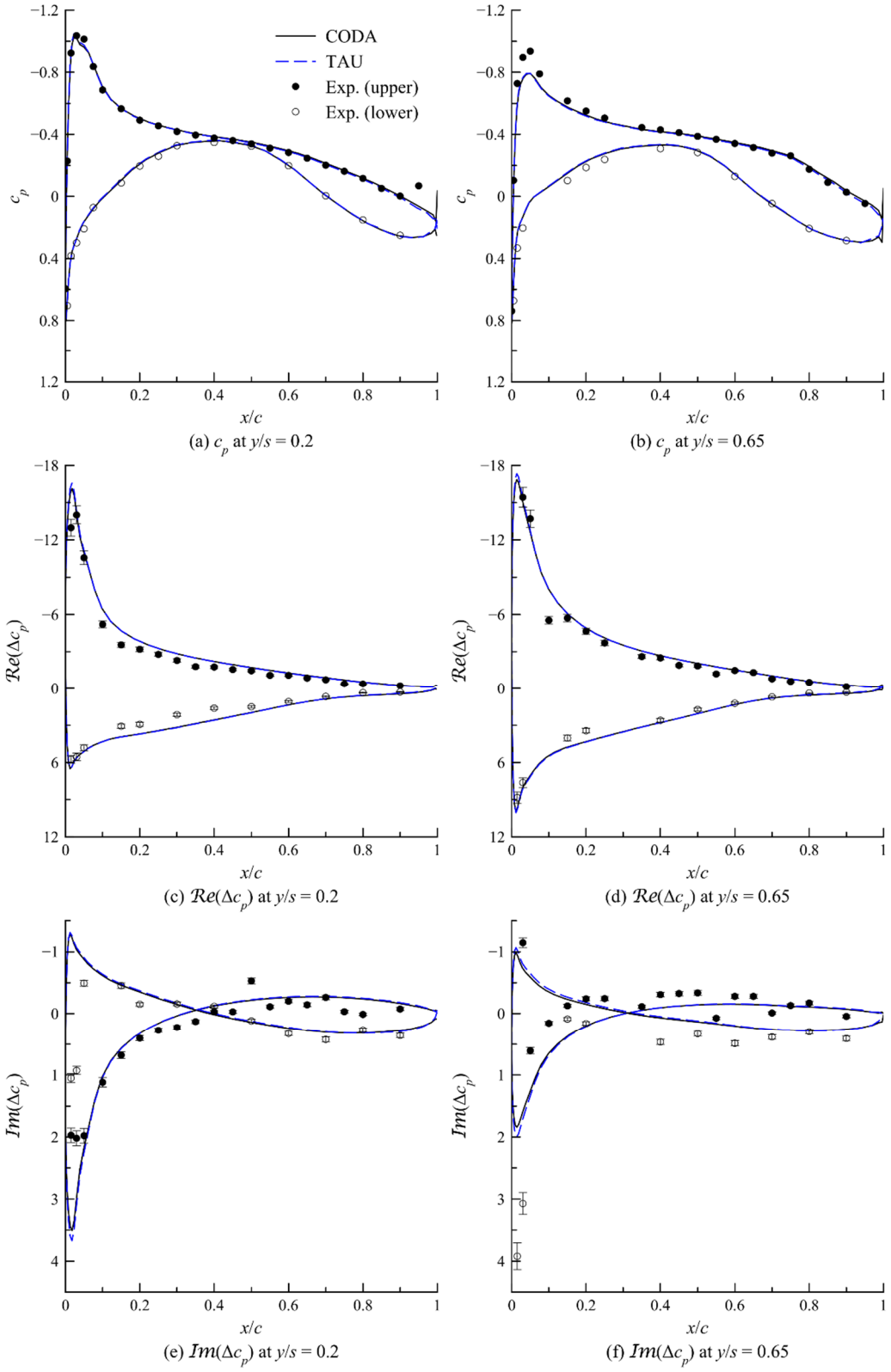


Figure 2: Steady and unsteady pressure coefficients for LANN wing case CT1.

As for the transonic case, it can be seen from Figure 3 that, although the general trends agree well between CODA and TAU, there are significant differences in the regions where the steady state solution exhibits abrupt changes. For instance, notice from Figure 3(a) that the shock position on the upper surface captured by experiment lies at $x/c \approx 0.48$ while the shock positions

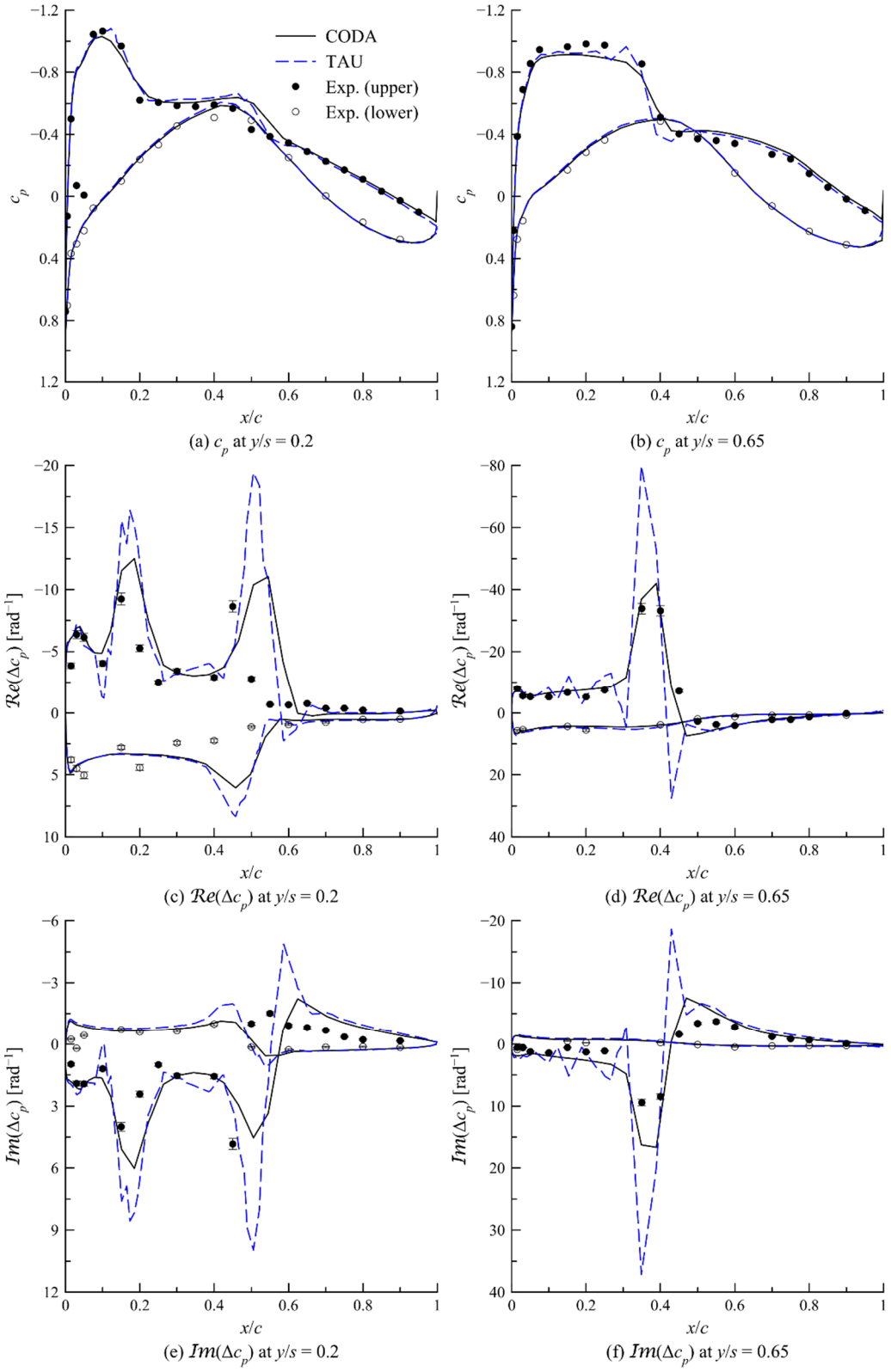


Figure 3: Steady and unsteady pressure coefficients for LANN wing case CT5.

captured by TAU and CODA lie at $x/c \approx 0.52$ and $x/c \approx 0.54$, respectively. This difference in shock positions clearly affects the locations of the peaks observed in Figure 3(c). In contrast, since CODA and TAU both agree with the experimental shock location at $x/c \approx 0.36$ in Figure 3(b), the corresponding peak locations also align well in Figure 3(d). The discrepancies can be

Table 2: NASA CRM case configuration.

Mach number	0.85
Mean angle of attack	3.0°
Reynolds number $Re = \rho_\infty U_\infty l_{ref}/\mu_\infty$	5.0 × 10^6
Reference length ($l_{ref} = c_{ref}$)	0.18915 m
Non-dimensional frequency $\omega^* = \omega l_{ref}/U_\infty$	1.0

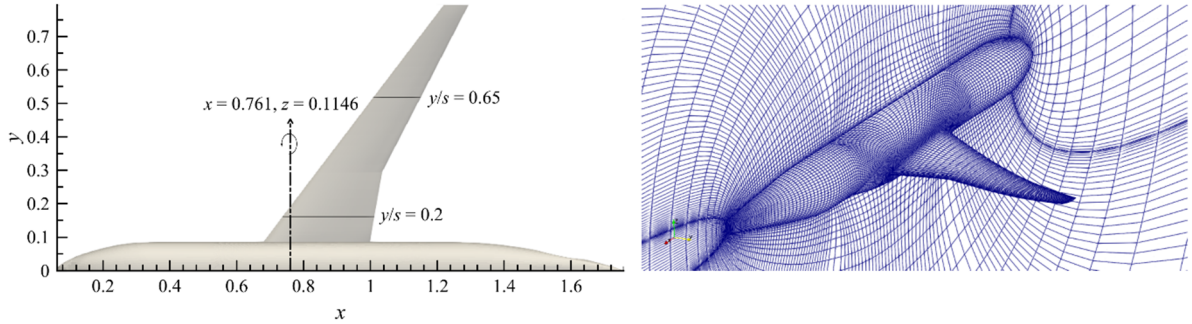


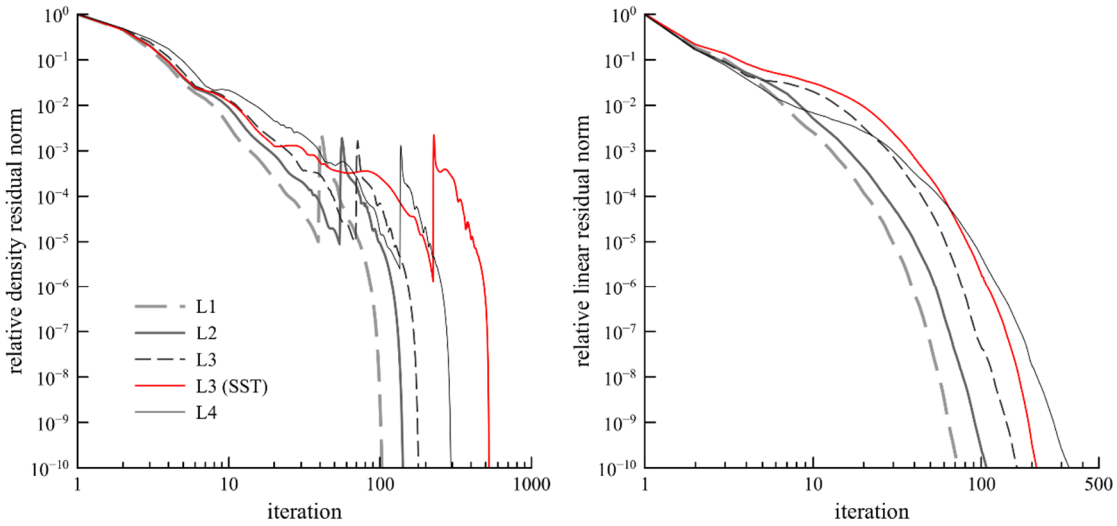
Figure 4: (left) CRM planform with the pitching axis & spanwise sampling locations and (right) wing-body surface and symmetry plane for grid L1.

attributed to the use of a rather coarse mesh which was chosen for testing and debugging purposes. On a different note, CODA consistently produces lower peak amplitudes compared to TAU. This is most likely due to the increased numerical dissipation of the Roe scheme compared to the JST scheme as evident from the more diffused shock profiles in Figures 3(a) and 3(b).

4.2 NASA CRM

The NASA Common Research Model (CRM) [14] is based on a typical civil transport aircraft. The case configuration is given in Table 2. The wing has a quarter-chord sweep angle of 35° and an aspect ratio of 9.0. It has a semi-span length of $s = 0.79337$ m. The Reynolds number and non-dimensional frequency are both defined based on the reference chord $c_{ref} = 0.18915$ m. Note that these dimensions correspond to the wind tunnel model used in the European Transonic Wind Tunnel campaign for this model. The pitching axis was chosen close to the wing root quarter chord position as indicated on the left of Figure 4 and the non-dimensional frequency was arbitrarily chosen to be 1.0.

The grids were obtained from the Fifth Drag Prediction Workshop (DPW5) grids database [15]. Out of a series of six structured grids of varying refinements, the first four grids (L1, L2, L3 & L4 consisting of respectively 0.64×10^6, 2.16×10^6, 5.11×10^6 & 17.3×10^6 hexahedral cells and a similar number of vertices) were used in this study. The grids were deformed according to the aeroelastic deflection of the wing at the chosen case configuration [16]. The aeroelastic deflections were measured in the European Transonic Wind Tunnel campaign of this model. The wing-body surface of the deformed grid L1 is shown on the right of Figure 4. The steady state and LFD computations were performed on the deformed grids. The discretization setup is similar to that described earlier for the LANN wing cases. The SA-negative turbulence model was used on all four grids while the k-ω SST (simply SST henceforth) turbulence model [17] was used only on grid L3. The problem was also computed using TAU with the SA-negative



(a) Steady state convergence (b) LFD convergence

Figure 5: Steady state and LFD residual convergence histories for NASA CRM case.

model on grid L3 to serve as a reference solution. Grid L3 was chosen for these two cases since it was found to yield sufficiently grid converged solutions for our purpose.

The steady state and LFD convergence histories are plotted in Figure 5. The steady states were computed in two stages – first with first order reconstruction until the density and turbulence-variables residual norms dropped by five orders of magnitude followed by second order reconstruction until the residual norms dropped by the desired ten orders of magnitude. The peaks observed midway during convergence in Figure 5(a) correspond to the restart with second order reconstruction. All cases converged to the required tolerance. Steady state convergence for the SST model on grid L3 required nearly three times the number of iterations as the SA-negative model on the same grid but LFD convergence required only slightly more iterations. The convergence of the SST model for a large-scale LFD problem is a promising development. It has not been possible previously to apply the SST model for LFD problems in TAU due to the stiffness of the original formulation of the model in which the specific dissipation rate ω could vary by many orders of magnitude throughout the flow field. In CODA, the SST model has been implemented with the transformed variable $\ln(\omega)$ instead which reduces the stiffness of the linear system. The reformulation and AD capability have enabled convergence of the LFD problem with the SST model. Similarly, we expect that more sophisticated turbulence/transition models will become available.

The steady and unsteady pressure coefficients were sampled at two spanwise locations $y/s = 0.2$ and $y/s = 0.65$. Results from CODA with the SA-negative model for all four grids are plotted in Figure 6. As expected, the size of the smallest features captured decrease and the peak amplitudes increase with grid refinement. CODA results for both turbulence models and TAU results for the SA-negative model, all computed on grid L3, are plotted in Figure 7. Steady state and LFD solutions from the SST model are similar to those computed with the SA-negative model with slight differences in the shock positions only. LFD results from TAU display larger peak amplitudes compared to CODA. This is, once again, an indication of the reduced numerical dissipation of the JST scheme used in TAU. However, insufficient numerical dissipation also leads to non-smooth wiggles in the LFD solutions from TAU at $y/s = 0.65$ (Figures 7(d) and 7(f)) which are highly reminiscent of Gibbs phenomenon near shocks. Indeed, one notices an undershoot and some minor oscillations in the steady state solution aft of the shock

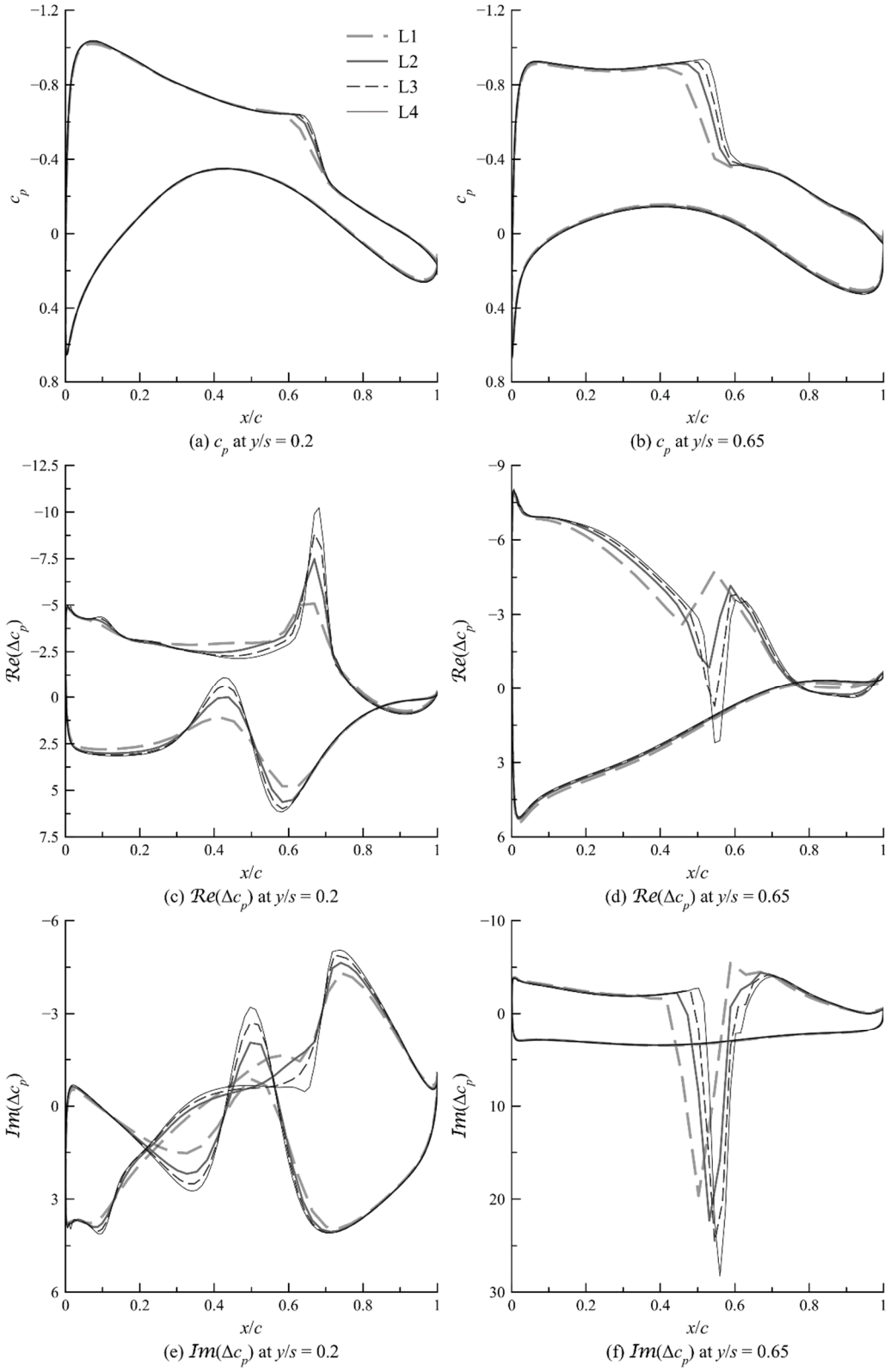


Figure 6: Grid convergence study for NASA CRM case on CODA with the SA-negative turbulence model.

in Figure 7(b). Since the LFD solution represents the perturbations about the steady state, any non-smoothness in the steady state solution becomes amplified in the LFD solution.

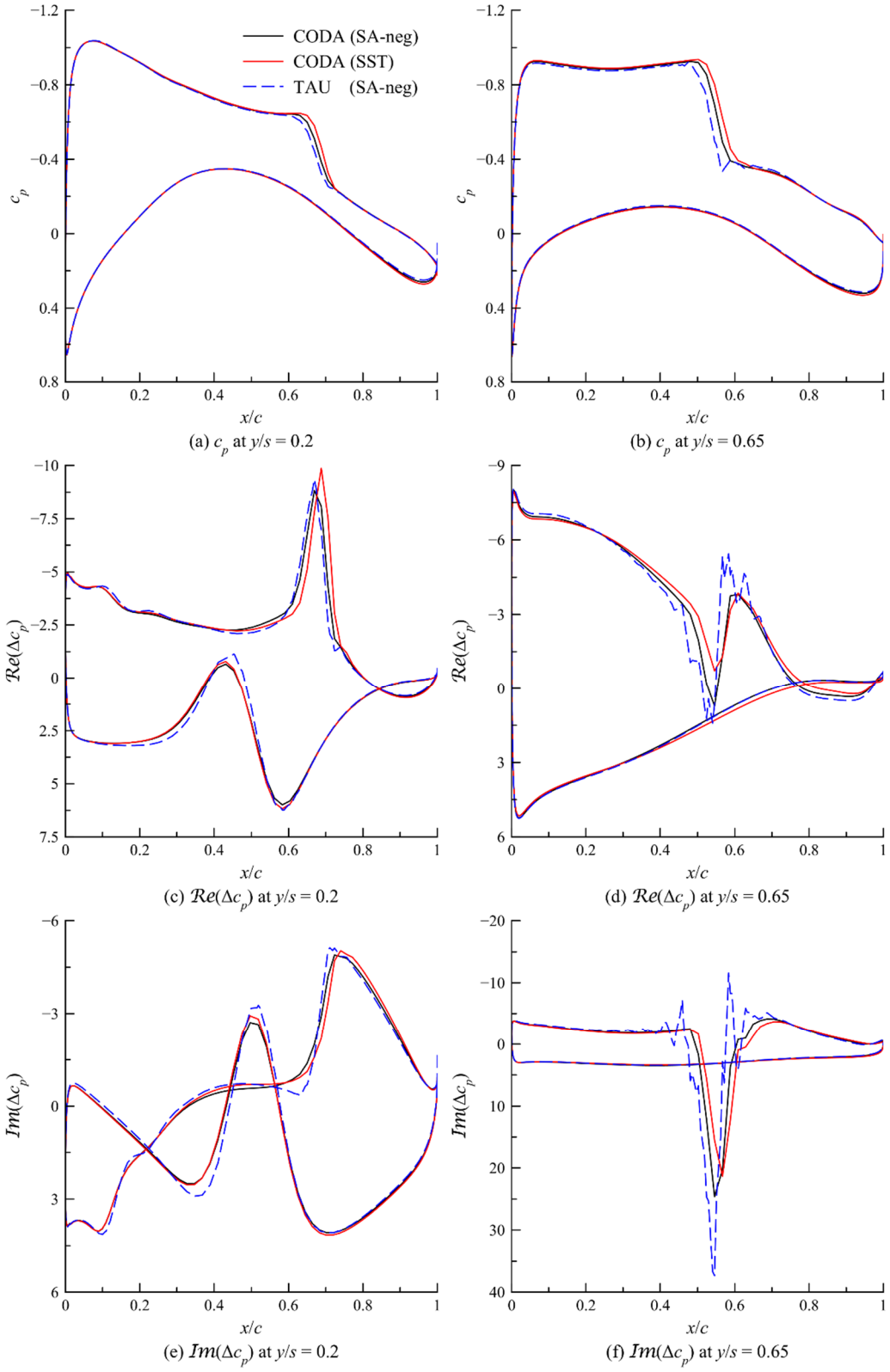


Figure 7: Steady and unsteady pressure coefficients for NASA CRM case on grid L3.

4.2.1 Computational Efficiency

The LFD problem using the SA-negative model on grid L3 was chosen for the scalability assessment. The problem was solved with Intel Xeon Gold 6230 CPU processors (2.10 GHz and 27.5 MB cache) which consist of 20 cores each. Each compute node comprises of two such

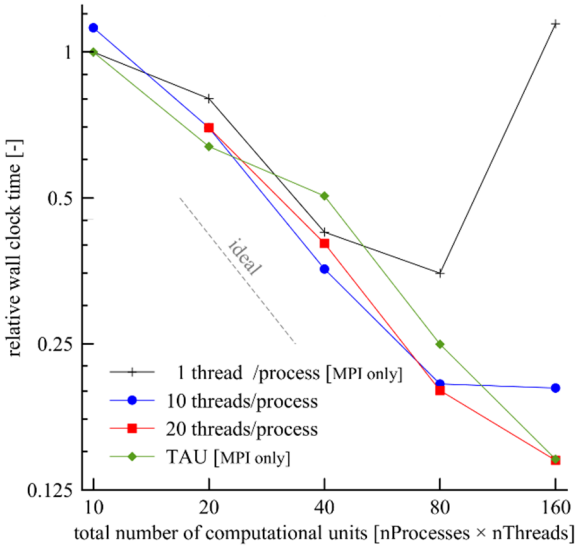


Figure 8: Scalability of LFD solvers for NASA CRM case on grid L3.

processors making a total of 40 cores available on each node. The problem was computed in CODA using MPI partitioning alone (one thread per MPI process) as well as the hybrid MPI/OpenMP partitioning (multiple threads per MPI process). Hybrid partitionings were done with 10, 20 and 40 threads per MPI process but 40 threads per MPI process was found to be a poor choice due to the node architecture. Hence, they are not included here. The problem was computed in TAU using MPI partitioning alone since it is the only one available. For conciseness, a particular partitioning is denoted as '(number of MPI processes) × (number of OpenMP threads per process)'. The product of these two numbers gives the total number of computational units. For instance, cases 40×1, 2×20 and 4×10 all use 40 computational units in total. Figure 8 shows the relative wall clock times plotted against the total number of computational units for the various cases tested. The wall clock times for TAU and CODA have been normalized with respect to the pure MPI cases 10×1 from the respective solvers.

TAU exhibits less-than-ideal scaling from case 10×1 to case 40×1 but improves thereafter and maintains near-ideal scaling up to cases 80×1 and 160×1. Since the ILU factorisation in TAU is only local to each core, it becomes more fragmented, and therefore less effective, with more partitions resulting in a greater number of linear iterations. While the scaling is reasonably good for the current case, the deterioration can be more rapid in other cases. CODA exhibits rather poor scaling with pure MPI partitioning from case 10×1 to case 20×1 which improves to near-ideal scaling from case 20×1 to case 40×1 before becoming extremely expensive and inefficient for a greater number of computational units. For greater than 10 computational units, hybrid partitionings take shorter times compared to pure MPI partitioning in CODA. With 10 threads per MPI process, the scaling improves considerably compared to pure MPI partitioning. It is less than ideal from case 1×10 to case 2×10 but becomes close to ideal from case 2×10 to case 4×10. Similarly, with 20 threads per MPI process, there is less-than-ideal scaling from case 1×20 to case 2×20 but becomes near-ideal from case 2×20 to case 4×20. Using 10 threads per process leads to some performance loss from case 4×10 to case 8×10 which becomes so excessive from case 8×10 to case 16×10 that there is almost no improvement achieved. Using 20 threads per process on the other hand does offer some improvement from case 4×20 to case 8×20 but it is less than ideal. The general trend of the CODA results seems to indicate that the loss of scalability encountered when using a large number of computational units can be somewhat offset with hybrid partitionings.

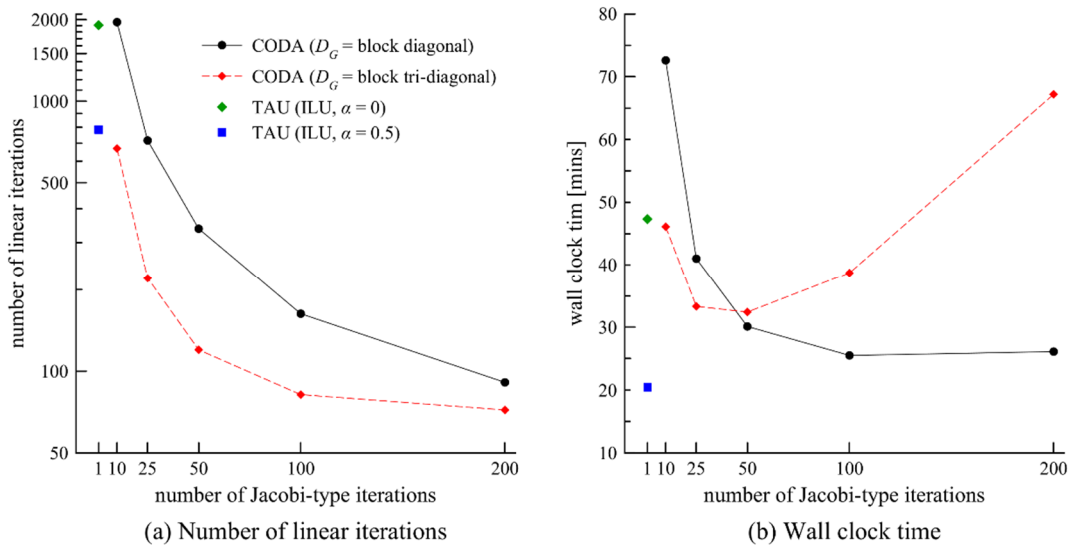


Figure 9: Effect of preconditioner for NASA CRM case on grid L3.

A preliminary profiling analysis of the CODA runs revealed that the majority of the time was spent during the preconditioning. Since the Jacobi-type iterations of the preconditioner were performed with a block diagonal decomposition, the preconditioner is perfectly parallel, i.e., its convergence is indifferent to how the problem is partitioned. Consequently, all the CODA LFD runs in the scalability test took identically 163 linear iterations to converge. In contrast, TAU required between 700 and 900 linear iterations to converge depending on the partitioning. The computation time in CODA is, therefore, solely determined by the efficiency of the preconditioner. Since the ILU factorisation is precomputed in TAU, each preconditioning step costs roughly as much as a single matrix-vector product (actually, less than this since it is process-local). On the other hand, with 100 Jacobi-type iterations, each preconditioning step in CODA requires 100 matrix-vector products with the approximate matrix and the precomputed block diagonal inversion and inter-process communication of 100 vectors. Hence, the preconditioning operation in CODA can currently be assumed to be roughly 100 times costlier than that in TAU. Note that preconditioning is only one operation in a single GMRES iteration and, therefore, the factor of 100 does not fully apply to a GMRES iteration. Despite the large difference in cost, it is remarkable that, even after accounting for the reduced number of linear iterations required in CODA, the computation times in CODA using hybrid partitionings were at most only twice greater than those of TAU. This implies that there is considerable advantage to be gained from the hybrid MPI/OpenMP parallelism since the use of shared memory by the OpenMP threads helps to keep the inter-process communication low.

Last but not least, the performances of different preconditioners in CODA for case 4×20 are compared in Figure 9 in terms of the wall clock times and the number of linear iterations. Both block diagonal and block tri-diagonal decompositions were tested with 10, 25, 50, 100 and 200 Jacobi-type iterations. The ILU factorisation in TAU is computed on a weighted average of two Jacobian matrices,

$$\mathrm{J} = \alpha \mathrm{J}_2 + (1-\alpha)\mathrm{J}_1 \tag{6}$$

where matrix J_1 is constructed with compact stencils considering only the immediate face neighbours and matrix J_2 is constructed with extended stencils considering the neighbours' neighbours as well. Using a weight of $\alpha = 0$ is analogous to the approximate Jacobian matrix in CODA while using a weight of $\alpha = 0.5$ represents the current best practice in TAU. Both

values of α were tested for case 80×1. TAU results are indicated in Figure 9 at one Jacobi-type iteration as the ILU factorisations were applied only once per GMRES iteration. Multiple Jacobi-type iterations were tested with the ILU factorisations in TAU but they produced worse results for this case which might be a result of the numerical instabilities arising from the matrix ordering [18]. It can be seen from Figure 9(a) that a single application of the ILU factorisation in TAU with the approximate and the weighted Jacobian matrices is equivalent to about 10 and 25 Jacobi-type iterations with block diagonal decomposition in CODA, respectively. Clearly, the choice of the matrix used for factorisation affects the efficiency of the preconditioner as demonstrated by McCracken*, et al.* [19]. Judging from the wall clock times shown in Figure 9(b), the ILU factorisations in TAU are about twice as fast as their equivalents in CODA. This shows that the efficiency of the preconditioner in CODA could also be improved by using more advanced factorisations. Focusing on the effect of the number of Jacobi-type iterations, it can be seen from Figure 9(a) that using more Jacobi-type iterations leads to a reduction in the number of linear iterations and for a given number of Jacobi-type iterations, block tri-diagonal decomposition results in a stronger preconditioner compared to block diagonal decomposition. However, beyond 100 Jacobi-type iterations, the reduction in the number of linear iterations becomes less significant and the cost of preconditioning becomes overwhelming. As seen from Figure 9(b), the wall clock times, which initially decrease with the number of Jacobi-type iterations, start to increase later. The results indicate that 100 Jacobi-type iterations with block diagonal decomposition is optimal in terms of the time taken. Future work will look into more advanced preconditioners.

5 CONCLUSIONS AND OUTLOOK

The recently implemented LFD solver in the next-generation flow solver CODA benefits from the AD capability that allows computation of the exact Jacobian matrix-vector product. The LFD results from CODA have been validated with experimental data for subsonic and transonic LANN wing cases. They have been demonstrated to be on par with the LFD results from DLR-TAU code for the LANN wing cases and a transonic CRM case. The k-ω SST turbulence model has been successfully used in computing the LFD solutions for the CRM case. A scalability assessment of the LFD solvers from CODA and TAU showed that, despite the use of much costlier preconditioner, the LFD solver in CODA took only slightly longer than TAU when using hybrid MPI/OpenMPI partitionings.

While some of these developments are a definite step forward from previous generation solvers, there are several areas which can be improved further. First of all, computation of the exact right-hand-side vector with AD would be more efficient and accurate compared to the finite difference approach used presently. Secondly, linear solver convergence could be improved by using multigrid methods which allow convergence independent of grid refinement. Thirdly, the performance of the preconditioner could be improved with the use of a more accurate, if possible exact, Jacobian matrix and more advanced factorisations. All three areas mentioned are being actively pursued.

6 ACKNOWLEDGEMENTS

The work leading to these results received funding through the UK project Development of Advanced Wing Solutions (DAWS) and the German collaborative research project DIGIfly. The DAWS project is supported by the Aerospace Technology Institute (ATI) Programme, a joint government and industry investment to maintain and grow the UK's competitive position in civil aerospace design and manufacture. The programme, delivered through a partnership between ATI, Department for Business, Energy & Industrial Strategy (BEIS) and Innovate UK,

addresses technology, capability and supply chain challenges. Adam Büchner recognizes and appreciates the support by Airbus Operations GmbH. We thank the University of Liverpool for computing time on the high-performance computing system. The simulation data that support the findings of this study are available from the authors upon reasonable request.

7 REFERENCES

[1] Albano, E. and Rodden, W.P. (1969). A doublet-lattice method for calculating lift distributions on oscillating surfaces in subsonic flows. *AIAA Journal*, 7(2), 279-285.

[2] Daumas, L., Forestier, N., Bissue, A., Broux, G., Chalot, F., Johan, Z., and Mallet, M. (2017). Industrial frequency-domain linearized Navier-Stokes calculations for aeroelastic problems in the transonic flow regime. *International Forum on Aeroelasticity and Structural Dynamics (IFASD) 2017*. Como, Italy.

[3] Allmaras, S.R. and Johnson, F.T. (Year). Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. *Seventh international conference on computational fluid dynamics (ICCFD7)*. Big Island, HI.

[4] Leicht, T., Jägersküpper, J., Vollmer, D., Schwöppe, A., Hartmann, R., Fiedler, J., and Schlauch, T. (2016). DLR-project Digital-X – Next generation CFD solver 'Flucs'. *Deutscher Luft- und Raumfahrtkongress 2016*. Braunschweig, Germany.

[5] Wagner, M., Jägersküpper, J., Molka, D., and Gerhold, T. (2021). Performance analysis of complex engineering frameworks, *Tools for High Performance Computing 2018 / 2019*. Mix, H., et al. (Eds.). Cham: Springer International Publishing. 123-138.

[6] Wagner, M. (2021). The CFD solver CODA and the sparse linear systems solver Spliss: evaluation of performance and scalability. *NHR CFD Workshop Day 2021*. Germany.

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

[8] 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.

[9] Sheng, C. and Allen, C.B. (2012). Efficient mesh deformation using radial basis functions on unstructured meshes. *AIAA Journal*, 51(3), 707-720.

[10] Jakobsson, S. and Amoignon, O. (2007). Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization. *Computers & Fluids*, 36(6), 1119-1136.

[11] Zwaan, R.J. (1985). LANN wing: pitching oscillation. AGARD-R-702, AGARD.

[12] Schwöppe, A. and Diskin, B. (2013). Accuracy of the cell-centered grid metric in the DLR TAU-code, *New Results in Numerical and Experimental Fluid Mechanics VIII: Contributions to the 17th STAB/DGLR Symposium Berlin, Germany 2010*. Dillmann, A., et al. (Eds.). Berlin, Heidelberg: Springer Berlin Heidelberg. 429-437.

[13] Jameson, A., Schmidt, W., and Turkel, E.L.I. (1981). Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes, *14th Fluid and Plasma Dynamics Conference*. American Institute of Aeronautics and Astronautics.

[14] Vassberg, J., Dehaan, M., Rivers, M., and Wahls, R. (2008). Development of a common research model for applied CFD validation studies. *26th AIAA Applied Aerodynamics Conference*. American Institute of Aeronautics and Astronautics.

[15] Morrison, J. (2021) DPW5 Grids and files from the Fifth Drag Prediction Workshop. Available from: https://dpw.larc.nasa.gov/DPW5/multiblock_grids.REV01/.

[16] Keye, S. and Gammon, M.R. (2018). Development of deformed computer-aided design geometries for the Sixth drag prediction workshop. *Journal of Aircraft*, 55(4), 1401-1405.

[17] Menter, F.R., Kuntz, M., and Langtry, R. (2003). Ten years of industrial experience with the SST turbulence model. *Turbulence, heat and mass transfer*, 4(1), 625-632.
[18] Anderson, W.K., Wood, S., and Jacobson, K.E. (2020). Node numbering for stabilizing preconditioners based on incomplete LU decomposition, *AIAA AVIATION 2020 FORUM*. American Institute of Aeronautics and Astronautics.
[19] McCracken, A., Da Ronch, A., Timme, S., and Badcock, K.J. (2013). Solution of linear systems in Fourier-based methods for aircraft applications. *International Journal of Computational Fluid Dynamics*, 27(2), 79-87.

COPYRIGHT STATEMENT