Recent Progress in OVERFLOW Convergence Improvements

Joseph M. Derlaga,* Charles W. Jackson,† Pieter G. Buning,‡
NASA Langley Research Center, Hampton, Virginia, 23681

Improvements have been made to the implicit symmetric successive overrelaxation algorithm in the OVERFLOW 2.3 structured, overset grid, computational fluid dynamics flow solver. These improvements, consisting of implicit boundary conditions, improved flux Jacobian linearizations, and CFL number ramping, are a series of evolutionary changes to the linear solver that have resulted in increased nonlinear convergence rates and faster time to solution. A series of test cases are presented that demonstrate the effect of the changes through comparison with the original SSOR path and other linear solver implementations within OVERFLOW.

I. Introduction

Computational engineers must balance the requirements of solution fidelity with cost (most often thought of as time to solution). At the fidelity and time extremes, both direct numerical simulations and panel methods provide useful engineering data, but one is too costly for day-to-day purposes and the other sacrifices too much physical fidelity for detailed flow analysis. Reynolds-averaged Navier-Stokes (RANS) simulations seem to have found the right balance in the fidelity to cost ratio, given the large number of government, academic, and commercial RANS solvers that exist. While fidelity can be improved by increasing the order of the discretization or by switching to a 'better' turbulence model, these can incur increased cost due to the complexity involved in providing more data to create a higher-order residual or the need to solve more closure equations. Cost can be addressed through computational kernel changes, such as rewriting code to better utilize newer hardware, or through algorithmic improvements to improve convergence.

The OVERFLOW flow solver [1] has been utilized across a variety of flow regimes and is well known to have a low cost per iteration and relatively small hardware footprint, i.e., a large number of computational grid points per CPU core. While a low cost per iteration is extremely important, the overall measure of cost is time to a converged solution, a metric that OVERFLOW has struggled with. The focus of this paper is on what has been done to improve the cost of using OVERFLOW, namely, what algorithmic changes have been made that will improve time to solution by improving convergence properties.

This paper will present some background information on the OVERFLOW flow solver, discuss various algorithmic changes that improve nonlinear convergence, and present examples of the improvements in practice.

II. Background

OVERFLOW is a structured, overset grid, RANS solver that has been used for simulating various configurations over a wide range of flow regimes. OVERFLOW has a long history, with its roots dating back almost 40 years to the F3D [2] and ARC2D/ARC3D [3] finite difference (FD) solvers. Since that time, OVERFLOW has been updated to include a variety of finite volume (FV) and Weighted Essentially Nonoscillatory (WENO) FD/FV options, not to mention an assortment of boundary conditions, turbulence models, and auxiliary equation sets.

A variety of solution evolution schemes are available in OVERFLOW, both explicit and implicit in time, with varying orders of time accuracy. The implicit schemes, which are the focus of this study, require the solution of a linearized problem, $Ax = b$ at each time step. This linear problem consists of a right-hand-side (RHS) vector, b, which contains the nonlinear residual and a left-hand-side (LHS) matrix, A, containing a linearization (potentially with approximations) of the RHS operator.

Spatial approximately factored (AF) implicit schemes, which perform the action of both forming and solving the linearized problem, are the workhorse linear solvers of OVERFLOW. The standard Beam-Warming 3-factor scheme [4] forms a series of easily inverted block-tridiagonal problems, based on a 3-point nearest neighbor stencil, with Jacobian entries for either central or Steger-Warming characteristic split finite difference schemes [1]. The Pulliam-Chaussee scalar pentadiagonal scheme [5] (aka the Diagonalized Alternating Direction Implicit (DADI) scheme [6]) modifies the

---

*Research Scientist, Computational AeroSciences Branch, 15 Langley Blvd. MS 128, AIAA Member
†Research Scientist, Computational AeroSciences Branch, 15 Langley Blvd. MS 128, AIAA Member
‡Senior Research Scientist, Computational AeroSciences Branch, 15 Langley Blvd. MS 128, AIAA Associate Fellow

Beam-Warming scheme by performing a local eigenvector factorization on a 5-point stencil, further simplifying the inversion process to a sequence of 3 scalar pentadiagonal solves while also allowing for an improved implicit treatment of artificial dissipation terms for the central schemes. All of the AF schemes, while efficient on a per iteration basis, are subject to factorization error that imposes a stability limit and requires a reduced Courant-Friedrichs-Lewey (CFL) number, as discussed in Nichols et al. [1].

The Successive Symmetric Overrelaxation (SSOR) nonfactored scheme introduced to OVERFLOW in Ref. [1] was designed to be a more robust option than the AF schemes. While more costly due to increased storage requirements, and requiring a sweeping method to approximately invert the LHS Jacobian matrix, the SSOR scheme has proven to be well matched to the upwind discretizations.

The original motivation for this work was to include implicit physical boundary condition contributions during the linear solve, something missing from all of the LHS approximations, with the goal of improving nonlinear convergence. While the factored implicit schemes have been extremely popular, adding implicit boundary conditions can be difficult due to the need for intermediate boundary specifications on the separate factors [7], or almost impossible in the case of the scalar pentadiagonal solver due to the underlying assumptions required to diagonalize the LHS. The SSOR scheme is not subject to these limitations, and is therefore the logical target for improvements, as discussed below.

III. Solver Improvements

The following subsections will discuss the addition of implicit physical boundary conditions, improved flux Jacobian linearizations, and CFL ramping. While not revolutionary, the sum of each evolutionary change has a remarkable impact on OVERFLOW's convergence behavior.

A. Implicit Physical Boundary Conditions

By treating physical boundary conditions in an explicit manner, the communication between boundary nodes and interior nodes lags during the linear solve, resulting in slower nonlinear convergence. Many of the boundary conditions within OVERFLOW are simply functions of Dirichlet and nearest neighbor data, resulting in a simple data dependency that fits within the current storage paradigm used by the SSOR solver, i.e., each node only has nearest neighbor support. Some boundary conditions, such as those that integrate or average over a computational face, do not satisfy the local support assumption but can be approximately linearized based on a nearest neighbor approximation.

As is true of all the implicit schemes in OVERFLOW, storage is already allocated for boundary terms (although only for nearest neighbor dependency), and a few simple modifications to the SSOR scheme were necessary to access those memory locations and ensure they were properly included within the sweeping algorithm.

Figure 1 shows the before and after convergence behavior of OVERFLOW for a simple 2D airfoil test case, as described in detail in Section IV A, using a physical time step equal to traveling $1/100^{th}$ of a chord length at freestream velocities and 10 Newton subiterations per timestep. While the convergence is slightly improved, the asymptotic convergence behavior is similar, pointing toward a more fundamental issue to be addressed.

B. Improved Linearizations

Given the results of Section III A, efforts were made to identify the root cause of the lackluster convergence behavior. Due to OVERFLOW's history as a finite difference solver, the linearizations of the mean flow flux terms were all based in a finite difference context. Per the work of MacCormack [8], the delta form linearization of a set of nonlinear equations are such that the RHS of the linear problem contains all of the desired physics, and the LHS matrix is simply numerics that are used to drive the delta form update to zero as the nonlinear problem evolves. A wide variety of approximations can be made to the LHS matrix in order to satisfy various requirements, e.g., storage size, computational effort, programming effort, linear solver convergence requirements. However, how well the LHS matrix approximates the true linearization of the nonlinear problem has a direct correlation on how well the nonlinear problem converges.* As noted, OVERFLOW has been upgraded over the course of many years to include new discretization schemes. The Steger-Warming flux Jacobians added by Nichols et al. [1], while an improvement on the central flux Jacobians previously used, created an obvious mismatch between the LHS matrix linearization and the true linearization of the residual operator.

---

*This ignores problems related to the realizability of the solution produced by the linear problem during transients. For example, see the work of Ceze and Fidkowski [9] for how penalization terms can be introduced to aid realizability.
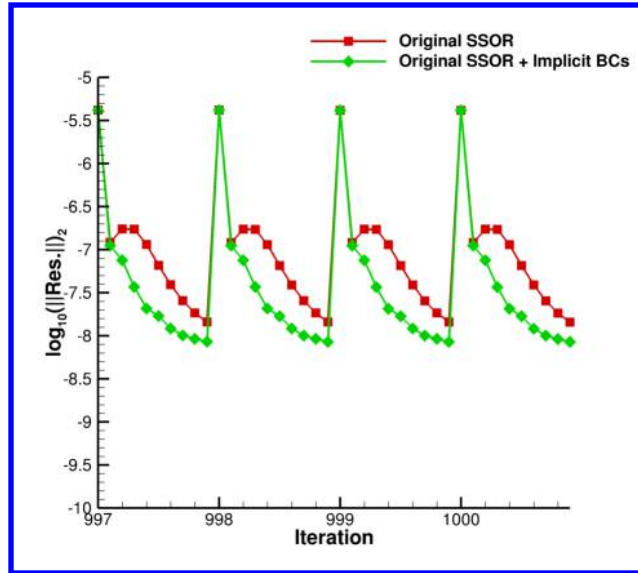
Fig. 1   Subiteration behavior with and without implicit boundary conditions.

Without further modifying any of the underlying SSOR methodology, a first-order finite volume linearization has been developed for the Roe [10], HLLC [11], and HLLE++ [12] flux schemes, both with and without low-Mach preconditioning [13]. Returning to the example case from Section III A, as shown by the line with triangles in Fig. 2, the convergence benefits of more appropriate linearizations and implicit boundary conditions are clear.
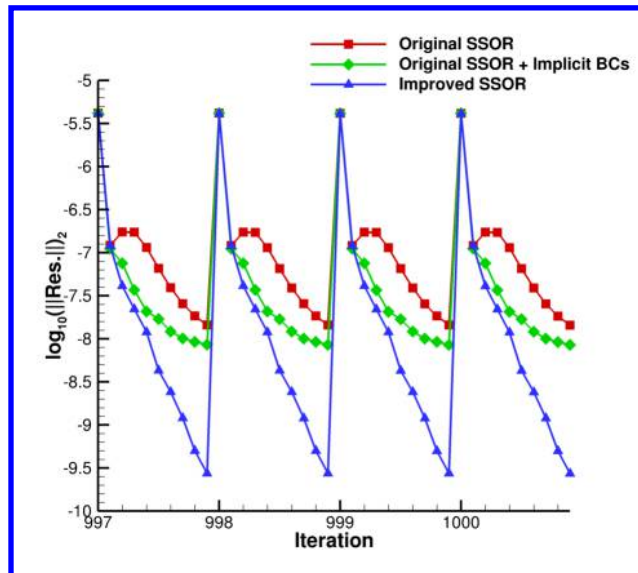

Fig. 2   Subiteration behavior with and without proper linearizations.

During testing, it was found that the improved linearizations could result in solution instabilities if not coupled with implicit boundary conditions. For example, before the addition of an implicit treatment of wake cuts, C-grid airfoil cases demonstrated solution divergence at moderate initial CFL numbers or global time steps. In general, any improvements made to the implicit boundary conditions improved the convergence behavior of the nonlinear residual. This presents a 'chicken or the egg' conundrum; inclusion of implicit boundary conditions does not yield large improvements in convergence when the flux linearizations are mismatched, but improving only the flux linearizations presents potential stability problems. Given that the approximate factorization schemes were the main linear solvers for many years, and the noted difficulty in including implicit boundary conditions, it is not surprising that the addition of implicit boundary

3

conditions and matched linearizations were not implemented sooner.

C. CFL Ramping

OVERFLOW can evolve a solution via either a physical timestep, where each node takes the same timestep, or in pseudotime, where a constant CFL is specified. For steady state simulations, where the time accuracy of intermediate solutions is unimportant, a global time step specification may be too restrictive given the wide range of scales present in a simulation. During initial startup, transient behavior can place restrictions on how aggressively a solution is allowed to evolve via the CFL number. By using a CFL evolution strategy, the CFL number is allowed to change, either in a predescribed manner or through a link to the nonlinear residual. The Switched Evolution Relaxation (SER) method of Mulder and van Leer [14] is perhaps the simplest evolution strategy that couples the CFL number with the residual convergence. If the nonlinear residual is converging (dropping), the CFL number is allowed to grow toward a preset maximum limit; if the nonlinear residual is diverging (growing), the CFL number is reduced toward a preset minimum limit.

Figure 3 demonstrates the cumulative effect of all the algorithmic improvements using the NACA 0012 test case distributed with OVERFLOW. The two lines with square and diamond symbols show, respectively, the central scheme with the scalar pentadiagonal solver and a Roe upwind scheme with the original SSOR path running with a fixed time step. For this case, the scalar pentadiagonal scheme appears to be converging better than the original SSOR path. However, by simply switching to the improved SSOR path, with its improved linearizations and implicit treatment of boundary conditions, the Roe scheme becomes more competitive, as shown by the line with the right facing triangles. Since this is a steady state case, the use of a constant or variable CFL number is of interest, and the former is shown as the line with the left facing triangles. Finding a stable constant CFL number can be a challenge; since every node is evolving at a different rate, the chance of instability increases. When running at a constant time step, the CFL number locally varies and allows near-wall points to evolve at a faster rate than what a constant CFL would normally allow in OVERFLOW. Upon enabling the CFL ramping, which was run with a maximum CFL limit of 1 million, we see impressive convergence behavior compared to any of the other options, as shown by the line with upward facing triangles. While not as efficient as the scalar pentadiagonal scheme on the time per iteration metric, the improved SSOR scheme is clearly superior when considering the time to solution metric. While any of the linear solvers in OVERFLOW can make use of the CFL ramping method, the presence of factorization error in the AF schemes means that CFL ramping is of limited use compared to the SSOR schemes, a point which will be revisited in Section IV C. During testing, it was noted that CFL ramping was not generally beneficial for unsteady problems as the number of iterations required to reach a large CFL number was typically greater than the number of subiterations that were practical for the computation. Because of this, using a constant time step method within the subiteration process is still preferred.
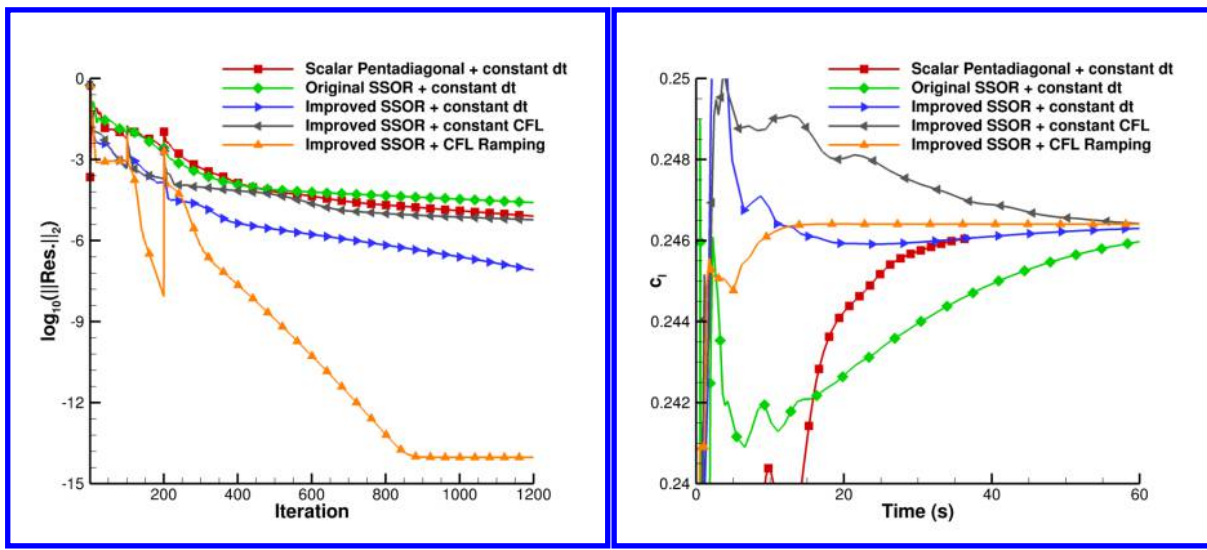

(a) Residual. Note the effect of grid sequencing. (b) Lift coefficient.

Fig. 3   NACA 0012 convergence comparison.

IV. Test Cases

Several test cases are presented below to demonstrate the capabilities of the improved SSOR scheme. It should be noted that the linear solvers used in this study have different stability characteristics and convergence behaviors, and that the settings that work best for one may not be the best for the other. For instance, the approximate factorization schemes generally have a lower stable CFL limit at which they will converge due to their inherent factorization error compared to the unfactored SSOR schemes, but this is balanced by the lower computational work per time step that the AF schemes incur. In order to make a fair comparison in terms of wall clock time, which is our most critical metric, the best effort was made at running each scheme with settings for best convergence or running with the same settings in order to compare and contrast convergence behavior. For the steady state cases, no attempt was made at a staged solution process that would improve transient convergence during startup, i.e., OVERFLOW was run from a single input file without restarts for each case discussed below. By using a single input file restart, glitches due to the recombination of split blocks are avoided as well, simplifying the analysis.

A. S809 Airfoil

A two-dimensional study allows us to compare and contrast the behavior of the improved SSOR scheme to the original SSOR scheme without having to account for explicit overset boundaries or grid splitting. A new O-grid version of the S809 airfoil [15] test case distributed with OVERFLOW (which use a C-grid) was created in order to focus on implicit physical boundary conditions in the absence of an artificial wake cut boundary. The wall spacing, as well as the spacing and number of nodes around the chord (601 points), were kept the same for the O- and C-grids.

These 2D tests are intended to examine the overall performance trends between the original and improved SSOR schemes and introduce some of the expected differences that could be seen in practice.

1. Compressible Steady State

A steady-state convergence study is examined first. Figure 4(a) is a linear-log plot of the residual convergence for the O-grid case, running with the HLLE++ flux scheme and negative Spalart-Allmaras (SA-neg) [16] turbulence model at an angle of attack of 1 degree, Mach number of 0.1, and chord Reynolds number of 2 million. Simply by switching from the original to the improved SSOR solver, a fully converged solution is reached in about half the number of iterations; by running with CFL ramping, the improved SSOR solver converged in $1/100^{th}$ of the number of iterations of the original scheme. In terms of wall clock time, the improved SSOR solver with CFL ramping fully converged this case in approximately three minutes as compared to the original SSOR scheme which required nearly 3.5 hours, as shown in Fig. 4(b).
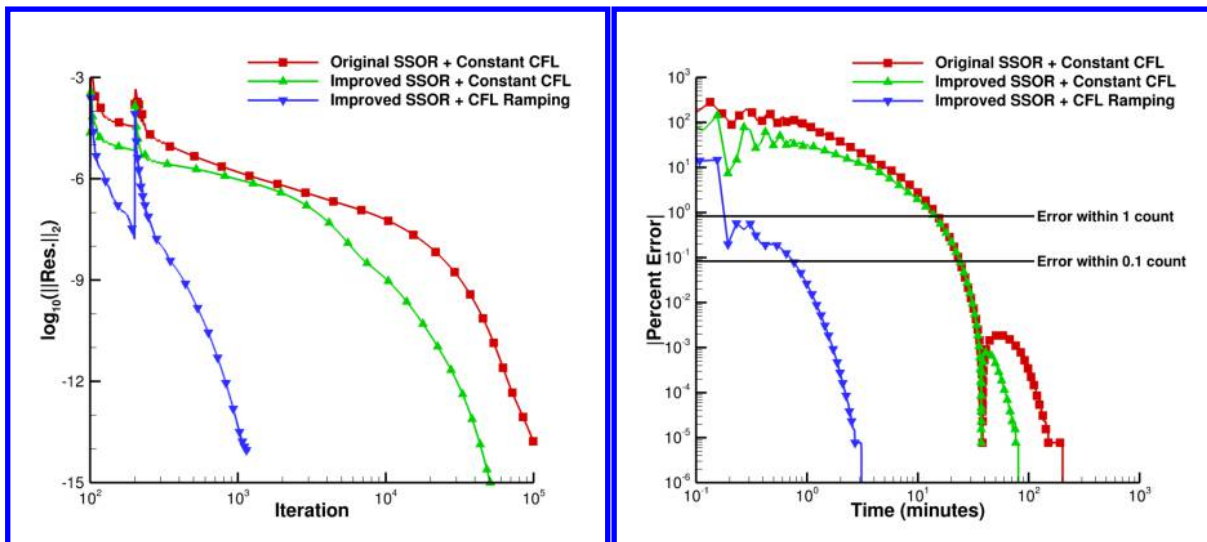
(a) Residual convergence. Note the linear-log plot and the removal of the first 100 iterations.

(b) Drag convergence. Note the log-log scale.

Fig. 4   Steady state S809 convergence comparison.

5

2. Low-Mach Preconditioning

As this case has a relatively low freestream Mach number, low-Mach preconditioning was also tested using the C-grid topology and the Langtry-Menter transition model [17]. Third-order MUSCL reconstruction [18] with Roe's scheme was used instead of HLLE++ as it proved to be the most robust option when using low-Mach preconditioning. This case uses a constant time-step with a time-step factor of 1000, no minimum CFL limit, and a maximum CFL limit of 1000. These settings were chosen to provide reasonable convergence with both the original and improved SSOR solvers. Of note, these settings were too aggressive for the compressible path with the original SSOR solver, causing divergence of the flow solver. Low-Mach preconditioning is certainly helpful for convergence, outperforming the compressible path with these solver settings, but, as shown previously in Fig. 4(a), the compressible path can perform well with alternate solver settings.
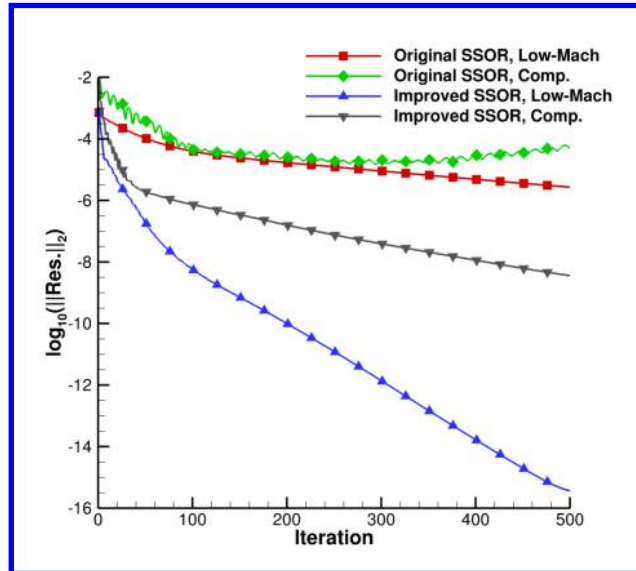

Fig. 5   Comparison of S809 convergence for compressible and low-Mach preconditioned paths.

In terms of time to solution, the low-Mach preconditioned case converged to within one count of drag within 30s, a tenth of a count within 40s, and to within a hundreth of a count within a minute for the improved SSOR scheme. In comparison, it took approximately 10 minutes for the original SSOR scheme to converge within 1 count of drag, approximately 30 minutes to be within a tenth of a count, and approximately 45 minutes to be within a hundreth of a count.

3. Time Accurate

In order to examine subiteration convergence, a somewhat contrived example was used. Using an impulsive start, OVERFLOW was run using a physical time step equal to $1/100^{th}$ of chord travel at freestream speed for a total of two hundred iterations, i.e., two chord lengths of travel time. Two subcases were examined, 1) using a fixed number of subiterations (10, plus an additional run with 25 for the orignal SSOR scheme), and 2) a two-order of magnitude convergence drop, a somewhat arbitrary, specified tolerance, with a maximum number of subiterations (50).

Figure 6 shows the order of magnitude of the residual drop during the subiteration process for all the cases and Fig. 7 shows the convergence behavior of the total drag coefficient during the subiteration process. Clearly, the improved SSOR scheme is superior, offering deeper convergence and a corresponding faster force convergence within the subiterations. Figure 6(a) shows the nonlinear residual at the start of each physical time step; no extraordinary differences in the residual levels exist that would account for the vast differences in the subiteration convergence behavior, i.e., the flow is somewhat similarly behaved between the different cases. Thus, it is safe to state that this is a valid comparison of the subiteration performance of the original and improved SSOR schemes despite the impulsive start.

Examining Fig. 6(c), we see how well each scheme is able to converge; the original SSOR scheme is almost never able to reach the required two-order of magnitude convergence drop, even with 50 subiterations, typically only converging just over one-and-a-half-orders of magnitude, while the improved SSOR scheme typically requires only 4 to

5 subiterations to meet the requirement.

   If we examine Fig. 7(a), which compares the drag convergence for all of the runs, we see that the original SSOR scheme with only 10 subiterations calculates a drag value at least 2 counts higher than the other schemes. Figure 7(b) removes this line so that we can better study the drag convergence during the subiteration process. The improved SSOR scheme shows essentially the same behavior between its two runs, showing that the 2 order of magnitude drop tolerance is sufficiently converged compared to the 10 subiteration case. The original SSOR scheme shows much different behavior. The lack of convergence of previous subiterations means that the starting solution at the beginning of each physical timestep (baseflow) is different enough to cause each scheme to converge to a different solution. Essentially, there is a build up of error in the baseflow that is carried through the time stepping process. This reinforces the impact that better convergence has on solution quality; we should always seek ways to improve the convergence of our discretization schemes.

   On close examination of the residual convergence for the original SSOR scheme, Fig. 7(c), which shows the drag vs. the absolute value of the residual drop for each of the test cases for the final iteration, the drag shows a high sensitivity to the residual drop, i.e., as the residual convergence stalls out, the drag convergence is still changing, in contrast to the improved SSOR scheme. This reinforces an important lesson that just because residual convergence appears to have stalled does not necessarily mean that the solution convergence has stalled as well.
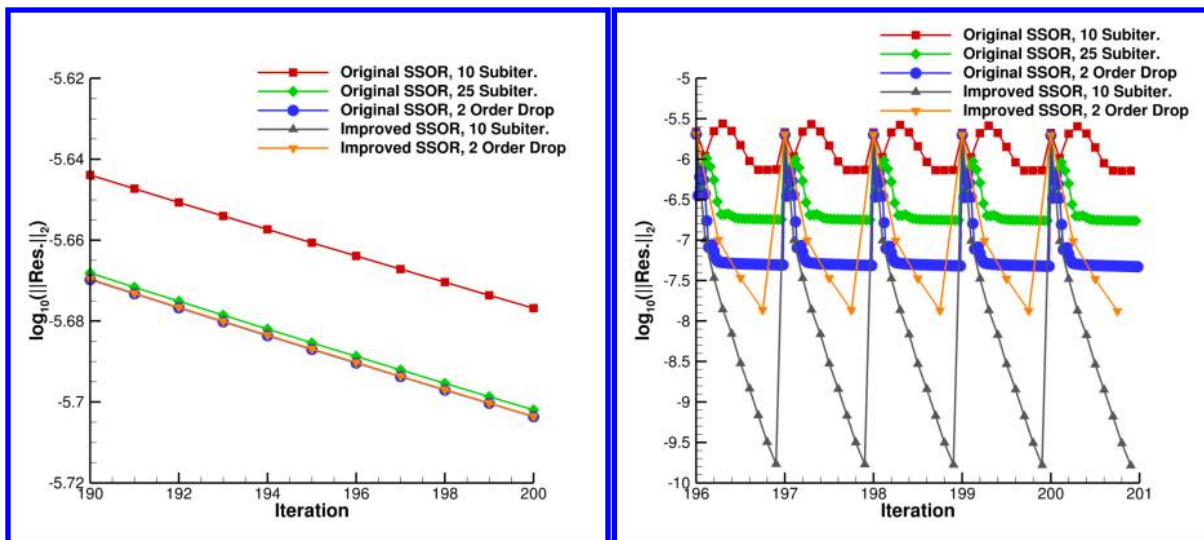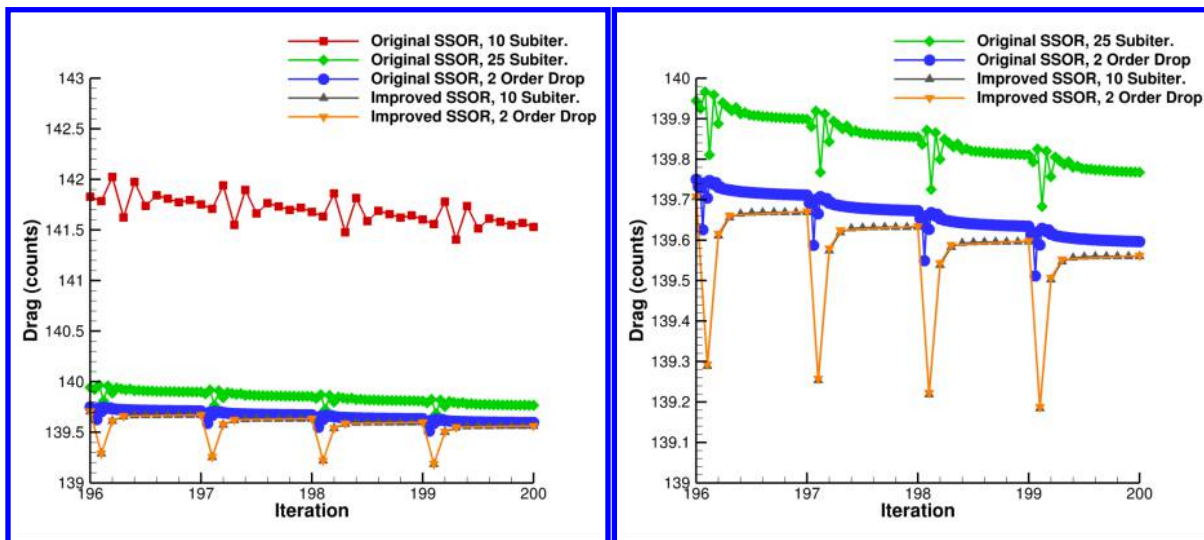
(a) Initial residual.
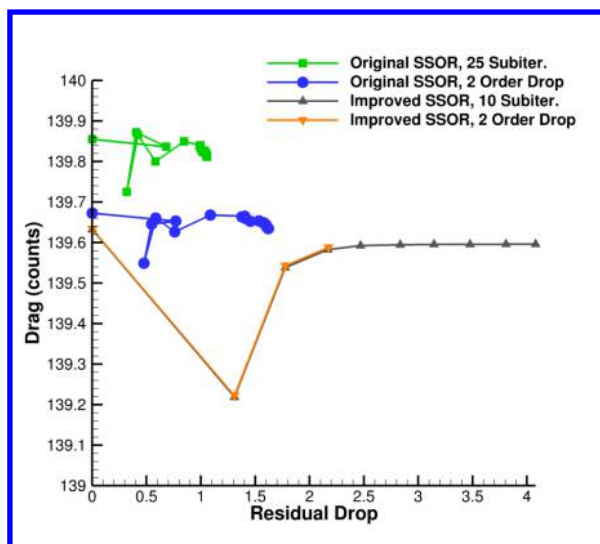
(b) Subiteration behavior.
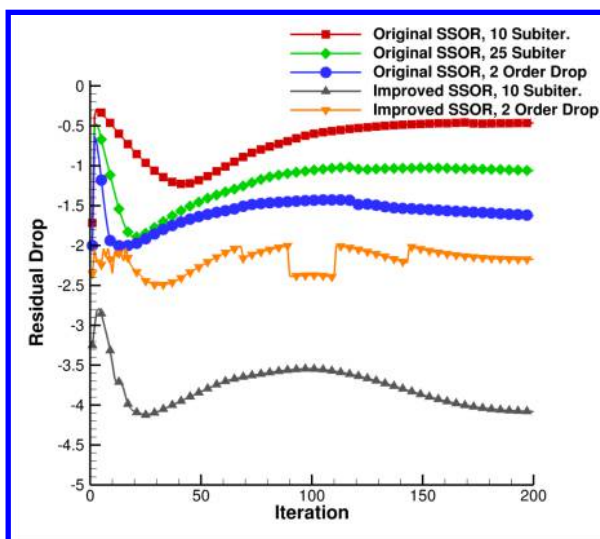
(c) Unsteady residual convergence at each time step.

Fig. 6 Initial residual level and subiteration convergence.

(a) All results.

(b) Selected results.

(c) Drag vs residual drop.

Fig. 7 Drag convergence during the subiteration process.

B. NACA 0015 Pitching Airfoil

An NACA 0015 airfoil oscillating in pitch [19] between 6.8 and 15.2 degrees at $M = 0.29$ with a chord Reynolds number of 1.95 million was studied in order to examine a time-varying case that includes grid motion. The HLLE++ flux scheme and SA-neg turbulence model were used, both with third-order MUSCL reconstruction. Unlike any of the other cases, this case was run in a staged mode in order to setup the meanflow at $\alpha = 6.8$ degrees, then two pitch cycles were performed at 10 subiterations per time step, and then details were examined on the third pitch cycle, with settings as discussed below.

Figure 8(a) shows the required number of subiterations to reach both a two and three order of magnitude residual drop per time step. Once more, the improved SSOR scheme is superior to the original SSOR scheme, with the 3 order of magnitude drop for the improved scheme requiring fewer subiterations than the 2 order of magnitude drop with the original scheme.
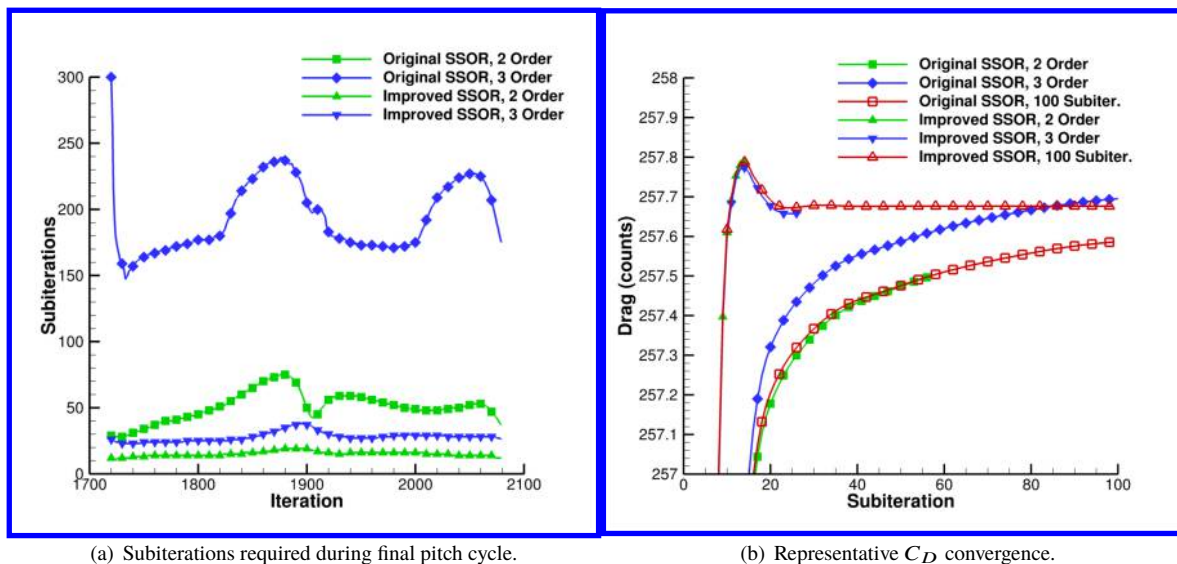

(a) Subiterations required during final pitch cycle.   (b) Representative C_D convergence.

Fig. 8   NACA 0015 results.

Figure 8(b) shows the drag convergence vs. subiteration for a representative iteration during the downstroke of the airfoil. As a control, a 100 subiteration case was also run for both linear solver schemes. Detailed examination of the results for the improved SSOR scheme show that the 2 order of magnitude drop solution insufficiently tracked the 100 iteration case, incurring an initial error in C_D of approximately a tenth of a drag count compared to the 100 subiteration case. The 3 order of magnitude case, with a maximum number of 40 subiterations for convergence, performed acceptably with an initial error on the order of a hundreth of a count. The original SSOR scheme faired much worse. Note how neither the 2 order drop nor the 100 subiteration case track the 3 order drop, indicating a build up in error over time, similar to the results of Section IV A. 3. This further reinforces the need for improved performance behavior and also demonstrates why an end user should always perform a convergence study to understand the computational effort needed to obtain a converged solution.

Overall, the solver improvements described in Section III certainly help single-grid, nonoverset convergence; this is useful, for example, for cases such as developing C81 airfoil tables for use with the rotor disk model [20] in OVERFLOW. The next examples present more challenging overset grid cases.

C. 3rd Drag Prediction Workshop

Case W1 of the 3rd AIAA Drag Prediction Workshop [21] provides an insightful test case to further highlight the linear solver improvements in OVERFLOW. The 'Boeing-overflow' grids provided on the workshop website[†] have multiple types of C-grid wake cuts, and in order to implicitly treat any type of C-grid with the improved SSOR scheme, grid blocks that contain C-grids are dynamically routed to the proper solver depending on the direction of the wake cut.

[†]https://aiaa-dpw.larc.nasa.gov/Workshop3/workshop3.html

An interesting feature of the grid system is that the wake cut for the wing is separated from the rest of the wing grid system by a trailing edge cap grid used to resolve the blunt trailing edge, as shown in Fig. 9(b).
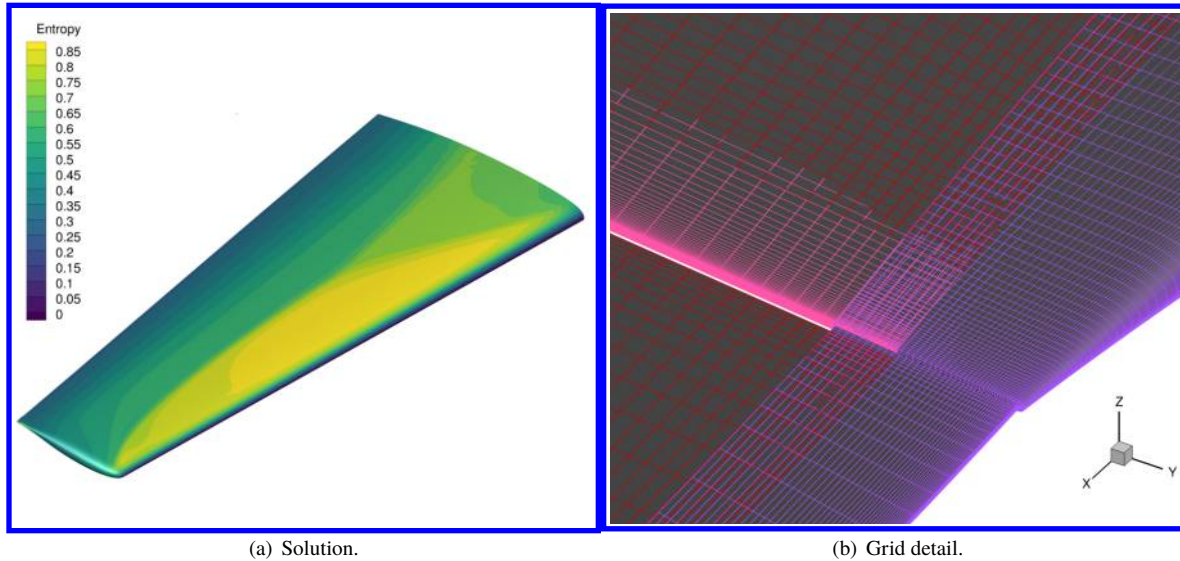

(a) Solution.　　　　　　　　　　　　　　　　　　　　　　　(b) Grid detail.

Fig. 9　DPW3 (a) solution and (b) detail of grid system near the wingtip trailing edge. Note the gap between the wing and wake-cut (red grid) due to the trailing edge cap (pink grid).

The 'medium' grid geometry of approximately 5 million nodes was studied at 0.5 degrees angle of attack, a Mach number of 0.76, fully turbulent SA-neg model, and Roe's flux with third-order MUSCL upwinding. Grid sequencing was used, but multigrid was not used due to the shock system on the upper surface of the wing, see Fig. 9(a). The scalar pentadiagonal, original SSOR, and improved SSOR schemes all used CFL ramping, starting from an initial CFL number of 1 and ramping to a maximum CFL number of 10 for the scalar pentadiagonal scheme and 50 for the SSOR schemes. The difference in maximum CFL is due to stability issues encountered by the scalar pentadiagonal scheme, which would cause it to slowly diverge; for this problem, it should be noted that this instability was present even at a maximum CFL number of 5. Dissipation settings for the scalar pentadiagonal scheme were not changed from the supplied input file. While the SSOR schemes are more robust, they can also suffer from instabilities that will result in stalled convergence at CFL numbers that are too high, but are generally less susceptible to divergence over long time periods.

As expected, the improved SSOR scheme reaches its maximum CFL value in fewer iterations than the the other schemes due to its improved convergence properties, as shown in Fig. 10(a). Because SER is used, the buzzing in the CFL number is directly related to the buzzing in the residual, as seen in Fig. 10(b). Note the behavior of the scalar pentadiagonal scheme; as mentioned, it slowly begins to diverge over time, which has an impact on the force and moment convergence.

For this case, the improved SSOR scheme, per iteration, is approximately 5% more expensive in terms of wall clock time than the original SSOR scheme, and both of the SSOR schemes are approximately 5 times more costly than the scalar pentadiagonal scheme. This means that even though the scalar pentadiagonal scheme required more iterations to reach its maximum CFL value, it started to display instabilities sooner than the other schemes.

The relative error in forces and pitching moment vs. time is plotted in Fig. 11, using an appropriately averaged value as the 'truth' value for each case. The C_D plots also include markers indicating convergence in terms of drag counts. The scalar pentadiagonal scheme initially converges very well, but as seen in the CFL number and residual plots, it becomes susceptible to an instability that eventually causes the forces and moments to oscillate. It is only through comparison with the solutions obtained with the SSOR schemes that it could be seen that the computed forces and moments were similar enough to be considered acceptable. The improved SSOR scheme once again presents better convergence and solution properties than the original SSOR scheme, reaching a 'steady-state' with a lower amplitude oscillation in the forces and moments than the other schemes. Clearly there is a trade-off between time and stability between the scalar pentadiagonal scheme and the improved SSOR scheme, but given the increased robustness of the SSOR scheme over long time integrations, and the cost associated with a failed run, the improved SSOR scheme is a

worthwhile competitor to the scalar pentagdiagonal scheme.
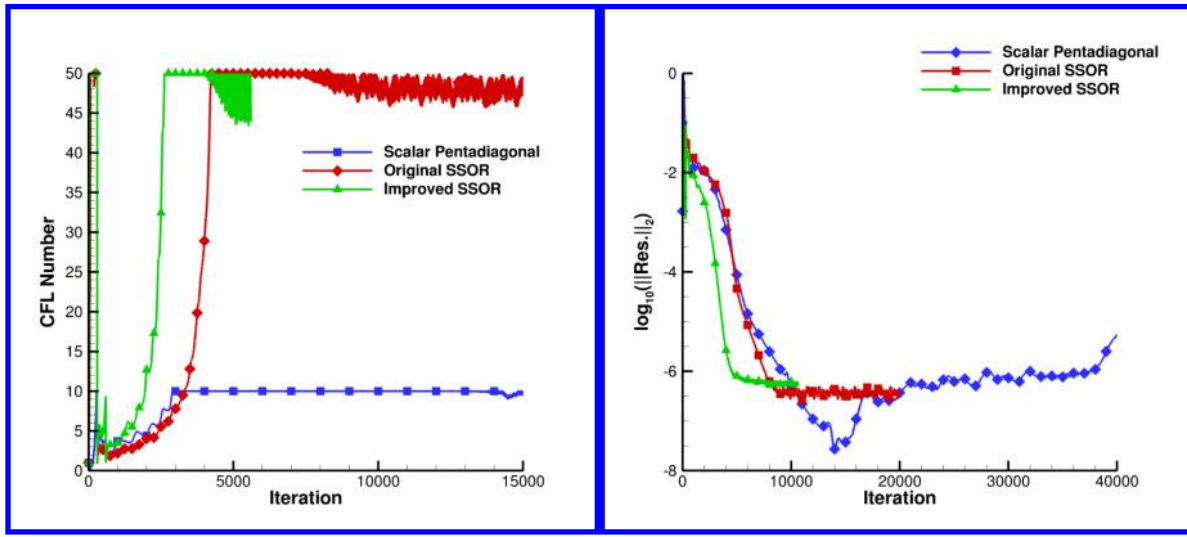
(a) CFL ramping. Some iterations have been removed for clarity.

(b) Residual convergence for wing grid.

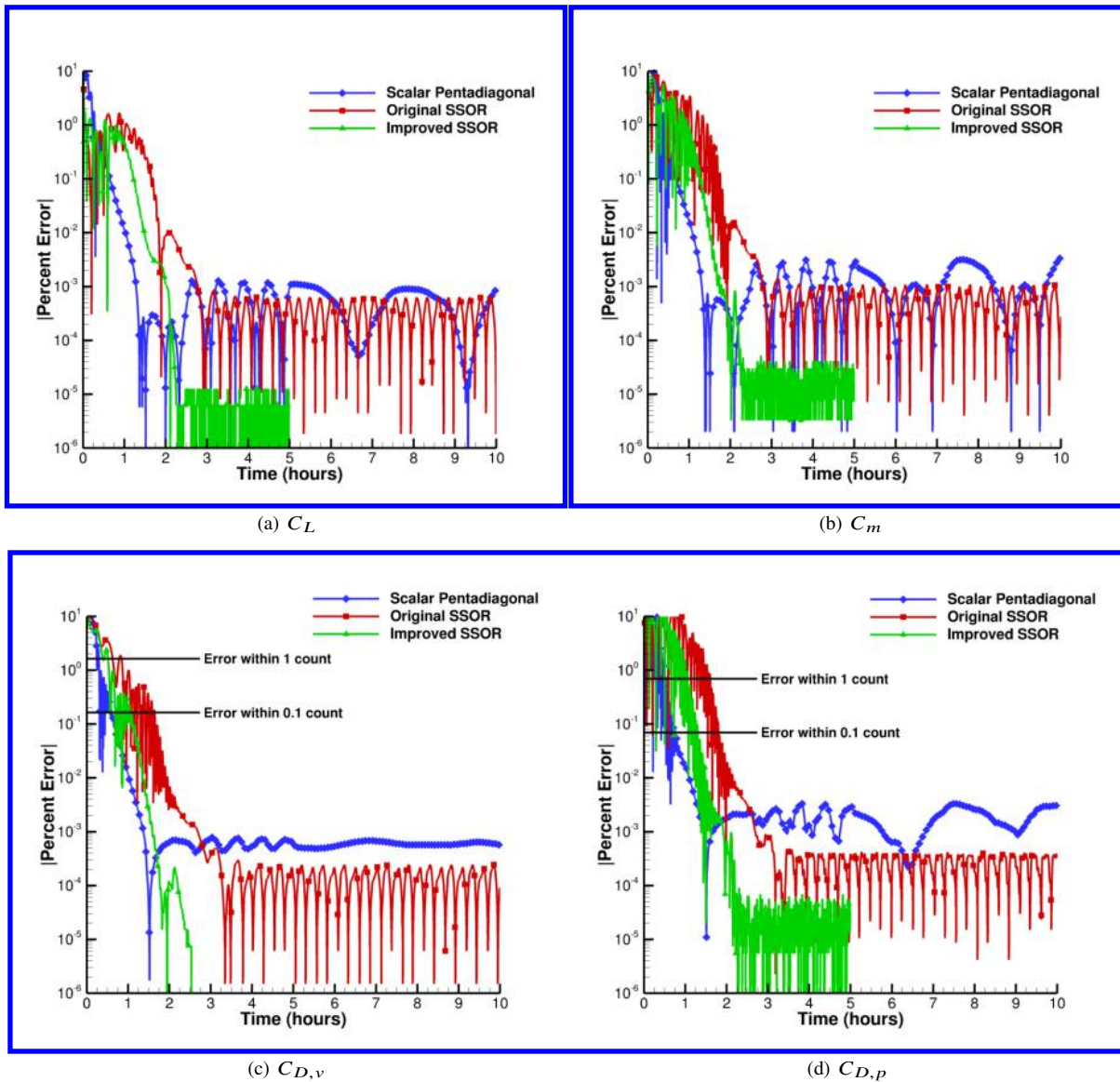**Fig. 10   DPW3 convergence behavior.**

(a) C_L

(b) C_m

(c) $C_{D,v}$

(d) $C_{D,p}$

Fig. 11 DPW3 force and moment percent error vs. time.

13

D. 6th Drag Prediction Workshop

Case 2A from the 6th AIAA Drag Prediction Workshop [22] is the last example for this work. The geometry consists of an aerodynamically deformed wing-body with freestream Mach number of 0.85 and a Reynolds number of 5 million based on the mean aerodynamic chord. The 'coarse' grid geometry, available from the workshop website[‡], consisting of approximately 14 million grid nodes, was used. An alpha seek was performed to match a desired condition of $C_L = 0.5$, and then OVERFLOW was run from an impulsive start using both the original and improved SSOR versions. The HLLC flux scheme, third-order MUSCL variable reconstruction, and fully turbulent SA-neg turbulence model were used. In additon, both schemes used grid sequencing, multigrid, and CFL ramping, with CFL capped at a maximum of 50. While the improved SSOR scheme is convergent at higher CFL numbers, the goal for this case is a head-to-head comparison. Because of the expectation of issues with side-of-body separation, the collar grid was only allowed to ramp from a CFL number of 0.1 to 5 to avoid causing oscillations/buzz in the convergence.

Figure 12 shows the residual convergence for both the original and improved SSOR schemes for grids with viscous wall boundary conditions. The improved SSOR scheme, with its better linearizations, shows faster convergence mainly due to being able to ramp to a higher CFL value at a faster rate than the original SSOR scheme. All of the grids, with the exception of the wing and collar grids, converge toward machine zero. The side-of-body separation bubble causes the convergence for the wing and collar grids to stall, indicating that this solution would be a good candidate for a time-accuracte comparsion.
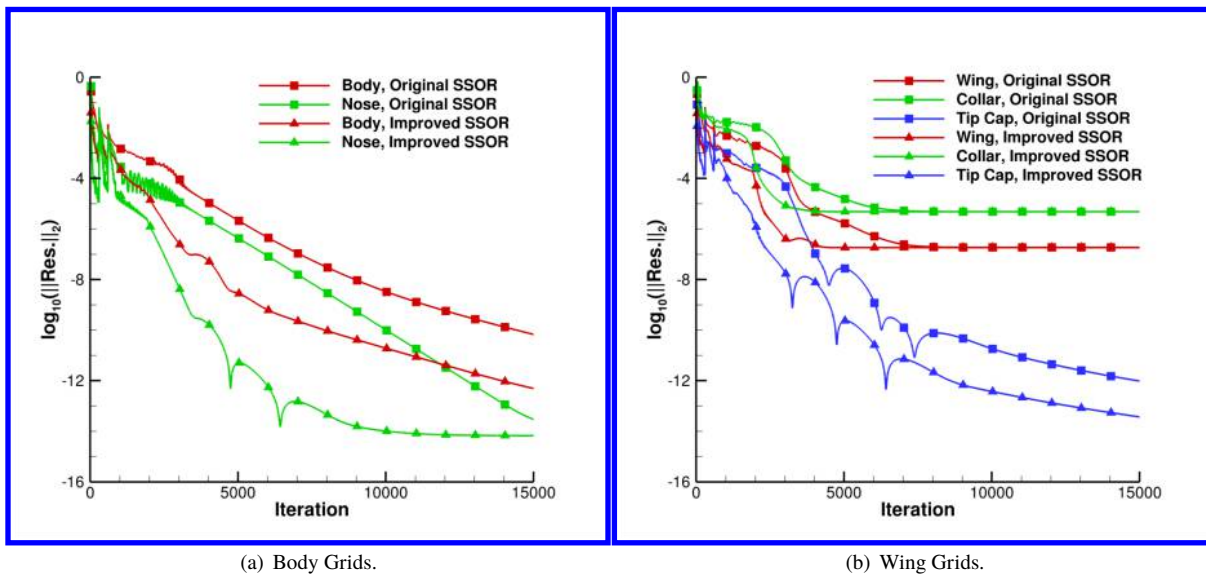

(a) Body Grids.

(b) Wing Grids.

Fig. 12   DPW6 convergence comparison.

Force and moment convergence as a percent error versus time is shown in Fig. 13. Both schemes converged to the same truth value, which was taken to be the value at the final iteration, after a runtime of approximately 30 hours. The plots of C_L, $C_{D,p}$, and $C_{D,v}$ include markers that indicate certain convergence tolerances. For this case, the improved SSOR scheme converged to the required $C_L = 0.5 \pm 0.0001$ condition in approximately 2.5 hours, while the original SSOR scheme requires 3.5 hours, as shown in Fig. 13(a). Unlike the previous example, there was no oscillation in the forces and moments at 'steady-state.' In the absence of CFL ramping, the original SSOR scheme would have required much longer to converge, as indicated by the previous examples in this work.
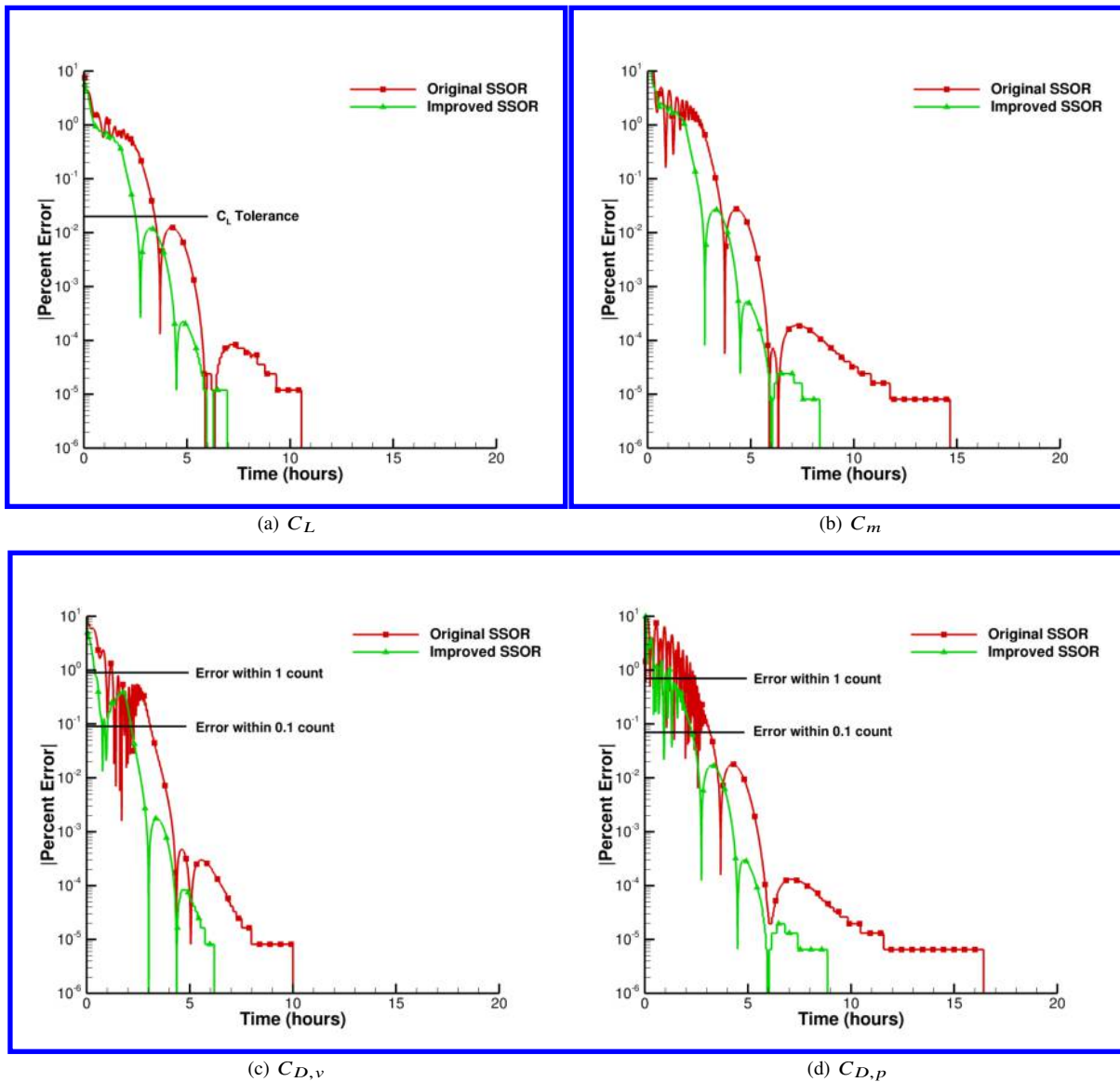

[‡]https://aiaa-dpw.larc.nasa.gov/

14

(a) C_L

(b) C_m

(c) $C_{D,v}$

(d) $C_{D,p}$

Fig. 13   DPW6 force and moment percent error vs. time.

15

V. Conclusions

Upgrades to the OVERFLOW flow solver have been presented whose cumulative effect is to greatly decrease the cost associated with the use of upwind schemes and the SSOR solver. While each upgrade is relatively minor, the sum of their effects greatly improve the convergence properties of OVERFLOW. In testing, time to convergence benefits were universally noted across a range of steady and unsteady problems. For unsteady problems, the CFL ramping feature was less useful due to the number of iterations required to ramp up to a higher CFL number, and so this feature has been highlighted in this work with respect to nominally steady state problems.

Each of the chosen examples highlight some of the benefits of the new linear solver options, but also hint at areas for future work. Updating the explicit overset boundaries to an implicit treatment, most likely through the use of a Krylov method, could possibly treat a majority of the remaining convergence problems suffered by OVERFLOW. In addition, not all boundary conditions have been linearized and this must be addressed through a combination of improved linear solvers and revisiting the boundary condition formulations. As these areas are addressed, issues with other turbulence models and species convection can be treated to improve the convergence behavior of the entire solver.

Overall, the improved SSOR solver is a useful addition to OVERFLOW, providing faster convergence (often on the order of 5× faster), improved convergence characteristics, and minimal cost per iteration impact compared to the original SSOR scheme.

Acknowledgments

References

[1] Nichols, R. H., Tramel, R. W., and Buning, P. G., "Solver and Turbulence Model Upgrades to OVERFLOW 2 for Unsteady and High-Speed Applications," AIAA Paper 2006–2824, 2006. doi:10.2514/6.2006-2824.

[2] Steger, J., and Warming, R., "Flux-Vector Splitting of the Inviscid Gasdynamics Equations with Application to Finite Difference Methods," *Journal of Computational Physics*, Vol. 40, No. 2, 1981, pp. 263 – 293. doi:10.1016/0021-9991(81)90210-2.

[3] Pulliam, T. H., and Steger, J. L., "Implicit Finite-Difference Simulations of Three-Dimensional Compressible Flow," *AIAA Journal*, Vol. 18, No. 2, 1980, pp. 159–167. doi:10.2514/3.50745.

[4] Beam, R., and Warming, R., "An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation-Law-Form," *Journal of Computational Physics*, Vol. 22, 1976, pp. 87 – 110. doi:10.1016/0021-9991(76)90110-8.

[5] Pulliam, T., and Chaussee, D., "A Diagonal Form of an Implicit Approximate-Factorization Algorithm," *Journal of Computational Physics*, Vol. 39, No. 2, 1981, pp. 347 – 363. doi:10.1016/0021-9991(81)90156-X.

[6] Klopfer, G. H., van der Wijngaart, R. F., Hung, C. M., and Onufer, J. T., "A Diagonalized Diagonal Dominant Alternating Direction Implicit (D3ADI) Scheme and Subiteration Correction," AIAA Paper 1998-2824, 1998. doi:10.2514/6.1998-2824.

[7] South, J., Hafez, M., and Gottlieb, D., "Stability analysis of intermediate boundary conditions in approximate factorization schemes," *Applied Numerical Mathematics*, Vol. 2, No. 3, 1986, pp. 181 – 192. doi:10.1016/0168-9274(86)90027-9, special Issue in Honor of Milt Rose's Sixtieth Birthday.

[8] MacCormack, R., "Current Status of Numerical Solutions of the Navier-Stokes Equations," AIAA Paper 1985-0032, 1985. doi:10.2514/6.1985-32.

[9] Ceze, M., and Fidkowski, K. J., "A Robust Adaptive Solution Strategy for High-Order Implicit CFD Solvers," AIAA Paper 2011–3696, 2011. doi:10.2514/6.2011-3696.

[10] Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372. doi:10.1016/0021-9991(81)90128-5.

[11] Toro, E. F., Spruce, M., and Speares, W., "Restoration of the Contact Surface in the HLL-Riemann Solver," *Shock Waves*, Vol. 4, No. 1, 1994, pp. 25–34. doi:10.1007/BF014146292.

[12] Tramel, R. W., Nichols, R. H., and Buning, P. G., "Addition of Improved Shock-Capturing Schemes to OVERFLOW 2.1," AIAA Paper 2009–3988, 2009. doi:10.2514/6.2009-3988.

[13] Pandya, S., Venkateswaran, S., and Pulliam, T., "Implementation of Preconditioned Dual-Time Procedures in OVERFLOW," AIAA Paper 2003–0072, 2003. doi:10.2514/6.2003-72.

[14] Mulder, W. A., and van Leer, B., "Experiments With Implicit Upwind Methods for the Euler Equations," *Journal of Computational Physics*, Vol. 59, 1985, pp. 232–246. doi:10.1016/0021-9991(85)90144-5.

[15] Somers, D. M., "Design and Experimental Results for the S809 Airfoil," NREL/SR 440-6918, 1997.

[16] Allmaras, S. R., Johnson, F. T., and Spalart, P. R., "Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model," *Seventh International Conference on Computational Fluid Dynamics (ICCFD7)*, 2012.

[17] Langtry, R. B., and Menter, F. R., "Correlation-Based Transition Modeling for Unstructured Parallelized Computational Fluid Dynamics Codes," *AIAA Journal*, Vol. 47, No. 12, 2009, pp. 2894–2906. doi:10.2514/1.42362.

[18] van Leer, B., "Towards the Ultimate Conservative Difference Scheme. V. A Second-Order Sequel to Godunov's Method," *Journal of Computational Physics*, Vol. 32, 1979, pp. 101–136.

[19] Piziali, R. A., "2-D and 3-D Oscillating Wing Aerodynamics for a Range of Angles of Attack Including Stall," NASA TM-4632, Ames Research Center, Sep. 1994.

[20] Chaffin, M., and Berry, J., "Helicopter Fuselage Aerodynamics Under a Rotor by Navier-Stokes Simulation," *Journal of the American Helicopter Society*, Vol. 42, No. 3, 1997, pp. 235–243.

[21] Vassberg, J. C., Tinoco, E., Mani, M., Brodersen, O., Eisfeld, B., Wahls, R., Morrison, J. H., Zickuhr, T., Laflin, K., and Mavriplis, D., "Summary of the Third AIAA CFD Drag Prediction Workshop," AIAA Paper 2007–260, 2007.

[22] "6th AIAA CFD Drag Prediction Workshop," , 2018. `http://aiaa-dpw.larc.nasa.gov`.