Unstructured Mesh Solution Techniques using the NSU3D Solver

Dimitri J. Mavriplis [*]
Karthik Mani [†]
Department of Mechanical Engineering, University of Wyoming, Laramie, WY 82071

NSU3D is an unstructured mesh Reynolds-averaged Navier-Stokes (RANS) solver designed for aerodynamic flow problems. This paper describes the basic solution techniques implemented and available within NSU3D for both steady-state aerodynamic problems and time-dependent simulations. The base solver consists of a multi-stage point-implicit approach that inverts small block-diagonal matrices at each mesh point. This is generalized to a line-Jacobi method, using lines generated within the unstructured mesh based on the strength of the edge connections between neighboring vertices. The line-Jacobi solver is used as a smoother for an agglomeration multigrid solver which makes use of coarse agglomerated mesh levels to accelerate the convergence of the overall solution procedure based on geometric and algebraic multigrid principles. The multigrid solver, in turn, can be used as a preconditioner for a GMRES algorithm that provides further robustness and convergence efficiency for large difficult cases. Different variants of these solvers are available as both linear and non-linear formulations and are applicable to steady as well as time-dependent cases. The numerical performance and parallel scalability of these solvers are illustrated on two representative steady-state aerodynamic RANS problems, and one unsteady time-implicit problem. The advantages and drawbacks of the various solvers are discussed, and prospects for future solver development are given.

I. Introduction

Current-day production level computational aerodynamics problems generally involve complex geometries, have high accuracy requirements, and must run on massively parallel computer architectures. The ability to handle complex geometries dictates the use of non-simple mesh topologies, which may include block-structured, overset, and unstructured meshes, and/or any combination of these techniques. Increasing accuracy requirements coupled with more capable computer hardware has led to consistent growth in grid sizes with grids in the range of 100 million points becoming feasible on a regular basis.[1–3] At the same time, hardware parallelism has expanded dramatically such that simulations using thousands of cores are common today, with numerous simulations using over 100,000 cores having been demonstrated on leading-edge hardware. These requirements place considerable strains on CFD solver technology. Competitive solvers must be able to converge efficiently and robustly the large systems of non-linear equations that arise from discretizations on complex mesh topologies while mapping effectively to the latest HPC hardware.

Current solvers are far from optimal, and considerable advances in efficiency and robustness remain possible that could greatly impact the cost effectiveness of CFD tools. Furthermore, the potential gains from solver technology will only increase in the future with growing problem sizes and hardware parallelism. Therefore, it is useful to examine the current state of various production level solvers, in order to assess their performance, scalability and robustness, and to identify the most promising areas for improvement.

In this paper, we describe and demonstrate the relative performance of the various solvers in the NSU3D unstructured Reynolds-averaged Navier Stokes (RANS) code. NSU3D has been in production use for over a decade and has been a regular participant code in the AIAA sponsored drag prediction workshop,[4–8] high-lift prediction workshop,[9,10] and aeroelastic prediction workshop series.[11] The solvers in NSU3D have been

[*]Professor, AIAA Associate Fellow; email: mavriplis@uwyo.edu
[†]Associate Research Scientist, Member AIAA; email: kmani@uwyo.edu

conceived over the years specifically for addressing high-Reynolds number steady-state[12] and more recently implicit-time dependent RANS aerodynamic problems.[13,14] These include a directional line-solver approach for addressing anisotropic mesh stiffness,[15,16] multigrid methods for maintaining good convergence rates on high resolution meshes,[12,17–19] and linear solver variants for improved performance and robustness.[20] Furthermore, all solvers have been designed to deliver identical convergence histories to machine precision regardless of the number of processors or mesh partitioning used in the solution process.

In the following section, the spatial and temporal discretizations used in the NSU3D code are briefly presented. In section III, a more detailed discussion of the various solver techniques available in the code is given. This is followed by the Results section, where the performance of the various solvers is examined on two steady-state cases, and one implicit time-dependent case. An argument in favor of the scalability of multigrid methods is also made in Section V, and conclusions are drawn in the final section.

II. Discretization

The equations to be solved can be written as:

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

where $\frac{\partial \mathbf{w}}{\partial t}$ represents the time derivative and $\mathbf{R}(\mathbf{w})$ represents the spatial discretization terms. The spatial discretization in NSU3D follows a vertex-centered approach where the unknown fluid flow and turbulence variables are stored at locations corresponding to the mesh points, and fluxes are computed on faces delimiting dual control volumes, with each dual face being associated with a mesh edge. This discretization operates on hybrid mixed-element meshes, generally employing prismatic elements in highly stretched boundary-layer regions, and tetrahedral elements in isotropic regions of the mesh away from the aircraft surfaces, with a small number of pyramidal elements at prismatic-tetrahedral element interfaces. A single edge-based data-structure is used to compute flux balances across all types of elements. The convective terms are discretized as central differences with added matrix dissipation. Second-order accuracy is achieved by formulating these dissipative terms as an undivided bi-harmonic operator, which is constructed in two passes of a nearest-neighbor Laplacian operator. In the matrix form, this dissipation is similar to that produced by a Riemann solver gradient-based reconstruction scheme, and is obtained by replacing the difference in the reconstructed states on each side of the control volume interface by the undivided differences along mesh edges resulting from the biharmonic operator construction. These differences are then multiplied by the characteristic matrix to obtain the final dissipation terms. A Roe upwind scheme using least-squares gradient reconstruction is also available in the NSU3D solver. The baseline NSU3D discretization employs an edge-based finite-difference scheme to approximate the thin-layer form of the viscous terms for the Navier-Stokes equations, although this is done in a multidimensional fashion, by computing a Laplacian of the velocity field.[12] The main approximation in this approach is the omission of the cross-derivative viscous terms, and the assumption of a locally constant viscosity, which corresponds to an incompressible form of the full Navier-Stokes terms. The discretization of the full Navier-Stokes terms based on a hybrid gradient-of-gradient approach is also available, where the second derivatives in the directions aligned with the incident mesh edges are computed using a nearest-neighbor stencil.[21] NSU3D incorporates the single equation Spalart-Allmaras turbulence model[22] as well as a standard Wilcox $k - \omega$ two-equation turbulence model[23] and the two-equation shear-stress transport (SST) model due to Menter.[24]

For steady-state problems, we seek the solution in the limit of vanishing time derivative terms, and thus the non-linear system of equations to be solved corresponds to:

$$\mathbf{R}(\mathbf{w}) = \mathbf{0} \tag{2}$$

For time-dependent problems, the time derivative term can be discretized in a variety of manners. In most cases, an implicit time-stepping formulation is preferred, in order to avoid the severe time-step stability limitations of explicit schemes. NSU3D incorporates first-order, second-order and third-order backwards difference schemes (BDF1, BDF2, BDF3),[25] a variety of diagonally implicit Runge-Kutta schemes,[13,26] and a time-spectral implementation.[27] The most commonly used temporal discretization is the BDF2 scheme, which can be written as:

$$\frac{\frac{3}{2}\mathbf{w}^{n+1} - 2\mathbf{w}^{n} + \frac{1}{2}\mathbf{w}^{n-1}}{\Delta t} + \mathbf{R}(\mathbf{w}^{n+1}) = 0 \tag{3}$$

where Δt denotes the physical time step, w^{n+1} represents the unknown values at the current time step and w^n and w^{n-1} correspond to the solution values known from the previous two time steps. Thus, the equations to be solved at each implicit time step can be written as:

$$\mathbf{S}(\mathbf{w}^{n+1}) = \frac{3\mathbf{w}^{n+1}}{2\Delta t} + \mathbf{R}(\mathbf{w}^{n+1}) - \frac{2\mathbf{w}^n}{\Delta t} + \frac{\mathbf{w}^{n-1}}{2\Delta t} = 0 \tag{4}$$

For all cases discussed in this work, the matrix dissipation scheme and the multidimensional thin-layer viscous term approximation are used. Additionally, the Spalart-Allmaras turbulence model is used exclusively in the following examples, and the BDF2 temporal discretization is used in the time-dependent case.

III. Solution Methodologies

The NSU3D unstructured mesh CFD code incorporates a variety of related solution techniques. A local implicit solution technique forms the baseline approach which can be used as a driver for an agglomeration multigrid method. These solvers can be used either in non-linear form where they are applied directly to the non-linear equations to be solved, or in linearized form, where they are used to solve the linear system arising at each step in a Newton scheme applied to the non-linear system to be solved.[20] Additionally, the linear form of the solver may be used as a preconditioner for a Newton-Krylov method (GMRES)[28,29] applied to the non-linear Reynolds averaged Navier-Stokes equations. Finally, all solvers may be applied to the solution of steady-state problems, or to the solution of the non-linear problem arising at each physical time step within the context of an implicit time-stepping problem. All solution techniques have been developed to operate with good scalability in a massively parallel computing environment and have been designed to produce identical convergence histories (to machine precision) regardless of the number of processors or mesh partitioning used.

A. Base Solver

In order to describe the solution techniques in the NSU3D code, we consider a large non-linear system of equations to be solved, denoted as:

$$\mathbf{R_i}(\mathbf{w_j}) = 0, \quad i = 1, 2, .., N \quad j = 1, 2, ..., N \tag{5}$$

where $\mathbf{R_i}$ represents the residual at the i^{th} grid point and $\mathbf{w_j}$ represent the unknowns at the j^{th} grid point, with N denoting the number of grid points or unknown locations in the mesh. In general, $\mathbf{w_j}$ represents a vector of 6 or 7 degrees of freedom at a single location j, since the RANS equations involve the determination of 5 conservation equations in three dimensions, and a single-equation or two-equation turbulence model. Similarly, $\mathbf{R_i}$ represents a vector of 6 or 7 coupled equations to be solved at the point i. Equation (5) is used generically to represent either equation (2) or (4) depending on whether the case to be solved corresponds to a steady-state or implicit time-stepping problem.

A Newton scheme applied to equation (5) can be written as:

$$\frac{\partial \mathbf{R}(\mathbf{w})}{\partial \mathbf{w}} \Delta w = -\mathbf{R}(\mathbf{w}) \tag{6}$$

A common simplification consists of replacing the exact Jacobian matrix $\frac{\partial \mathbf{R}(\mathbf{w})}{\partial \mathbf{w}}$ by an approximate matrix corresponding to the Jacobian of a first-order discretization as:

$$\frac{\partial \mathbf{R_{1st}}(\mathbf{w})}{\partial \mathbf{w}} \Delta w = -\mathbf{R}(\mathbf{w}) \tag{7}$$

Due to the nearest neighbor stencil of the corresponding first-order discretization, the resulting Jacobian matrix has a simple 6×6 or 7×7 block structure, where each diagonal block $[D_i]$ is associated with a grid vertex or control volume, and all off-diagonal blocks $[O_{ij}]$ are associated with mesh edges that join two neighboring vertices i and j.[17]

The baseline iterative solver in NSU3D can be viewed as a further simplification where all off-diagonal terms in the Jacobian are dropped resulting in the following point implicit iterative scheme:

$$\mathbf{\Delta w_i} = -[D]_i^{-1} \mathbf{R_i}(\mathbf{w}) \quad i = 1, 2, ..., N \tag{8}$$

which requires only the inversion of the local block matrix $[D_i]$ at each point or control volume i. Since the above approach reduces to an explicit scheme for scalar equations, the block diagonal inverse matrix $[D_i]^{-1}$ may be interpreted as a matrix time step. This suggests a formal extension of multi-stage explicit Runge-Kutta schemes to systems of equations as:

$$\begin{aligned}
\mathbf{w}^{(0)} &= \mathbf{w}^{(n)} \\
\mathbf{w}^{(1)} &= \mathbf{w}^{(0)} - CFL\,\alpha_1\,[D_i]^{-1}\,\mathbf{R}(\mathbf{w^{(0)}}) \\
\mathbf{w}^{(2)} &= \mathbf{w}^{(0)} - CFL\,\alpha_2\,[D_i]^{-1}\,\mathbf{R}(\mathbf{w^{(1)}}) \\
\mathbf{w}^{(3)} &= \mathbf{w}^{(0)} - CFL\,\alpha_3\,[D_i]^{-1}\,\mathbf{R}(\mathbf{w^{(2)}}) \\
\mathbf{w}^{(n+1)} &= \mathbf{w}^{(3)}
\end{aligned} \tag{9}$$

for a three-stage scheme, where α_i denote the Runge-Kutta coefficients, and the CFL number is given as CFL. Multi-stage Runge-Kutta schemes were initially used by Jameson to solve the Euler equations[30] and have been extensively refined to provide good high frequency error damping properties for multigrid algorithms.[31,32] The baseline solver in the NSU3D code corresponds to a three-stage point-implicit Runge-Kutta scheme as shown in equation (9) optimized for high-frequency error damping. This solver requires the inversion of the block diagonal matrices, which is performed using a simple LU factorization algorithm. The solver either performs factorization on the full blocks, or on simplified blocks where the coupling between mean flow and turbulence equations is dropped, resulting in a simpler 5×5 block inversion, along with an additional scalar inversion or 2×2 block inversion for single-equation and two-equation turbulence models, respectively. In order to save computational effort, the block matrices are factorized once and frozen for all three stages of the Runge-Kutta scheme.

B. Line Solver

The use of highly stretched meshes in boundary-layer and wake regions for high-Reynolds number RANS problems induces considerable stiffness in the discrete equations causing slower convergence. In order to alleviate this stiffness, additional implicitness in the local solver is required in the direction of strong coupling in regions of high mesh anisotropy. For structured mesh problems, this can be achieved by solving simultaneously for all variables along normal lines in boundary-layer and wake regions, requiring the inversion of individual tridiagonal matrices for each normal mesh line. For unstructured meshes, similar benefits can be obtained provided line structures can be extracted from the underlying graph of the mesh. For vertex-based discretizations, this can be achieved by grouping together contiguous edges in the mesh that join tightly coupled vertices. In order to preserve the local tridiagonal Jacobian matrix structure, individual edge sets must be non-intersecting and topologically equivalent to line segments.

In principle, line structures can be extracted from arbitrary unstructured meshes using a weighted graph algorithm that is initiated in regions of high mesh anisotropy and proceeds in a greedy fashion towards regions of lower anisotropy, with lines terminating when nearly isotropic mesh elements are encountered.[15] For hybrid unstructured meshes that employ prismatic elements in boundary-layer regions and tetrahedral elements in inviscid flow regions, the current NSU3D implementation constructs lines by grouping together contiguous edges joining only prismatic element vertices in the direction normal to the boundary-layer surface, as identified by the element type and wall boundary condition. Although a graph algorithm approach offers more generality and the potential for superior solver efficiency, the current approach was adopted due to robustness considerations.

Generation of line structures in unstructured meshes results in a set of lines of varying length that do not span the entire mesh, as illustrated in Figure 1. For each identified edge joining points ij in the line set, the corresponding off-diagonal block matrix entries $[O_{ij}]$ and $[O_{ji}]$ are retained in the simplified Jacobian matrix, while all other off-diagonal entries are dropped, as previously. The mesh vertices and edges are then reordered resulting in a local tridiagonal Jacobian matrix structure for each line, and the local solver can then proceed analogously as described in equations (8) and (9), where the block diagonal matrix inversion is replaced with the corresponding locally inverted or factorized tridiagonal matrix. The implementation naturally handles lines of varying length, where lines of length 0 (1 vertex, zero edges) reduce to the previously described point implicit solver. As previously, the local tridiagonal matrices are factorized and frozen for all stages of the Runge-Kutta solver.

Tridiagonal line solvers scale linearly with the number of unknowns (or line length)[33] and the current line solver implementation incurs minimal additional computational expense compared to the point solver, while

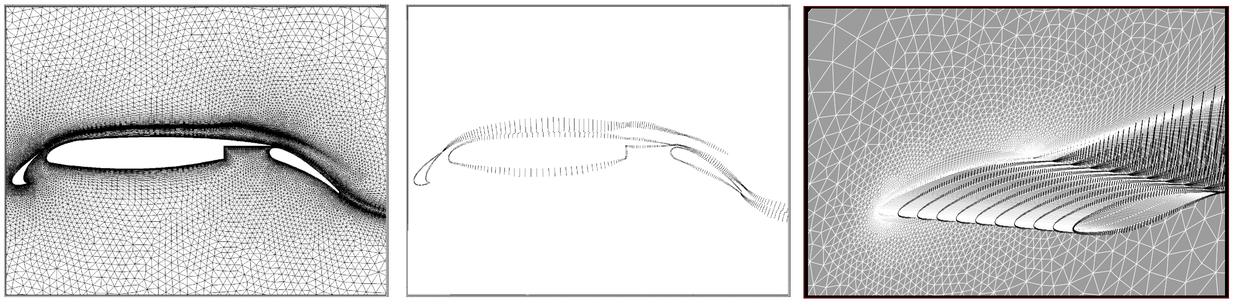


Figure 1. Illustration of implicit line structures extracted from unstructured meshes in two and three dimensions

providing faster convergence in regions of high mesh anisotropy. However, the factorization of each individual local line solver is an inherently sequential (non-parallelizable) process. This results in the requirement that all lines be fully contained within an individual mesh partition and do not cross processor boundaries in a parallel computing environment. This is achieved at run time by partitioning "around" line structures, using a contracted weighted graph, where all vertices along individual lines are contracted into a single weighted vertex prior to the graph partitioning process.[16,34] In this manner, lines can never be broken across partition boundaries, and good partitioning quality is maintained due to the associated weighting used in the partition graph. A simpler approach consists of partitioning the original uncontracted mesh graph while placing very large weights on individual line edges, in order to penalize the splitting of these edges in the partitioner. However, experience has shown that this approach consistently results in a small number of split line edges, with the consequence that identical solver characteristics and convergence behavior can no longer be guaranteed for parallel runs using different numbers of processors.

C. Agglomeration Multigrid

Because the line and point solvers described above are local methods, their performance does not scale optimally with problem size and significant convergence degradation is observed for larger problems. In general, a more global solution strategy is required to maintain favorable convergence characteristics with increasing problem size. Multigrid methods provide an optimal solution strategy for large problems and theoretically are capable of delivering convergence rates that are independent of the number of unknowns.[35,36] The idea of a multigrid algorithm is to use coarser grid levels to accelerate the solution of the fine grid equations. A multigrid method typically begins by partially solving the problem on the fine grid. The solution and residuals are then transferred up to a coarser grid, generating a new coarse grid problem that is again partially solved. The process is repeated recursively on a sequence of coarser grids until the coarsest grid of the sequence is reached, at which point the solution corrections are interpolated back to the finest grid, constituting a complete multigrid cycle, after which the process is repeated. Effective multigrid methods make use of local solvers designed to reduce high frequency error components on each grid level.[37] Thus, in the present context, the local line and point solvers, in conjunction with the three-stage Runge-Kutta scheme designed for high-frequency damping properties, are used as drivers on each level of the multigrid sequence.

An important aspect of multigrid algorithms is that the coarser level problems do not simply correspond to lower resolution versions of the fine grid physical problem, but rather are formulated as correction equations for the fine level problem, and thus the solution of modified equations is required on the coarse grid. Multigrid methods may be applied directly to non-linear problems, as in the Full Approximation Storage (FAS) multigrid method.[19,35–37] Given the fine level non-linear equations to be solved written as

$$\mathbf{R_h}(\mathbf{w_h}) = 0 \tag{10}$$

where h denotes the fine grid level, a non-linear coarse grid equation is formulated as:

$$\mathbf{R_H}(\mathbf{w_H}) = \mathbf{R_H}(\hat{I}_h^H \mathbf{w_h}) - I_h^H \mathbf{R_h}(\mathbf{w_h}) = \mathbf{D_H} \tag{11}$$

where H denotes the coarse grid level, and I_h^H and $\hat{I}_h^H$ denote fine-to-coarse grid transfer operators for residuals and solution variables, respectively (which in general are not the same). After the coarse grid equation is solved, the corrections are interpolated or prolongated back to the fine grid following:

$$\mathbf{w_h} = \mathbf{w_h} + I_H^h(\mathbf{w_H} - \hat{I}_h^H \mathbf{w_h}) \tag{12}$$

where I_H^h denotes the coarse-to-fine grid transfer or prolongation operator. The term on the right-hand side of equation (11) is often referred to as the defect-correction and ensures that the coarse grid equations are driven by the fine grid residuals.[31,35] For example, when the fine grid residuals are identically zero, it is easily seen that the correction $(w_H - \hat{I}_h^H w_h)$ prolongated back to the fine grid vanishes, thus ensuring consistency between fine and coarse grid equations.

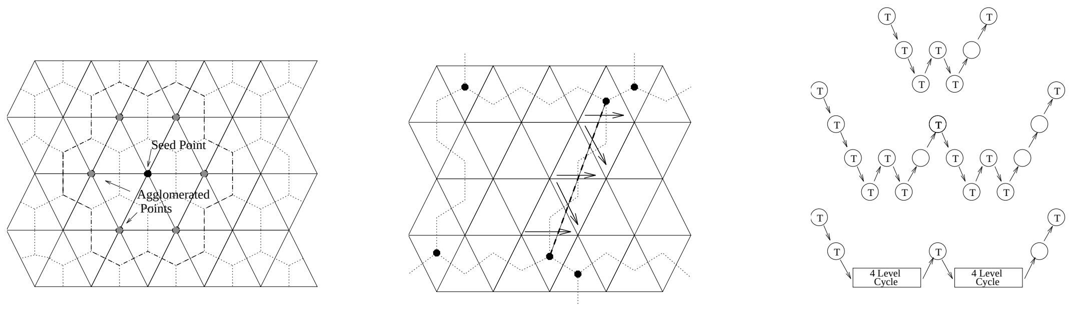


Figure 2. Illustration of (left) agglomeration of dual mesh control volumes to form (center) larger control volumes on coarse level and (right) definition of multigrid W-cycle

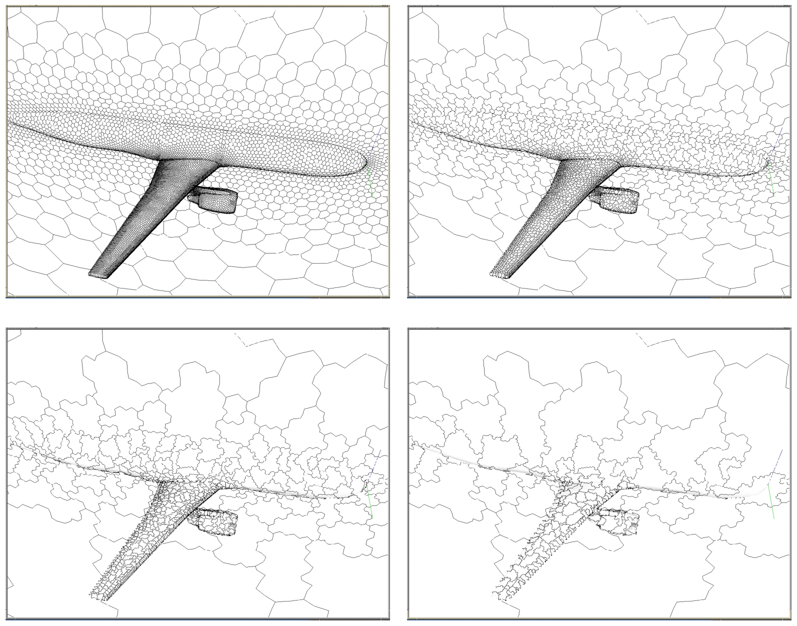


Figure 3. Sequence of four coarse agglomerated levels of unstructured mesh over aircraft configuration

Multigrid methods were originally conceived for structured mesh problems, where coarse grid levels could be constructed easily by removing every second grid line in each coordinate direction. For unstructured mesh problems, construction and use of suitable coarse levels has been a principle challenge. In practice, some type of merging or agglomeration approach is used, and this may be done in an algebraic sense, operating directly on the stencil of the discrete equations or resulting matrix, or geometrically, by merging mesh elements or control volumes together. The approach adopted in the NSU3D solver is somewhat of a hybrid method. A graph algorithm is used to merge together neighboring control volumes in the dual representation of the unstructured mesh used for the vertex-based discretization, as shown in Figure 2. Merging is based on a combination of topological considerations and weights based on grid metrics. The agglomeration process begins on solid wall boundaries, and then proceeds outwards along the implicit line structures, after which the inviscid regions are merged based on a minimal independent set graph algorithm implementation.[18] The process contains checks and modifications to avoid pathological configurations such as singleton agglomerated cells, and islands (cells completely surrounded by another single cell). An example of a three-dimensional agglomerated coarse level sequence is shown in Figure 3. The coarse level equations are discretized only to first-order accuracy to maintain robustness and stencil simplicity, and intergrid transfers are performed using piecewise constant interpolation, although the coarse-to-fine prolongation of corrections is followed by a smoothing operation. During the agglomeration process, boundary cells with multiple composite boundary conditions are formed. Therefore, the boundary condition implementation follows closely an algebraic multigrid strategy, where the composite boundary condition is formed as the sum of the constituent fine level boundary conditions. In fact, the entire coarse level discretization that is used can be recovered from an algebraic multigrid viewpoint using Galerkin projection as described in reference.[19]

Efficiency of the multigrid algorithm is closely tied to the complexity of the generated coarse levels. In three dimensions, uniform 2:1 structured mesh coarsening in each coordinate direction leads to coarse levels that contain 8 times fewer points or cells compared to the generating fine level. Using a V-cycle multigrid strategy where each mesh level is visited once during a single multigrid cycle leads to a total multigrid cycle complexity of:

$$1 + 1/8 + 1/64 + ... = 8/7 \tag{13}$$

The multigrid W-cycle is a recursive strategy where coarse grids are visited 2^{k-1} times, where k represents the grid level (finest grid : k=1), as illustrated in Figure 2. Thus the complexity of a structured mesh multigrid W-cycle is given as:

$$1 + 2/8 + 4/64 + ... = 4/3 \tag{14}$$

Note that the complexity of the W-cycle increases rapidly with lower coarsening ratios and becomes unbounded if the ratio is below 4:1. Because the multigrid W-cycle generally produces more effective convergence rates, suitably aggressive coarsening is desirable in the agglomeration process. Thus, the NSU3D agglomeration algorithm targets an 8:1 coarsening ratio, although in practice ratios of 7.5:1 for the finer levels are usually obtained, with lower ratios observed for the coarsest mesh levels. Currently, multigrid agglomeration is performed sequentially on a single processor. This is done as an integral part of the parallel preprocessor that performs all other preprocessing operations in parallel, using a memory-lean implementation of the agglomeration graph algorithm. This strategy enables the use of the same coarse levels for different mesh partitionings, thus guaranteeing identical convergence behavior on different numbers of processors. A parallel agglomeration strategy is under development, although at present it is not clear if efficient parallel agglomeration that is invariant to mesh partitioning will be feasible.

D. Linear Multigrid Solver

While the FAS multigrid scheme described above is designed to be applied directly to non-linear problems, a slightly modified version of the agglomeration multigrid method can be used to solve the linear problem arising at each step of a global Newton method applied to the flow equations. As previously, we choose to formulate an approximate Newton solver using a simplified first-order accurate Jacobian as shown in equation (7). Additionally, we retain the option of further modifying this Jacobian through the addition of a diagonal term simulating the effect of a pseudo time step $\Delta\tau$, resulting in the following linear system to be solved at each non-linear update:

$$\left[\frac{I}{\Delta\tau} + \frac{\partial \mathbf{R_{1st}}(\mathbf{w})}{\partial \mathbf{w}}\right] \Delta w = -\mathbf{R}(\mathbf{w}) \tag{15}$$

In practice the pseudo time step is taken as some constant factor (CFL) times an estimate of the local maximum stable time step for the corresponding explicit scheme. Clearly, when $\Delta\tau$ becomes very large equation (7) is recovered and fast non-linear convergence can be expected, whereas for small pseudo time steps, the Jacobian matrix becomes more diagonally dominant and thus easier to invert iteratively, although non-linear convergence efficiency is compromised.

In order to apply multigrid to the above linear system, we first write the residual of the linear system as:

$$\mathbf{r}_{linear} = \left[\frac{I}{\Delta\tau} + \frac{\partial \mathbf{R_{1st}}(\mathbf{w})}{\partial \mathbf{w}} \right] \Delta w + \mathbf{R}(\mathbf{w}) \tag{16}$$

Using the multigrid correction scheme (CS),[19,36,37] the coarse grid correction equation can be written as:

$$\left[\frac{I}{\Delta\tau} + \frac{\partial \mathbf{R_{1st}}(\mathbf{w})}{\partial \mathbf{w}} \right]_H \Delta w_H = -I_h^H \mathbf{r}_{linear} \tag{17}$$

where the subscript H denotes coarse grid values, and the operator I_h^H corresponds to the restriction operator transferring fine grid residuals to the coarse grid level. Once the coarse level corrections Δw_H have been solved, they are prolongated back to the fine level using the I_H^h operator and used to update the fine level solution as:

$$\begin{aligned} \Delta w_h &= \Delta w_h + I_H^h \Delta w_H \\ w_h &= w_h + \Delta w_h \end{aligned} \tag{18}$$

In practice, the same restriction and prolongation intergrid transfer operators are used for the non-linear (FAS) and linear (CS) multigrid algorithms. The coarse level Jacobian matrix on the left hand side of equation (17) can either be assembled by summing the elements of the fine level Jacobian matrix following the control-volume agglomeration patterns in an algebraic multigrid fashion, or by rediscretizing the equations on the coarse level as is done in the FAS multigrid scheme, although using frozen state variables at the given fine level non-linear state. We have investigated both approaches and found little difference in performance, and thus the latter approach is generally used since this leverages the implementation done for the FAS multigrid scheme.

Although we have described the use of multigrid to solve equation (15), in practice any linear iterative solver may be used including the baseline line and point solvers in the absence of multigrid. One of the advantages of the linear solver approach is that the cost of a linear solver cycle depends only on the stencil of the Jacobian, whereas the cost of a non-linear solver iteration depends on various factors including the cost of the non-linear residual evaluation.[38] For second-order finite-volume RANS problems, linear solver cycles are computationally cheaper than non-linear cycles, enabling a greater number of mesh sweeps to be performed at fixed cost compared to non-linear solvers.[20]

On the other hand, the linear solver formulation described herein only solves an approximate Newton problem, due to the use of a first-order Jacobian and the inclusion of a diagonal pseudo-time step term. Thus, solving each linear problem to completion is wasteful in that it does not produce justifiable gains in overall non-linear convergence. This is particularly true for small values of the pseudo-time step, which require large numbers of outer non-linear updates to converge the non-linear problem. Additionally, in these cases, the linear problem is highly diagonally dominant or local in nature, and global solvers such as multigrid are not required and may even be more expensive than the use of simple local solvers such as the line-implicit and point-implicit solvers. However, even for large (near infinite) values of the pseudo-time step, the approximation incurred through the use of a first-order Jacobian places an upper limit on the overall speed of non-linear convergence, requiring several hundred non-linear cycles to reach machine zero in the best of circumstances.[20]

The principal issue with the linear solver approach consists of finding an optimal balance between the level of convergence of the linear system and a suitable choice of the pseudo-time step in order to achieve good overall non-linear convergence at optimal cost. This is complicated by the fact that small pseudo-time steps are required in the initial phases of convergence when non-linear effects dominate, while large values are preferred in the final phases in order to accelerate convergence. Although significant efforts have been expended in devising robust and optimal pseudo-time step scheduling strategies in the literature,[39–43] the current implementation relies on a simple exponential growth approach with prescribed minimum and maximum values.

E. Newton Krylov Solver

The most rapid non-linear convergence is generally achieved using an exact Newton method. This involves solving equation (6) where the left-hand side represents the exact linearization of the residual in equations

(5) or (6). Assembling and inverting the exact Jacobian matrix can be overly complex and costly due to its size and sparsity pattern, which stems from the extended stencil of the full second-order residual. However, Krylov methods such as GMRES may be used to solve large linear systems such as equation (6) without explicitly forming and inverting the Jacobian matrix. For example, the application of GMRES to equation (6) consists of first forming the residual of the linear problem as

$$\mathbf{r}_{linear} = \mathbf{v}_0 = \frac{\partial \mathbf{R}(\mathbf{w})}{\partial \mathbf{w}} \Delta w + \mathbf{R}(\mathbf{w}) \tag{19}$$

and then forming a set of Krylov vectors or search directions following

$$\mathbf{v}_{j+1} = \frac{\partial \mathbf{R}(\mathbf{w})}{\partial \mathbf{w}} \mathbf{v}_j \tag{20}$$

with the final solution given as some linear combination of the $\mathbf{v}_j$ vectors.[28] Thus, only Jacobian-vector products are required in the GMRES algorithm, obviating the need to assemble and store the exact second-order Jacobian matrix. A completely matrix-free implementation of a Newton-Krylov method can be achieved by approximating the Jacobian-vector products with finite differences of the residual vector.[29,44] Although this approach is simple and has been implemented in previous work,[20] convergence to tight tolerances is often hindered by the accuracy of the finite-difference approximation which is sensitive to machine roundoff error. Since the NSU3D code incorporates an exact discrete adjoint and forward linearization capability,[45] the forward linearization can be used to produce exact Jacobian vector products. The current implementation relies on a hand-coded linearization of each individual subroutine, which has been verified to machine precision using the complex step method.[46] This implementation is fully equivalent to the application of automatic differentiation to the code, although the current approach can be expected to be more computationally efficient.

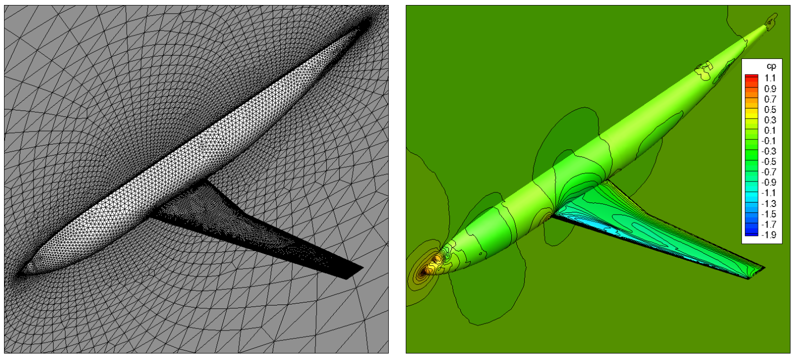
In order to be effective, Krylov methods require efficient preconditioning. Therefore, the line-implicit linear multigrid solver described in the previous section is used as a preconditioner for the Newton-Krylov scheme. In general, Newton-Krylov methods are most effective when used with good initial solutions. Therefore, convergence is usually initiated with the linear multigrid method used as a solver in order to drive the residual down by several orders of magnitude, after which the solution strategy is switched to the Newton-Krylov method using the linear multigrid method as a preconditioner.

IV. Example Test Cases

Three test cases have been chosen to examine the performance of the various solvers described in the previous section. These consist of two steady-state cases and one time-dependent case. The two steady-state cases are taken from previous AIAA sponsored drag prediction and high-lift prediction workshop cases. The grids used for these cases are available at the respective workshop web sites[47,48] and thus the cases are easily reproducible. Of these two cases, the first one is considered to be a standard well-behaved test case, while the second one is considered to be representative of a more difficult convergence problem. Mixed-element unstructured meshes are used for all cases, involving highly stretched prismatic elements in boundary-layer regions, with normal spacing of the order of $y^+ = 1$ at viscous walls, tetrahedral elements in inviscid flow regions, and with a small number of pyramidal elements in prismatic to tetrahedral transition regions.

A. Drag Prediction Workshop Case

This test case consists of the solution of transonic flow over a wing-body configuration at cruise condition. The geometry corresponds to the DLR-F6 configuration which was the subject of the second and third drag prediction workshops. The Mach number is 0.75, the incidence is 0 degrees, and the Reynolds number based on mean aerodynamic chord is 3 million. A mixed-element unstructured mesh is used for this case consisting of approximately 1.2 million points with 1,172,066 prismatic elements, 30,46,727 tetrahedral elements and 8,464 pyramidal elements as shown in Figure 4(a). This mesh constituted the coarse mesh for the second drag prediction workshop and is available as a fully tetrahedral mesh (which can be merged into a mixed-element mesh) labeled *f6wbnc* at the DPW2 web site.[47] At these flow conditions, the flow is transonic with a weak shock on the wing upper surface and the flow is mostly attached as illustrated in the sample NSU3D solution on this mesh in Figure 4(b). This case is considered representative of a standard on-design RANS simulation problem that should not exhibit significant convergence difficulties for most solvers.



(a) Mesh (b) Solution

Figure 4. Illustration of 1.2 million point mixed-element unstructured grid and computed surface pressure coefficient values at Mach=0.75, Incidence=0^o, and Reynolds number = 3 million for DLR-F6 wing body configuration

Figure 5 depicts the convergence in terms of the RMS average of the flowfield density residuals, turbulence residuals and lift coefficient, as a function of the number of non-linear iterations for the various solvers. These convergence plots are reproduced in terms of wall clock time for the solver running in parallel on 128 cpu cores in Figure 6. For all these cases, the multigrid solver uses 5 mesh levels.

The first observation is that the single grid solver, using the line-implicit solver without multigrid is the slowest of all, requiring well over 50,000 iterations to reach machine zero. The non-linear FAS multigrid solver using the line-implicit solver on each level converges considerably faster, reaching machine zero in approximately 5000 multigrid cycles. Even though a single FAS multigrid cycle using 5 mesh levels requires approximately twice as much cpu time as one cycle of the single grid solver, the multigrid solver is over an order of magnitude faster than the single grid solver in terms of cpu time to convergence, as shown in Figure 6. In order to illustrate the effect of the line solver, the FAS multigrid solver is also run using only the point solver on each level (i.e. without the line solver). Although the initial convergence is similar, significant degradation is seen in the asymptotic convergence rate of the point-implicit multigrid solver versus the line-implicit multigrid solver. In previous work it has been shown that the line solver is capable of maintaining convergence rates that are independent of the mesh aspect ratio in boundary-layer regions, and the degradation of the point-implicit multigrid solver with respect to the line-implicit multigrid solver can be expected to increase for higher Reynolds number cases.[15] Interestingly, the cpu time required for a single multigrid cycle of the line-implicit multigrid solver is not significantly different than that required for the point-implicit multigrid solver. This is due to the efficiency of the line solver combined with the fact that major portions of the cpu time are expended in other components of the code. Thus the line-implicit multigrid solver compares similarly versus the point-implicit multigrid solver both in terms of number of cycles as well as cpu time.

Two linear multigrid solver cases are shown in the figures. The first case employs three linear multigrid cycles on five mesh levels with four block line-Jacobi smoothing passes on each mesh level for each non-linear update. The second case employs a similar strategy but using ten linear multigrid cycles per non-linear update. In both cases, the pseudo-time step CFL number is initialized at $CFL = 2$, and grows exponentially by the factor 1.2 up to the maximum value of $CFL = 10^{10}$. The first case is designed to optimize cpu time to convergence, while the second case is used to determine the maximum non-linear convergence rate of the linear solver approach in terms of non-linear iterations, since in this case the linear system is converged to tight tolerances at each non-linear iteration.

As can be seen from Figure 5, both cases deliver roughly equivalent convergence rates in terms on non-linear cycles, indicating that three multigrid cycles per non-linear update are sufficient to obtain good non-linear convergence rates. Furthermore, the linear multigrid approach converges over 4 to 5 times faster than the non-linear FAS multigrid method in terms of non-linear cycles. In term of cpu time, the three-linear-multigrid cycle case retains a 4 to 5:1 advantage over the FAS multigrid scheme, since the cost of a single non-linear update is roughly equivalent for both solvers. The ten-linear-multigrid cycle case is not as competitive in terms of cpu costs, due to over-convergence of the linear system at each non-linear cycle. It is interesting to note that in terms of overall number of multigrid cycles (linear or non-linear) the FAS multigrid scheme is roughly equivalent to the three-cycle linear-multigrid scheme, with the overall advantage of the latter scheme being derived from the lower cost of linear versus non-linear multigrid cycles.[20] The most effective solver overall is the linear multigrid solver using three multigrid cycles per non-linear update. Convergence to machine zero is achieved in roughly 800 non-linear cycles using this approach, which requires approximately 300 seconds on 128 cores of the NCAR-Wyoming Supercomputer using 2.6 GHz Intel Sandy Bridge processors.[49] With regards to engineering accuracy, lift and drag coefficients are converged to engineering accuracy in approximately 30 seconds of wall clock time using 128 cores, as shown in the rightmost plot in Figure 6.

It is noteworthy that the asymptotic rate of non-linear convergence of the linear multigrid scheme represents a limitation due to the use of an inexact (first-order accurate) Jacobian, since the pseudo-time term is vanishingly small in this case, and since solving the linear system to higher tolerances has no effect on the non-linear convergence rate. The observed rate corresponds to a residual reduction of 0.96 per non-linear cycle (800 cycles to machine zero). This rate is remarkably similar to that observed over a decade ago for two-dimensional problems using the equivalent discretization[20] and appears to be a characteristic of the approximate Jacobian used for this discretization. Thus, achieving faster non-linear convergence requires the formulation of a more exact Newton method.

Figure 7 illustrates the convergence of the GMRES solver used to solve the exact Newton problem for this case, which is initiated after 200 cycles of the linear multigrid solver. In terms of number of non-linear iterations, the GMRES solver converges very quickly to machine zero, as expected from an exact Newton approach. However, each non-linear update consists of the computation of 50 preconditioned Krylov vectors, such that the comparison in terms of cpu time is less dramatic for this solver. Nevertheless, GMRES offers further acceleration for convergence to machine precision as well as significant advantages in terms of robustness.[28, 42]

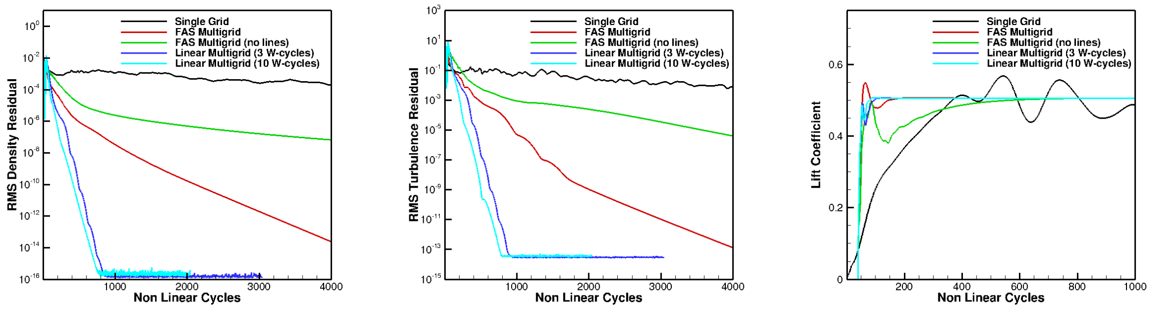


Figure 5. Convergence in terms of non-linear cycles for viscous turbulent transonic flow over wing-body configuration on mesh of 1.2 million points

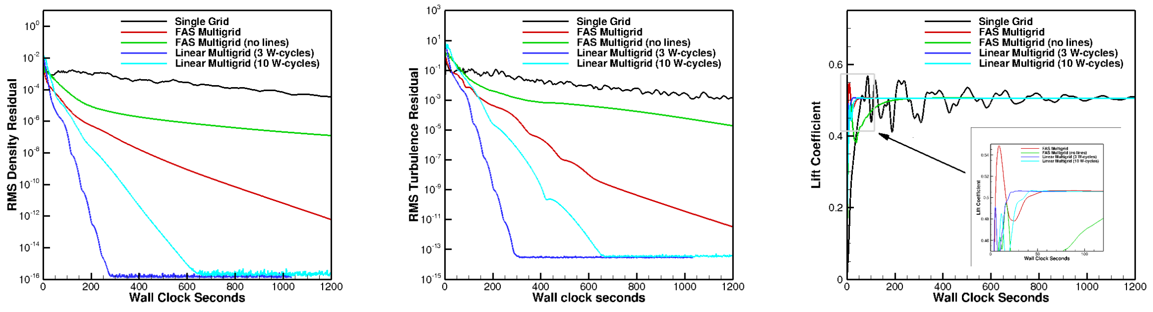


Figure 6. Convergence in terms of wall clock seconds for viscous turbulent transonic flow over wing-body configuration on mesh of 1.2 million points running on 128 cpu cores

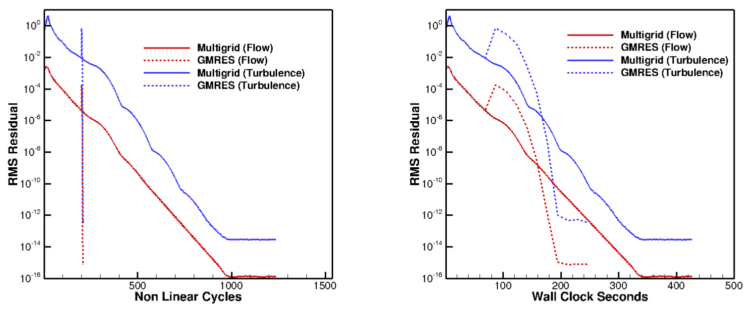


Figure 7. Comparison of linear multigrid algorithm used as a solver and a preconditioner for GMRES for viscous turbulent transonic flow over wing-body configuration

B. High-Lift Prediction Workshop Case

The next test case is taken from the second AIAA High Lift Prediction Workshop and consists of the subsonic compressible flow over a modern aircraft high-lift configuration. The subject geometry is the DLR-F11 wing body with high-lift system deployed in a landing configuration, and includes the slat brackets and flap-track fairings. The mesh for this case contains a total of 41.5 million points, with 58 million prismatic elements and 71 million tetrahedral elements, and is available as mesh *D_uns_mix_Case2Config4_v2* at the workshop web site.[48] The mesh is illustrated in Figure 8, along with a sample NSU3D flow solution in terms of computed surface pressure coefficient, and skin friction coefficient. For this case the Mach number is 0.175, the Reynolds number is 15.1 million based on mean aerodynamic chord, and the incidence is 16 degrees. This case lies within the linear range of the C_L - α curve, and is not close to the $C_{L_{MAX}}$ point. Therefore, it does not represent one of the most difficult cases of the workshop. However, significant regions of flow separation are evident behind the flap-track fairings and near the wing-fuselage junction for this case, as can be seen in Figure 8. These flow characteristics, combined with the increased geometric complexity and larger mesh size make this a significantly more difficult convergence test case.

Figure 9(a) illustrates the convergence obtained for this case using the line-implicit FAS non-linear multigrid scheme. The density residual converges approximately three orders of magnitude over 1000 cycles, but it is evident that the multigrid convergence has stalled by this point. However, the force coefficients are converged to within 1% of their final values, which may be considered sufficient for engineering accuracy depending on the application. This calculation required a total of 28 minutes running on 1024 cores of the NCAR-Wyoming Supercomputer.

In order to converge this case to machine zero, significant additional effort is required. The simplest approach consists of restarting this solution using the single grid solver, and running for a large number of cycles, as shown in Figure 9(b). One of the problems with the non-linear multigrid scheme is that when convergence stalls, there is little recourse for restoring convergence short of reducing the number of multigrid levels. However, in the linear solver approach, the pseudo-time step can be scaled back to trade robustness for increased computational cost. Figure 9(c) depicts such a strategy where the stalled FAS multigrid solution was restarted using the linear multigrid scheme with a pseudo-time-step CFL number of 200. The residuals initially spike up and then resume convergence, although residual stagnation is again observed. However, even as the residuals stall, the lift coefficient convergence continues as can be seen in Figure 10(a). This solver was run for 2000 cycles, after which the CFL number was reduced to 20. At this low CFL value, multigrid is ineffective, and thus a single level line-implicit linear solver with 30 subiterations per non-linear update was used in the linear solver. As can be seen, the residuals resume convergence and are reduced by another three orders of magnitude over 2000 cycles.

For comparison, the same single level line-implicit solver using 30 subiterations with a CFL of 20 was used to converge this test case starting from initial freestream conditions. As seen in Figure 9(c), this approach requires a greater number of non-linear iterations and cpu time than the hybrid convergence case, and also does not achieve equivalent asymptotic convergence rates as the hybrid approach, suggesting that some error components have been effectively removed by the earlier application of the multigrid algorithm.

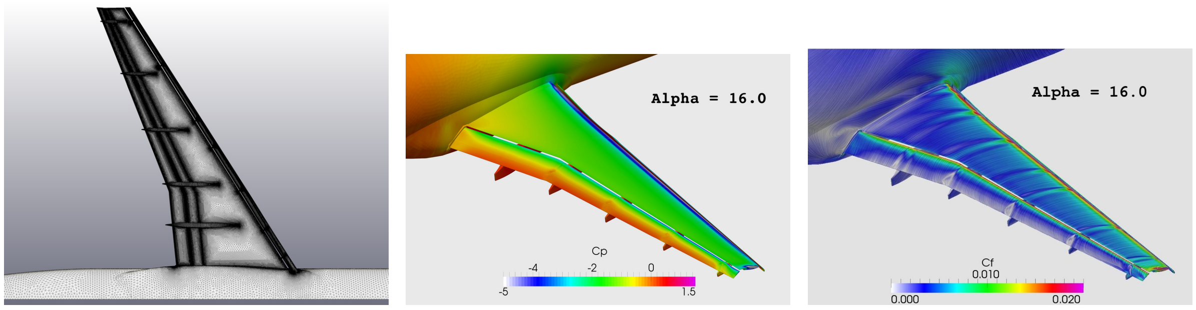
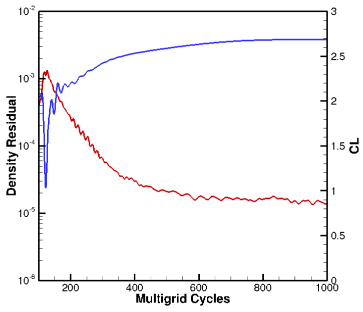
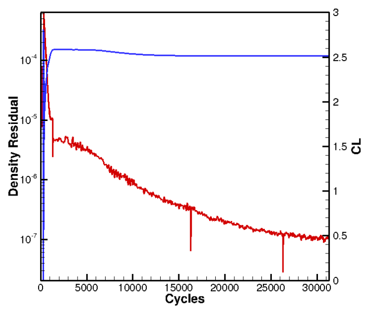


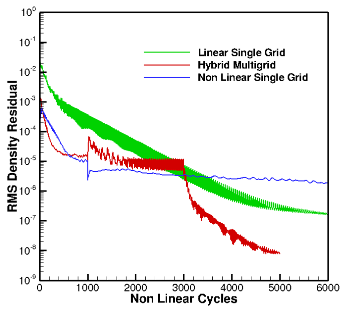
Figure 8. Illustration of mesh and solution for high-lift prediction workshop test case on mesh of 41.5 million points for flow conditions Mach = 0.175, Incidence = 16^o, Reynolds number = 15.1 million



(a) Non-linear multigrid

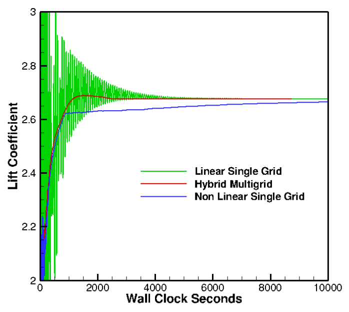


(b) MG followed with single grid

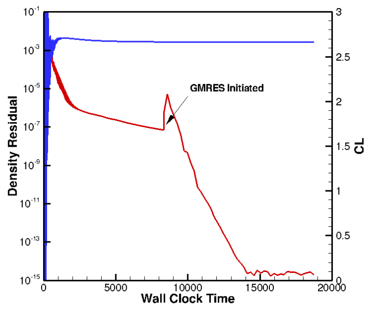


(c) Hybrid convergence histories

Figure 9. Convergence of high-lift test case using non-linear multigrid, single grid, and hybrid restarted methods



(a) Hybrid convergence histories



(b) GMRES solver

Figure 10. Hybrid convergence histories for high-lift test case using linear multigrid and single grid solvers and GMRES solver

Comparisons of convergence efficiency in terms of wall-clock time are similar to those in terms of non-linear cycles since the time per non-linear cycle of all schemes is approximately the same (except for the single level non-linear line solver which is approximately half the cost of the other schemes). Figure 10(a) provides a comparison in terms of wall-clock time for the lift coefficient convergence, illustrating the superior efficiency of the hybrid multigrid strategy, where it can be seen that convergence for this case requires several hours of wall clock time. Furthermore, these comparisons are mostly qualitative, since no attempt has been made to optimize the number of subiterations and number of non-linear cycles for each phase of the hybrid case. However, this test case points out the relative strengths and weaknesses of the various solver approaches.

The robustness advantages of the GMRES solver are illustrated in Figure 10(b), where it is shown that the residuals for this case can be driven to machine zero using the linear multigrid scheme as a preconditioner for GMRES. In this case, the GMRES algorithm with 50 Krylov vectors and 10 restarts per non-linear update was used after several orders reduction in the residuals were obtained through application of the linear multigrid method without GMRES. While this convergence history is far from optimal, it demonstrates the possibility of driving the residuals to zero for difficult problems where the standard multigrid algorithm stalls.

C. Time-Dependent Rotor Test Case

The final test case consists of a time-dependent problem where we examine the efficiency of the various solvers at converging the non-linear problem arising at each implicit time step using a BDF2 time discretization. The geometry consists of a four bladed HART2 rotor, which is a benchmark configuration for the rotorcraft CFD community.[50] The blades are given a constant collective pitch angle of 5 degrees and the rotor is impulsively started in quiescent flow (Mach = 0). The rotation rate is such that the blade tip Mach number is 0.637, and the Reynolds number based on the blade chord is 2.15 million. A mixed-element unstructured mesh of 2.3 million points (3.3 million prisms, 3.9 million tetrahedra and 35,684 pyramids) is used to model the entire rotor and the mesh-rotor system is rotated by 2 degrees at each physical time step, as illustrated in Figure 11. This case has been computed previously for rigid and flexible blade analysis and optimization in references.[51, 52] The convergence study is performed on a restarted solution after 45 time steps (after 90 degrees azimuthal rotation), in a region of the simulation where the thrust value of the rotor is changing significantly, as shown in Figure 12(a). Figure 12(b) depicts the convergence rate using the single grid and FAS multigrid solvers at the 46th, 47th, and 48th time steps in terms of non-linear iterations for the density residuals and thrust coefficient using a 2 degree time step. As shown in the figure, the single grid line-implicit solver converges more slowly than the non-linear multigrid solver. However, in this case, the difference between the multigrid solver and the single grid solver is not as dramatic as in the previous steady-state cases. This is partly due to the characteristic of the time-implicit system, which is more local in nature than a steady-state problem. In fact, it is well known that for small enough physical time steps, global solvers such as multigrid are not required to obtain rapid convergence of the resulting time-implicit problem. This is demonstrated in Figure 12(c) where the convergence for a 0.5 degree time step is examined using the non-linear single and multigrid solvers, showing even less difference between the two approaches. In Figure 13 the convergence of both schemes is compared in terms of wall-clock time, which favors the single grid solver, since this solver requires approximately half the cpu time per cycle compared to the multigrid method. For the two degree time-step case, the residual convergence plots are similar for both solvers, while in the case of the 0.5 degree case, the single grid solver appears to be more efficient than the multigrid solver for residual convergence. However, close examination of the initial convergence of the thrust coefficient plots shows that this value approaches its final converged value substantially faster in the 2 degree case using the multigrid solver. For the 0.5 degree case, the multigrid solver still holds a slight advantage over the single grid solver in terms of the initial thrust coefficient convergence, illustrating the importance of examining more than one measure of convergence.

Figure 14 compares the convergence obtained for the 2 degree time step using the previous solvers with the addition of the linear multigrid solver and the GMRES solver. As expected, both these solvers converge much more quickly to machine precision both in terms of non-linear iterations and wall-clock time. However, a close examination of the thrust coefficient convergence history shows that these methods lag the non-linear FAS multigrid approach in the initial phases of convergence. This is due to the CFL ramping used in these methods. The standard CFL ramping used in the linear multigrid method is seen to produce considerably slower initial convergence, even though the final asymptotic rate of this method is very fast. For the GMRES solver, which begins with the same linear multigrid method prior to the initiation of GMRES, a more aggressive CFL strategy was used, producing faster initial convergence, although the non-linear FAS multigrid method still produces the fastest initial convergence of the thrust coefficient value as can be seen in the right-most plot of Figure 14. The application of GMRES is effective in rapidly reducing the residuals to machine zero, although by the time this technique is initiated, sufficient convergence for most engineering simulations is already achieved.

Finally, although 2 degrees represents a relatively large time step, the coarse nature of the grid used in this test case indicates that the larger time step example may be more representative of production runs that use a smaller time step but in conjunction with a finer grid.

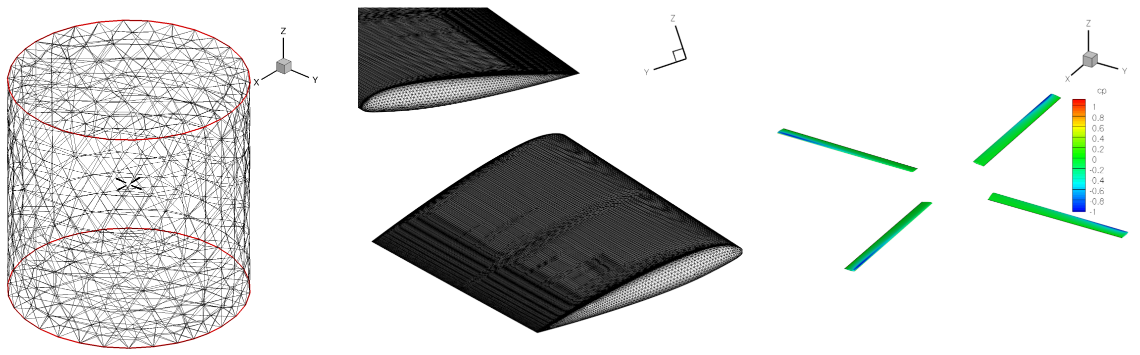
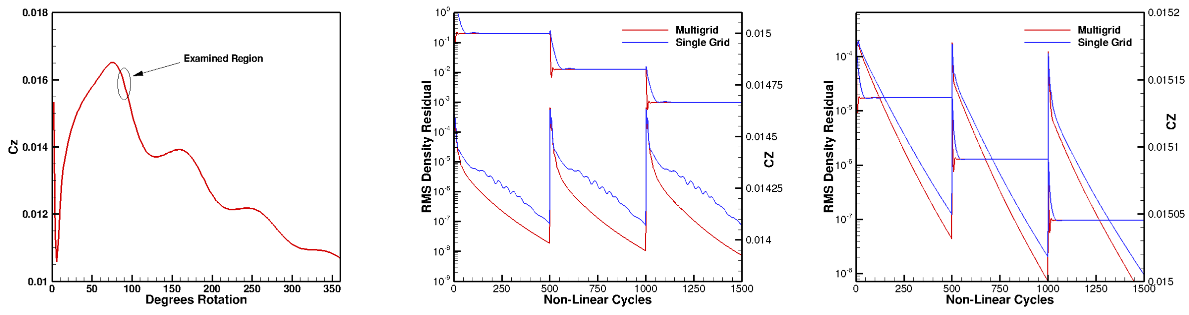
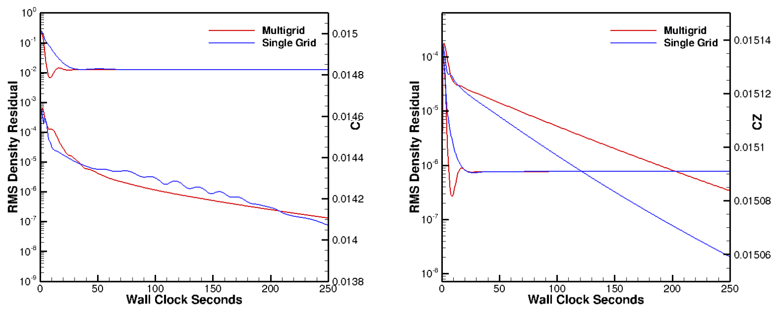


Figure 11. Illustration of implicit time-dependent test case based on HART2 four-bladed rotor configuration on mesh of 2.3 million points



(a) Global time history of thrust coefficient (b) Convergence using 2^o time steps (c) Convergence using 0.5^o time steps

Figure 12. Time history and convergence of implicit time-dependent HART2 four-bladed rotor test case



(a) Convergence using 2^o time steps (b) Convergence using 0.5^o time steps

Figure 13. Residual and thrust coefficient convergence at 47th time step in terms of wall-clock time

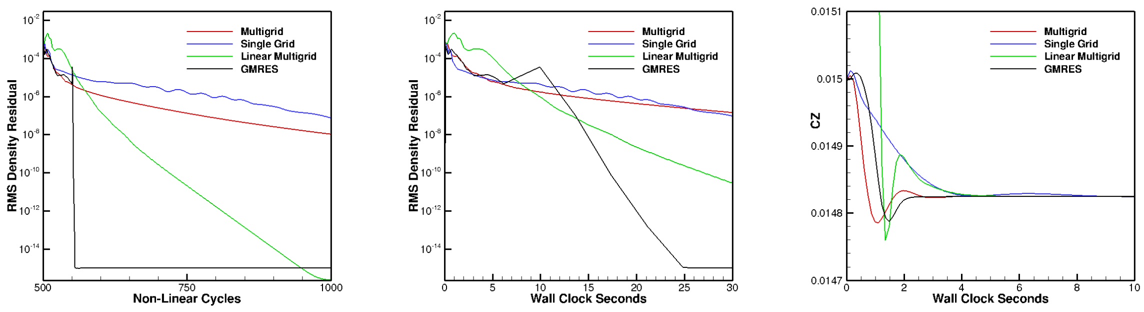


Figure 14. Comparison of convergence at 47th time step using 2 degree time step for various solvers in terms of residual and thrust coefficient

V. Scalability

All the solvers described in this work have been designed to deliver identical convergence rates that are independent of the number of mesh partitions or parallel computing cores used in the solution process. This is unlike many other popular approaches, such as for example ADI schemes or ILU preconditioned GMRES solvers, that do not parallelize entirely and are thus applied in a partition local sense, leading to degrading numerical convergence with increasing hardware parallelism. In the current implementation, complete solver parallelism is achieved firstly by ensuring that line structures are never broken during the partitioning process, and secondly by constructing coarse multigrid levels that are independent of the mesh partitioning. For the multigrid solver, this leads to mismatch between coarse and fine multigrid parent-to-child control-volume partitions, although this is easily handled through the use of an additional MPI communication routine. In practice approximate matching of fine-to-coarse partitions is performed through a greedy algorithm. Furthermore, it has been found that, for best overall scalability, it is more important to minimize intra-grid communication on each level rather than inter-grid communication between levels. The scalability of the multigrid algorithm has held up remarkably well in going to very large numbers of processors. Figure 15 illustrates the scalability achieved on up to 32,768 cores of the NCAR-Wyoming Supercomputer of the FAS multigrid solver on a 75 million point mesh used for the second high-lift prediction workshop.[3, 10, 48] The scalability is seen to decrease somewhat for the multigrid algorithm using four mesh levels compared to the single solver. However, overall scalability remains good especially considering the size of this mesh, which when spread across 32,768 cores contains less than 2300 points per core for the fine level, and of the order of 10 control volumes per core for the fourth multigrid level. Furthermore, these results are obtained using the multigrid W-cycle, which spends more time on the coarse levels than the simpler V-cycle, but which delivers faster convergence rates.

Because the coarse multigrid levels contain fewer control volumes but are spread over the same number of cores, the ratio of communication to computation increases for the coarser levels. However, a simple argument[53] can be used to show that the ratio of communication to computation for the entire multigrid algorithm is asymptotically constant for weak scaling problems (where the problem size grows with the number of processors or cores). Given a mesh of N points or control volumes distributed over P processors, an estimate of the work per partition can be taken as:

$$Work\,per\,partition = \frac{N}{P}. \tag{21}$$

Using the volume to surface argument, which assumes that the communication cost is governed by the number of cut edges at partition boundaries, an estimate of the communication costs is given as:

$$Communication\,per\,partition = (\frac{N}{P})^{\frac{2}{3}} \tag{22}$$

for three-dimensional problems. Thus, the ratio of communication to computation for the entire mesh is given as:

$$Ratio = (\frac{P}{N})^{\frac{1}{3}} \tag{23}$$

illustrating how the ratio increases for strong scaling problems (increasing P, constant N), but remains constant for weak scaling problems (increasing P and N at same rate). Although this ratio is much worse for each individual coarse mesh of a multigrid sequence (constant P, decreasing N), the ratio for a complete multigrid cycle can be shown to be bounded. For example, considering a multigrid V-cycle, the total work per partition becomes:

$$V\,cycle\,Work = \frac{N}{P} + \frac{N}{8P} + \frac{N}{64P} + ... = (1 + \frac{1}{8} + \frac{1}{64} + ...)(\frac{N}{P}) = \frac{8}{7}(\frac{N}{P}) \tag{24}$$

while the total communication cost becomes:

$$V\,cycle\,Comm = (\frac{N}{P})^{\frac{2}{3}} + (\frac{N}{8P})^{\frac{2}{3}} + (\frac{N}{64P})^{\frac{2}{3}} + ... = (1 + \frac{1}{4} + \frac{1}{16} + ...)(\frac{N}{P})^{\frac{2}{3}} = \frac{4}{3}(\frac{N}{P})^{\frac{2}{3}} \tag{25}$$

leading to an estimate of the ratio of communication to computation of

$$Ratio = \frac{7}{6}(\frac{P}{N})^{\frac{1}{3}} \tag{26}$$

Similarly for the multigrid W-Cycle, the total work per cycle becomes:

$$W\,cycle\,Work = \frac{N}{P} + \frac{2N}{8P} + \frac{4N}{64P} + ... = (1 + \frac{1}{4} + \frac{1}{16} + ...)(\frac{N}{P}) = \frac{4}{3}(\frac{N}{P}) \tag{27}$$

while the total communication cost becomes:

$$W\,cycle\,Comm = (\frac{N}{P})^{\frac{2}{3}} + (\frac{2N}{8P})^{\frac{2}{3}} + (\frac{4N}{64P})^{\frac{2}{3}} + ... < (1 + \frac{1}{2} + \frac{1}{4} + ...)(\frac{N}{P})^{\frac{2}{3}} = 2(\frac{N}{P})^{\frac{2}{3}} \tag{28}$$

leading to an estimate of the ratio of communication to computation of

$$Ratio < \frac{3}{2}(\frac{P}{N})^{\frac{1}{3}} \tag{29}$$

demonstrating, at least theoretically, that weak scaling of multigrid algorithms should be optimal given sufficiently aggressive coarsening strategies. This argument is supported by our current and previous experience over the years in benchmarking the FAS non-linear multigrid algorithm for increasingly large problems on leading edge hardware.[10, 34, 54, 55]

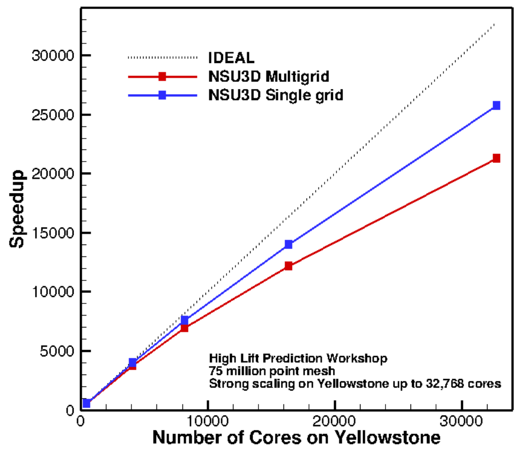


Figure 15. Strong scaling of multigrid solver in NSU3D using up to 32,786 cores for high-lift test case on 75 million point mesh

VI. Conclusions and Future Work

The various solvers in the NSU3D unstructured mesh RANS code have been described in this paper and demonstrated on three representative test cases. All solvers have been designed to produce identical convergence histories running on different numbers of processors. The benefits of the line solver over the point solver for highly anisotropic meshes have been demonstrated, along with the convergence acceleration achieved by the non-linear and linear multigrid solvers, while the relative benefits of non-linear versus linear solution techniques have been discussed. In general, the linear multigrid solver is found to be the most efficient approach, delivering over an order of magnitude increase in solution efficiency compared to single level solvers. However the linear multigrid solver requires a suitable CFL scheduling strategy while the non-linear FAS multigrid approach can be applied directly to non-linear problems with poor initial guesses. On the other hand, the flexibility provided by the CFL scheduling can be used to increase the robustness of the linear multigrid approach at the expense of computational efficiency. Although faster rates may be achievable for simple problems, empirical evidence shows that approximate Newton methods that make use of a first-order Jacobian are limited to asymptotic convergence rates of 0.96 residual reduction per non-linear cycle, at least for the current discretization. Thus, enabling faster non-linear convergence requires the formulation of a more exact Newton method. Using the linear multigrid solver as a preconditioner for GMRES to solve the exact Newton method introduces additional tunable solver parameters, but can provide an effective mechanism for improving solution robustness, particularly for tight convergence tolerances.

The benefits of multigrid decrease as the problem character becomes more local, for example for implicit-time problems using relatively small physical time steps. However, multigrid methods that are implemented in a parallel partitioning agnostic manner are asymptotically optimal solvers, meaning that as the problem size increases, multigrid methods should maintain more favorable convergence rates compared to other solvers. Future planned improvements to the current solvers in the NSU3D code include the extension of the block Jacobi line and point solvers to a colored Gauss-Seidel strategy, and the implementation of a more robust CFL scheduling strategy for the linear solvers.[39–43] For time-dependent problems, since convergence to low tolerance levels is rarely affordable, solvers that provide rapid initial convergence (perhaps at the expense of asymptotic convergence rates) should be favored.

Although multigrid methods have enjoyed early success in many CFD codes, more recently there has been a trend away from multigrid methods in favor of matrix-based solvers such as different ILU variants, in the interest of creating more robust solvers for difficult cases.[42] Because multigrid has better numerical scaling properties and can be made to deliver convergence rates that are independent of the level of parallelism, advances in multigrid methods still hold great potential for accelerating the solution of very large problems on massively parallel computer architectures in the future. The key will be to develop multigrid algorithms than maintain these favorable properties while retaining the robustness of other matrix-based schemes.

VII. Acknowledgments

We are grateful for computer time provided by the NCAR-Wyoming Supercomputer Center (NWSC) and by the University of Wyoming Advanced Research Coomputing Center (ARCC).

References

[1]Vassberg, J. C., Tinoco, E. N., Mani, M., Zickuhr, T., Levy, D., Brodersen, O. P., Eisfeld, B., Wahls, R. A., Morrison, J. H., Mavriplis, D. J., and Murayama, M., "Summary of the Fourth AIAA CFD Drag Prediction Workshop," AIAA Paper 2010-4547.

[2]Levy, D. W., Laflin, K. R., Vassberg, J. C., Tinoco, E., Mani, M., Rider, B., Brodersen, O., Crippa, S., Rumsey, C. L., Wahls, R. A., Morrison, J., Mavriplis, D. J., and Murayama, M., "Summary of Data from the Fifth AIAA CFD Drag Prediction Workshop," AIAA Paper 2013-046, presented at the 51st Aerospace Sciences Meeting, Grapevine TX.

[3]Rumsey, C. R. and Slotnick, J., "Overview and Summary of HiLiftPW-2," AIAA Paper 2014-xxx presented at the AIAA SciTech 2014 Conference, National Harbor, MD.

[4]Mavriplis, D. J. and Levy, D. W., "Transonic Drag Prediction using an Unstructured Multigrid Solver," *AIAA Journal of Aircraft*, Vol. 42, No. 4, 2003, pp. 887–893.

[5]Mavriplis, D. J., "Grid Resolution Study of a Drag Prediction Workshop Configuration Using the NSU3D Unstructured Mesh Solver," AIAA-Paper 2005-4729.

[6]Mavriplis, D. J., "Third Drag Prediction Workshop Results using the NSU3D Unstructured Mesh Solver," *AIAA Journal of Aircraft*, Vol. 45, No. 3, May 2008, pp. 750–761.

[7]Mavriplis, D. J. and Long, M., "NSU3D Results from the Fourth AIAA CFD Drag Prediction Workshop," AIAA Paper 2010-4364.

[8]Park, M., Laflin, K. R., Chaffin, M., Powell, N., and Levy, D. W., "CFL3D, FUN3D and NSU3D contributions to the fifth drag prediction workshop," AIAA Paper 2013-0050, 51st AIAA Aerospace Sciences Meeting, Grapevine TX.

[9]Long, M. and Mavriplis, D. J., "NSU3D Results from the First AIAA CFD High-Lift Prediction Workshop," AIAA Paper 2011-863 presented at the 49th AIAA Aerospace Sciences Meeting.

[10]Mavrplis, D. J., Long, M., Lake, T., and Langlois, M., "NSU3D Results for the second AIAA High-Lift Prediction Workshop," AIAA Paper 2014-xxx presented at the AIAA SciTech 2014 Conference, National Harbor, MD.

[11]Mavriplis, D. J., Z.Yang, and Long, M., "Results using NSU3D for the First AIAA Aeroelastic Prediction Workshop," AIAA Paper 2013-0786, 51st AIAA Aerospace Sciences Meeting, Grapevine TX.

[12]Mavriplis, D. J. and Venkatakrishnan, V., "A Unified Multigrid Solver for the Navier-Stokes Equations on Mixed Element Meshes," *International Journal for Computational Fluid Dynamics*, Vol. 8, 1997, pp. 247–263.

[13]Yang, Z. and Mavriplis, D. J., "High-Order Time Integration Schemes for Aeroelastic Applications on Unstructured Meshes," *AIAA Journal*, Vol. 45, No. 1, 2007, pp. 138–150.

[14]Sitaraman, J., Wisssink, A., Sankaran, V., Jayaraman, B., Datta, A., Yang, Z., Mavriplis, D., Saberi, H., Potsdam, M., O'Brien, D., Cheng, R., Hariharan, N., and Strawn, R., "Application of the HELIOS Computational Platform to Rotorcraft Flowfields," AIAA Paper 2010-1230, Presented at the 48th AIAA Aerospace Sciences Meeting, Orlando FL.

[15]Mavriplis, D. J., "Multigrid Strategies for Viscous Flow Solvers on Anisotropic Unstructured Meshes," *Journal of Computational Physics*, Vol. 145, No. 1, Sept. 1998, pp. 141–165.

[16]Mavriplis, D. J., "Directional Agglomeration Multigrid Techniques for High-Reynolds Number Viscous Flow Solvers," *AIAA Journal*, Vol. 37, No. 10, Oct. 1999, pp. 1222–1230.

[17]Mavriplis, D. J., "A Three-dimensional Multigrid Reynolds-averaged Navier-Stokes Solver for Unstructured Meshes," *AIAA Journal*, Vol. 33, No. 3, March 1995, pp. 445–4531.

[18]Venkatakrishnan, V. and Mavriplis, D. J., "Agglomeration Multigrid for the Three-dimensional Euler Equations," *AIAA Journal*, Vol. 33, No. 4, April 1995, pp. 633–640.

[19]Mavriplis, D. J., "Multigrid Techniques for Unstructured Meshes," *VKI Lecture Series VKI-LS 1995-02*, March 1995.

[20]Mavriplis, D. J., "An Assessment of Linear versus Non-Linear Multigrid Methods for Unstructured Mesh Solvers," *Journal of Computational Physics*, Vol. 175, Jan. 2002, pp. 302–325.

[21]Mavriplis, D. J., "Unstructured Mesh Discretizations and Solvers for Computational Aerodynamics," *AIAA Journal*, Vol. 46, No. 6, 2008, pp. 1281–1298.

[22]Spalart, P. R. and Allmaras, S. R., "A One-equation Turbulence Model for Aerodynamic Flows," *La Recherche Aérospatiale*, Vol. 1, 1994, pp. 5–21.

[23]Wilcox, D. C., *Turbulence Modeling for CFD*, DCW Industries Inc., La Canada, CA, 1998, Second edition.

[24]Menter, F. R., "Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications," *AIAA Journal*, Vol. 32, No. 8, 1994, pp. 1598–1605.

[25]Lambert, J. D., *Numerical methods for ordinary differential systems*, Wiley, Chicester, UK, 1991.

[26]Butcher, J. C., *Numerical methods for ordinary differential equations*, Wiley, Chicester, UK, 2003.

[27]Mavriplis, D. J. and Yang, Z., "Time Spectral Method for Periodic and Quasi-Periodic Unsteady Computations on Unstructured Meshes," *Mathematical Modeling of Natural Phenomena*, Vol. 6, No. 3, May 2011, pp. 213–236, DOI: http://dx.doi.org/10.1051/mmnp/20116309.

[28]Saad, Y., *Iterative Methods for Sparse Linear Systems*, PWS Series in Computer Science, PWS Publishing Company, Boston, MA, 1996.

[29]Mousseau, V. A., Knoll, D. A., and Rider, W. J., "Physics-based preconditioning and the Newton-Krylov method for non-equilibrium radiation diffusion," *Journal of Computational Physics*, Vol. 160, No. 2, 2000, pp. 743–765.

[30]Jameson, A., Schmidt, W., and Turkel, E., "Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes," AIAA Paper 81-1259.

[31]Jameson, A., "Solution of the Euler Equations by a Multigrid Method," *Applied Mathematics and Computation*, Vol. 13, 1983, pp. 327–356.

[32]Tai, C. H., Sheu, J. H., and van Leer, B., "Optimal multistage schemes for Euler equations with residual smoothing," *AIAA Journal*, Vol. 33, 1995, pp. 1008–1016.

[33]Isaacson, E. and Keller, H. B., *Analysis of Numerical Methods*, John Wiley and Sons, New York, NY, 1966.

[34]Mavriplis, D. J. and Pirzadeh, S., "Large-Scale Parallel Unstructured Mesh Computations for 3D High-Lift Analysis," *AIAA Journal of Aircraft*, Vol. 36, No. 6, Dec. 1999, pp. 987–998.

[35]Hackbush, W., *Multigrid Methods and Applications*, Springer-Verlag, Berlin, Germany, 1985.

[36]Trottenberg, U., Schuller, A., and Oosterlee, C., *Multigrid*, Academic Press, London, UK, 2000.

[37]Briggs, W. L., Henson, V. E., and McCormick, S. F., *A Multigrid Tutorial*, SIAM, Philadelphia, PA, 2000.

[38]Mavriplis, D. J., "Multigrid Approaches to non-linear diffusion problems on unstructured meshes," *Journal of Numerical Linear Algebra with Applications*, Vol. 8, No. 8, 2001, pp. 499–512.

[39]Kelley, C. T. and Keyes, D. E., "Convergence analysis of pseudo-transient continuation," *SIAM Journal on Numerical Analysis*, Vol. 35, No. 2, April 1998, pp. 508–523.

[40]Kelley, C. T., Liao, L. Z., Qi, L., Chu, M. T., Reese, J., and Winton, C., "Projected Pseudo-Transient Continuation," *SIAM Journal on Numerical Analysis*, Vol. 46, No. 6, 2008, pp. 3071–3083.

[41]Bucker, H. M., Pollul, B., and Rasch, A., "On CFL evolution strategies for implicit upwind methods in linearized Euler equations," *International Journal for Numerical Methods in Fluids*, Vol. 59, 2009, pp. 1–18.

[42]Kamenetskiy, D., Bussoletti, J., Hilmes, C., Johnson, F., Venkatakrishnan, V., and Wigton, L., "Numerical Evidence of Multiple Solutions for the Reynolds-Averaged Navier-Stokes Equations for High-Lift Configurations," AIAA Paper 2013-663 presented at the 51st AIAA Aerospace Sciences Meeting, Grapevine, TX.

[43]Ceze, M. and Fidkowski, K. J., "Pseudo-transient Continuation, Solution Update Methods, and CFL Strategies for DG Discretizations of the RANS-SA Equations," AIAA Paper 2013-2686 presented at the 21st AIAA Computational Fluid Dynamics Conference, San Diego CA.

[44]Yu, L. B. W. N. J. and Young, D. P., "GMRES Acceleration of Computational Fluid Dynamic Codes," *Proceedings of the 7th AIAA CFD Conference*, July 1985, pp. 67–74, AIAA Paper 85-1494-CP.

[45]Mavriplis, D. J., "A Discrete Adjoint-Based Approach for Optimization Problems on Three Dimensional Unstructured Meshes," *AIAA Journal*, Vol. 45, No. 4, April 2007, pp. 741–750.

[46]Lyness, J. N., "Numerical Algorithms based on the Theory of Complex Variables," Proceedings ACM 22nd National Conference, Thomas Book Company, Washington DC.

[47]"Second AIAA Drag Prediction Workshop." Orlando, FL. http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw.

[48]"Second AIAA High Lift Prediction Workshop." San Diego, CA. http://highliftpw.larc.nasa.gov.

[49]"NCAR-Wyoming Supercomputer Center," http://nwsc.ucar.edu/.

[50]Yu, Y. H., Tung, C., van der Wall, B., Pausder, H.-J., Burley, C., Brooks, T., Beaumier, P., Delrieux, Y., Mercker, E., and Pengel, K., "The HART-II Test: Rotor Wakes and Aeroacoustics with Higher-Harmonic Pitch Control (HHC) Inputs -The Joint German/French/Dutch/US Project-," *58th American Helicopter Society Annual Forum, Montreal, Canada*, June 11–13 2002.

[51]Mani, K. and Mavriplis, D. J., "Geometry Optimization in Three-Dimensional Unsteady Flow Problems using the Discrete Adjoint," AIAA Paper 2013-0662, 51st Aerospace Sciences Meeting, Grapevine TX.

[52]Mishra, A., Mani, K., Mavriplis, D. J., and Sitaraman, J., "Time Dependent Adjoint-based Optimization for Coupled Aeroelastic Problems," AIAA Paper 2013-2906, Presented at the 21st AIAA CFD Conference, San Diego, CA.

[53]Mavriplis, D. J., "A Highly Scalable Unstructured Aggolomeration Multigrid Algorithm for Viscous Turbulent Flows," Proceedings of the 1999 Copper Mountain Multigrid Conference, Copper Mountain, CO. Available at http://ww.mgnet.org.

[54]Mavriplis, D. J., Das, R., Saltz, J., and Vermeland, R. E., "Implementation of a parallel unstructured Euler solver on shared and distributed memory machines," *The J. of Supercomputing*, Vol. 8, No. 4, 1995, pp. 329–344.

[55]Mavriplis, D. J., Aftosmis, M., and Berger, M., "High-Resolution Aerospace Applications using the NASA Columbia Supercomputer," *International Journal of High Performance Computing Applications*, Vol. 21, No. 1, 2007, pp. 106–126.