AIAA 2010-4731

40th Fluid Dynamics Conference and Exhibit, 28 Jun - 1 Jul, 2010, Chicago, IL

Development and Application of Agglomerated Multigrid Methods for Complex Geometries

Hiroaki Nishikawa*, Boris Diskin†
National Institute of Aerospace, Hampton, VA 23666
and
James L. Thomas‡
NASA Langley Research Center, Hampton VA 23681

We report progress in the development of agglomerated multigrid techniques for fully unstructured grids in three dimensions, building upon two previous studies focused on efficiently solving a model diffusion equation. We demonstrate a robust fully-coarsened agglomerated multigrid technique for 3D complex geometries, incorporating the following key developments: consistent and stable coarse-grid discretizations, a hierarchical agglomeration scheme, and line-agglomeration/relaxation using prismatic-cell discretizations in the highly-stretched grid regions. A significant speed-up in computer time over state-of-art single-grid computations is demonstrated for a model diffusion problem, the Euler equations, and the Reynolds-averaged Navier-Stokes equations for 3D realistic complex geometries.

I. Introduction

Multigrid techniques [1] are used to accelerate convergence of current Reynolds-Averaged Navier-Stokes (RANS) solvers for both steady and unsteady flow solutions, particularly for structured-grid applications. Mavriplis et al. [2, 3, 4, 5] pioneered agglomerated multigrid methods for large-scale unstructured-grid applications. During the present development, a serious convergence degradation in some of the state-of-the-art multigrid algorithms was observed on highly-refined grids. To investigate and overcome the difficulty, we critically studied agglomerated multigrid techniques [6, 7] for two- and three-dimensional isotropic and highly-stretched grids and developed quantitative analysis methods and computational techniques to achieve grid-independent convergence for a model equation representing laminar diffusion in the incompressible limit. It was found in Ref. [6] that it is essential for grid-independent convergence to use consistent coarse-grid discretizations. In the later Ref. [7], it was found that the use of prismatic cells and line-agglomeration/relaxation is essential for grid-independent convergence on fully-coarsened highly-stretched grids. In this paper, we extend and demonstrate these techniques for inviscid and viscous flows over complex geometries.

The paper is organized as follows. Finite-volume discretizations employed for target grids are described. Details of the hierarchical agglomeration scheme are described. Elements of the multigrid algorithm are then described, including discretizations on coarse grids. Multigrid results for complex geometries are shown for a model diffusion equation, the Euler equations, and the RANS equations. The final section contains conclusions and recommendations for future work.

*Senior Research Scientist (hiro@nianet.org), National Institute of Aerospace, 100 Exploration Way, Hampton, VA 23666 USA.
†Associate Fellow (bdiskin@nianet.org), National Institute of Aerospace, 100 Exploration Way, Hampton, VA 23666 USA.; Visiting Associate Professor, Mechanical and Aerospace Engineering Department, University of Virginia, Charlottesville, Senior Member AIAA
‡Senior Research Scientist (James.L.Thomas@nasa.gov), Computational AeroSciences Branch, Mail Stop 128, Fellow AIAA.

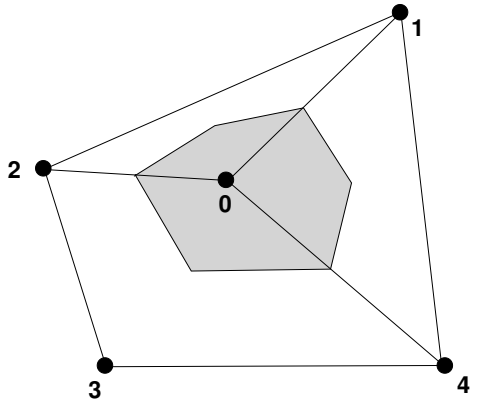


Figure 1. Illustration of a node-centered median-dual control volume (shaded). Dual faces connect edge midpoints with primal cell centroids. Numbers 0-4 denote grid nodes.

II. Discretization

The discretization method is a finite-volume discretization (FVD) centered at nodes. It is based on the integral form of governing equations of interest:

$$\oint_{\Gamma} (\mathcal{F} \cdot \hat{\mathbf{n}}) \, d\Gamma = \iint_{\Omega} s \, d\Omega, \tag{1}$$

where $\mathcal{F}$ is a flux tensor, s is a source term, Ω is a control volume with boundary Γ, and $\hat{\mathbf{n}}$ is the outward unit normal vector. For the model diffusion (Laplace) equation, the boundary conditions are taken as Dirichlet, i.e., specified from a known exact solution over the computational boundary. Tests are performed for a constant manufactured solution, $U(x, y, z) = 10.0$, with a randomly perturbed initial solution. For inviscid flow problems, the governing equations are the Euler equations. Boundary conditions are a slip-wall condition and inflow/outflow conditions on open boundaries. For viscous flow problems, the governing equations are the RANS equations with the Spalart-Allmaras one-equation model [8]. Boundary conditions are non-slip condition on walls and inflow/outflow conditions on open boundaries. The source term, s, is zero except for the turbulence-model equation (see Ref. [8]).

The general FVD approach requires partitioning the domain into a set of non-overlapping control volumes and numerically implementing Equation (1) over each control volume. Node-centered schemes define solution values at the mesh nodes. In 3D, the primal cells are tetrahedra, prisms, hexahedra, or pyramids. The *median-dual* partition [9, 10] used to generate control volumes is illustrated in Figure 1 for 2D. These non-overlapping control volumes cover the entire computational domain and compose a mesh that is dual to the primal mesh.

The main target discretization of interest for the model diffusion equation and the viscous terms of the RANS equations is obtained by the Green-Gauss scheme [11, 12], which is a widely-used viscous discretization for node-centered schemes and is equivalent to a Galerkin finite-element discretization for tetrahedral grids. For mixed-element cells, edge-based contributions are used to increase the h-ellipticity of the operator [11, 12]. The inviscid terms are discretized by a standard edge-based method with unweighted least-squares gradient reconstruction and Roe's approximate Riemann solver [13]. Limiters are not used for the problems considered in this paper. The convection terms of the turbulence equation are discretized with first-order accuracy.

III. Agglomeration Scheme

As described in the previous papers [6,7], the grids are agglomerated within a topology-preserving framework, in which hierarchies are assigned based on connections to the computational boundaries. *Corners* are identified as grid points with three or more boundary-condition-type closures (or three or more boundary slope discontinuities). *Ridges* are identified as grid points with two boundary-condition-type closures (or two boundary slope discontinuities). *Valleys* are identified as grid points with a single boundary-condition-type closure. *Interiors* are identified as grid points without any boundary condition. The agglomerations proceed hierarchically from seeds within the topologies — first corners, then ridges, then valleys, and finally interiors. Rules are enforced

Hierarchy of Agglomeration	Hierarchy of Added Volume	Agglomeration Admissibility
corner	any	disallowed
ridge	interior	disallowed
ridge	valley	disallowed
ridge	ridge	conditional
valley	interior	disallowed
valley	valley	conditional
interior	interior	allowed

Table 1. Admissible agglomerations.

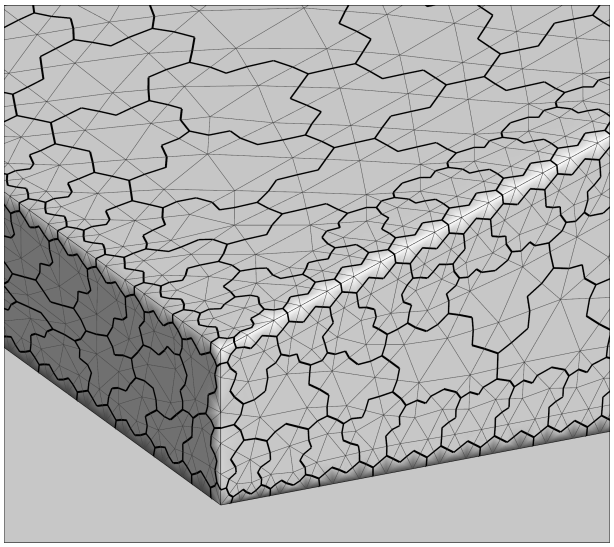


Figure 2. Trailing-edge area of a 3D wing agglomerated by the hierarchical scheme. Primal grid is shown by thin lines; agglomerated grid is shown by thick lines.

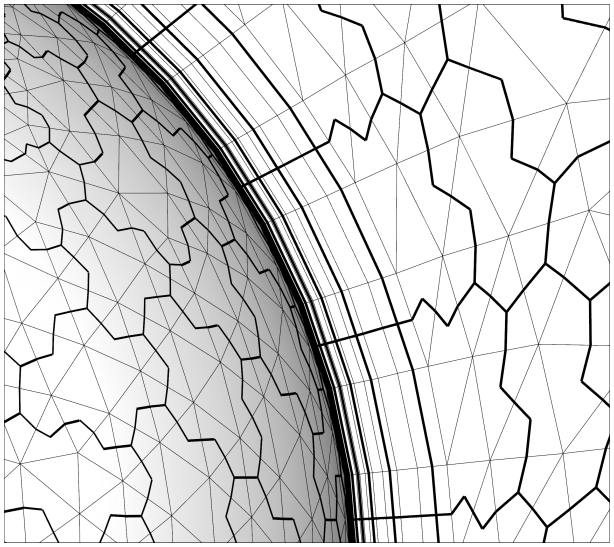


Figure 3. Typical implicit line-agglomeration showing a curved solid body surface on the left and a symmetry plane on the right. The projection of the line-agglomerations can be seen on the symmetry plane.

to maintain the boundary condition types of the finer grid within the agglomerated grid. Candidate volumes to be agglomerated are vetted against the hierarchy of the currently agglomerated volumes. In this work, we use the rules summarized in Table 1. In order to enable a valid non-degenerate stencil for linear prolongation and least-squares gradients near boundaries [7], the rules reflect less agglomerations near boundaries than in the interior. Corners are never agglomerated, ridges are agglomerated only with ridges, and valleys are agglomerated only with valleys. A typical boundary agglomeration generated by the above rules is shown in Figure 2. The conditional entries denote that further inspection of the connectivity of the topology must be considered before agglomeration is allowed. For example, a ridge can be agglomerated into an existing ridge agglomeration if the two boundary conditions associated with each ridge are the same. For valleys or interiors, all available neighbors are collected and then agglomerated one by one in the order of larger number of edge-connections to a current agglomeration until the maximum threshold of agglomerated nodes (4 for valleys; 8 for interiors) is reached. The prolongation operator P_1 is modified to prolong only from hierarchies equal or above the hierarchy of the prolonged point. Hierarchies on each agglomerated grid are inherited from the finer grid.

For the results reported in this paper, we employ agglomeration scheme II described in previous papers [6,7]. It has been modified to deal with viscous meshes using implicit-line agglomeration. It performs the agglomeration in the following sequence:

1. Agglomerate viscous boundaries (bottom of implicit lines).
2. Agglomerate prismatic layers through the implicit lines (implicit-line agglomeration).
3. Agglomerate the rest of the boundaries.

4. Agglomerate the interior.

The second step is a line-agglomeration step where volumes are agglomerated along implicit lines starting from the volume directly above the boundary volume. Specifically, we first agglomerate volumes corresponding to the second and third entries in the implicit-line lists associated with each of the fine-grid volumes contained in a boundary agglomerate. The line agglomeration continues to the end of the shortest line among the lines associated with the boundary agglomerate. This line-agglomeration process preserves the boundary agglomerates. Figure 3 illustrates typical implicit line-agglomeration near a curved solid body. The implicit line-agglomeration preserves the line structure of the fine grid on coarse grids, so that line-relaxations can be performed on all grids to address the grid anisotropy. If no implicit lines are defined, typical for inviscid grids, the first two steps are skipped.

In each boundary agglomeration (steps 1 and 3), agglomeration begins with corners, creates a front list defined by collecting volumes adjacent to the agglomerated corners, and proceeds to agglomerate volumes in the list (while updating the list as agglomeration proceeds) in the order of ridges and valleys. During the process, a volume is selected from among those in the same hierarchy that has the least number of non-agglomerated neighbors, thereby reducing the occurrences of agglomerations with small numbers of volumes. A heap data-structure is utilized to efficiently select such a volume. The agglomeration continues until the front list becomes empty. Finally, for both valleys and interiors, agglomerations containing only a few volumes (typically one) are combined with other agglomerations.

Figures 4 and 5 show primal grids and agglomerations for the F6 wing-body combination and the DPW-W2 [14] grids. These grids are viscous grids; the primal grid has prismatic viscous layers around the body and the wing. Coarsening ratios are indicated by r_k $(k = 1, 2, 3, 4)$ in the parenthesis. Line agglomeration was applied in these regions. Figures 6, 7, and 8 show primal grids and agglomerations for a wing-body combination, a wing-flap combination, and a 3D wing with a blunt trailing edge — all are pure-tetrahedral inviscid grids.

IV. Single-grid Iterations

Single-grid iteration scheme is based on the implicit formulation:

$$\left(\frac{\Omega}{\Delta\tau} + \frac{\partial \hat{R}^*}{\partial U}\right)\delta U = -\hat{R}(U), \tag{2}$$

where $\hat{R}(U)$ is the target residual computed for the current solution U, $\Delta\tau$ is a pseudo-time step, $\frac{\partial \hat{R}^*}{\partial U}$ is an exact/approximate Jacobian, and δU is the change to be applied to the solution U. An approximate solution to Equation (2) is computed by a certain number of iterations on the linear system (linear-sweeps). Update of U completes one nonlinear iteration. The RANS equations are iterated in a loosely-coupled formulation, updating the turbulence variables after the mean-flow variables at each nonlinear iteration. The left-hand-side operator of Equation (2) includes an exact linearization of the viscous (diffusion) terms and a linearization of the inviscid terms involving first-order contributions only. Thus, the iterations represent a variant of defect correction. Typically in single-grid FUN3D RANS applications, the first-order Jacobian corresponds to the linearization of Van Leer's flux-vector splitting. For inviscid cases, we consider using the linearization of Roe's approximate Riemann solver. Jacobians are updated after each iteration. The linear sweeps performed before each nonlinear update include ν_p sweeps of the point multi-color Gauss-Seidel relaxation performed through the entire domain followed by ν_l line-implicit sweeps in stretched regions. The line-implicit sweeps are applied only when solving the model diffusion or the RANS equations. In a line-implicit sweep, unknowns associated with each line are swept simultaneously by inverting a block tridiagonal matrix [7]. For RANS simulations, $\nu_p = \nu_l = 15$ for the mean-flow equations and $\nu_p = \nu_l = 10$ for the turbulence equation. For the model diffusion equation, only one linear sweep is performed per nonlinear iteration, i.e., $\nu_p = \nu_l = 1$, and the exact Jacobian computed only once at the beginning of the entire calculation. In spite of linearity of the model diffusion equation, computations of $\hat{R}(U)$ in Equation (2) do not employ the exact Jacobian, thus, providing a better similarity to nonlinear computations.

V. Multigrid

Elements of the multigrid algorithm are presented in this section. In this study, we do not explore various algorithmic options, relying on the methods that proved effective from the previous studies.

V.A. Multigrid V-Cycle

The multigrid method is based on the full-approximation scheme (FAS) [1, 15] where a coarse-grid problem is solved/relaxed for the solution approximation. A correction, computed as the difference between the restricted fine-grid solution and the coarse-grid solution, is prolonged to the finer grid to update the fine-grid solution. The two-grid FAS is applied recursively through increasingly coarser grids to define a V-cycle. A V-cycle, denoted as $V(\nu_1, \nu_2)$, uses ν_1 relaxations performed at each grid before proceeding to the coarser grid and ν_2 relaxations after coarse-grid correction. On the coarsest grid, relaxations are performed to bring two orders of magnitude residual reduction or until the maximum number of relaxations, 10, is reached.

V.B. Inter-Grid Operators

The control volumes of each agglomerated grid are found by summing control volumes of a finer grid. An operator that performs the summation is given by a conservative agglomeration operator, R_0, which acts on fine-grid control volumes and maps them onto the corresponding coarse-grid control-volumes. Any agglomerated grid can be defined, therefore, in terms of R_0 as

$$\Omega^c = R_0 \Omega^f, \tag{3}$$

where superscripts c and f denote entities on coarser and finer grids, respectively. On the agglomerated grids, the control volumes become geometrically more complex than their primal counterparts and the details of the control-volume boundaries are not retained. The directed area of a coarse-grid face separating two agglomerated control volumes, if required, is found by lumping the directed areas of the corresponding finer-grid faces and is assigned to the virtual edge connecting the centers of the agglomerated control volumes.

Residuals on the fine grid, $\hat{R}^f$, corresponding to the integral equation (1) are restricted to the coarse grid by the conservative agglomeration operator, R_0, as

$$\hat{R}^c = R_0 \hat{R}^f, \tag{4}$$

where $\hat{R}^c$ denotes the fine-grid residual restricted to the coarse grid.

The fine-grid solution approximation, U^f, is restricted as

$$U_0^c = \frac{R_0(U^f \Omega^f)}{\Omega^c}, \tag{5}$$

where U_0^c denotes the fine-grid solution approximation restricted to the coarse grid. The restricted approximation is then used to define the forcing term to the coarse-grid problem as well as to compute the correction, $(\delta U)^c$:

$$(\delta U)^c = U^c - U_0^c, \tag{6}$$

where U^c is an updated coarse-grid solution obtained directly from the coarse-grid problem.

The correction to the finer grid is prolonged typically through the prolongation operator, P_1, that is exact for linear functions, as

$$(\delta U)^f = P_1 (\delta U)^c. \tag{7}$$

The operator P_1 is constructed locally using linear interpolation from a tetrahedra defined on the coarse grid. The geometrical shape is anchored at the coarser-grid location of the agglomerate that contains the given finer control volume. Other nearby points are found by the adjacency graph. An enclosing simplex is sought that avoids prolongation with non-convex weights and, in situations where multiple geometrical shapes are found, the first one encountered is used. Where no enclosing simplex is found, the simplex with minimal non-convex weights is used.

V.C. Coarse-Grid Discretizations

For inviscid coarse-grid discretization, a first-order edge-based scheme is employed. For the model equation and the viscous term in the RANS equations, two classes of coarse-grid discretizations were previously studied [6, 7]: the Average-Least-Squares (Avg-LSQ) and the edge-terms-only (ETO) schemes. The consistent Avg-LSQ schemes are constructed in two steps: first, LSQ gradients are computed at the control volumes; then, the average of the control-volume LSQ gradients is used to approximate a gradient at the face, which is augmented with the edge-based directional contribution to determine the gradient used in the flux. There are two variants

	Inviscid	Viscous (Diffusion)
Primal grid	Second-order edge-based reconstruction	Green-Gauss
Coarse grids	First-order edge-based reconstruction	Face-Tangent Avg-LSQ

Table 2. Summary of discretizations used to define the residual, $\hat{R}$.

	Inviscid	Viscous (Diffusion)
Primal grid	Approximate (first-order scheme)	Exact ($\hat{R}^* = \hat{R}$)
Coarse grids	Exact or Approximate	Approximate (edge-terms only)

Table 3. Summary of Jacobians, $\frac{\partial \hat{R}^*}{\partial U}$.

of the Avg-LSQ scheme. One uses the average-least-squares gradients in the direction normal to the edge (edge-normal gradient construction). The other uses the average-least-squares gradients along the face (face-tangent gradient construction).

The ETO discretizations are obtained from the Avg-LSQ schemes by taking the limit of zero Avg-LSQ gradients. The ETO schemes are often cited as a thin-layer discretization in the literature [2, 3, 4]; they are positive schemes but are not consistent (i.e., the discrete solutions do not converge to the exact continuous solution with consistent grid refinement) unless the grid is orthogonal [13, 16]. As shown in the previous papers [6, 7], ETO schemes lead to deterioration of the multigrid convergence for refined grids, and therefore are not considered in this paper. For practical applications, the face-tangent Avg-LSQ scheme was found to be more robust than the edge-normal Avg-LSQ scheme. It provides superior diagonal dominance in the resulting discretization [6, 7]. In this study, therefore, we employ the face-tangent Avg-LSQ scheme as a coarse-grid discretization for the model equation and the viscous term.

For excessively-skewed faces (over 90° angle between the outward face normal and the corresponding outward edge vector), which can arise on agglomerated grids, the gradient is computed by the Avg-LSQ scheme and edge contributions are ignored. The Galerkin coarse-grid operator [1], which was considered in a previous study, is not considered here since the method was found to be grid-dependent and slowed down the multigrid convergence for refined grids [6]. For inviscid discretization, we employ a first-order edge-based discretization on coarse grids. Table 2 shows a summary of discretizations used.

V.D. Relaxations

Relaxation scheme is similar to the single-grid iteration described in Section IV with the following important differences. On coarse grids, the Avg-LSQ scheme used for viscous terms has a larger stencil than the Green-Gauss scheme implemented on the target grid and its exact linearization has not been used; instead relaxation of the Avg-LSQ scheme relies on an approximate linearization, which consists of edge terms only. For inviscid cases, the first-order Jacobian is constructed based on Roe's approximate Riemann solver, and thus it is exact on coarse grids where the first-order scheme is used for the residual. For RANS cases, the first-order Jacobian is constructed based on Van Leer's flux-vector splitting, but the inviscid part of the residual is computed by Roe's approximate Riemann solver. Therefore, the Jacobian is approximate on both the primal and coarse grids. Table 3 summarizes the Jacobians used for inviscid and viscous (diffusion) terms on the primal and coarse grids. In multigrid nonlinear applications, Jacobians are evaluated at the beginning of a cycle and frozen during the cycle. For inviscid and RANS flow simulations, significantly fewer linear sweeps are used in a multigrid relaxation than in a single-grid iteration: $\nu_p = \nu_l = 5$ for both the mean flow and turbulence relaxations. For the model diffusion equation, still only one sweep is performed per relaxation.

V.E. Cost of Multigrid V-Cycle

All of the computations in the paper use FAS multigrid. For the linear model diffusion equation, the computer time would be reduced if the corresponding correction scheme (CS) cycle is used. To estimate relative cost of multigrid cycles in comparison with single-grid iterations, the cost of nonlinear residual evaluations, relaxation updates, and Jacobian evaluations needs to be taken into account. Suppose that a nonlinear relaxation and

Model	W_{SG}^{MG}	J	σ^{SG}	σ^{MG}
Diffusion	4.5	0	1.16(1, 1)	1.16(1, 1)
Inviscid	1.8	2.0	3.70(15, 0)	1.90(5, 0)
RANS	1.5	3.0	6.12(15, 15):(10, 10)	3.16(5, 5):(5, 5)

Table 4. Summary of costs and typical numbers of linear-sweeps. The multigrid cycle is a 5-level $V(2,1)$ with a typical coarsening ratio 8. The numbers in parenthesis denote the number of point and line sweeps, respectively, and the second set for RANS denotes the number of point and line sweeps of the turbulent equation.

a Jacobian evaluation cost σ and J times a nonlinear residual evaluation, respectively. Then, the cost of a single-grid iteration relative to the cost of a nonlinear residual evaluation is given by

$$W^{SG} = \sigma^{SG} + J, \tag{8}$$

where the superscript SG denotes single-grid iterations. On the other hand, a multigrid cycle involves $\nu_1 + \nu_2$ nonlinear relaxations, a nonlinear residual evaluation before restriction, and a Jacobian evaluation per cycle per grid. A residual evaluation on coarse grids is also required to form the FAS forcing term. The cost of a multigrid cycle, MG, relative to the cost of a fine-grid nonlinear residual evaluation is given by

$$W^{MG} = C\left[(\nu_1 + \nu_2)\sigma^{MG} + J + 1\right] + C - 1, \tag{9}$$

where C is a coarse-grid factor,

$$C = 1 + \frac{1}{r_1} + \frac{1}{r_1 r_2} + \frac{1}{r_1 r_2 r_3} + \cdots. \tag{10}$$

Here, r_k is the agglomeration ratio of the k-th agglomerated grid. The relative cost, W_{SG}^{MG}, of a V-cycle is therefore given by

$$W_{SG}^{MG} = \frac{W^{MG}}{W^{SG}}. \tag{11}$$

Table 4 shows values of W_{SG}^{MG}, σ and J for each equation set within the single-grid iteration and the multigrid method. The values for σ and J are based on measured computer times associated with residual evaluation, Jacobian evaluation, and linear-sweeps on the primal grid for particular configurations. The corresponding values on a per node basis vary from the tabulated values on the coarser grids and across configurations. Thus, Equation (11) serves as a reasonable approximation to the expected code performance. Note that the Jacobian computation has been ignored for the model diffusion equation. This is because the Jacobian is constant for the linear problem and therefore it is computed only once and never updated. Observe also that σ is much smaller in the multigrid cycle than in the single-grid iteration for the nonlinear cases. This saving comes from much fewer linear-sweeps in the multigrid method. We experimentally found that the multigrid convergence did not depend heavily on the number of linear-sweeps. Increasing them further does not reduce the number of cycles for convergence, but it merely increases the CPU time. The numbers of linear-sweeps shown for the single-grid method are typical numbers considered sufficient for robust computations with a reasonably large CFL number. The relative cost (11) computed based on the measured σ and J are shown in the third column of the table. Considering a 5-level $V(2,1)$ cycle with a coarsening ration of 8, the relative cost is found to be 4.5 for the diffusion equation, 1.8 for the inviscid equation, and 1.5 for the RANS equations.

VI. Results for Complex Geometries

All calculations presented in this paper were performed with a single processor. Parallelization of the multigrid algorithm is currently underway. The multigrid cycle is a 5-level $V(2,1)$ for all cases.

VI.A. Model Diffusion Equation

The multigrid method was applied on grids generated for two practical geometries: the F6 wing-body and the DPW-W2 wing-alone cases [14]. Both grids are tetrahedral, but prisms are used in a highly-stretched viscous layer near the solid boundary. Pyramidal cells are also present around the transitional region. The multigrid $V(2,1)$ cycle is applied and compared with single-grid iterations. The CFL number is set to infinity. For the F6

Grid	Flow Model	Cost of V-cycle (W_{SG}^{MG})	Expected Speed-up (Equation (12))	Actual Speed-up (Using actual CPU time)
F6WB	Diffusion	4.7	69.4	62.6
DPW-W2	Diffusion	4.8	24.8	22.4
Wing-Body	Inviscid	1.8	5.2	5.2
Wing-Flap	Inviscid	1.8	2.0	1.9
NACA15 Wing	Inviscid	1.8	2.9	2.8
F6WB	RANS	1.6	5.4	5.0

Table 5. Cost of V-cycle relative to a single-grid iteration and speed-up factor. The expected speed-up factors have been computed with the actual coarsening ratio.

Grid	Size (nodes)	Inflow Mach number	Angle of Attack	N_{ramp}
Wing-Body	1,012,189	0.3	0°	10
Wing-Flap	1,184,650	0.3	2°	10
NACA15 Wing	2,039,914	0.3	2°	100

Table 6. Summary of grid sizes and parameters for the inviscid cases.

wing-body grid (1,121,301 nodes), the grids and convergence results are shown in Figure 4. The speed-up factor is 63 in CPU time. A similar result was obtained for the DPW-W2 grid (1,893,661 nodes) as shown in Figure 5. The speed-up factor is nearly 22 in this case. The cost of one V-cycle computed according to Equation (11) with actual coarsening ratios is shown for each case in the fourth column of Table 5. It shows that one V-cycle costs nearly 4 single-grid iterations. The fifth column is an expected speed-up factor based on the number of single-grid iterations (N_{SG}), the number of multigrid cycles (N_{MG}), and the factor W_{SG}^{MG}:

$$\frac{N_{SG}}{N_{MG}W_{SG}^{MG}}. \tag{12}$$

The last column is the actual speed-up factor computed as a ratio of the total single-grid CPU time to the total multigrid CPU time. A fairly good agreement can be observed between the expected and actual speed-up factors.

VI.B. Inviscid Flows

The multigrid method was applied to three inviscid cases: low-speed subsonic flows over a wing-body configuration, a wing-flap configuration, and a NACA15 wing with a blunt trailing edge. Table 6 shows a summary of grid sizes and parameters. N_{ramp} denotes the number of first iterations/cycles over which the CFL number is ramped from 10 to 200 for single-grid/multigrid calculations. The multigrid cycle is $V(2,1)$ for these cases.

Figure 6 shows the grids and convergence results for the wing-body configuration case. As Figure 6(f) shows, the multigrid converges (to machine zero) 5 times faster in CPU time than the single-grid iterations. The convergence results for the wing-flap configuration is given in Figure 7(f). It shows that the multigrid converges (to machine zero) nearly 2 times faster in CPU time than the single-grid iterations. For the NACA15 wing case, the solution does not fully converge in either single-grid or multigrid computations apparently due to an unsteady behavior near the blunt trailing edge. However, as shown in Figure 8(f), the multigrid drives the residual more rapidly down to the level of 10^{-9} than the single-grid iteration.

In all three cases, the ratio of the number of multigrid cycles to the number of single-grid iterations is about twice the speed-up factor in terms of the CPU time. It implies that the cost of one multigrid $V(2,1)$ cycle is close to the cost of two single-grid iterations. These results are in good agreement with the estimates of the cost of one V-cycle computed according to Equation (11) and shown in the fourth column of Table 5. The estimated cost of one V-cycle is 1.8 of the single-grid iteration cost for all inviscid cases. The estimated speed-up shown in the fifth column agrees well with the actual speed-up shown in the last column.

VI.C. Turbulent Flows (RANS)

We applied the multigrid algorithm to a RANS simulation on the F6 wing-body grid shown in Figure 4. The inflow Mach number is 0.3, the angle of attack is 1 degree, and the Reynolds number is 2.5 million. For this case, a prolongation operator that is exact for a constant function is used. The P_1 prolongation operator encountered a difficulty on a boundary for this particular configuration, and it is currently under investigation. The CFL number is not ramped in this case, but set to 200 for the mean-flow equations and 30 for the turbulence equation.

Convergence results are shown in Figure 9. As can be seen, the multigrid achieved four orders of reduction in the residual 5 times faster in CPU time than the single-grid iteration. For this case, neither the multigrid nor single-grid method fully converges seemingly due to a separation near the wing-body junction. Four orders of magnitude reduction is just about how far a single-grid is run in practice for this particular configuration. The comparison of the number of cycles with the number of single-grid iterations in the figure implies that the CPU time for a multigrid $V(2,1)$ cycle is less than the CPU time for two single-grid iterations. As shown in Table 5, one multigrid V-cycle actually costs 1.6 single-grid iterations, indicating a good agreement between the expected and actual speed-up factors.

VII. Concluding Remarks

An agglomerated multigrid algorithm has been applied to inviscid and viscous flows over complex geometries. A robust fully-coarsened hierarchical agglomeration scheme was described for highly-stretched viscous grids, incorporating consistent viscous discretization on coarse grids. Results for practical simulations show that impressive speed-ups can be achieved for realistic flows over complex geometries.

Parallelization of the developed multigrid algorithm is currently underway to expand the applicability of the developed technique to larger-scale computations and to demonstrate grid-independent convergence of the developed multigrid algorithm.

Acknowledgments

All results presented were computed within the FUN3D suite of codes at NASA Langley Research Center (`http://fun3d.larc.nasa.gov/`). This work was supported by the NASA Fundamental Aeronautics Program through NASA Research Announcement Contract NNL07AA23C.

References

[1]Trottenberg, U., Oosterlee, C. W., and Schüller, A., *Multigrid*, Academic Press, 2001.

[2]Mavriplis, D. J., "Multigrid Techniques for Unstructured Meshes," *VKI Lecture Series VKI-LS 1995-02, Von Karman Institute for Fluid Dynamics, Rhode-Saint-Genese, Belgium*, 1995.

[3]Mavriplis, D. J., "Unstructured Grid Techniques," *Annual Review of Fluid Mechanics*, Vol. 29, 1997, pp. 473–514.

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

[5]Mavriplis, D. J., "An Assessment of Linear versus Non-Linear Multigrid Methods for Unstructured Mesh Solvers," *Journal of Computational Physics*, Vol. 275, 2002, pp. 302–325.

[6]Nishikawa, H., Diskin, B., and Thomas, J. L., "Critical Study of Agglomerated Multigrid Methods for Diffusion," *AIAA Journal*, Vol. 48, No. 4, 2010, pp. 839–847.

[7]Thomas, J. L., Diskin, B., and Nishikawa, H., "A Critical Study of Agglomerated Multigrid Methods for Diffusion on Highly-Stretched Grids," *Computers and Fluids*, 2010, to appear.

[8]Spalart, P. R. and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA paper 92-0439, 1992.

[9]Barth, T. J., "Numerical Aspects of Computing Viscous High Reynolds Number Flows on Unstructured Meshes," AIAA Paper 91-0721, 1991.

[10]Haselbacher, A. C., *A Grid-Transparent Numerical Method for Compressible Viscous Flow on Mixed Unstructured Meshes*, Ph.D. thesis, Loughborough Universit, 1999.

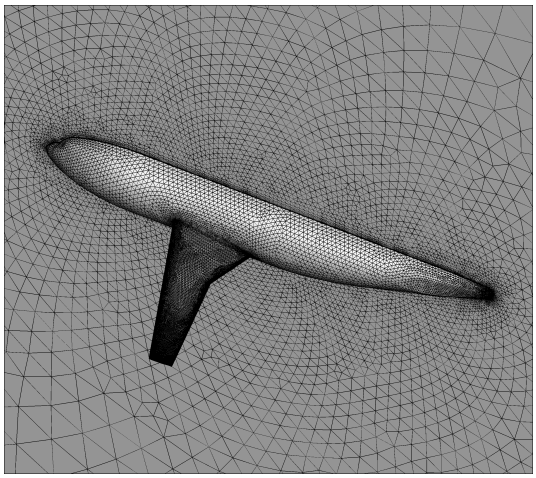
[11]Anderson, W. K. and Bonhaus, D. L., "An implicit upwind algorithm for computing turbulent flows on unstructured grids," *Computers and Fluids*, Vol. 23, 1994, pp. 1–21.

[12]Diskin, B., Thomas, J. L., Nielsen, E. J., Nishikawa, H., and White, J. A., "Comparison of Node-Centered and Cell-Centered Unstructured Finite-Volume Discretizations. Part I: Viscous Fluxes," *47th AIAA Aerospace Sciences Meeting*, AIAA Paper 2009-597, January 2009.

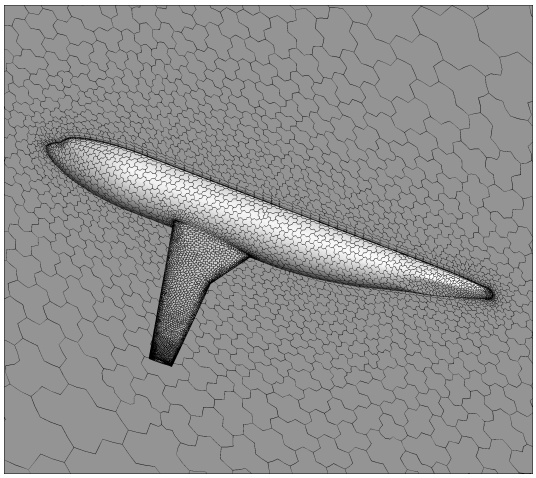
[13]Diskin, B. and Thomas, J. L., "Accuracy Analysis for Mixed-Element Finite-Volume Discretization Schemes," *NIA Report No. 2007-08*, 2007.

[14]"The DPW-II Workshop for the geometry," `http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/Workshop2/workshop2.html`.

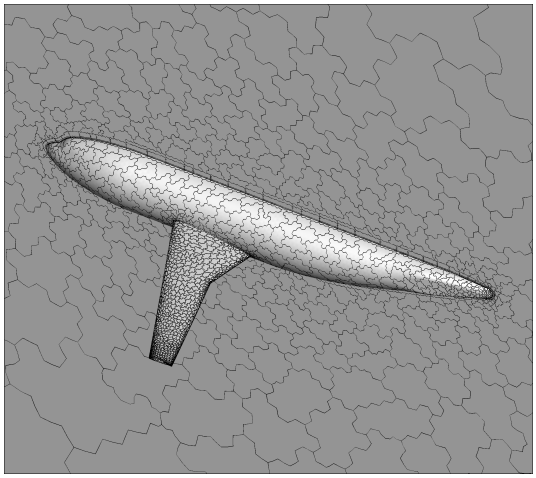
[15]Briggs, W. L., Henson, V. E., and McCormick, S. F., *A Multigrid Tutorial*, SIAM, 2nd ed., 2000.
[16]Thomas, J. L., Diskin, B., and Rumsey, C. L., "Towards Verification of Unstructured Grid Methods," *AIAA Journal*, Vol. 46, No. 12, December 2008, pp. 3070–3079.



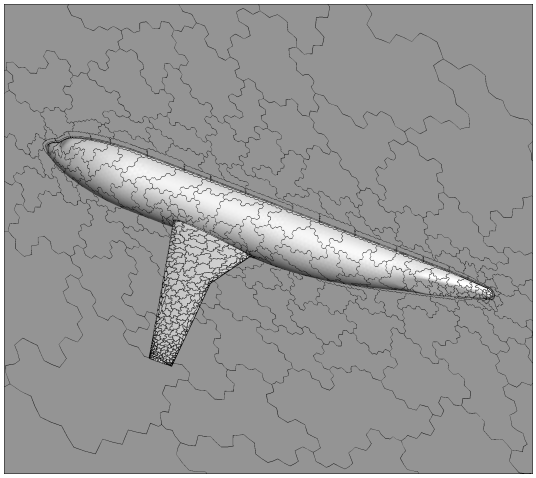
(a) Level 1: primal grid.



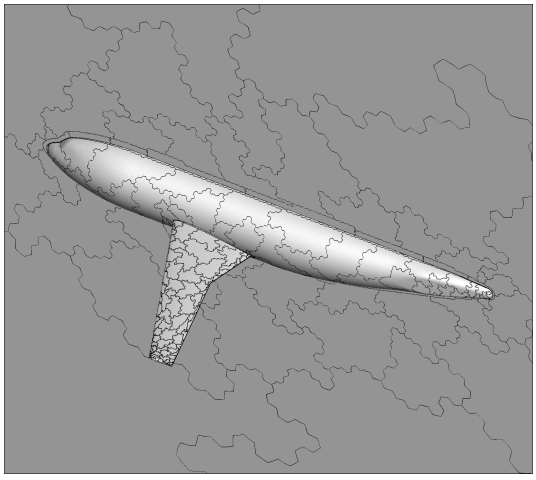
(b) Level 2: coarse grid ($r_1 = 6.5$).



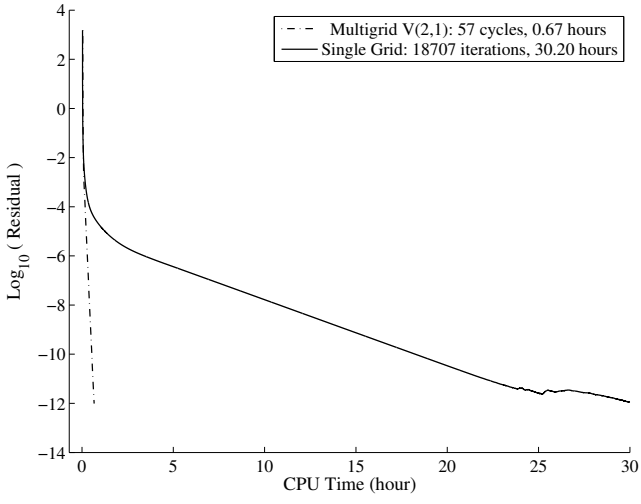
(c) Level 3: coarse grid ($r_2 = 6.2$).



(d) Level 4: coarse grid ($r_3 = 5.5$).

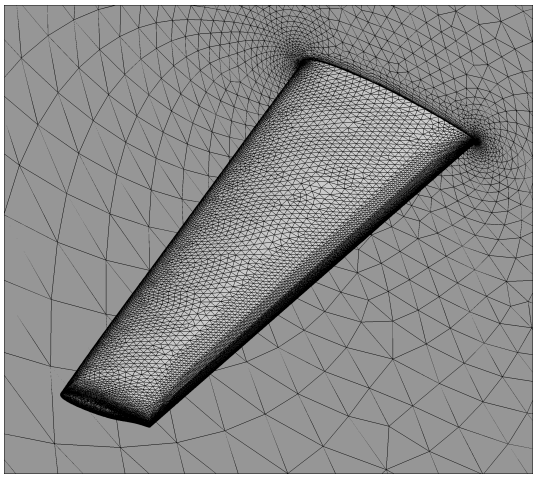


(e) Level 5: coarse grid ($r_4 = 4.6$).

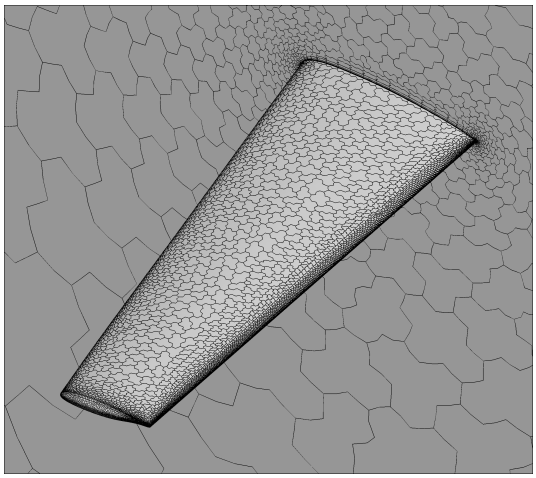


(f) Convergence history: residual versus CPU time.

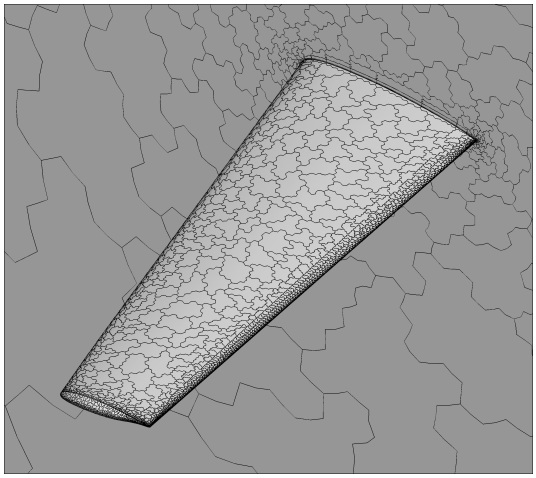
Figure 4. Grids and convergence of the model diffusion equation for the F6 wing-body combination.



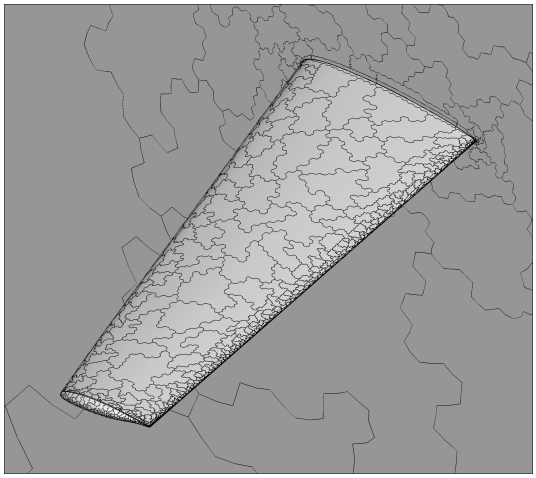
(a) Level 1: primal grid.



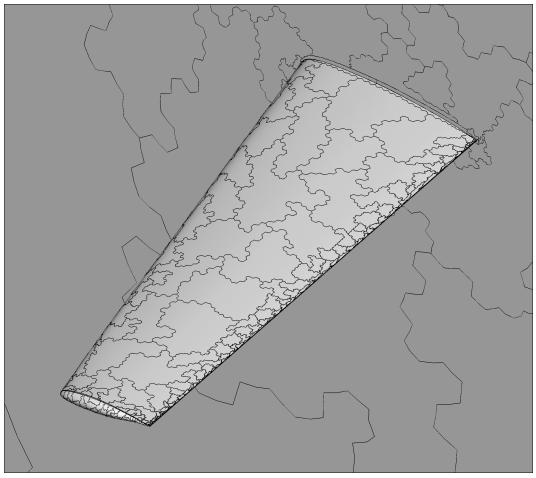
(b) Level 2: coarse grid ($r_1 = 6.3$).



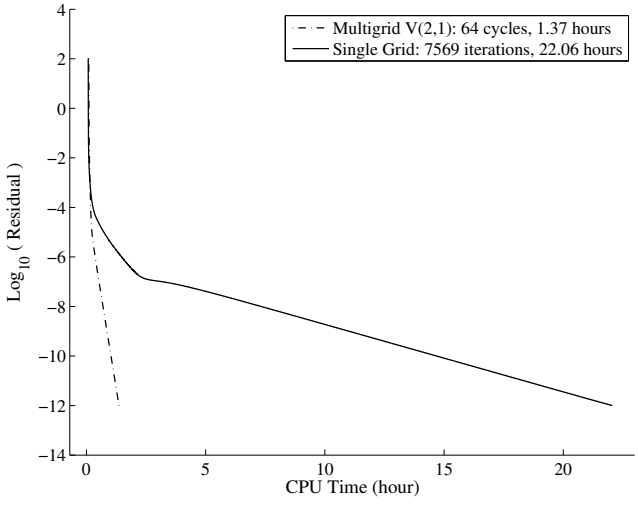
(c) Level 3: coarse grid ($r_2 = 5.8$).



(d) Level 4: coarse grid ($r_3 = 5.3$).

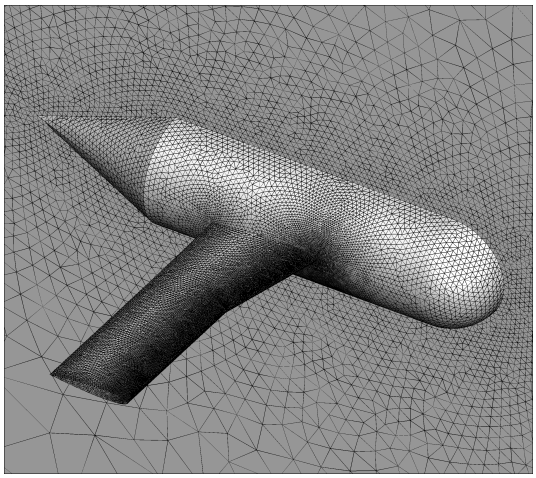


(e) Level 5: coarse grid ($r_4 = 4.7$).

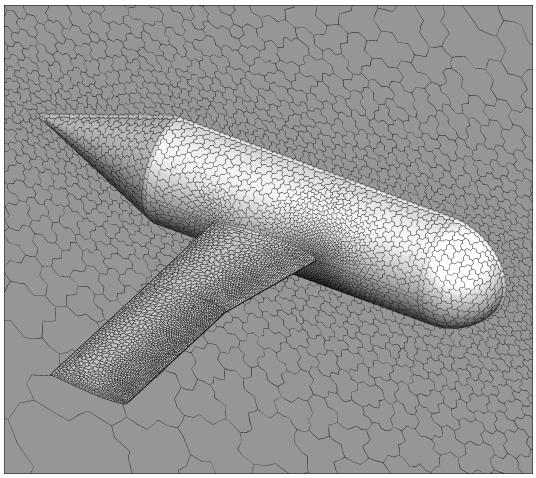


(f) Convergence history: residual versus CPU time.

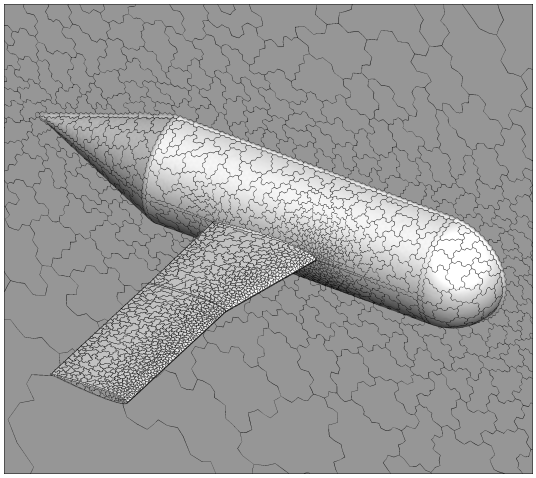
Figure 5. Grids and convergence of the model diffusion equation for the DPW-W2 case



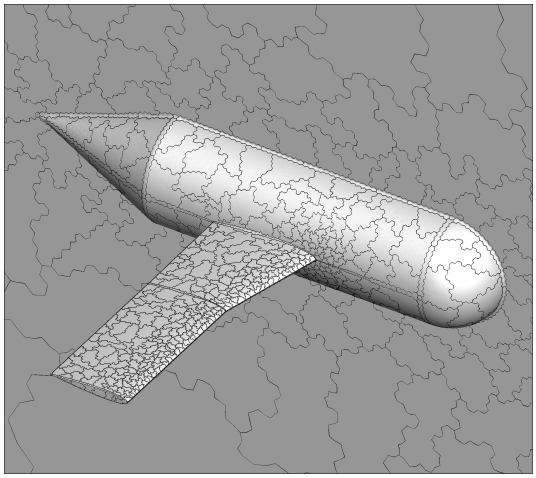
(a) Level 1: primal grid.



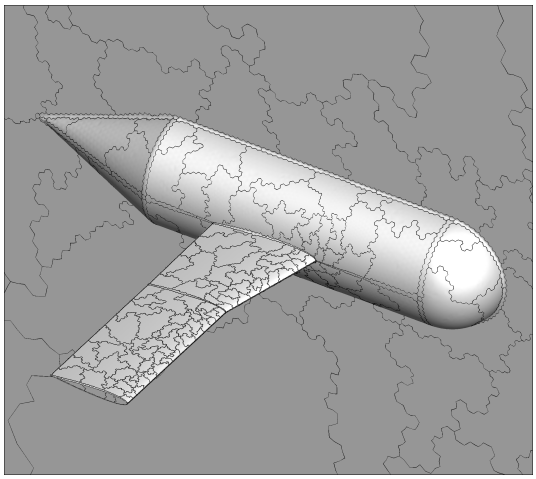
(b) Level 2: coarse grid ($r_1 = 7.0$).



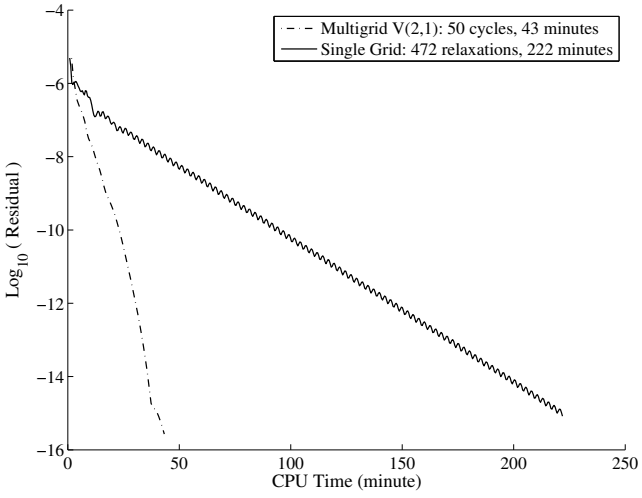
(c) Level 3: coarse grid ($r_2 = 6.8$).



(d) Level 4: coarse grid ($r_3 = 6.2$).

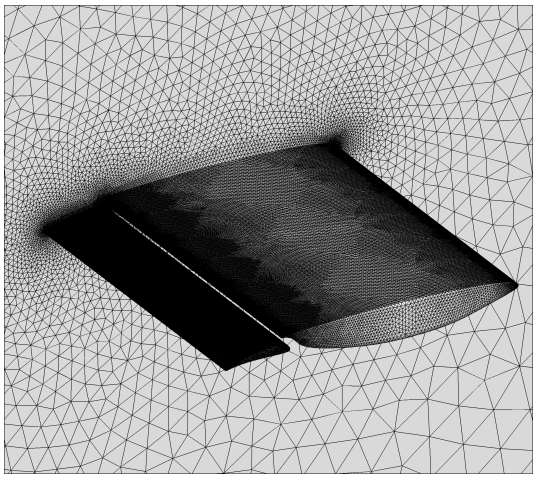


(e) Level 5: coarse grid ($r_4 = 4.9$).

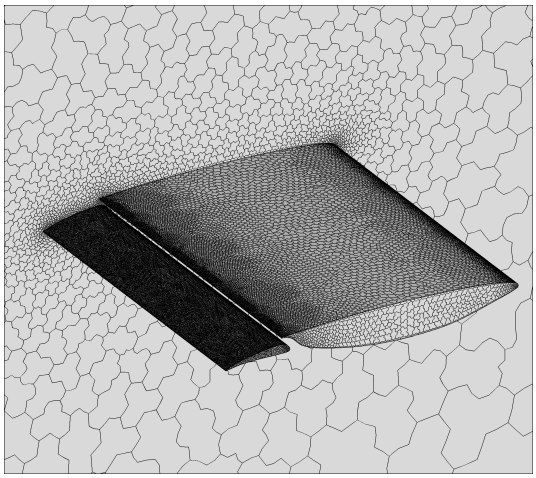


(f) Convergence history: residual versus CPU time.

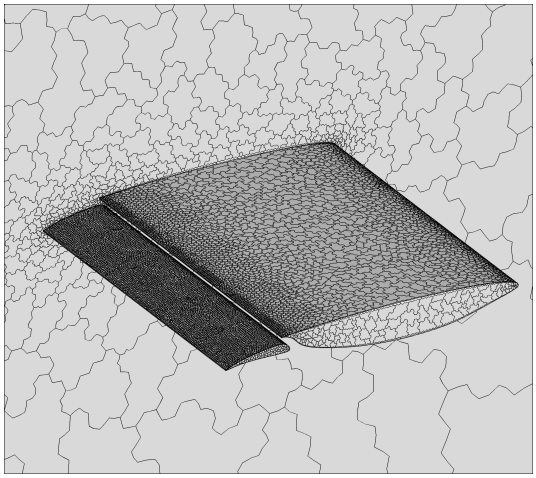
Figure 6. Grids and convergence for the wing-body inviscid case.



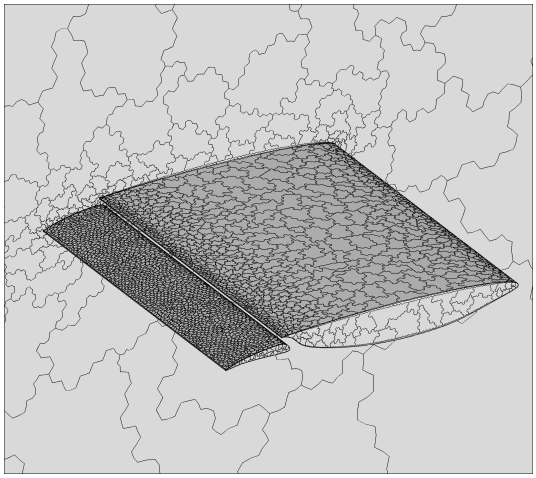
(a) Level 1: primal grid.



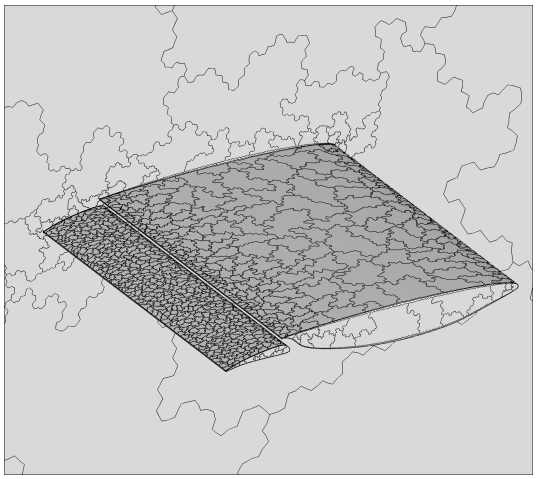
(b) Level 2: coarse grid ($r_1 = 5.9$).



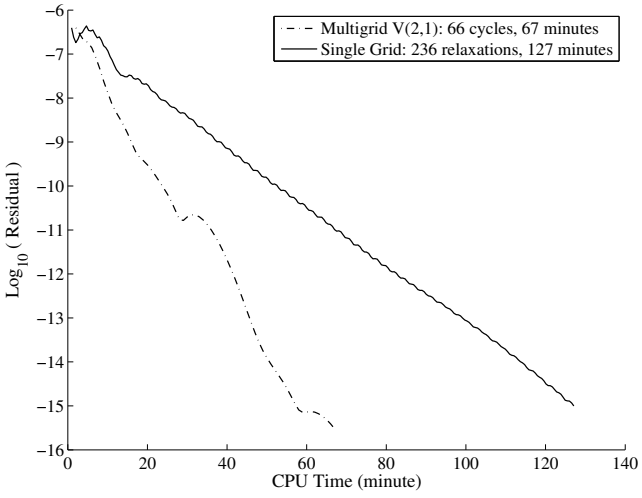
(c) Level 3: coarse grid ($r_2 = 5.1$).



(d) Level 4: coarse grid ($r_3 = 4.4$).

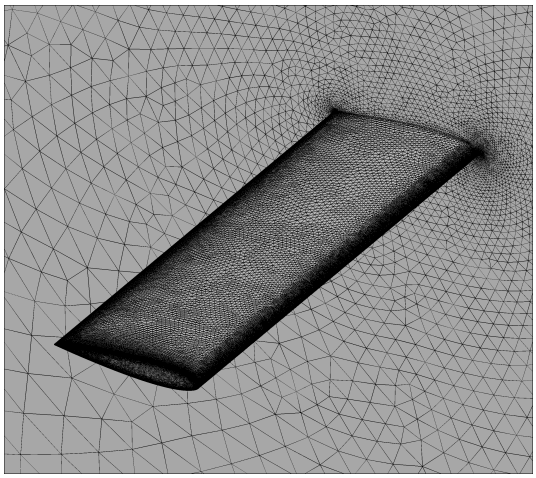


(e) Level 5: coarse grid ($r_4 = 3.8$).

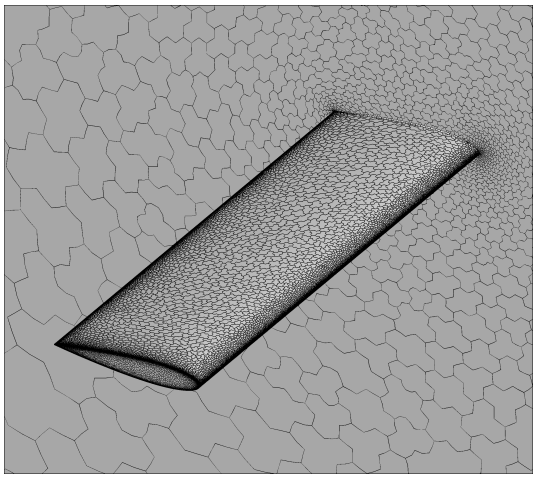


(f) Convergence history: residual versus CPU time.

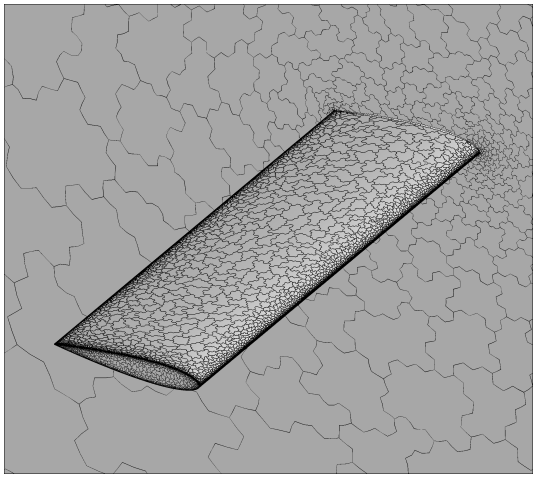
Figure 7. Grids and convergence for the wing-flap inviscid case.



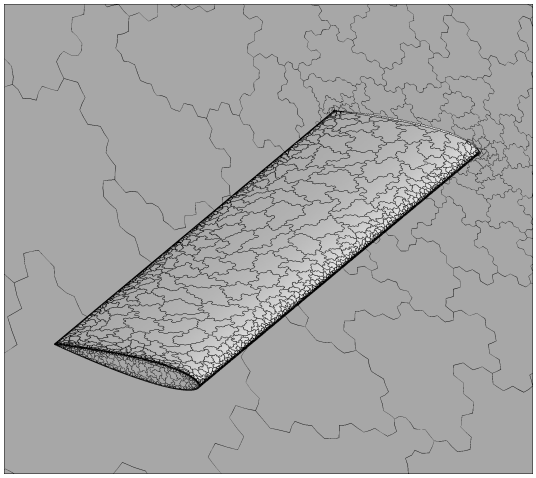
(a) Level 1: primal grid.



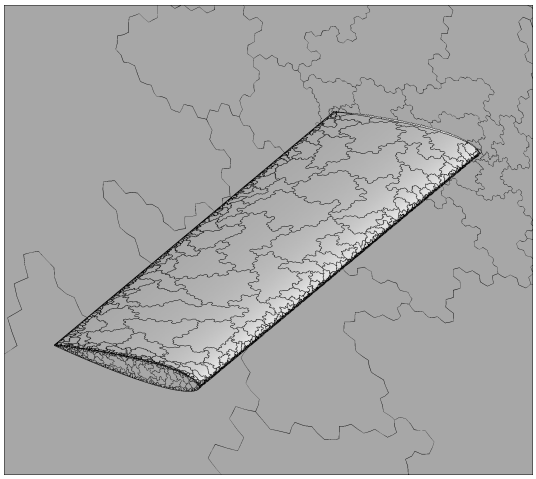
(b) Level 2: coarse grid ($r_1 = 6.3$).



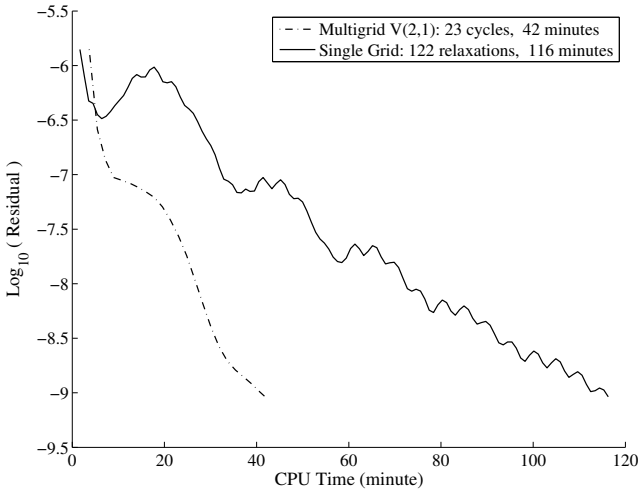
(c) Level 3: coarse grid ($r_2 = 5.6$).



(d) Level 4: coarse grid ($r_3 = 4.3$).



(e) Level 5: coarse grid ($r_4 = 3.3$).



(f) Convergence history: residual versus CPU time.

Figure 8. Grids and convergence for the NACA15-wing inviscid case.

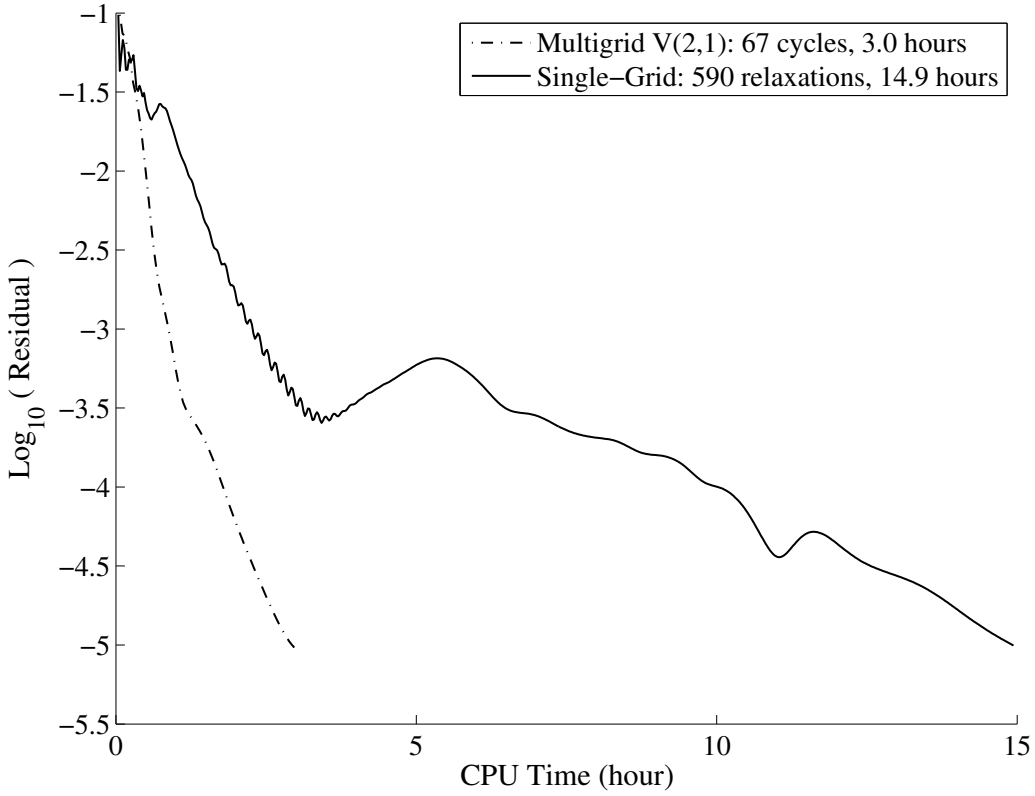


Figure 9. Residual versus CPU time for the F6 wing-body case (RANS).