Assessment of Preconditioner for a USM3D Hierarchical Adaptive Nonlinear Method (HANIM) (Invited)

Mohagna J. Pandya[1]
NASA Langley Research Center, Hampton, Virginia 23681, USA

Boris Diskin[2]
National Institute of Aerospace, Hampton, Virginia 23666, USA

and

James L. Thomas[3], Neal T. Frink[4]
NASA Langley Research Center, Hampton, Virginia 23681, USA

Enhancements to the previously reported mixed-element USM3D Hierarchical Adaptive Nonlinear Iteration Method (HANIM) framework have been made to further improve robustness, efficiency, and accuracy of computational fluid dynamic simulations. The key enhancements include a multi-color line-implicit preconditioner, a discretely consistent symmetry boundary condition, and a line-mapping method for the turbulence source term discretization. The USM3D iterative convergence for the turbulent flows is assessed on four configurations. The configurations include a two-dimensional (2D) bump-in-channel, the 2D NACA 0012 airfoil, a three-dimensional (3D) bump-in-channel, and a 3D hemisphere cylinder. The Reynolds Averaged Navier Stokes (RANS) solutions have been obtained using a Spalart-Allmaras turbulence model and families of uniformly refined nested grids. Two types of HANIM solutions using line- and point-implicit preconditioners have been computed. Additional solutions using the point-implicit preconditioner alone (PA) method that broadly represents the baseline solver technology have also been computed. The line-implicit HANIM shows superior iterative convergence in most cases with progressively increasing benefits on finer grids.

I. Introduction

Government agencies and industry rely heavily on computational aerodynamic analysis and design tools for conducting fundamental and applied aerodynamic research, and for cost-effective modification/development of air vehicles. One benefit of computational aerodynamic tools is a reduced, more strategic utilization of wind tunnel test resources. Another is their enabling capability for analyzing many parametric variations of a new vehicle concept with physics-based models. The range of applications is typically very diverse with many requirements that result in the need for flexible approaches to addressing problems.

The advancement of computational fluid dynamic (CFD) tools for aerodynamic analysis is an ongoing, vibrant process. Much CFD research is directed at fundamental method developments, and similarly, much is directed toward its effective utilization in application-driven research and production environments. After numerous collaborative technology evaluations, such as the NASA Drag Prediction Workshop [1], NASA High Lift Workshop [2], and international NATO task groups for assessing the CFD state-of-the-art for predicting aircraft stability and control [3], it is broadly accepted that large solution uncertainties are inherent from current CFD tools. Hence, it is widely recognized among the applied aerodynamics community that sponsoring organizations should encourage the

[1]Research Aerospace Engineer, Configuration Aerodynamics Branch, Mail Stop 499, Senior Member AIAA.
[2]NIA Research Fellow and Research Associate Professor, MAE Department, University of Virginia, Charlottesville, VA, Associate Fellow AIAA.
[3]Distinguished Research Associate, Computational Aerosciences Branch, Fellow AIAA.
[4]Senior Aerospace Engineer, Configuration Aerodynamics Branch, Mail Stop 499, Associate Fellow AIAA.

development and utilization of multiple CFD tools and approaches, in concert with closely coordinated experimental validation campaigns, in order to mature the tools more rapidly and to quantify their levels of uncertainty.

A wide range of numerical approaches is being explored and utilized throughout the CFD community. Many tools are based on the unstructured, general-cell-topology methods due to the ease of discretizing the domain. Within the unstructured framework, some approaches are based on a cell-vertex reconstruction, such as FUN3D [4], NSU3D [5], TAU [6], and the EDGE [7] codes. Others are based on a cell-centered reconstruction, such as USM3D [8], Kestrel [9], and Cobalt [10]. There are also successful tools developed around the hexahedral cell topology, such as the OVERFLOW overset grid solver [11], and the Cart3D Cartesian grid solver [12]. Efforts are also underway to bring higher-order finite volume [13] and Lattice Boltzmann [14] methods into the mainstream. These tools have been used effectively for critical applications, and each is mutually dependent on one another for spawning new advancements and for quantifying solution uncertainty.

The contribution of the present work is directed toward further improving the efficiency and accuracy of the 2nd-order unstructured cell-centered finite volume approach within an application-driven environment. The extensions are implemented into the USM3D Navier-Stokes flow solver, which is a part of the NASA Tetrahedral Unstructured Software System (TetrUSS) [15]. This tool has endured the multi-year rigor of extensive configuration aerodynamic research within NASA [15] to [17], and for product development within major airframe companies [18] to [20]. An overriding attribute of this solver has been its speed and robustness in providing solutions, often numbering in thousands.

Since many recent USM3D applications have ventured into predominately-separated flow regimes, attention is warranted on the sensitivity to near-wall cell topology of smooth surface flow separation, such as a wing leading edge near stall. A similar case can be made for computing heat transfer on high-speed bodies. The numerical simulations for such flows can be improved by using more flow-aligned anisotropic hexahedral or prismatic cells in the boundary layer and isotropic tetrahedral cells away from the surface. Motivated by this observation, a capability to support different cell topologies was previously implemented in the USM3D mixed-element version and verified on benchmark two-dimensional (2D) turbulent flow cases [21]. To improve the convergence and robustness of USM3D solutions, a Hierarchical Adaptive Nonlinear Iteration Method (HANIM) was recently implemented in the mixed-element USM3D [22].

The goals of the present work are to 1) make mixed-element USM3D [21], [22] significantly faster, thereby enabling even more rapid advancement of new aerodynamic concepts, and 2) provide the CFD community with the experiences and metrics of another benchmark study of state-of-the-art CFD methodology. The work in this paper builds upon the preceding version of mixed-element USM3D enhanced with a Hierarchical Adaptive Nonlinear Iteration Method (HANIM) framework [22]. This paper reports several new extensions to USM3D that are implemented to further improve robustness, efficiency, and accuracy of the Reynolds-Averaged Navier Stokes (RANS) solutions. The key extensions include a line-implicit preconditioner, a general discretely-consistent implementation of symmetry boundary condition, and an improved discretization of turbulence model source terms. The USM3D iterative convergence has been assessed for turbulent flows on four configurations. The configurations include a 2D bump-in-channel, the 2D NACA 0012 airfoil, a three-dimensional (3D) bump-in-channel, and a 3D hemisphere cylinder.

The RANS solutions have been obtained using the Spalart-Allmaras (SA) turbulence model and families of uniformly refined nested grids. Two types of HANIM solutions using line- and point-implicit preconditioners have been computed. Additional solutions using the point-implicit preconditioner alone (PA) method that broadly represents the baseline solver technology have also been computed. This paper focuses on the assessment of the iterative convergence of the line-implicit HANIM. In a separate study, the USM3D line-implicit HANIM is also used to establish reference solutions for 3D benchmark turbulent flows including the 3D bump-in-channel and the 3D hemisphere cylinder cases. An accompanying paper [23] reports on this reference solution study and verifies the accuracy of the line-implicit HANIM solutions.

The material in this paper is organized in the following order. Section II overviews components of the preceding version of mixed-element USM3D, including HANIM, that was reported a year ago [22]. Section III provides a detailed description of the newly implemented and used features for the USM3D computations reported in Section IV. Concluding remarks and future directions are presented in Sections V and VI, respectively.

II. Overview of Preceding USM3D Mixed-Element HANIM Solver

The salient features of the preceding version of mixed-element USM3D [22] are described here. USM3D solves a system of nonlinear flow equations that can be formally represented as

$$\boldsymbol{R}(\boldsymbol{Q}) = 0. \tag{1}$$

The discrete nonlinear operator, $\boldsymbol{R}(\boldsymbol{Q})$, represents a cell-centered discretization of steady-state RANS equations. The term "cell-centered" means that the flow variables are solved at the centroid of each cell.

A fully implicit formulation is used implying that the solution variables at the grid nodes and boundary faces as well as the cell gradients are computed solely from the current solution variables defined at the cell centers. Solution at a boundary node that is not on the symmetry plane is computed using extrapolation from the interior of the computational domain. In the preceding USM3D version, ghost cells were used at the symmetry boundaries and intersecting symmetry boundaries were precluded. This restriction is removed from the current USM3D version used for the computations reported in this paper.

A reconstruction process based on solution gradients computed within cells accomplishes the USM3D second-order spatial discretization of inviscid fluxes. The cell gradients used for inviscid fluxes are computed by a Green-Gauss integration procedure. Inviscid fluxes are computed at each cell face using various upwind schemes, such as Roe's Flux Difference Splitting (FDS), van Leer's Flux Vector Splitting (FVS), Harten, Lax, van Leer, Einfeldt (HLLE), or Harten, Lax, van Leer – Contact (HLLC) schemes, among others. The standard [24] and negative variants [25] of the SA one-equation turbulence model are available. The convective term of the SA turbulence model equation can be approximated with either second- or first-order accuracy. Face gradients are needed to compute diffusive fluxes for the mean flow and turbulence model equations. A face gradient is evaluated from the Mitchell's stencil [26], [8]. A weighted average of face gradients is used to compute cell based velocity gradients for the SA source term.

The cell-centered solution values are required to satisfy the realizability constraints, such as positive values of density and pressure. For second-order spatial accuracy, a realizability check is implemented for the reconstructed solution to avoid a catastrophic failure (an underflow condition associated with a square root of a negative number) in computing mean flow inviscid fluxes. If a realizability violation is detected at a face during any nonlinear iteration, the face is temporarily designated as "first-order". The "first-order" designation for a face is removed after the second-order reconstruction values have been realizable for 20 consecutive nonlinear iterations.

The USM3D preconditioner uses a defect correction scheme

$$\frac{V}{\Delta\tau}\Delta\boldsymbol{Q} + \frac{\widehat{\partial \boldsymbol{R}}}{\partial \boldsymbol{Q}}\Delta\boldsymbol{Q} = -\boldsymbol{R}(\boldsymbol{Q}^n), \tag{2}$$

$$\boldsymbol{Q}^{n+1} = \boldsymbol{Q}^n + \Delta\boldsymbol{Q}. \tag{3}$$

Here, $\boldsymbol{Q}^n$ and $\boldsymbol{Q}^{n+1}$ are the solutions at iterations n and $n+1$, respectively; $\frac{\widehat{\partial \boldsymbol{R}}}{\partial \boldsymbol{Q}}$ is an approximation to Jacobian $\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{Q}}$; V is a control volume; and $\Delta\tau$ is a pseudo-time step, which is set through a Courant-Friedrichs-Lewy number, CFL, specification. The mean flow and turbulence model preconditioner equations are loosely coupled. The approximate Jacobian for the mean flow equations is formed using the linearization of the first-order FVS or FDS inviscid fluxes and a thin-layer approximation for the viscous fluxes. The approximate Jacobian for a turbulence-model equation includes the contributions from the advection, diffusion, and source terms. The advection term is linearized with a first-order approximation. A thin-layer approximation is used for the diffusion term. The entire contribution from the linearized source term is added to the diagonal. A positivity check for the diagonal values is conducted before adding the pseudo-time term. Negative diagonal values are substituted by their absolute values. An option to use single precision for the approximate Jacobian off-diagonal terms and for the solution updates is available to reduce the memory footprint.

In the preceding USM3D version, only point-implicit Gauss-Seidel (G-S) iterations were available to solve the linear preconditioner equation (Eq. 2). The G-S iterations were terminated if the root mean square (rms) norm of the residuals (Eq. (2)) reduced by one order of magnitude, any time a minimum of 10 G-S iterations were performed. The G-S iterations could terminate even before completing 10 G-S iterations if the rms norm of the residuals reduced by two orders of magnitude.

The HANIM is available for nonlinear iterations. The HANIM is an enhanced solver for Eq. (1) that seeks to improve (i) robustness of iterations by providing a mechanism to treat difficult (e.g., transient) solution states, (ii) efficiency by iterating at higher CFL, and (iii) automation by reducing the number of ad hoc parameters. Figure 1 provides the schematic representations of the PA and HANIM nonlinear iterations.

The HANIM extends the PA method by providing two additional hierarchies around the preconditioner. The hierarchies are a matrix-free linear solver for the exact linearization of RANS equations and a nonlinear control of the solution update. An economical version of the Generalized Conjugate Residual (GCR) method [27] is used as the matrix-free linear solver. The HANIM mainly uses GCR to control the stability of linear iterations, and therefore only a few (or just one) search directions are used in GCR. The nonlinear control is used as a mechanism to optimize the reduction of nonlinear residuals. The nonlinear solution update strategy automatically adapts the under-relaxation parameter and pseudo-time step. The three HANIM hierarchies are interrelated. The preconditioner generates a new search direction for the GCR matrix-free linear solver. GCR suggests updates for the current nonlinear solution. The updates are checked for the solution realizability and passed on to the nonlinear control where they may be scaled.

The HANIM specifies the maximum number of G-S iterations allowed in preconditioner and the maximum number of search directions allowed in GCR. Furthermore, the HANIM prescribes the residual reduction targets for the preconditioner, GCR, and nonlinear solution updates. The HANIM increases CFL if all the HANIM hierarchies have reported complete success. On the other hand, if either preconditioner, or GCR, or nonlinear solution update has not succeeded, the HANIM discards the suggested correction and aggressively reduces the CFL.

III. New Developments in USM3D

A. Preconditioner Enhancements

1. Line-implicit preconditioner

High frequency errors are typically main contributors to the residuals of discretized partial difference equations. Such errors are expected to be reduced in preconditioner iterations. Efficiency of high-frequency error reduction is characterized by a smoothing factor [28]. Point-implicit iterations update preconditioner solutions one cell at a time. On grids with high aspect ratio elements, point-implicit iterations can reduce effectively the errors with high-frequency variations in the direction of strong coupling (small mesh sizes), but reduce poorly the errors that strongly vary in the directions of weak coupling (large mesh sizes). The smoothing factor of point-implicit iterations on highly anisotropic grids is poor, approaching unity. Block iterations, in which strongly coupled solutions are updated simultaneously, are a well-known way to improve the smoothing factor. Line-implicit iterations are a specific type of block iterations. Line-implicit iterations simultaneously update preconditioner solution at all cells of a grid line that approximately aligns with the direction of strong coupling. On grids used for turbulent flow computations, extremely small mesh spacing is typical in the direction normal to the viscous surfaces. Line-implicit iterations mitigate the shortcomings of point-implicit iterations on such grids and reduce effectively the errors with high-frequency variations in all directions.

The line generator used for this study relies on the concept of advancing grid layers and takes advantage of the prismatic and/or hexahedral cells characteristic of boundary layers. The lines are constructed based only on local grid connectivity information and do not use geometric (node position, cell size, aspect ratio, node-to-boundary mapping, etc.) and/or discretization information. As such, lines do not follow precisely the directions of strong coupling in discrete equations. The motivation for these lines is more limited to mitigating the possible inefficiencies of point-implicit preconditioner due to grid-induced anisotropies within boundary layers. While the benefit of lines is mainly expected within boundary layers, the line generation algorithm generates lines throughout the entire domain.

A line can be viewed as a sequence of ordered face-connected cells. It can be composed of the same type of cells, e.g., hexahedral cells or prismatic cells. Different lines of the same grid can be composed of different type cells. For example, computations reported in Section IV.D use the lines composed of hexahedral cells and the lines composed of prismatic cells. A line also can be described as an ordered sequence of faces, starting from a bottom face and ending with a terminal face. Note that not all faces of cells in a line are line faces. For example, if a line segment is a part of a prismatic advanced-layer grid, then the triangular faces of the prisms in the segment are line faces, but the quadrilateral faces are not. An interior line face in a given line is paired with the previous and the next line face; bottom and terminal line faces are paired with one interior line face. Different line faces of the same line do not intersect. All line faces in a line are of the same type, triangles or quadrilaterals.

The line generation algorithm starts from a set of boundary faces designated as bottom-of-the-line faces and proceeds recursively from layer to layer. The bottom-of-the-line faces are typically viscous boundary faces, however, farfield boundary faces can be used as bottom-of-the-line faces as well. During the recursive step, each line face at the current layer is paired with a face on the next layer. The pair of faces satisfies two conditions: the faces belong to the same prismatic or hexahedral cell and do not intersect. The interior current-layer line face has two cells associated with the face; one has already been assigned to a line and the other has not. The next-layer line face is found by inspecting all faces of the cell associated with the current-layer line face that has not been assigned to a line. After pairing, the cell between two paired faces is assigned to the line.

The Jacobian, $J = \frac{\partial \widehat{\boldsymbol{R}}}{\partial \boldsymbol{Q}}$, consists of the diagonal and off-diagonal blocks. The diagonal $n_{eq} \times n_{eq}$ block contains all contributions associated with a cell where RANS equations are solved, and the off-diagonal $n_{eq} \times n_{eq}$ blocks contain contributions from all neighboring cells. Here, n_{eq} is the number of equations at a cell. The diagonal and off-diagonal blocks are stored separately. The diagonal block is always stored with the double precision, while off-diagonal blocks are often stored with the single precision to reduce the memory footprint. Storing the diagonal block with the double precision is important for point-implicit preconditioner because it limits the round-off error magnification in matrix inversion. The round-off errors effects can be contained if pivoting is used. However, since the matrix inversion in the preconditioner is the main user of the computing resources, pivoting may lead to a noticeable increase in time to solution. The line-implicit preconditioner simultaneously computes corrections in all cells of the same line. Thus for a given line of N cells, the following block tri-diagonal linear equation is to be solved

$$\begin{cases} \boldsymbol{J}_{1,1}\Delta\boldsymbol{Q}_1 + \boldsymbol{J}_{1,2}\Delta\boldsymbol{Q}_2 = \boldsymbol{G}_1, \\ \boldsymbol{J}_{n,n-1}\Delta\boldsymbol{Q}_{n-1} + \boldsymbol{J}_{n,n}\Delta\boldsymbol{Q}_n + \boldsymbol{J}_{n,n+1}\Delta\boldsymbol{Q}_{n+1} = \boldsymbol{G}_n, \quad 1 < n < N, \\ \boldsymbol{J}_{N,N-1}\Delta\boldsymbol{Q}_{N-1} + \boldsymbol{J}_{N,N}\Delta\boldsymbol{Q}_N = \boldsymbol{G}_N, \end{cases} \tag{4}$$

where $\boldsymbol{J}_{n,n}$ is the diagonal block of cell n, and $\boldsymbol{J}_{n,n-1}$, $\boldsymbol{J}_{n,n+1}$ are off-diagonal blocks of cell n corresponding to its neighboring cells $n-1$ and $n+1$. Correction $\Delta\boldsymbol{Q}_n$ is to be applied to the solution quantities at cell n; $\boldsymbol{G}_n$ is the right-hand side of the preconditioner equation corresponding to the cell n minus the sum of all off-diagonal blocks, which are not in the line, multiplied by corresponding $\Delta\boldsymbol{Q}$. A standard version of the Thomas algorithm is used for each line to solve Eq. (4)

```
Allocate matrix $\boldsymbol{C}_n$ by size $(N-1)n_{eq}^2$ and array $\boldsymbol{D}_n$ by size $N\ n_{eq}$
$\boldsymbol{C}_1 = \boldsymbol{J}_{1,1}^{-1}\boldsymbol{J}_{1,2}$
$\boldsymbol{D}_1 = \boldsymbol{J}_{1,1}^{-1}G_1$
do $n = 2, N-1$
        $\boldsymbol{C}_n = (\boldsymbol{J}_{n,n} - \boldsymbol{J}_{n,n-1}\boldsymbol{C}_{n-1})^{-1}\boldsymbol{J}_{n,n+1}$
        $\boldsymbol{D}_n = (\boldsymbol{J}_{n,n} - \boldsymbol{J}_{n,n-1}\boldsymbol{C}_{n-1})^{-1}(\boldsymbol{G}_n - \boldsymbol{J}_{n,n-1}\boldsymbol{D}_{n-1})$
enddo
$\boldsymbol{D}_N = (\boldsymbol{J}_{N,N} - \boldsymbol{J}_{N,N-1}\boldsymbol{C}_{N-1})^{-1}(\boldsymbol{G}_N - \boldsymbol{J}_{N,N-1}\boldsymbol{D}_{N-1})$
$\Delta\boldsymbol{Q}_N = \boldsymbol{D}_N$
do $n = N-1, 1, -1$
        $\Delta\boldsymbol{Q}_n = \boldsymbol{D}_n - \boldsymbol{C}_n\Delta\boldsymbol{Q}_{n+1}$
enddo
```

The Thomas algorithm is found to be stable in many practical computations; however, it is not guaranteed to be stable in general. It is proven to be stable in several special cases, such as when the matrix corresponding to the line equations is diagonally dominant (either by rows or columns) or symmetric positive definite. Note that for line-implicit preconditioner, storing only diagonal block coefficients with the double precision is not enough to prevent round-off error magnification. The inverted matrices include not only diagonal terms, $\boldsymbol{J}_{n,n}$; they are combinations of diagonal and some off-diagonal terms of the same line, $(\boldsymbol{J}_{n,n} - \boldsymbol{J}_{n,n-1}\boldsymbol{C}_{n-1})$. Using the double precision for some off-diagonal blocks in the line can help to reduce the effects of round-off errors, but significantly increases the memory footprint. If proven stability is required, a partial or complete pivoting should be applied. Precision of the inverted blocks may be a significant factor in determining sensitivity of the computed correction to round-off errors. USM3D currently does not have pivoting for the line solver. However, a simple pivoting for the point-implicit preconditioner is available in USM3D.

The efficiency and stability of preconditioner iterations may depend on the order in which cells are visited during preconditioner iterations. A coloring algorithm defines the cell visiting order. Each cell is assigned a color, which is a positive integer. Color 1 cells are visited first, color 2 cells are visited next, and so on. The goal of the coloring algorithm is to ensure that neighboring cells do not have the same color. Such coloring (order of updates) improves scalability and smoothing properties of preconditioner iterations. The latter is important for fast multigrid convergence. The coloring is also important for the stability of line-implicit iterations of convection-dominated equations. Without coloring, line-implicit iterations for such equations may become unstable when lines are not aligned with the grid induced anisotropy.

The USM3D coloring algorithm is implemented as follows. At the beginning, all cells are assigned color 0. The algorithm goes through cells one at a time. At each cell, the algorithm first attempts to assign color *n = 1*. If any of the cell neighbors has already been assigned color *n*, the attempts fails. Algorithms increases the color index by one, *n = n+1*, and repeats attempts until a color is successfully assigned to the cell. The algorithm ensures that at the end of the coloring process no two neighbors have the same color.

The USM3D coloring algorithm treats a line as a super cell that has many component cells. A super cell may be a neighbor either to another super cell or to an individual cell (not belonging to any line) that shares a face with any component cell. For the purpose of coloring, super cells and individual cells are referred as cells. In the computations reported in this paper, coloring is applied for line-implicit iterations only.

2. *Optimization of preconditioner*

Static residual

The modifications described in this section apply to line- as well as point-implicit preconditioner iterations. In the previous work [22], the convergence of preconditioner iterations was monitored in terms of a dynamic linear residual. The dynamic residual evaluated within each iteration is based on the most recent solution approximation; at this stage the solution has been updated at some stencil cells but not at all the cells. The dynamic residual is used in computing local correction and therefore it is readily available. Its monitoring is computationally efficient because it does not require additional residual evaluations. The main disadvantage of the dynamic residual is that it does not accurately represent the residual either at the beginning or at the end of an iteration.

In the current study, the static residual is monitored after completion of an iteration. The static residual is computed based on a solution approximation, in which the solution at all cells has been updated. To monitor convergence of the static residual, one needs to compute the norm of the static residual before and after an iteration. The disadvantage of the static residual is an additional computational cost. To compute static residuals one needs to conduct additional residual computations. To mitigate this cost, USM3D introduces a monitoring schedule: the static residuals are evaluated after each of the first five iterations, the tenth iteration, and then every tenth iteration.

Some of the criteria for the success or failure of preconditioner iterations have also been modified in comparison to those used in the previous study [22] (see Section II). The target residual reduction has been set to 0.5 (previously 0.1). If the ratio of the rms norms of the current and initial static residuals is less than the target residual reduction, the preconditioner iterations are terminated with success. If at the end of iterations, the rms norm of the static residual has increased by more than a factor of 2 or has exceeded the 95% of the largest value observed in the iterations, the preconditioner iterations are declared as failure; otherwise they are declared as a success. In another distinction from the previous study [22], the iteration control is also incorporated for the turbulence model iterations.

Pivoting for LU decomposition of the mean-flow preconditioner

The LU decomposition method is a memory-efficient way to implement the Gaussian elimination algorithm to compute matrix inverse, $\boldsymbol{A}^{-1}$, where $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$ and $\boldsymbol{L}$ and $\boldsymbol{U}$ are lower and upper triangular matrices, respectively. Matrix L has *1*s on the diagonal. As a variant of Gaussian elimination, matrix inversion by the LU decomposition method can become unstable. The instabilities can be remedied by partial or complete pivoting. The LU decomposition presents a convenient way to detect instability of the inversion before conducting computations: if any entry on the diagonal of $\boldsymbol{U}$ is too small (close to zero) or any off-diagonal value of $\boldsymbol{L}$ is too big (the inverse of this value is too close to zero), then the inversion is unstable. The big/small assessments are in comparison with computational precision, that is a value of 10^{-7} is probably too small for the single precision, but is acceptable for the double precision. This test is useful and accurate, but is computationally expensive. In practical computations, matrix inversion instabilities do not occur often, so this additional computational expense is not viewed as necessary. In USM3D, this test is implemented as a diagnostic tool, which is applied when convergence instabilities are detected/suspected.

However, in practical computations, there is a scenario, in which matrix inversion instabilities are more probable than in other cases. This scenario may occur for inversion of diagonal block of the mean-flow Jacobian. To describe this scenario, we make two observations.

The first observation relates to the computing of LU decomposition of a diagonal block of a Jacobian matrix. The $(1, 1)$ upper-left element of the Jacobian diagonal block is the same as the $(1, 1)$ element of the corresponding matrix $\boldsymbol{U}$ in the LU decomposition. That is if the $(1, 1)$ element of the matrix is too small then the matrix inversion is likely to be unstable.

The second observation relates to the way of computing Jacobian. The mean-flow Jacobian matrix is computed for five conservative variables $\boldsymbol{Q} = [\rho, \rho\boldsymbol{u}, \rho E], \boldsymbol{u} = [u, v, w]$, which correspond to density, momentum, and energy. The linearized equations (rows in the Jacobian block) are placed in the matrix in the following order: conservation of mass, momentum, and energy. In particular, the linearization of the continuous mass conservation equation, $\boldsymbol{\nabla} \cdot \rho\boldsymbol{u} = 0$, does not depend on ρ. Thus, all contributions to the $(1, 1)$ element of the diagonal block are due to artificial contributions of a particular discretization scheme (e.g., numerical dissipation). For consistent discretization schemes, these contributions will approach zero in the limit of grid refinement.

These two observations may explain the matrix inversion instability (division by zero). Note that the instability is unlikely on relatively coarse grids and for relatively low CFL. However, this type of instability was observed for a converged solution on a coarse tetrahedral grid around a wing-body-tail configuration used for the 4th AIAA Drag Prediction Workshop. The specific solution was computed using the HANIM, and van Leer flux linearization for mean flow Jacobian. The preconditioner iterations produced not-a-number (NaN) when operating on a converged solution with a very high (10^{17}) CFL.

A remedy was implemented that changed the order of the variables and the equations in the Jacobian matrix. With the new order of variables ($\boldsymbol{Q} = [\rho, \rho\boldsymbol{u}, \rho E]$) and equations (mass conservation equation is the last following the energy conservation equation), the $(1, 1)$ element of the diagonal block was not too small, there were no "too small" values on the diagonal of matrix $\boldsymbol{U}$, and the preconditioner iterations were able to preserve the zero-residual solution.

B. Enhancements Unrelated to Preconditioner

1. Nodal averaging stencil augmentation

In the current version of USM3D, solution variables at a node are computed as weighted averages of the corresponding solutions defined at the centers of the cells surrounding the node. A 3D pseudo-Laplacian method is used to compute the weights [29] to [32]. All the cells connected to that node form an initial stencil. This stencil is augmented if it is not sufficient to support a least-squares system for a linear fit, i.e., if all stencil points are placed nearly on the same linear subspace (2D plane). A threshold parameter, ε, defines the minimum recognizable distance. For example, if the distance between a plane and a point is less than ε, the point is considered as located on the plane. For the simulations reported in the paper, ε is the minimum distance from a cell center to a viscous surface.

A nodal averaging stencil is marked for augmentation if any of the following four conditions cannot be satisfied. Here, $\boldsymbol{r}_{nk}$ is the vector connecting point n with point k, $\|\boldsymbol{r}_{nk}\|$ is the magnitude of the vector, point $n = 0$ corresponds to the node, and points with $k > 0$ correspond to the stencil cell centers.

1) There is a stencil point 1 that satisfies
$$\|\boldsymbol{r}_{01}\| > \varepsilon. \tag{5}$$
2) There is a second stencil point 2 that satisfies
$$\|\boldsymbol{r}_{12}\| > \varepsilon. \tag{6}$$
3) There is a third stencil point 3 that is located farther than ε from the line defined by the vector $\boldsymbol{r}_{12}$, that is
$$\sqrt{\|\boldsymbol{r}_{13}\|^2 - \frac{(\boldsymbol{r}_{13} \cdot \boldsymbol{r}_{12})^2}{\|\boldsymbol{r}_{12}\|^2}} > \varepsilon. \tag{7}$$
4) There is a fourth stencil point 4 that is located farther than ε from the plane defined by vectors $\boldsymbol{r}_{12}$ and $\boldsymbol{r}_{13}$, that is
$$\left|\boldsymbol{r}_{14} \cdot \frac{\boldsymbol{r}_{12} \times \boldsymbol{r}_{13}}{\|\boldsymbol{r}_{12} \times \boldsymbol{r}_{13}\|}\right| > \varepsilon. \tag{8}$$

If all four conditions have been satisfied, the stencil requires no augmentation. The stencil that has been marked for augmentation is expanded by adding face-neighbors of each stencil cell to the original stencil. The expanded stencil is evaluated again. The augmentation process continues until a stencil satisfies all four conditions.

2. Grid sequencing

For the grids used in the current computations, a highly specialized grid sequencing methodology has been applied. The methodology relies on the assumptions that all grids in a family are nested and that specific rules for cell numbering are known a priori. On such grids, a mapping between a coarse-grid cell and the corresponding fine grid cells occupying the same physical space can be established. In the grid sequencing procedure, the initial solution on a fine grid is obtained by interpolation of the coarse-grid solution. In the current computations, a first-order (piece-wise constant) interpolation is used, i.e., solution value from a coarse-grid cell is assigned to the fine-grid mapped cells. While grid sequencing does not improve the asymptotic convergence rate, it noticeably reduces time to solution and improves robustness. Note that the current grid sequencing is limited in its scope and not suitable for general unstructured grids. A cell agglomeration approach is needed to enable general grid sequencing.

3. *Symmetry boundary condition*

The goal of implementing symmetry boundary (SB) conditions is to improve efficiency by allowing simulations of symmetric flows (e.g., axisymmetric flows) on reduced computational domains with fewer degrees of freedom. Similar to farfield, inflow, and outflow boundary conditions, the SB conditions are defined at exterior boundaries of computational domains, not at physical surfaces. However, SB conditions differ from other exterior boundary conditions, which use the local exterior solution. The effect of inaccuracies in approximating farfield, inflow, and outflow boundary conditions can be mitigated if the corresponding boundaries are moved far enough from the region of interest. SB conditions are set in the interior of the computational domain, and they do not use exterior solutions. SB conditions are global in the sense that if a SB patch is defined at a certain location, the entire computational domain, all boundary conditions, and the entire solution must be symmetric with respect to the infinite symmetry plane containing the SB patch. The SB condition cannot be moved away from the physical surfaces and other regions of interest. Inaccuracies in approximating SB conditions have stronger effects on solutions in important parts of the domain.

The current implementation relies on properties of SB patches. As any usual boundary patch, an SB patch is a collection of connected boundary faces. There is an outward directed area vector associated with each face. The boundary face values are computed at the boundary face centers. An SB patch has a single unit normal (the normalized directed area). In other words, any SB patch is a subset of a 2D infinite symmetry plane implying that all boundary faces belonging to the SB patch have identical normal directions. Neither interior nor boundary faces anywhere in the computational domain are allowed to cross the symmetry plane. All boundary faces within the symmetry plane that do not belong to a SB patch have to have boundary conditions compatible with the symmetry (e.g., no-slip or slip, but not inflow/outflow/farfield) boundary conditions.

For any given discretization scheme, there is only one discretely consistent approximation to the SB conditions. Discrete consistency relates residuals and solutions computed on the reduced domain with SB conditions to residuals and solutions on the extended domain reflected with respect to the symmetry plane. The zero-residual solution computed with the discretely consistent SB conditions on the reduced domain can be extended by reflection to produce the zero-residual solution on the extended domain. Specifically, the extended solution produces zero interior residuals at and near the symmetry plane. Thus, the discretization scheme used in the interior of the computational domain uniquely and fully defines the discretely consistent SB conditions.

Some of popular methods used in practical codes approximate SB conditions on a continuous level as a perfect-slip (zero flux) boundary or by replacing some of discrete flow equations defined at (or near) SB patches with explicit conditions setting the velocity component normal to the symmetry plane to zero and/or requiring zero gradients of scalar quantities in the direction normal to the symmetry plane. These methods require modifications in discretization stencils at (and near) the SB patches. While such approximations may produce reasonable solutions in many cases, they violate the discrete consistency requirement. Symmetric extension of zero-residual solutions computed with such SB conditions would generate nonzero residuals on the extended domain at (and near) the SB surface.

The SB conditions implemented in USM3D are discretely consistent with the interior USM3D discretization. In particular, gradient, nodal averaging, and flux reconstruction stencils associated with SB conditions are not biased, similar to the interior stencils. The discrete solution computed on the reduced domain with this implementation of SB conditions is identical to the solution on the extended domain.

The implementation of discretely consistent SB conditions reported previously [22] used ghost cells across the symmetry plane and did not allow intersection of SB patches. The general USM3D implementation reported in the current paper does not use ghost cells and allows intersection of up to three SB patches. The intersection angle between two SB patches can be any divisor of 360 degrees. This option allows efficient simulations of arbitrary bodies of revolution. In case of three intersecting SB patches, the third SB patch has to be orthogonal to other two SB patches.

A new procedure is implemented to compute nodal solution at SB nodes as well as fluxes and their linearization at SB face centers. The procedure is described below.

Computing nodal averaging

The nodal averaging procedure is based on a 3D pseudo-Laplacian method [32], which is equivalent to a least-squares method [33]. By default, the averaging stencil includes all cells that connect to the node. The stencil cell weights depend on the position vectors (relative coordinates) of the cell centers with respect to the node. If necessary, a stencil augmentation procedure described in Section III.B.1 is applied.

In the setup stage, all boundary nodes are classified as per their position on various SB patches. For the classification purpose, no distinction is made between intersecting collinear SB patches, and they are treated as a single monolithic SB patch. All boundary nodes are grouped in four different types:

type 0 nodes that are not on any SB patch
type 1 nodes that are on just one SB patch
type 2 nodes that are at an intersection of two SB patches
type 3 nodes that are at an intersection of three SB patches

The four conditions for node-averaging stencil augmentation described in Section III.B.1 are reduced to three conditions for SB nodes. If any of the three conditions has not been satisfied, the stencil is marked for augmentation. For an SB node of *type 1*, the "no augmentation" conditions are as follows:

1) There is a stencil point 1 that is located farther than ε from the SB plane,
$$|\boldsymbol{r}_{01} \cdot \boldsymbol{n}_{SB}| > \varepsilon, \tag{9}$$
where $\boldsymbol{n}_{SB}$ is the outward normal associated with the SB patch.
2) There is a second stencil point 2 whose projection on the SB plane is farther than ε from the projection of point 1 on the SB plane,
$$\|\boldsymbol{r}_{12} - (\boldsymbol{r}_{12} \cdot \boldsymbol{n}_{SB})\boldsymbol{n}_{SB}\| > \varepsilon. \tag{10}$$
3) There is a third stencil point 3 whose projection on the SB plane is farther than ε from the line connecting the projections of points 1 and 2,
$$\sqrt{\|\boldsymbol{v}_{13}\|^2 - \|\boldsymbol{p}\|^2} > \varepsilon, \quad \boldsymbol{e}_{12} = \frac{\boldsymbol{v}_{12}}{\|\boldsymbol{v}_{12}\|}, \quad \boldsymbol{p} = (\boldsymbol{v}_{13} \cdot \boldsymbol{e}_{12})\boldsymbol{e}_{12}, \tag{11}$$
$$\boldsymbol{v}_{12} = \boldsymbol{r}_{12} - (\boldsymbol{r}_{12} \cdot \boldsymbol{n}_{SB})\boldsymbol{n}_{SB}, \quad \boldsymbol{v}_{13} = \boldsymbol{r}_{13} - (\boldsymbol{r}_{13} \cdot \boldsymbol{n}_{SB})\boldsymbol{n}_{SB}.$$
where $\boldsymbol{v}_{12}$ and $\boldsymbol{v}_{13}$ are projections of vectors $\boldsymbol{r}_{12}$ and $\boldsymbol{r}_{13}$ on the SB plane, $\boldsymbol{e}_{12}$ is the unit vector in the $\boldsymbol{v}_{12}$ direction, and $\boldsymbol{p}$ is the projection of the vector $\boldsymbol{v}_{13}$ on the direction defined by the vector $\boldsymbol{e}_{12}$.

For an SB node of *type 2* located at the intersection of SB patches SB1 and SB2 with corresponding normal $\boldsymbol{n}_{SB1}$ and $\boldsymbol{n}_{SB2}$, the "no augmentation" conditions are as follows:

1) There is a stencil point 1 that is located farther than ε from the SB1 plane,
$$|\boldsymbol{r}_{01} \cdot \boldsymbol{n}_{SB1}| > \varepsilon. \tag{12}$$
2) There is a stencil point k (can be point 1 again) that is farther than ε from the SB2 plane,
$$|\boldsymbol{r}_{0k} \cdot \boldsymbol{n}_{SB2}| > \varepsilon. \tag{13}$$
3) The maximum distance between projections, p_i, of the stencil points on the intersection line between SB1 and SB2 planes is greater than ε,
$$\max_i p_i - \min_i p_i > \varepsilon, \quad p_i = (\boldsymbol{r}_{0i} \cdot \boldsymbol{e}), \quad \boldsymbol{e} = \frac{\boldsymbol{n}_{SB1} \times \boldsymbol{n}_{SB2}}{\|\boldsymbol{n}_{SB1} \times \boldsymbol{n}_{SB2}\|}. \tag{14}$$
where $\boldsymbol{e}$ is the unit vector along the intersection line.

For a boundary node of *type 3* that is located at the intersection of SB patches SB1, SB2, and SB3 with corresponding normal $\boldsymbol{n}_{SB1}$, $\boldsymbol{n}_{SB2}$, and $\boldsymbol{n}_{SB3}$, the "no augmentation" conditions are as follows:

1) There is a stencil point 1 that is located farther than ε from the SB1 plane,
$$|\boldsymbol{r}_{01} \cdot \boldsymbol{n}_{SB1}| > \varepsilon. \tag{15}$$
2) There is a stencil point k (can be point 1 again) that is farther than ε from the SB2 plane,
$$|\boldsymbol{r}_{0k} \cdot \boldsymbol{n}_{SB2}| > \varepsilon. \tag{16}$$
3) There is a stencil point n (can be point 1 or k again) that is farther than ε from the SB2 plane,
$$|\boldsymbol{r}_{0n} \cdot \boldsymbol{n}_{SB3}| > \varepsilon. \tag{17}$$

Computation of averaging weights for SB nodes involves the pseudo Laplacian method in lower dimensions. For an SB node of type 1, the position vectors used in computing averaging weights are in-plane vectors from the node to the projection of the cell centers on the SB patch plane. An in-plane position vector, $\boldsymbol{v}_i$, is computed as,
$$\boldsymbol{v}_{0i} = \boldsymbol{r}_{0i} - (\boldsymbol{r}_{0i} \cdot \boldsymbol{n}_{SB})\boldsymbol{n}_{SB}. \tag{18}$$
The averaging weights are computed according to a 2D pseudo-Laplacian procedure. For an SB node of *type 2*, the position vectors used in computing averaging weights are in-line vectors from the node to the projection of the cell centers on the intersection line between two SB patches. An in-line position vector, $\boldsymbol{w}_{0i}$, is computed as,
$$\boldsymbol{w}_{0i} = p_i\,\boldsymbol{e}, \tag{19}$$
where p_i and $\boldsymbol{e}$ are the in-line stencil point projections and the unit vector along the intersection line, respectively, defined in Eq. (14). The averaging weights are computed according to a 1D pseudo-Laplacian procedure. For SB nodes of *type 3*, all averaging weights are unity.

A vector quantity at an SB node is modified such that its components orthogonal to the SB patches are subtracted. For a boundary node of *type 1*, the modified vector $\boldsymbol{w}_{0,M}$ is computed as:

$$\boldsymbol{w}_{0,M} = \boldsymbol{w}_0 - (\boldsymbol{w}_0 \cdot \boldsymbol{n}_{SB})\boldsymbol{n}_{SB}, \tag{20}$$

where $\boldsymbol{w}_0$ is the unmodified vector averaged from the vectors defined at the stencil cells. For boundary nodes of *type 2*, the modified vector, $\boldsymbol{w}_{0,M}$, is computed in a similar two-step procedure as:

$$\boldsymbol{w}_{0,M} = \boldsymbol{w}_{0,m} - (\boldsymbol{w}_{0,m} \cdot \boldsymbol{n}_{SB2})\boldsymbol{n}_{SB2}, \qquad \boldsymbol{w}_{0,m} = \boldsymbol{w}_0 - (\boldsymbol{w}_0 \cdot \boldsymbol{n}_{SB1})\boldsymbol{n}_{SB1}. \tag{21}$$

For the boundary nodes of *type 3*, the averaged vector quantity is set to 0.

Computing symmetric quantities

Flux computations at control volume faces require solution quantities defined at cells on the two sides of a face. To describe flux contributions computed at a SB face, an analogy of a ghost cell symmetric to the real cell attached to the SB face is introduced. The values defined at actual cell are referenced as "left" values, while values at the ghost cell are referenced as "right" values. In the actual implementation, no ghost cells are generated; the symmetry effects are accounted through targeted modifications in the flux computation procedures at SB faces. There are two general rules for computing symmetric quantities:

1) Right scalar quantities, s_r, are the same as the left quantities, s_l,

$$s_r = s_l. \tag{22}$$

2) Right vector quantities, $\boldsymbol{v}_r$, are modified from the left vector quantities, $\boldsymbol{v}_l$, as

$$\boldsymbol{v}_r = \boldsymbol{v}_l - 2(\boldsymbol{v}_l \cdot \boldsymbol{n}_{SB})\boldsymbol{n}_{SB}. \tag{23}$$

For computing inviscid fluxes at an SB face, the left scalar density (ρ), energy (ρE), and SA turbulence model ($\hat{\nu}$) variables are reflected according to Eq. (22); the left vector momentum variables $[\rho u, \rho v, \rho w]$ are reflected according Eq. (23). For the mean flow diffusive fluxes, one needs the face values of laminar viscosity, eddy viscosity, and temperature as well as solution values at the cell centers across the face. In case of Mitchell diffusive fluxes, solution values at the SB nodes are needed too. In case of averaged least-squares diffusive fluxes, solution gradients are required at the cell centers across the face. The viscosity values at the SB face are set to be equal the corresponding values at the left cell center. Cell centered gradients of scalar quantities (e.g., ρ) are vectors (e.g., $\boldsymbol{\nabla}\rho = [\partial_x\rho, \partial_y\rho, \partial_z\rho]$) and they are reflected according to Eq. (23). The gradient of velocity at the left cell center is a tensor, $\boldsymbol{G}_l$,

$$\boldsymbol{G}_l = \boldsymbol{\nabla}^T[u, v, w] = \begin{bmatrix} \partial_x u & \partial_x v & \partial_x w \\ \partial_y u & \partial_y v & \partial_y w \\ \partial_z u & \partial_z v & \partial_z w \end{bmatrix}, \tag{24}$$

the superscript T denotes vector transposition. The corresponding reflected tensor, $\boldsymbol{G}_r$, is computed as

$$\boldsymbol{G}_r = [\boldsymbol{I} - \boldsymbol{n}_{SB}^T\boldsymbol{n}_{SB}]\boldsymbol{G}_l[\boldsymbol{I} - \boldsymbol{n}_{SB}^T\boldsymbol{n}_{SB}], \tag{25}$$

where $\boldsymbol{I}$ is the identity matrix. For Jacobian computation, flux linearization at the SB faces uses specific relationships between left and right states, as shown in the Eqs. (22) and (23).

4. Improved turbulence-model source discretization

Accuracy of velocity gradients used for computing turbulence model source terms is critically important for efficiency and robustness of RANS solutions. It is well known that accuracy of gradient reconstruction degrades on curved highly anisotropic grids typical of advancing layer grids used in turbulent flow simulations. The accuracy degradation mostly affects the gradient component in the direction orthogonal to the curved grid layers. Various methods have been proposed to improve accuracy of gradient reconstruction for cell-centered formulations on curved anisotropic grids. One of the most popular methods is a weighted least-squares method [34], [35]. In this method, the gradient of a scalar function is reconstructed at a given cell through a linear fit to a least squares minimization problem that uses the function representations at the neighboring cells with weights that are inversely proportional to the distance between the centers of the neighboring cells and the given cell. This method often produces accurate gradients on prismatic and hexahedral advancing layer grids, where cell centers in the advancing layer structures form a straight line normal to the curved grid layers. However, the method is sensitive to small perturbations in the alignment of cells at neighboring grid layers. It can generate large errors if cell centers of cells in the same line are not on a straight line. The method is not effective on tetrahedral advancing layer grids.

Another method to recover accurate gradients is based on approximate mapping (AM) methodology [36], [37]. This method uses the distance function mapping typically available for RANS simulations to account for grid curvature. The AM method computes accurate gradients on a grid composed of arbitrary elements and does not rely on an advancing layer grid structure. The main disadvantage of this method is its reliance on the distance function being a smooth mapping from interior cells to boundary locations. In computations involving complex geometries, such a mapping is not smooth, for example, when the distance function for neighboring interior cells is computed from different surfaces.

An alternative method to improve gradient accuracy on curved anisotropic grids is a line mapping (LM) method. This method needs the identification of grid lines. With this method, one can use any available gradient reconstruction method to compute the initial gradient, $\boldsymbol{g}_n$, at a line cell. The position vector of the cell center is $\boldsymbol{x}_n$. Accuracy of $\boldsymbol{g}_n$ may degrade in the direction approximately aligning with the line direction. The LM correction replaces the gradient component along the line direction with a divided difference between solutions at line cells above and below the cell centered at $\boldsymbol{x}_n$.

$$\boldsymbol{g}_{LM} = \boldsymbol{g}_n - \left((\boldsymbol{g}_n \cdot \boldsymbol{e}) - \frac{U_{n+1} - U_{n-1}}{|\boldsymbol{x}_{n+1} - \boldsymbol{x}_{n-1}|} \right), \quad \boldsymbol{e} = \frac{\boldsymbol{x}_{n+1} - \boldsymbol{x}_{n-1}}{|\boldsymbol{x}_{n+1} - \boldsymbol{x}_{n-1}|}. \tag{26}$$

Here, g_{LM} is the LM gradient of function U computed at $\boldsymbol{x}_n$; $\boldsymbol{e}$ is the unit vector representing the local line direction. The quantities $\boldsymbol{x}_{n-1}$ and U_{n-1} are the position vector of the cell center and the solution at the below cell. The quantities $\boldsymbol{x}_{n+1}$ and U_{n+1} are the corresponding quantities of the above cell.

IV. Results and Discussion

The performance of the newly implemented line-implicit HANIM is assessed in comparison with the point-implicit HANIM and with the PA method. The methods compute solutions on families of uniformly refined nested grids generated for four benchmark turbulent flow cases. The cases are a 2D bump-in-channel configuration, the 2D NACA 0012 airfoil, a 3D bump-in-channel configuration, and a 3D hemisphere cylinder configuration. A multi-color line-implicit preconditioner is used for the line-implicit HANIM. A G-S point-implicit preconditioner is used for the point-implicit HANIM and for the PA method. Only one search direction is used for the HANIM solutions reported below.

The SA *negative* turbulence model [25] is used for all solutions reported herein. An identical CFL is used for both mean flow and SA model solutions. The start up CFL is 1 for all solution methods. In a PA solution the CFL is specified a priori for any iteration. The two HANIM methods update CFL adaptively.

A detailed comparative study of the performance of the PA, point-implicit HANIM, and line-implicit HANIM for each of the four cases mentioned earlier is presented in this section. Unless specified otherwise, solutions are obtained using the same set of common input parameters. Some of these parameters are identified based on past precursor studies and experience [22] as well as judgment. A brief summary of these input parameters is provided below:

1) Mean flow convective flux: Roe's FDS, second order
2) SA turbulence model convection term: upwind, first-order
3) SA model formulation: *negative* variant
4) Cell gradients for mean flow convective flux: Green-Gauss and nodal averaging scheme
5) Face gradients for mean flow and SA model diffusive flux: Mitchell's method
6) Diffusion terms in mean flow: full Navier-Stokes, 2nd-order
7) Diffusion terms in SA model: full Navier-Stokes
8) Approximate Jacobian convection terms: Roe's FDS, 1st order
9) Approximate Jacobian diffusion terms: thin-layer approximation
10) Cell gradients for SA model source term: area-weighted face gradients; these gradients are subsequently modified using line-mapping for the line-implicit HANIM
11) Maximum number of G-S iterations for preconditioner: 500
12) Linear static residual reduction target for preconditioner: 0.5
13) Linear residual reduction target (for HANIM): 0.92
14) Nonlinear residual reduction target (for HANIM): 0.96
15) Preconditioner failure condition (for HANIM): Linear residuals increase by more than factor 2 or the linear residual at the end of iterations is more than 0.95 times the maximum residual recorded during iterations
16) CFL: Ramped from 1 to 150 over 150 iterations for PA; updated adaptively for HANIM with no limit. The HANIM CFL is increased by factor 2 if all hierarchies report success, decreased by factor 10 if any hierarchy reports failure.
17) Initial conditions: grid sequencing
18) Farfield value of the SA model variable: $\hat{\nu}_{farfield} = 3\nu_{\infty}$
19) Prandtl number for diffusion fluxes: 0.72 for meanflow, 0.90 for turbulence model

For each of the four cases discussed below, solution convergence is monitored by tracking the RANS residuals as well as certain field and integrated surface quantities. Iterative convergence is demonstrated in a separate figure for each grid. A figure includes plots for the run time variations of four quantities, namely, (a) combined mean flow and SA model residuals, (b) CFL, (c) maximum eddy viscosity, and (d) drag coefficient. A more quantified convergence evaluation is presented in tables (a) through (d). The tables summarize the Central Processing Unit (CPU) time taken by each solution method to converge the rms norm of the combined residuals to the target level of 10^{-13} and the aerodynamic coefficients to six significant digits. Relative speedup of the line-implicit HANIM over the point-implicit HANIM and the PA methods is also shown in these tables. Note that solutions computed with the point- and line-implicit preconditioners are slightly different. The difference in the solutions is due to different discretization of the turbulence model source terms. For solutions computed with line-implicit preconditioner, line structures are readily available, and hence line mapping is used to improve the accuracy of the velocity gradients contributing to the turbulence model source terms. The aerodynamic coefficients computed with the line- and point-implicit methods are summarized in tables (e).

The computational efficiency of the HANIM solutions is expected to be derived from high CFL during the solution phase after the initial transients but before the target level to which residuals have converged. The specific interval between the points when initial residuals are reduced by two orders and when the residuals are above the level 10^{-11} level is identified as CFL operational range.

A. 2D Bump-in-Channel

Five nested hexahedral grids available at the NASA Turbulence Modeling Resource (TMR) website [38] have been used for this case. The specific grids, denoted by the number of grid points in the spanwise, streamwise, and body-normal directions, are 2x89x41, 2x177x81, 2x353x161, 2x705x321, and 2x1409x641. Figure 2 presents a schematic view of the bump configuration, computational domain and the boundary conditions in the X-Z plane. Symmetry boundary condition is applied on the two spanwise planes.

The PA and HANIM solutions have been computed for a flow at freestream Mach number of 0.2, Reynolds number of $3\text{x}10^6$ per unit grid length, and angle of attack of 0°. The solutions have been computed on the NASA Langley Research Center (LaRC) mid-range compute cluster *k3* (Sandy Bridge) nodes consisting of Intel Xeon E5-2670 processors.

Figures 3-7 show the iterative convergence for the five grids, coarsest to finest, respectively. The line-implicit HANIM outperforms the point-implicit HANIM and the PA method on all grids in all convergence metrics. The line-implicit preconditioner benefits increase on finer grids. The PA solution exhibited a limit-cycle behavior on the 2x705x321 grid. A separate study (not shown here) revealed that the limit-cycle behavior could be redressed if preconditioner equations are solved with higher accuracy. This was achieved by changing the linear residual reduction target from 0.5 to 0.1. It can be seen from Fig. 7 that on the finest 2x1409x641 grid, the PA solution is yet to converge to the target residual level even after 3 million CPU seconds.

The HANIMs operate at higher CFL levels on all grids, as compared to the specified CFL of 150 for the PA method. On the three coarse grids, average CFL in the HANIMs is comparable within the operational range. The peak CFL values achieved by the line-implicit HANIM are 1 to 2 orders of magnitude higher than the CFL peaks achieved by the point-implicit HANIM. This observation may explain a better convergence of the line-implicit HANIM on coarse grids. On the two finest grids, average CFL of the line-implicit HANIM is about an order of magnitude higher than the corresponding CFL of the point-implicit HANIM.

Tables 1a-1d quantify iterative convergence characteristics for the 2D bump-in-channel configuration. On all grids, the line-implicit HANIM yields significantly faster convergence as compared to the PA method. The speedup across different grids and metrics ranges from 9.9 to 63.8. Note that, in spite of incomplete residual convergence, the PA force coefficients on the finest grid have converged up to six significant digits. The line-implicit HANIM speedup over the point-implicit HANIM ranges from 1.7 to 15.3. The speedup increases on finer grids. On the finest grid, the line-implicit HANIM shows at least a factor 10 speedup over the point-implicit HANIM in all metrics. The speedup factor over the PA method is greater than 24.9 on the finest grid.

Table 1e shows the lift and drag coefficients. The aerodynamic coefficients computed with and without line mapping converge to the same limit in grid refinement. The difference between coefficients computed with and without line mapping is very small. The maximum relative difference of 2.3% is observed for the viscous drag coefficient on the coarsest grid. In grid refinement, the total and viscous drag coefficients decrease, whereas the lift coefficient increases. The drag coefficients computed with line mapping are closer to the grid-converged values than the corresponding coefficients computed without line mapping. For the lift coefficient, the values computed without line mapping are closer to the grid converged value than the values computed with line mapping.

Table 1a. Residual convergence time for 2D bump-in-channel solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
2x89x41	475	64	38	12.5	1.7
2x177x81	3524	372	216	16.3	1.7
2x353x161	32648	3958	1463	22.3	2.7
2x705x321	limit-cycle	67211	8774	N/A	7.7
2x1409x641	3277440*	746485	65183	> 50.3	11.5

*residuals converged to $2x10^{-11}$ level

Table 1b. Drag coefficient convergence time for 2D bump-in-channel solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
2x89x41	243	39	19	12.8	2.1
2x177x81	1034	197	104	9.9	1.9
2x353x161	11699	1779	668	17.5	2.7
2x705x321	limit-cycle	30645	5649	N/A	5.4
2x1409x641	1257230	493124	45977	27.3	10.7

Table 1c. Lift coefficient convergence time for 2D bump-in-channel solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
2x89x41	201	34	20	10.1	1.7
2x177x81	1381	184	84	16.4	2.2
2x353x161	16235	1700	553	29.4	3.1
2x705x321	limit-cycle	38308	3817	N/A	10.0
2x1409x641	2228430	534538	34930	63.8	15.3

Table 1d. Viscous drag coefficient convergence time for 2D bump-in-channel solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
2x89x41	206	35	19	10.8	1.8
2x177x81	1110	215	99	11.2	2.2
2x353x161	7687	1557	674	11.4	2.3
2x705x321	limit-cycle	27916	6261	N/A	4.5
2x1409x641	754957	342938	30304	24.9	11.3

Table 1e. Aerodynamic coefficients for 2D bump-in-channel converged solutions.

Grid	Point-implicit			Line-implicit		
	total drag	lift	viscous drag	total drag	lift	viscous drag
2x89x41	0.00491507	0.0243718	0.00324692	0.00484552	0.0243641	0.00317841
2x177x81	0.00375154	0.0247993	0.00321704	0.00373286	0.0247970	0.00319987
2x353x161	0.00359722	0.0248893	0.00320079	0.00359253	0.0248887	0.00319653
2x705x321	0.00357730	0.0249184	0.00319472	0.00357613	0.0249182	0.00319365
2x1409x641	0.00357321	0.0249431	0.00319201	0.00357292	0.0249431	0.00319175

B. 2D NACA 0012 Airfoil

Nested hexahedral grids from the TMR website [38] “Cases and Grids for Turbulence Model Numerical Analysis” section have been used for this case. Grids of three families, namely, Family I, Family II, and Family III are available. The families differ in the trailing edge streamwise spacing. Family II grids provide the highest resolution around the trailing edge and produce the most accurate solutions [39]. The current study uses six of the seven nested structured grids of the Family II. The specific grids, denoted by the number of grid points in the spanwise, streamwise, and body-normal directions, are 2x113x33, 2x225x65, 2x449x129, 2x897x257, 2x1793x513, and 2x3585x1025. The current study does not use the finest grid (2x7169x2049) in the seven grid series. Figure 8 presents a schematic view of the NACA 0012 airfoil, computational domain and the boundary conditions in the X-Z plane. Symmetry boundary condition is applied on the two spanwise planes.

The PA and HANIM solutions have been computed for a flow at freestream Mach number of 0.15, Reynolds number of $6x10^6$ per unit chord length, and angle of attack of 10°. The solutions have been computed on the NASA LaRC mid-range compute cluster *k3* (Sandy Bridge) nodes consisting of Intel Xeon E5-2670 processors.

Figures 9-14 show the iterative convergence for the six grids, coarsest to finest, respectively. The line-implicit HANIM outperforms the PA method on all grids and the point-implicit HANIM on the three finer grids in all metrics. The improved efficiency of the line-implicit HANIM relative to the PA method and the point-implicit HANIM is more evident as the grid is refined. It can be seen from Fig. 14a that on the 2x3585x1025 grid, the PA method and the point-implicit HANIM are yet to converge to the target residual level after 2.5 million CPU seconds. For this grid, solutions computed by the PA and point-implicit HANIM are deemed as not converged in all metrics. The drag coefficient computed by the point-implicit HANIM is noticeably more oscillatory than the drag coefficient in the PA finest-grid solution (Fig. 14d). Note that the line-implicit HANIM solution is fully converged to the target residual level in 285,492 CPU seconds. It is evident that on the rest of the grids, both types of the HANIM solutions converged substantially faster than the PA solutions.

The HANIMs operate at higher levels of CFL on all grids, as compared to the specified CFL of 150 for the PA method. The average CFL in the point- and line-implicit HANIM is quite similar within the operational range on all grids. The magnitude and frequency of high CFL peaks appear as better predictors of residual convergence. On the two coarser grids, the peak CFL of the point-implicit HANIM is 1 to 4 orders of magnitude higher than the peak CFL of the line implicit HANIM. On these grids, the point-implicit HANIM is noticeably faster to converge in all metrics. On the 2x449x129 grid, the line-implicit HANIM has higher CFL peaks, while the average CFL appears to be somewhat higher for the point-implicit HANIM. This is the coarsest grid on which the line-implicit HANIM converges to the target residual level faster than the point-implicit HANIM. On the three finer grids, the CFL peaks in the line-implicit HANIM are about 2 orders of magnitude higher than in the point-implicit HANIM. On these grids, the line-implicit HANIM outperforms the point-implicit HANIM in all metrics.

Tables 2a-2d quantify iterative convergence characteristics for the 2D NACA0012 airfoil configuration. On all grids, the HANIMs yield faster convergence as compared to the PA method. The line-implicit HANIM speedup across different grids and metrics ranges from 1.6 to 37.4. The speedup over the point-implicit HANIM ranges from 0.4 to 1.4 on the three coarser grids and from 2.1 to 7.8 on the two finer grids. Recall that only the line-implicit HANIM meets the convergence targets on the finest grid.

Table 2e shows the lift and drag coefficients. The aerodynamic coefficients computed with and without line mapping converge to the same limit in grid refinement. The difference between coefficients computed with and without line mapping is small. The maximum relative difference of 7% is observed for the viscous drag coefficient on the coarsest grid. On the 2x1793x513 grid, the relative difference between the viscous drag coefficients is less than 0.03%. In grid refinement, the total drag coefficient decreases, whereas the lift coefficient and the viscous drag coefficient increase. The lift and total drag coefficients computed with line mapping are closer to the grid-converged values than the corresponding coefficients computed without line mapping. The viscous drag coefficients computed without line mapping are closer to the grid-converged value than those computed with line mapping.

Table 2a. Residual convergence time for NACA 0012 airfoil solutions.

Grid	**Time (seconds) to converge**			**HANIM line-implicit speedup over**	
	PA	**HANIM point-implicit**	**HANIM line-implicit**	**PA**	**HANIM point-implicit**
2x113x33	133	17	34	3.9	0.5
2x225x65	895	85	152	5.9	0.6
2x449x129	7223	1291	945	7.6	1.4
2x897x257	86983	24207	5778	15.1	4.2
2x1793x513	1346080	281795	36007	37.4	7.8
2x3585x1025	2644810*	2554120**	285492	>9.3	>8.9

*residuals converged to $3x10^{-8}$ level; ** residuals converged to $7x10^{-11}$ level

Table 2b. Drag coefficient convergence time for NACA 0012 airfoil solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
2x113x33	33	9	21	1.6	0.4
2x225x65	278	49	96	2.9	0.5
2x449x129	2836	616	596	4.8	1.0
2x897x257	31103	9323	4239	7.3	2.2
2x1793x513	513421	98593	21165	24.3	4.7
2x3585x1025	2644810*	2554120*	184412	>14.3	>13.9

*yet to converge

Table 2c. Lift coefficient convergence time for NACA 0012 airfoil solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
2x113x33	51	9	18	2.8	0.5
2x225x65	235	35	86	2.7	0.4
2x449x129	1798	385	500	3.6	0.8
2x897x257	18345	6103	2942	6.2	2.1
2x1793x513	380069	82194	15258	24.9	5.4
2x3585x1025	2644810*	2554120*	147039	>18.0	>17.4

*yet to converge

Table 2d. Viscous drag coefficient convergence time for NACA 0012 airfoil solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
2x113x33	69	12	18	3.8	0.7
2x225x65	276	49	86	3.2	0.6
2x449x129	2602	663	522	5.0	1.3
2x897x257	37569	10530	2759	13.6	3.8
2x1793x513	362085	100361	17787	20.4	5.6
2x3585x1025	2644810*	2554120*	137584	>19.2	>18.6

*yet to converge

Table 2e. Aerodynamic coefficients for NACA 0012 converged solutions.

Grid	Point-implicit			Line-implicit		
	total drag	lift	viscous drag	total drag	lift	viscous drag
2x113x33	0.0209708	1.01120	0.00587982	0.0195204	1.02241	0.00550868
2x225x65	0.0145285	1.07610	0.00602064	0.0141849	1.07808	0.00591435
2x449x129	0.0127287	1.08777	0.00614234	0.0126448	1.08822	0.00611424
2x897x257	0.0123448	1.08947	0.00617505	0.0123241	1.08958	0.00616798
2x1793x513	0.0122646	1.09001	0.00618732	0.0122596	1.09004	0.00618556
2x3585x1025	N/A	N/A	N/A	0.0122557	1.09054	0.00619432

C. 3D Bump-in-Channel

This case is a 3D version of the 2D bump-in-channel case presented earlier, with spanwise variation added. The *x*-location of any position on the bump varies in the spanwise direction according to $x = x_0 + 0.3(\sin(\pi y))^4, -0.5 \leq y \leq 0$. In the preceding formula, x is the streamwise and y is the spanwise coordinate direction, x_0 is a given *x*-location on the 2D bump at *y* = 0 location. A family of six nested uniformly refined 3D hexahedral grids generated from a FORTRAN program available under the TMR website [38] "Cases and Grids for Turbulence Model Numerical Analysis" section, "3D Modified Bump" subsection have been used for this case. The specific grids, denoted by the number of grid points in the spanwise, streamwise, and body-normal directions, are 3x45x21, 5x89x41, 9x177x81, 17x353x161, 33x705x321, and 65x1409x641. Figure 15 presents a schematic view of the bump configuration, computational domain and the boundary conditions. Symmetry boundary condition is applied on the two spanwise planes.

The PA and HANIM solutions have been computed for a flow at freestream Mach number of 0.2, Reynolds number of $3x10^6$ per unit grid length, and angle of attack of 0°. The solutions have been computed on the NASA Advanced Supercomputing (NAS) facility's Pleiades supercomputer Haswell nodes consisting of Intel Xeon E5-2680v3 processors.

Figures 16-21 show the iterative convergence for the six grids, coarsest to finest, respectively. The line-implicit HANIM outperforms the PA method in all metrics on all grids, where both methods were attempted. It either matches or outperforms the point-implicit HANIM in all metrics on all grids, where both methods were attempted. The improved efficiency of the line-implicit HANIM relative to the PA method and the point-implicit HANIM is more evident as the grids are refined. For example, in order to converge the residuals to the target level on the 33x705x321 grid, the point-implicit HANIM required almost 3 million CPU seconds, whereas the line-implicit HANIM needed less than 400,000 CPU seconds (Fig. 20a). On this grid, the PA method diverged in the initial transient phase. No further attempts to stabilize the PA method, such as specification of lower CFL or a stricter tolerance for preconditioner residual reduction (e.g., 0.1 instead of 0.5), were made. Due to these reasons, only the line-implicit HANIM was run on the finest grid.

The HANIMs operate at higher levels of CFL on all grids, as compared to the specified CFL of 150 for the PA method. On the three coarser grids, the average CFL in the two sets of HANIM solutions is comparable within the operational range. On the coarsest 3x45x21 grid, the peak CFL of the line-implicit HANIM is higher than the peak CFL of the point-implicit HANIM (Fig. 16b). However, the point-implicit HANIM shows a slightly faster or comparable convergence in all metrics. On all other grids, the line-implicit HANIM has higher peak CFL in the operational range and converges faster than the point-implicit HANIM in all metrics.

Tables 3a-3d quantify iterative convergence characteristics for the 3D bump-in-channel configuration. The line-implicit HANIM yields significantly faster convergence as compared to the PA method on all grids where PA solutions are available. The speedup across different grids and metrics ranges from 3.8 to 13.7. Note that the PA solution could not be obtained on the 33x705x321 grid and was not attempted on the finest grid. The speedup of the line-implicit HANIM over the point-implicit HANIM ranges from 0.6 to 30.3. The line-implicit HANIM provides increasingly larger speedup on finer grids. The point-implicit HANIM solution was not attempted on the finest grid due to large anticipated CPU time requirement.

Table 3e shows the lift and drag coefficients. The aerodynamic coefficients computed with and without line mapping converge to the same limit in grid refinement. The difference between coefficients computed with and without line mapping is small. The maximum relative difference of 9% is observed for the viscous drag coefficient on the coarsest grid. On the 33x705x321 grid, the relative difference between the viscous drag coefficients is less than 0.04%. In grid refinement, the total drag coefficient decreases, whereas the lift coefficient and the viscous drag coefficient increase. The total drag coefficient computed with line mapping is marginally closer to the grid-converged value than the corresponding coefficient computed without line mapping. For the lift and viscous drag coefficients, the values computed without line mapping are closer to the grid-converged values than the values computed with line mapping.

Table 3a. Residual convergence time for 3D bump-in-channel solutions.

Grid	**Time (seconds) to converge**			**HANIM line-implicit speedup over**	
	PA	**HANIM point-implicit**	**HANIM line-implicit**	**PA**	**HANIM point-implicit**
3x45x21	72	13	13	5.5	1.0
5x89x41	937	191	145	6.5	1.3
9x177x81	19336	4129	1839	10.5	2.2
17x353x161	313419	130853	22923	13.7	5.7
33x705x321	diverged	2618500	181679	N/A	14.4
65x1409x641	did not run	did not run	1925800	N/A	N/A

Table 3b. Drag coefficient convergence time for 3D bump-in-channel solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
3x45x21	23	5	6	3.8	0.8
5x89x41	366	100	87	4.2	1.1
9x177x81	8616	2208	934	9.2	2.4
17x353x161	140472	75642	12149	11.6	6.2
33x705x321	diverged	2498690	83067	N/A	30.1
65x1409x641	did not run	did not run	1036340	N/A	N/A

Table 3c. Lift coefficient convergence time for 3D bump-in-channel solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
3x45x21	31	5	8	3.9	0.6
5x89x41	479	102	81	5.9	1.3
9x177x81	10378	2301	936	11.1	2.5
17x353x161	160145	68310	12445	12.9	5.5
33x705x321	diverged	2741610	90609	N/A	30.3
65x1409x641	did not run	did not run	1594950	N/A	N/A

Table 3d. Viscous drag coefficient convergence time for 3D bump-in-channel solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
3x45x21	32	6	6	5.3	1.0
5x89x41	306	103	79	3.9	1.3
9x177x81	6714	2156	843	8.0	2.6
17x353x161	133272	72187	11970	11.1	6.0
33x705x321	diverged	1503640	80352	N/A	18.7
65x1409x641	did not run	did not run	1130640	N/A	N/A

Table 3e. Aerodynamic coefficients for 3D bump-in-channel converged solutions.

Grid	Point-implicit			Line-implicit		
	total drag	lift	viscous drag	total drag	lift	viscous drag
3x45x21	0.0126652	0.0222017	0.00318379	0.0124428	0.0221633	0.00292423
5x89x41	0.00615079	0.0244426	0.00315625	0.00608194	0.0244329	0.00308985
9x177x81	0.00411583	0.0249748	0.00318860	0.00409758	0.0249726	0.00317180
17x353x161	0.00366276	0.0250624	0.00320331	0.00365813	0.0250619	0.00319908
33x705x321	0.00359240	0.0250798	0.00320510	0.00359124	0.0250797	0.00320404
65x1409x641	did not run	did not run	did not run	0.00358259	0.0250874	0.00320451

D. 3D Hemisphere Cylinder

A family of three nested unstructured grids generated from a FORTRAN program available under the TMR website [38] "Cases and Grids for Turbulence Model Numerical Analysis" section, "3D Hemisphere Cylinder" subsection have been used for this case. The grids are designed such that they do not have a polar singularity. The FORTRAN program generates the grid in the entire 360° domain in circumferential direction. The unstructured grids are periodic with respect to 60° rotations and reflection-symmetric with respect to circumferential planes with the circumferential angles in the multiples of 30°. Each grid is composed of prismatic and hexahedral cells. The computational domain around hemisphere is gridded with prismatic cells, whereas, the part of the domain around the cylinder is gridded with hexahedral cells. For the present study, using a special purpose utility developed by Ed Parlette of ViGYAN, Inc., 60° circumferential sector grids were extracted from the entire 360° domain grids. The present grids use symmetry planes in the circumferential direction. The coarse grid, identified as "Grid.3", has

355,200 cells and 344,318 nodes. The medium grid, identified as "Grid.2", has 2,841,600 cells and 2,679,075 nodes. The fine grid, identified as "Grid.1", has 22,732,800 cells and 21,134,309 nodes. Figure 22 presents schematic views of the hemisphere cylinder configuration, its computational domain, and the boundary conditions. A close-up view of the Grid.2 surface grid near the interface of the hemisphere and cylinder is also provided in Fig. 22.

The PA and HANIM solutions have been computed for a flow at freestream Mach number of 0.6, Reynolds number of 3.5×10^5 per unit grid length, and angle of attack of 0°. The solutions have been computed on the NAS facility's Pleiades supercomputer Ivy Bridge nodes consisting of Intel Xeon E5-2680v2 processors.

In distinction from the other three cases reported in this paper, the computations for the hemisphere cylinder do not use the grid sequencing procedure. On each grid, the solution is initialized using the freestream conditions. Figures 23-25 show the iterative convergence for the coarse (Grid.3), medium (Grid.2), and fine (Grid.1) grids, respectively. It is evident that the point-implicit HANIM converges faster than the PA method on all grids. The line-implicit HANIM shows faster residual convergence relative to the PA method, although this benefit is not reflected in the figures for the other metrics. On all grids, the line-implicit HANIM converges slower than the point-implicit HANIM. There is a noticeable similarity in convergence patterns shown by the point- and line-implicit HANIMs for this case. Although not shown, the convergence of the point- and line-implicit HANIMs is very similar with respect to iterations. Moreover, the average number of iterations required to meet the preconditioner residual reduction target is also comparable. It is recognized that the CPU time required for each preconditioner iteration is at least twice as much for the line-implicit method as compared to the point-implicit method, which results in the inferior convergence of the line-implicit HANIM with respect to the CPU time.

The HANIMs operate at higher average CFL as compared to the specified CFL of 150 for the PA method. On all grids, CFL in the point-implicit HANIM initially exceeds CFL in the line-implicit HANIM by several orders. The line-implicit HANIM has higher CFL peak later in the convergence process.

Tables 4a-4d quantify iterative convergence characteristics for the 3D hemisphere cylinder configuration. The line-implicit HANIM yields somewhat modest convergence improvement as compared to the PA on all grids. The speedup across different grids and metrics ranges from 1.2 to 2.2. The point-implicit HANIM outperforms the line-implicit HANIM. The line-implicit HANIM speedup over the point-implicit HANIM ranges from 0.4 to 0.8. A separate study (not shown) that used the maximum of 10 search directions for the two types of HANIM solutions on the coarse grid, did not improve the convergence of the line-implicit HANIM relative to the point-implicit HANIM. Further investigation is needed to understand the unexpected convergence trends.

Table 4e shows lift and drag coefficients for the 3D hemisphere cylinder configuration. The aerodynamic coefficients computed with and without line mapping converge to the same limit in grid refinement. The difference between coefficients computed with and without line mapping is small. The maximum relative difference of 1% is observed for the viscous drag coefficient on Grid.3. On Grid.1, the relative difference between the viscous drag coefficients is less than 0.06%. In grid refinement, the total drag coefficient decreases whereas the lift coefficient increases. The viscous drag coefficient computed without line mapping monotonically increases in grid refinement. The viscous drag coefficient computed with line mapping is not monotonic in grid refinement. All aerodynamic coefficients computed with line mapping are marginally closer to the grid-converged values than the corresponding coefficients computed without line mapping.

Table 4a. Residual convergence time for 3D hemisphere cylinder solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
Grid.3	2911	888	1753	1.7	0.5
Grid.2	47016	18966	24470	1.9	0.8
Grid.1	883722	338919	400973	2.2	0.8

Table 4b. Drag coefficient convergence time for 3D hemisphere cylinder solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
Grid.3	1342	512	1109	1.2	0.5
Grid.2	24738	9158	18395	1.3	0.5
Grid.1	603555	265792	383554	1.6	0.7

Table 4c. Lift coefficient convergence time for 3D hemisphere cylinder solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
Grid.3	2299	512	1091	2.1	0.5
Grid.2	31831	11695	24280	1.3	0.5
Grid.1	736113	312131	391011	1.9	0.8

Table 4d. Viscous drag coefficient convergence time for 3D hemisphere cylinder solutions.

Grid	Time (seconds) to converge			HANIM line-implicit speedup over	
	PA	HANIM point-implicit	HANIM line-implicit	PA	HANIM point-implicit
Grid.3	1764	557	1281	1.4	0.4
Grid.2	25105	8561	17536	1.4	0.5
Grid.1	570746	255525	329698	1.7	0.8

Table 4e. Aerodynamic coefficients for 3D hemisphere cylinder converged solutions.

Grid	Point-implicit			Line-implicit		
	total drag	lift	viscous drag	total drag	lift	viscous drag
Grid.3	0.00203971	0.00919826	0.00172103	0.00202247	0.00919923	0.00170475
Grid.2	0.00191615	0.00939251	0.00171314	0.00191206	0.00939265	0.00170916
Grid.1	0.00189090	0.00942997	0.00171001	0.00188989	0.00942998	0.00170902

E. HANIM Convergence Deterioration in Grid Refinement

Figure 26 compares the point- and line-implicit HANIM normalized CPU time to target residual level for each case. The CPU time is normalized by degrees of freedom. The HANIM iterative convergence is not grid independent. A grid-independent method is expected to have constant time to solution per degree of freedom. The point-implicit HANIM normalized time approximately triples in grid refinement for 2D cases and approaches factor 2 for 3D cases. The line-implicit HANIM normalized time approximately doubles in grid refinement for 2D cases and for the 3D hemisphere cylinder case. For the 3D bump-in-channel case, the normalized time is almost constant.

Figure 27 compares the normalized CPU time to solution of all cases, separately for the point- and line-implicit HANIM. The line-implicit HANIM shows less variation in the normalized time among various cases as compared to the point-implicit HANIM. Also, it is evident that the performance of the line-implicit HANIM is less sensitive to increase in degrees of freedom than the performance of the point-implicit HANIM. An unexpected observation is that the normalized time for the 3D hemisphere cylinder case is lower than the normalized time for other cases, including 2D cases, at comparable degrees of freedom. A low normalized time indicates that the case is easy to converge. This is unexpected for the 3D hemisphere cylinder case because this is the only case that does not use grid sequencing. One possible explanation for the low normalized time is that, due to axial symmetry, the case is similar to 2D cases. Indeed, it appears that the 3D hemisphere cylinder normalized time variation is similar to the 2D NACA 0012 airfoil normalized time variation on coarser grids. Another interesting observation is that for the comparable degrees of freedom, the time to solution of the point-implicit HANIM is higher by an order of magnitude for the 3D bump-in-channel case as compared to other cases. An additional investigation is required to understand the reasons for the disparity in the normalized times observed for the 3D cases.

V. Concluding Remarks

The Hierarchical Adaptive Nonlinear Iteration Method (HANIM) was recently implemented in USM3D [22]. This HANIM used a point-implicit preconditioner. Additional enhancements have been made in the current study to further improve iterative convergence, robustness, and accuracy of the mixed-element USM3D solutions. Newly added features include a multi-color line-implicit preconditioner, a general discretely consistent symmetry boundary condition, and an improved discretization of the source term in the turbulence model equations, among others. Four benchmark turbulent flow cases have been used to assess the enhanced methodology. Numerical solutions for flows around a two-dimensional (2D) bump-in-channel, the 2D NACA 0012 airfoil, a three-dimensional (3D) bump-in-channel, and a 3D hemisphere cylinder configurations have been computed on families of consistently refined grids. The iterative convergence of the line-implicit HANIM is compared with the convergence of the point-implicit HANIM and the convergence of the preconditioner-alone (PA) method representative of the current Reynolds-

Averaged Navier Stokes (RANS) solvers. The PA method uses a point-implicit preconditioner. In this study, solutions for RANS equations with a negative Spalart-Allmaras (SA) turbulence model are required to satisfy the following convergence targets: (1) the rms norm of the combined meanflow and turbulence model residuals is less than 10^{-13}, and (2) the aerodynamic coefficients have converged to at least six significant digits. The CPU times to achieve these targets are compared. The following observations are made from the current study:

1) The line-implicit HANIM met the convergence targets in all cases. This solver outperformed the other two iterative methods for the 2D cases and the 3D bump-in-channel case on fine grids. The speedup in the CPU time to achieve the convergence targets on the three finer grids in the corresponding families is at least a factor of 2.1 in comparison to the point-implicit HANIM, and at least a factor of 6.2 in comparison to the PA method. On many grids, the speedup factor is higher than 10. On the finest grids corresponding to the 2D NACA 0012 airfoil and the 3D bump-in-channel cases, only the line-implicit HANIM met the convergence targets. Convergence of the other two methods was slow, and the corresponding solutions were either terminated before convergence or not even attempted.
2) On coarse grids, the convergence acceleration benefits of the line-implicit HANIM are less significant. While the line-implicit HANIM always outperforms the PA method for 2D cases, its time to solution is comparable (and sometimes inferior) to the corresponding time of the point-implicit HANIM.
3) For the 3D hemisphere cylinder, the point implicit HANIM is superior on all grids presently considered.
4) An assessment of time to solution normalized by degrees of freedom shows that the case-to-case variation in the performance of the line-implicit HANIM is significantly less than the corresponding variation in the performance of the point-implicit HANIM. The performance of the line-implicit HANIM is also less sensitive to increase in degrees of freedom. The normalized time to solution for the HANIMs is lower for the 3D hemisphere cylinder than for all other cases at comparable degrees of freedom. The normalized time of the point-implicit HANIM for the 3D bump-in-channel case is an order of magnitude higher than for all other cases at comparable degrees of freedom. An additional investigation is required to understand the reasons for the disparity in the normalized times for the 3D cases.
5) The average levels of Courant-Friedrichs-Lewy number, CFL, attained by point- and line-implicit HANIMs are similar in most computations. It appears that the HANIM solution with CFL that achieves high peaks converges faster than the HANIM solution with CFL that oscillates with smaller amplitude, even if average CFL is similar or even higher for the latter solution.
6) Implementation of a general discretely consistent symmetry boundary condition allowed for an economical formulation for flows around bodies of revolution. The flow around the 3D hemisphere cylinder at the zero degree angle of attack has been simulated on a 60-degree sector with two intersecting symmetry planes using only one sixth of the total number of degrees of freedom.
7) The effect of line mapping for cell-centered velocity gradients that contribute to the SA source term was smaller than expected. This observation indicates that the gradients computed with the baseline method (face area average of Mitchel's face gradients) were sufficiently accurate.

VI. Future Directions

Automation of the HANIM has been identified as an important goal in the previous study [22]. To improve automation, the number of user prescribed parameters has to be decreased. Currently the HANIM is controlled by many parameters. For example, the preconditioner has a residual reduction target and the maximum number of Gauss-Seidel iterations as parameters. A linear multigrid solver should completely eliminate these dependences, as a single multigrid cycle is expected to be sufficient to meet a reasonable residual reduction target in preconditioner computations on any grid.

Iterative performance achieved by line-implicit HANIM is not grid independent. A nonlinear multigrid solver is envisioned as the method that is capable to approach a fast and grid independent convergence.

Acknowledgments

The NASA Aeronautics Research Mission Directorate Transformative Aeronautics Concepts Program Transformative Tools and Technologies project and Airspace Operations and Safety Program Airspace Technology Demonstrations project funded the work reported herein. The authors would like to thank Dr. Hiroaki Nishikawa (National Institute of Aerospace) and Dr. Ed Parlette (ViGYAN, Inc.) for providing programs to generate and extract grids for one of the cases considered in the present study. All computations were performed on NASA LaRC k3 cluster and NASA Advanced Supercomputing facility's Pleiades supercomputer.

References

[1] Levy, D.W., Laflin, K.R., Tinoco, E.N., Vassberg, J.C., Mani, M. Rider, B., Rumsey, C.L., Wahls, R.A., Morrison, J.H., Brodersen, O.P. Crippa, S., Mavriplis, D.J., and Murayama, M., "Summary of Data from the Fifth Computational Fluid Dynamics Drag Prediction Workshop", *Journal of Aircraft*, 2014, Vol.51: 1194-1213, 10.2514/1.C032389

[2] Rumsey, C.L. and Slotnick, J.P., "Overview and Summary of the Second AIAA High-Lift Prediction Workshop", *Journal of Aircraft*, Vol. 52, Special Section on Second High Lift Prediction Workshop (2015), pp. 1006-1025. doi: 10.2514/1.C032864

[3] Cummings R.M., and Schütte, A., "Integrated Computational/Experimental Approach to Unmanned Combat Air Vehicle Stability and Control Estimation", *Journal of Aircraft*, Vol. 49, No. 6, 2012, pp. 1542-1557. doi: 10.2514/1.C031430

[4] Anderson, W. K., and Bonhaus, D. L., "An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids," *Computers and Fluids*, Vol. 23, No. 1, 1994, pp. 1–22. doi:10.1016/0045-7930(94)90023-X

[5] 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, No. 4, 1997, pp. 247–263. doi:10.1080/10618569708940807

[6] Gerhold, T., "Overview of the Hybrid RANS Code TAU," MEGAFLOW — Numerical Flow Simulation for Aircraft Design, edited by Kroll, N., and Fassbender, Vol. 89, Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Springer, Berlin, 2005, pp. 81–92.

[7] Eliasson, P., "EDGE, a Navier–Stokes Solver for Unstructured Grids," Finite Volumes for Complex Applications III, Hermes Penton Ltd., London, 2002, pp. 527–534.

[8] Frink, N. T.: Tetrahedral unstructured Navier-Stokes method for turbulent flows. *AIAA Journal*, Vol. 36, No. 11, November 1998, pp. 1975-1982. doi: 10.2514/2.324

[9] Morton, S. A., McDaniel, D. R., Sears, D. R., Tillman, B., and Tuckey, T. R., "Kestrel: A Fixed Wing Virtual Aircraft Product of the CREATE Program," AIAA Paper 2009–0338, January 2009.

[10] Strang, W. Z., Tomaro, R. F., and Grismer, M. J., "The Defining Methods of Cobalt60: A Parallel, Implicit, Unstructured Euler/Navier Stokes Flow Solver," AIAA Paper 99-0786, 37th AIAA Aerospace Sciences Meeting and Exhibit, Jan. 1999.

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

[12] Aftosmis, M. J., Berger, M. J., and Adomavicius, G., "A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries," AIAA 2000-0808, Jan. 2000.

[13] Newman, III, J.C. and Anderson, W.K., "Investigation of Unstructured Higher-Order Methods for Unsteady Flow and Moving Domains", AIAA 2015-2917, June 2015.

[14] Xu, A.-G., Zhang, G.-C., Gan, Y.-B., Chen, F., and Yu, X.-J., "Lattice Boltzmann Modeling and Simulation of Compressible Flows," *Frontiers of Physics*, Vol. 7, No. 5, 2012, pp. 582–600. doi:10.1007/s11467-012-0269-5

[15] Pandya, M.J., Frink, N.T., Abdol-Hamid, K.S., Samareh, J.A., Parlette, E.B., and Taft, J.R., "Enhancements to TetrUSS for NASA Constellation Program", *Journal of Spacecraft and Rockets*, Vol. 49, No. 4, 2012, pp. 617-631. doi: 10.2514/1.A32089

[16] Bauer, S. X., Krist, S. E., and Compton, W. B. "Generation of the Ares I-X Flight Test Vehicle Aerodynamic Data Book and Comparison To Flight," AIAA Paper 2011–0011.

[17] Abdol-Hamid, K. S., Ghaffari, F., and Parlette, E. B., "Ares I Vehicle Computed Turbulent Ascent Aerodynamic Data Development and Analysis," *Journal of Spacecraft and Rockets*, Vol. 49, No. 4, 2012, pp. 596-608.

[18] Warwick, G., "Lockheed Martin Refines Hybrid Wing-Body Airlifter Concept", *Aviation Week & Space Technology*, February 17, 2014, p.40.

[19] Wick, A.T., Hooker, J.R., and Zeune, C.H., "Integrated Aerodynamic Benefits of Distributed Propulsion", AIAA Paper 2015-1500, January 2015.

[20] Hooker, J. R., Wick, A., Zeune, C., Agelastos, A., "Over Wing Nacelle Installations for Improved Energy Efficiency", AIAA Paper 2103-2920.

[21] Pandya, M. J., Frink, N. T., Ejiang Ding, and Edward Parlette, "Toward Verification of USM3D Extensions for Mixed Element Grids," AIAA Paper 2013-2541.

[22] Pandya, M. J., Diskin, B., Thomas, J. L., and Frink, N. T., "Improved Convergence and Robustness of USM3D Solutions on Mixed Element Grids (Invited), AIAA Paper 2015-1747.

[23] Diskin, B., Thomas, J. L., Pandya, M. J., and Rumsey, C. L., "Reference Solutions for Benchmark Turbulent Flows in Three Dimensions," To be presented at the AIAA SciTech 2016.

[24] Spalart, P. R. and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," Recherche Aerospatiale, No. 1, 1994, pp. 5–21.

[25] Allmaras, S. R., Johnson, F. T., and Spalart, P. R., "Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model," Seventh International Conference on Computational Fluid Dynamics, Big Island, Hawaii, 2012.

[26] Mitchell, C. R., "Improved Reconstruction Schemes for the Navier-Stokes Equations on Unstructured Meshes," AIAA Paper 1994-0642.

[27] Van der Vorst H. A. and Vuik C. "GMRESR: A Family of Nested GMRES Methods," Numerical Linear Algebra with Applications, Vol. 1, pp. 369-386, 1994.

[28] Brandt, A., "Multigrid Techniques: 1984 Guide, with Applications to Fluid Dynamics," 1984. ISBN-3-88457-081-1.

[29] Courant, R. and Hilbert, D., Methods of Mathematical Physics, Interscience, New York, 1962.

[30] Holmes, D. G. and Connell, S. D., "Solution of the 2D Navier-Stokes equations on unstructured adaptive grids," AIAA Paper 1989-1932.

[31] Rausch, R. D., Batina, J. T., and Yang, H. T., "Spatial adaptation procedures on unstructured meshes for accurate unsteady aerodynamic flow computation," *AIAA Journal*, Vol. 30, No. 5, 1992, pp. 1243–1251.

[32] Frink, N. T., "Recent Progress Toward a. Three-Dimensional Unstructured. Navier-Stokes Flow Solver," AIAA Paper 1994-0061.

[33] Haselbacher, A. and Blazek, J., "Accurate and Efficient Discretization of Navier-Stokes Equations on Mixed Grids," *AIAA Journal*, Vol. 38, No. 11, 2000, pp. 2094–2102.

[34] Mavriplis, D. J., "Revisiting the Least-Square Procedure for Gradient Reconstruction on Unstructured Meshes," AIAA Paper 2003-3986.

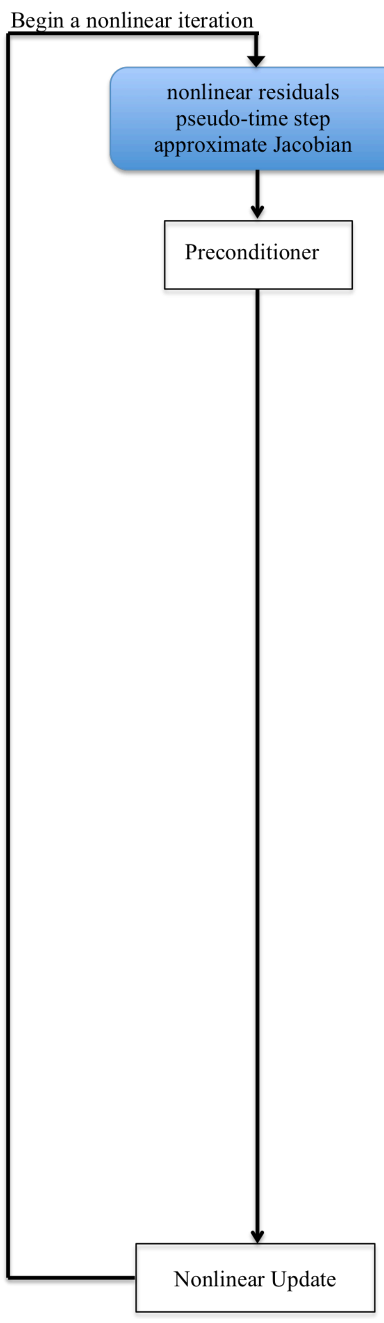
[35] Petrovskaya, N. V., "The Choice of Weight Coefficients for Least-Square Gradient Approximation," *Journal of Mathematical Modeling*, Vol. 16, No. 5, 2004, pp. 83–93.

[36] Diskin, B. and Thomas, J. L., "Comparison of Node-Centered and Cell-Centered Unstructured Finite-Volume Discretizations: Inviscid Fluxes," *AIAA Journal*, Vol. 49, No. 4, 2011, pp. 836–854.

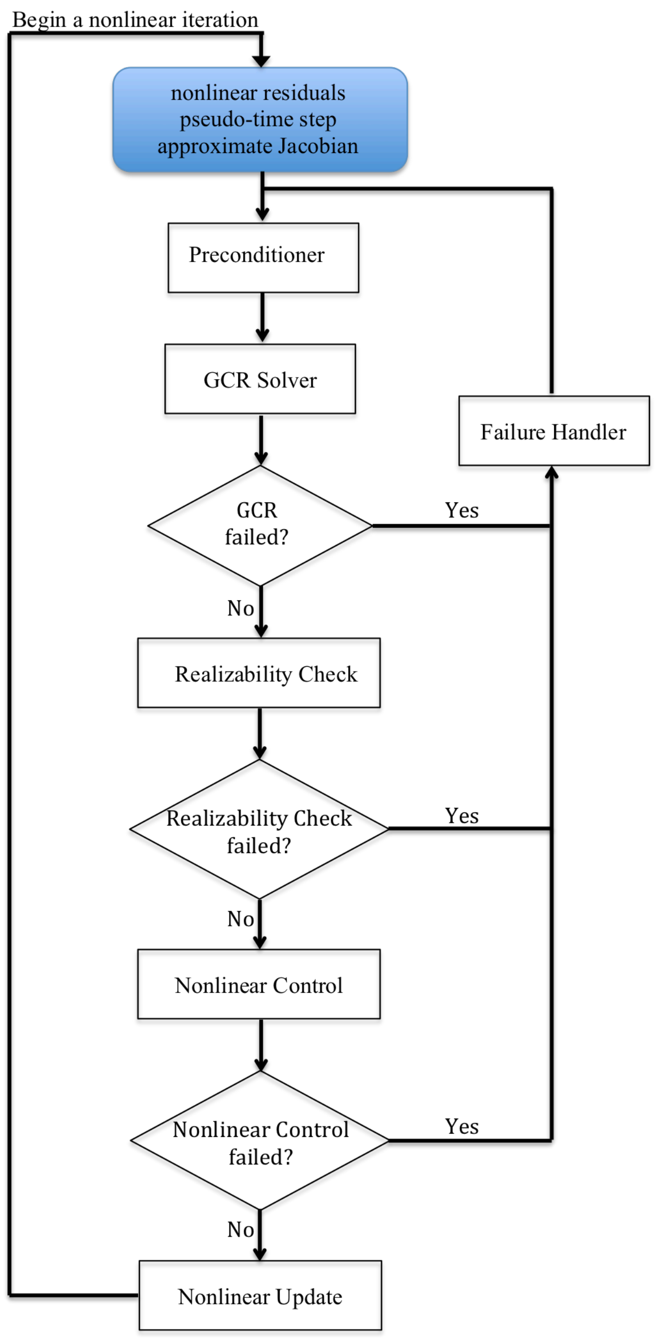
[37] Diskin, B. and Thomas, J. L., "Comparison of Node-Centered and Cell-Centered Unstructured Finite-Volume Discretizations: Inviscid Fluxes. (Composition erratum)," *AIAA Journal*, Vol. 51, No. 1, 2013, pp. 277–277.

[38] Turbulence Modeling Resource website, URL http://turbmodels.larc.nasa.gov [cited 3 November 2015].

[39] Diskin, B., Thomas, J. L., Rumsey, C. L., and Schwoeppe, A., "Grid Convergence for Turbulent Flows (Invited)," AIAA Paper 2015-1746.



(a) PA iterations



(b) HANIM iterations

Figure 1. Flowchart of two nonlinear iteration schemes.

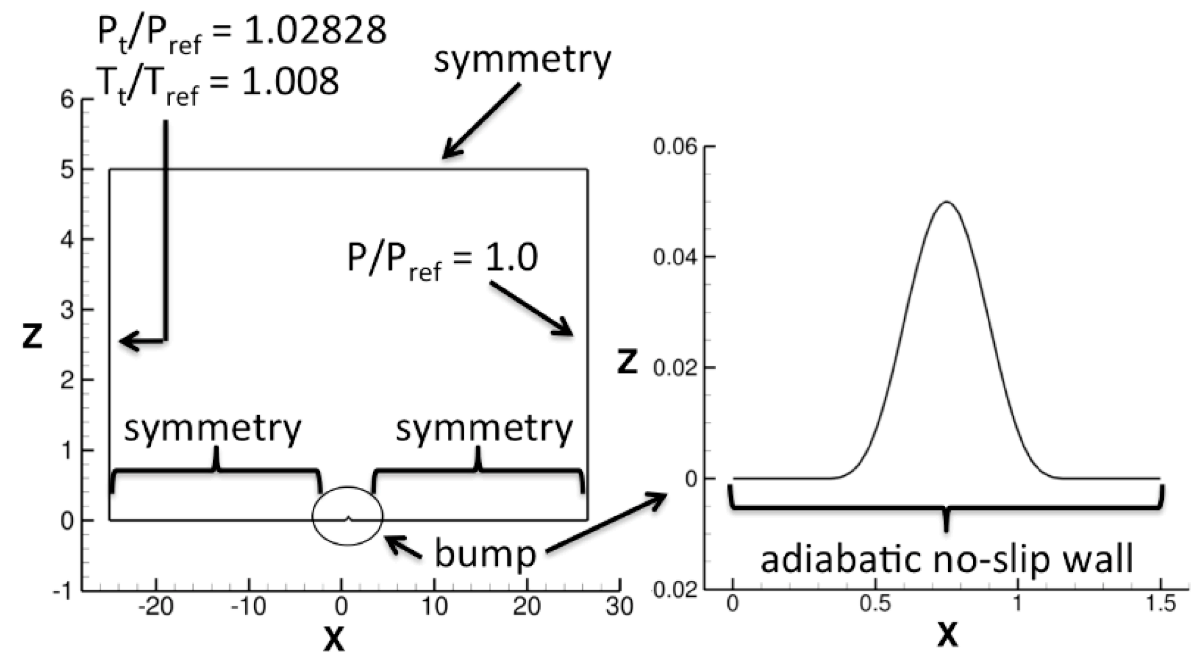
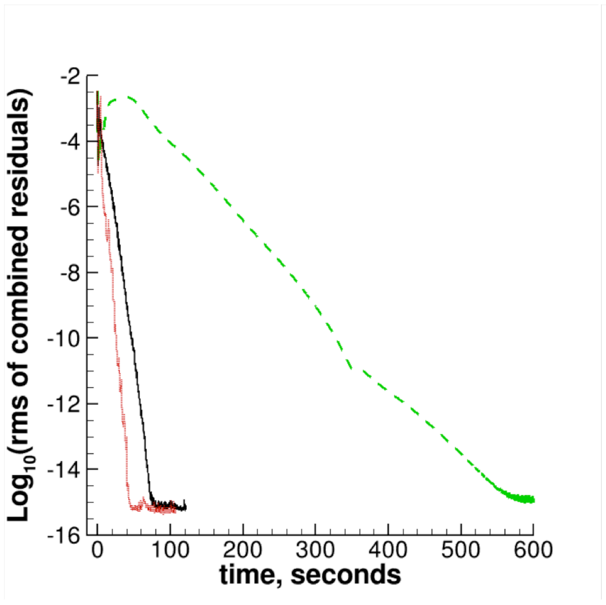
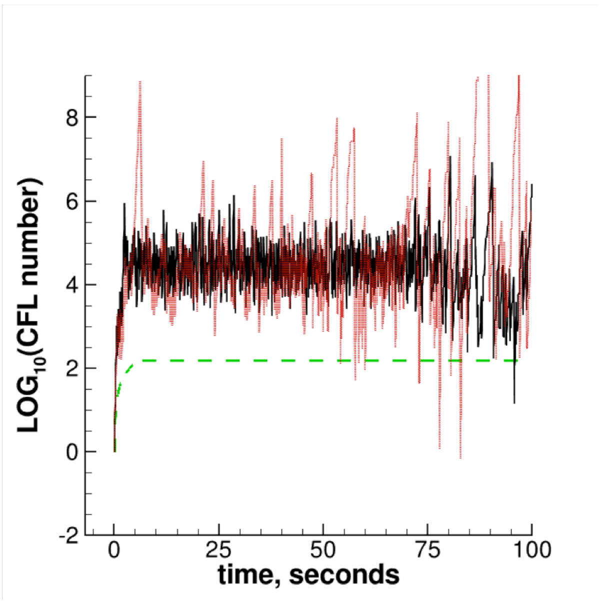


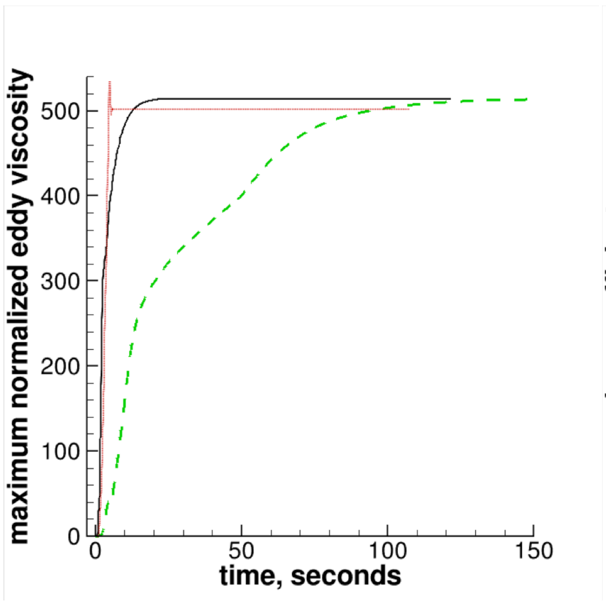
Figure 2. Schematic representation of the 2D bump-in-channel configuration and its computational domain with boundary conditions (BC).



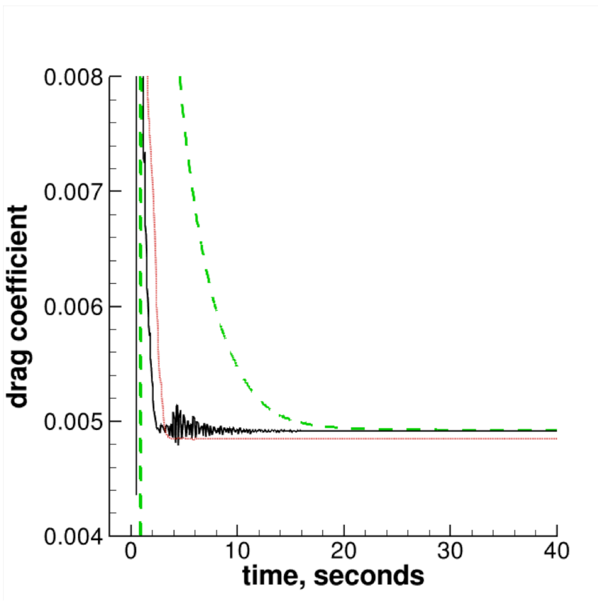
(a) rms of combined mean flow and SA model residuals



(b) CFL

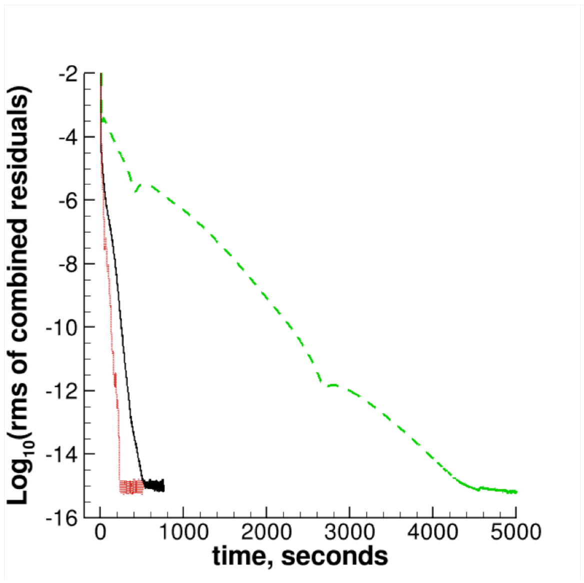


(c) maximum normalized eddy viscosity

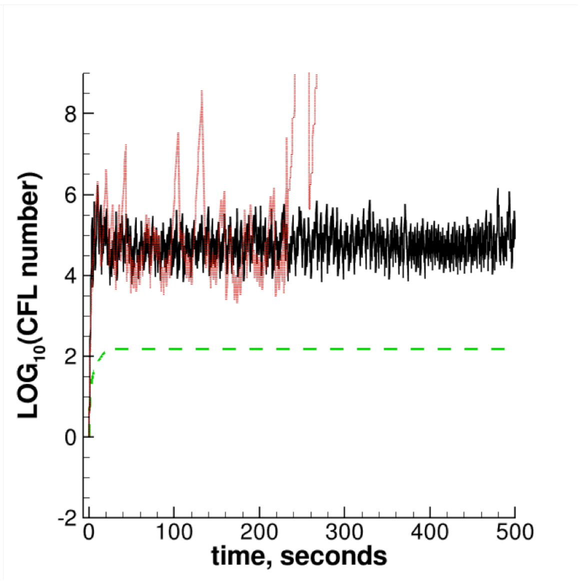


(d) total drag coefficient

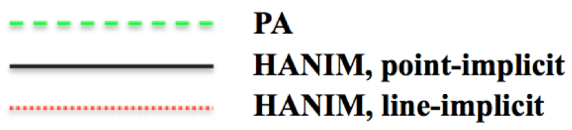
Figure 3. 2D Bump-in-channel solution convergence history using 2x89x41 hexahedral grid. M_∞ = 0.2, $Re_L = 3x10^6$.

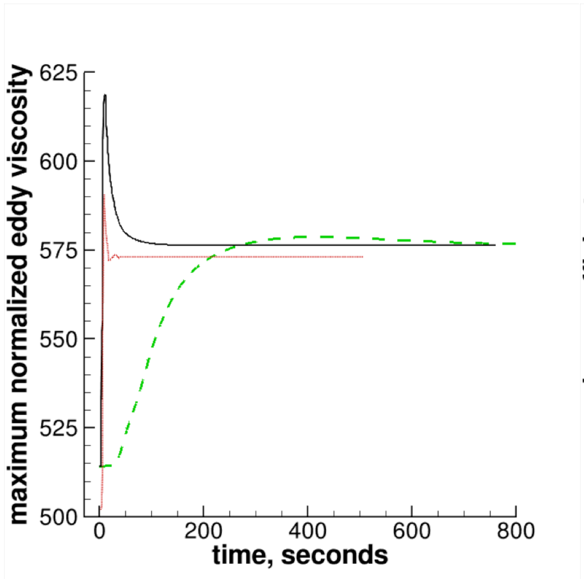


(a) rms of combined mean flow and SA model residuals

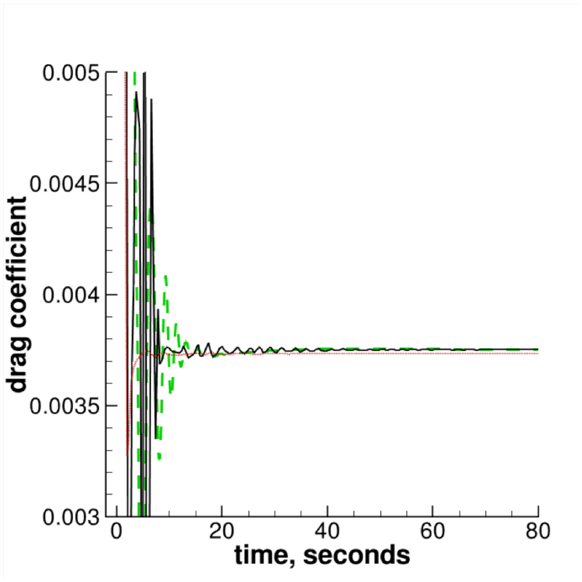


(b) CFL



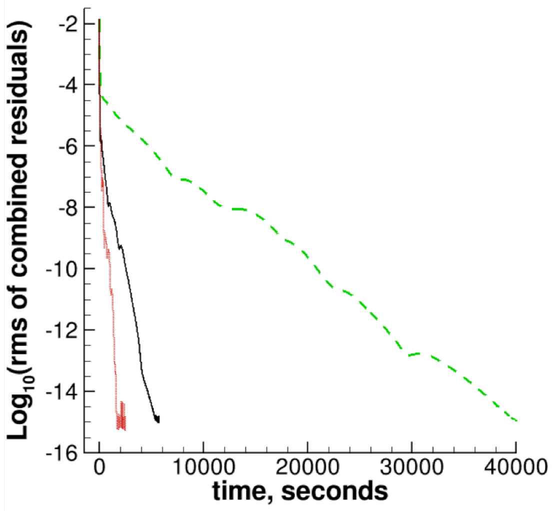


(c) maximum normalized eddy viscosity

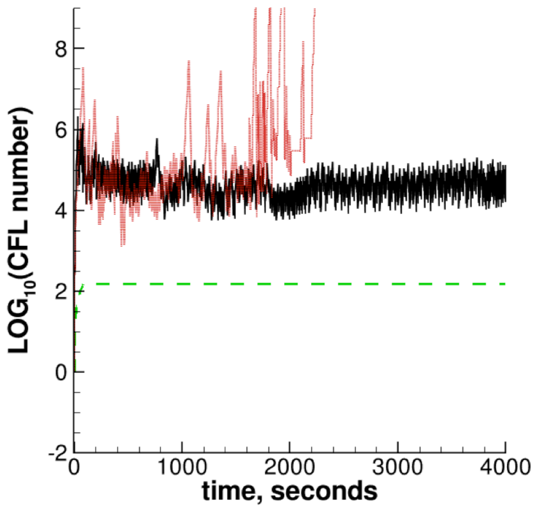


(d) total drag coefficient

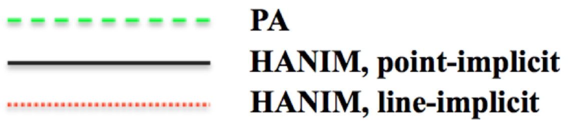
Figure 4. 2D Bump-in-channel solution convergence history using 2x177x81 hexahedral grid. $M_\infty = 0.2$, $Re_L = 3x10^6$.

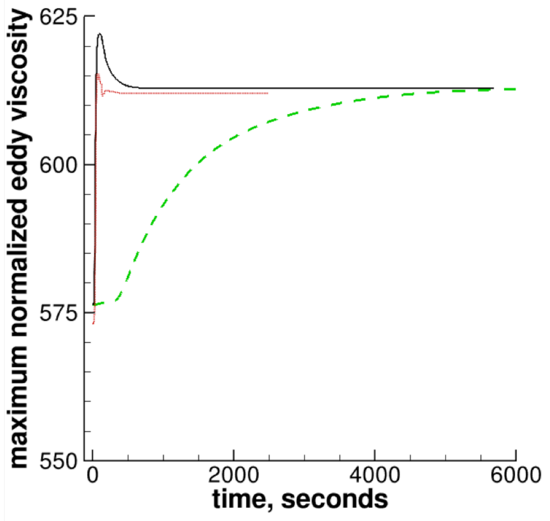


(a) rms of combined mean flow and SA model residuals

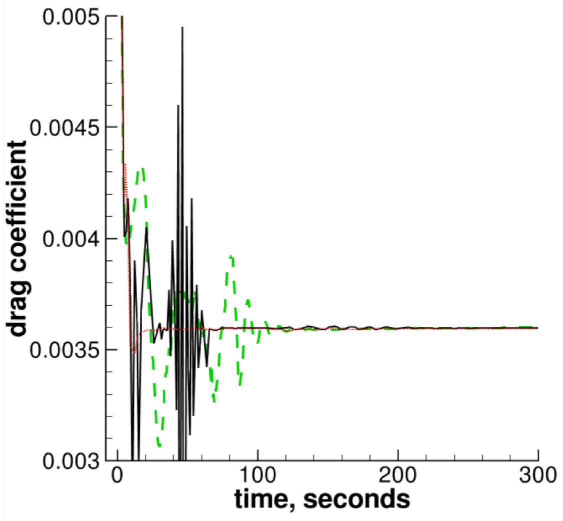


(b) CFL



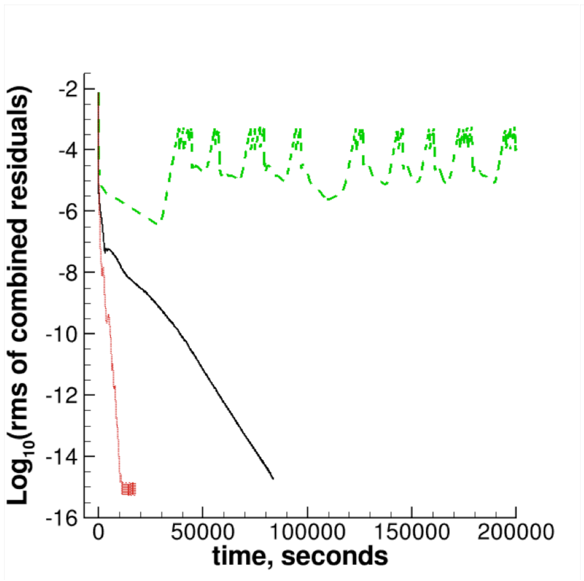


(c) maximum normalized eddy viscosity

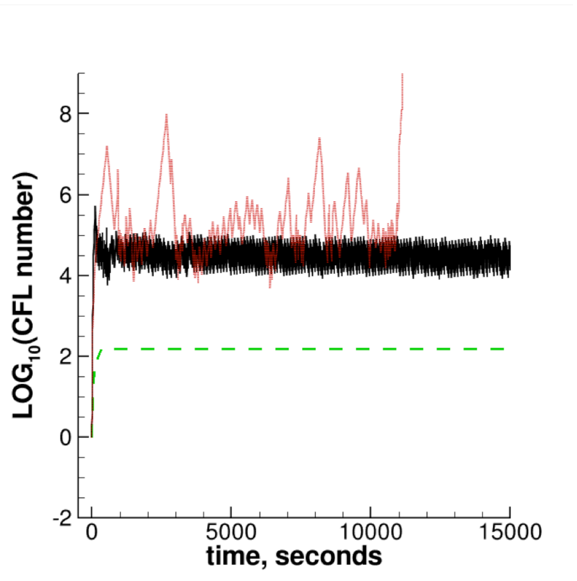


(d) total drag coefficient

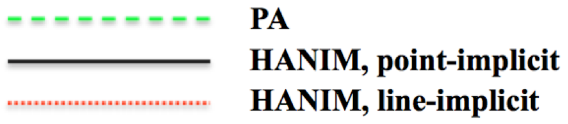
Figure 5. 2D Bump-in-channel solution convergence history using 2x353x161 hexahedral grid. $M_\infty = 0.2$, $Re_L = 3x10^6$.

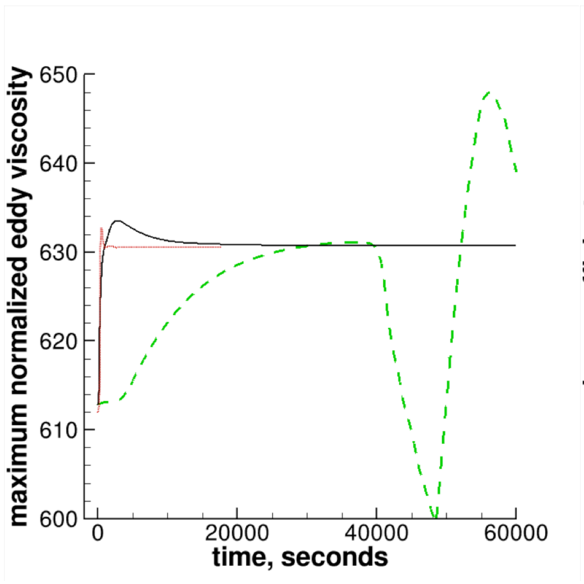


(a) rms of combined mean flow and SA model residuals

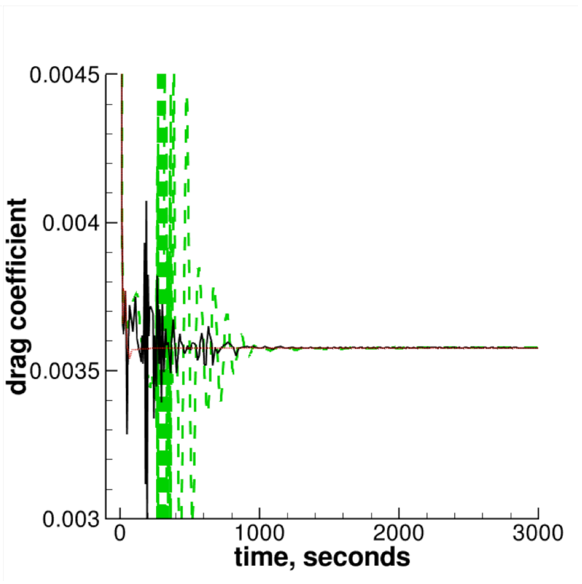


(b) CFL



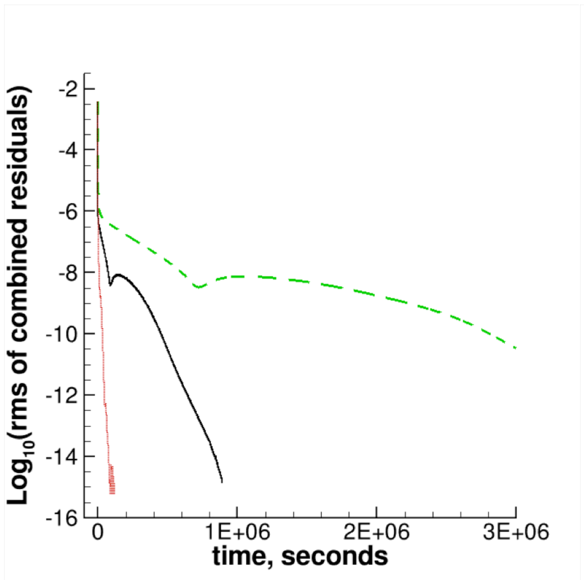


(c) maximum normalized eddy viscosity

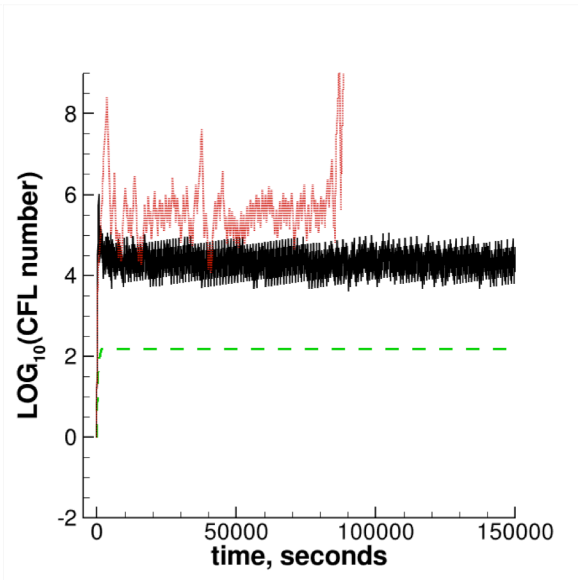


(d) total drag coefficient

Figure 6. 2D Bump-in-channel solution convergence history using 2x705x321 hexahedral grid. $M_\infty = 0.2$, $Re_L = 3x10^6$.



(a) rms of combined mean flow and SA model residuals

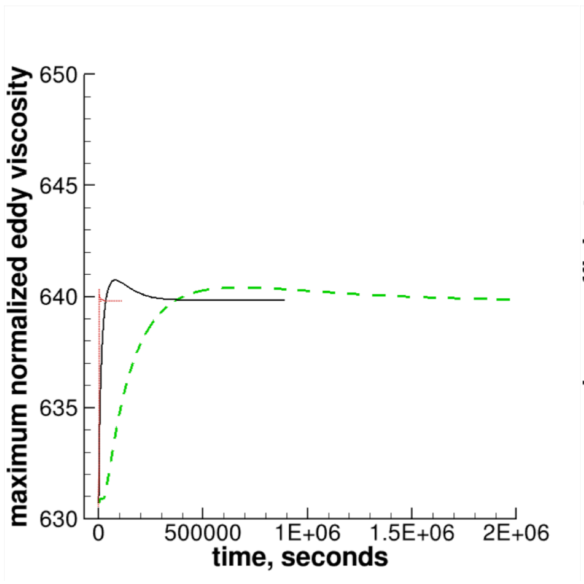


(b) CFL

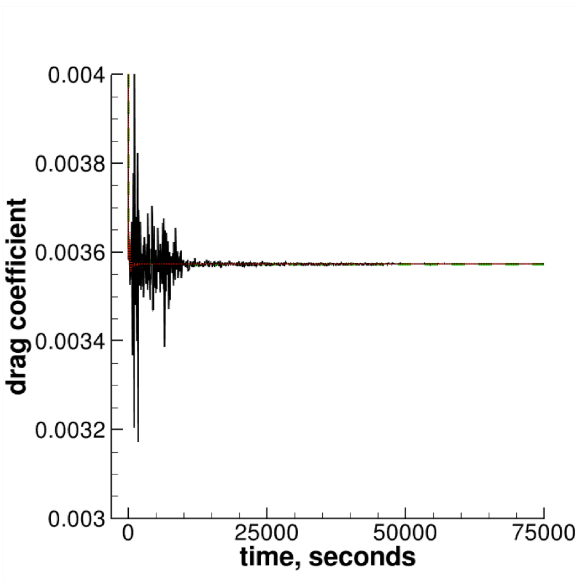
PA

HANIM, point-implicit

HANIM, line-implicit



(c) maximum normalized eddy viscosity



(d) total drag coefficient

Figure 7. 2D Bump-in-channel solution convergence history using 2x1409x641 hexahedral grid. M_∞ = 0.2, $Re_L = 3x10^6$.

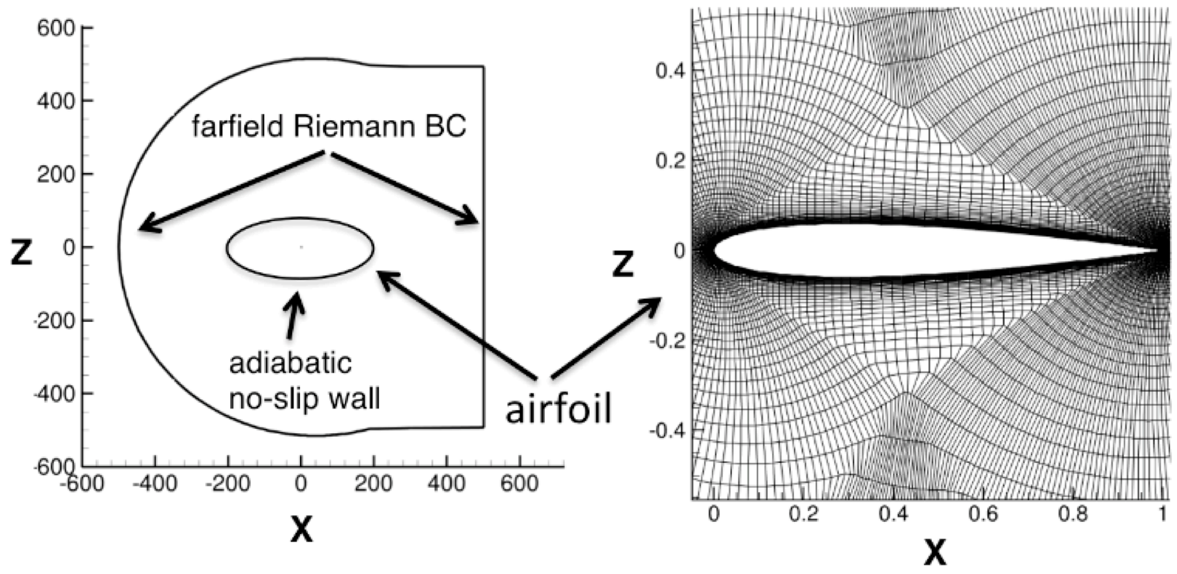
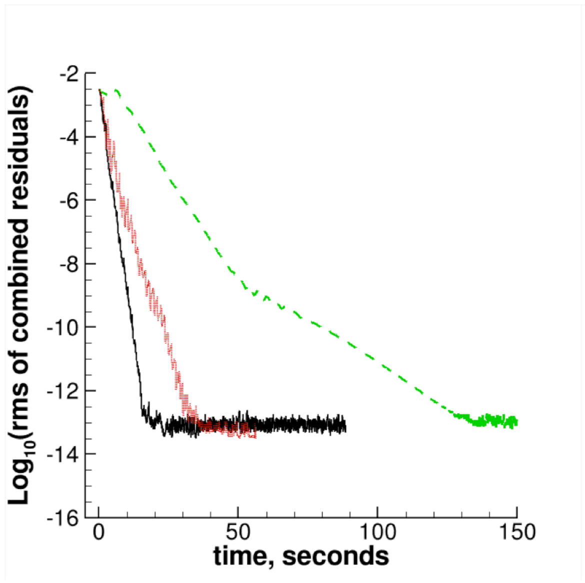
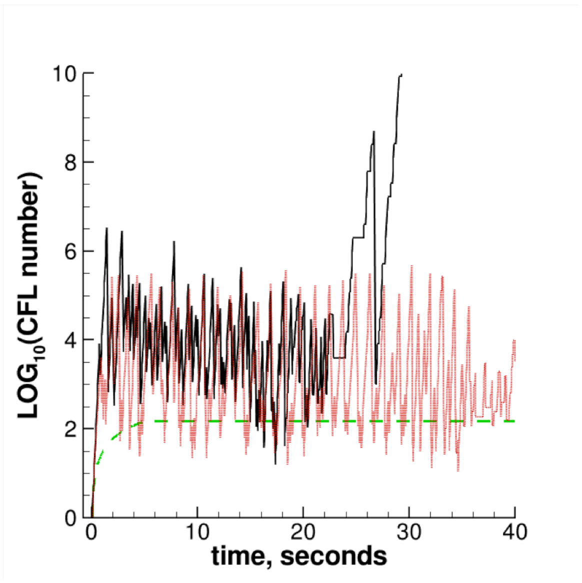


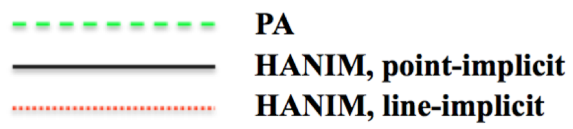
Figure 8. Schematic representation of the 2D NACA 0012 airfoil configuration and its computational domain with boundary conditions (BC).

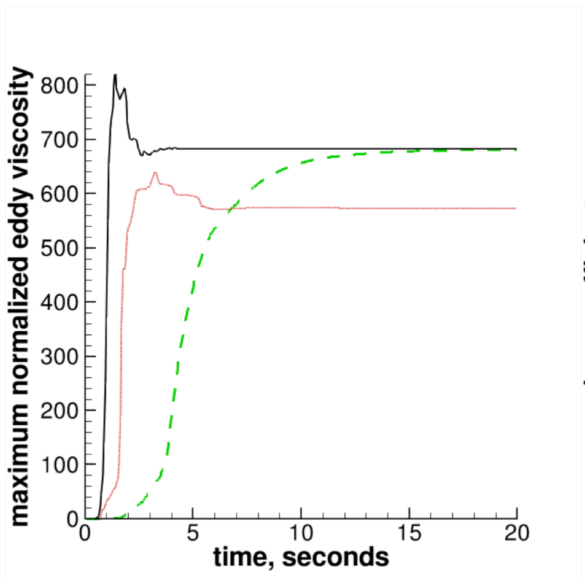


(a) rms of combined mean flow and SA model residuals

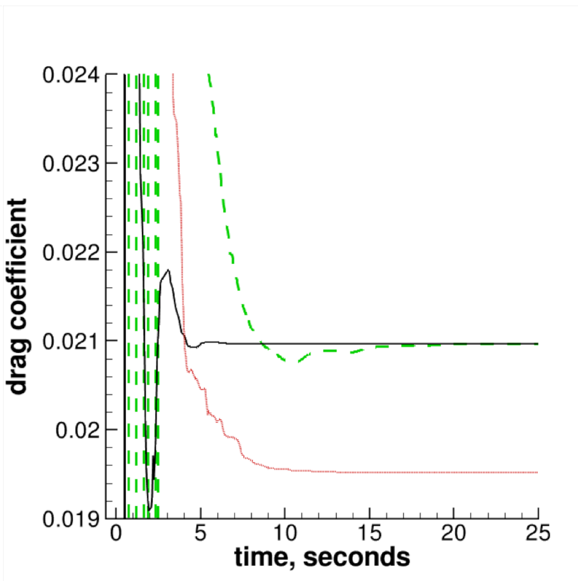


(b) CFL



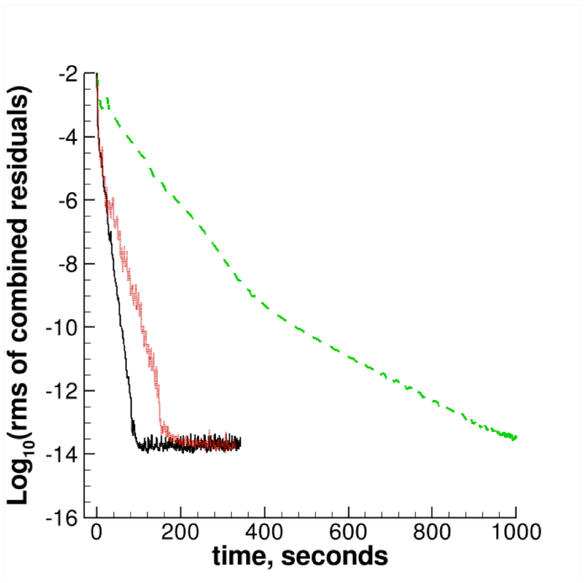


(c) maximum normalized eddy viscosity

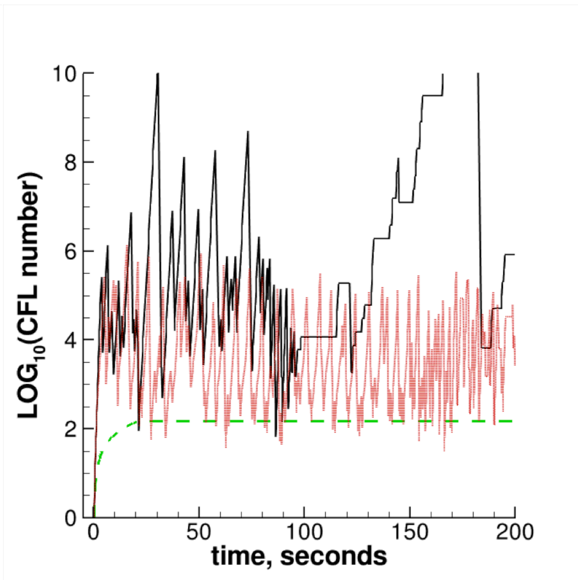


(d) total drag coefficient

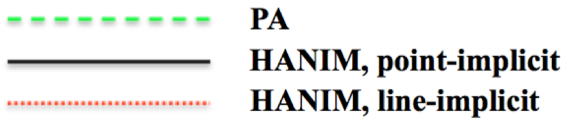
Figure 9. NACA 0012 solution convergence history using 2x113x33 hexahedral grid. M_∞ = 0.15, α = 10°, $Re_c = 6x10^6$.

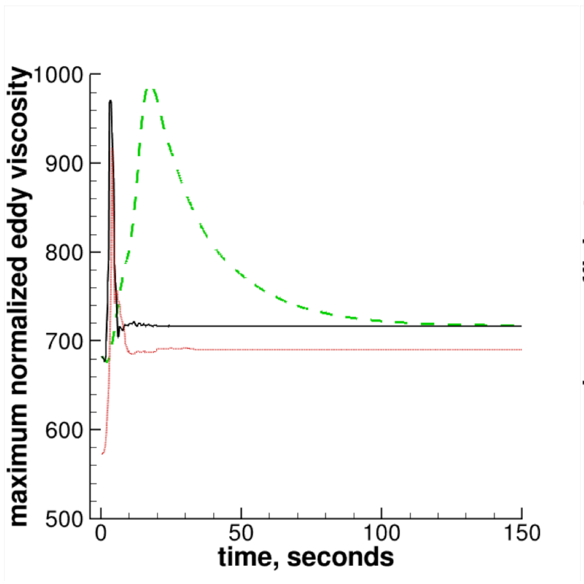


(a) rms of combined mean flow and SA model residuals

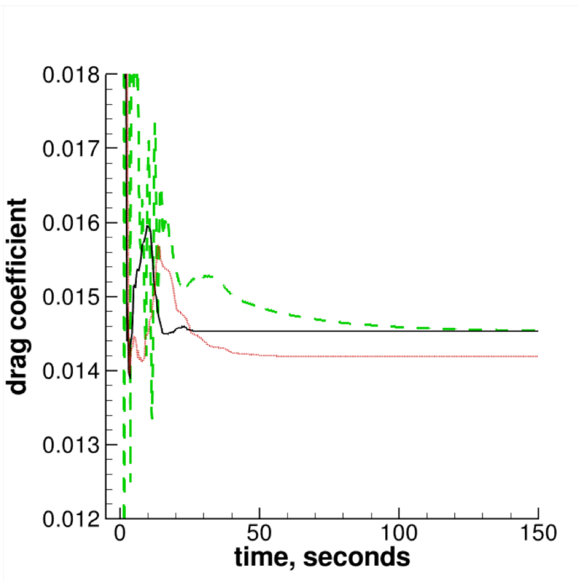


(b) CFL



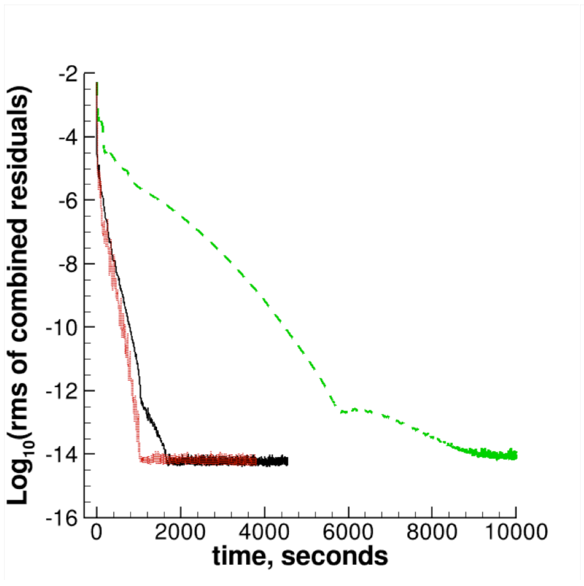


(c) maximum normalized eddy viscosity

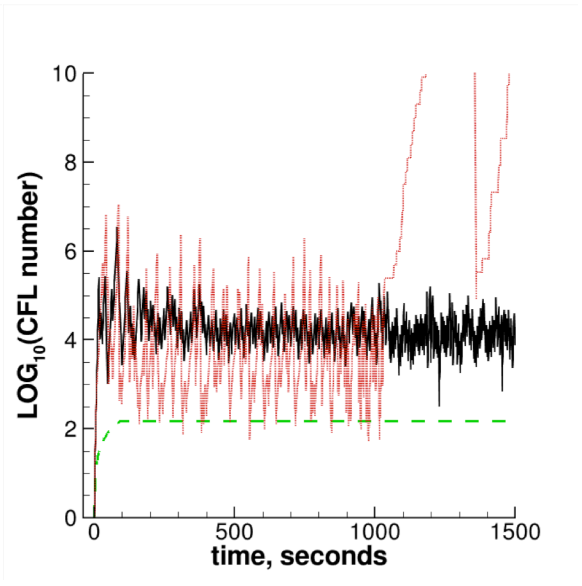


(d) total drag coefficient

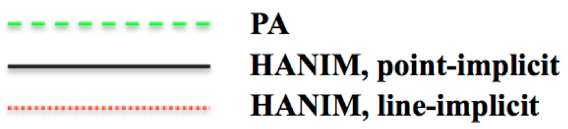
Figure 10. NACA 0012 solution convergence history using 2x225x65 hexahedral grid. $M_\infty = 0.15$, $\alpha = 10°$, $Re_c = 6x10^6$.

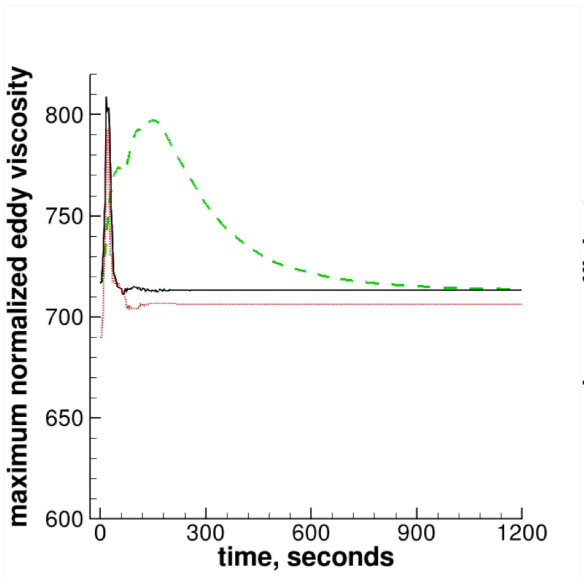


(a) rms of combined mean flow and SA model residuals

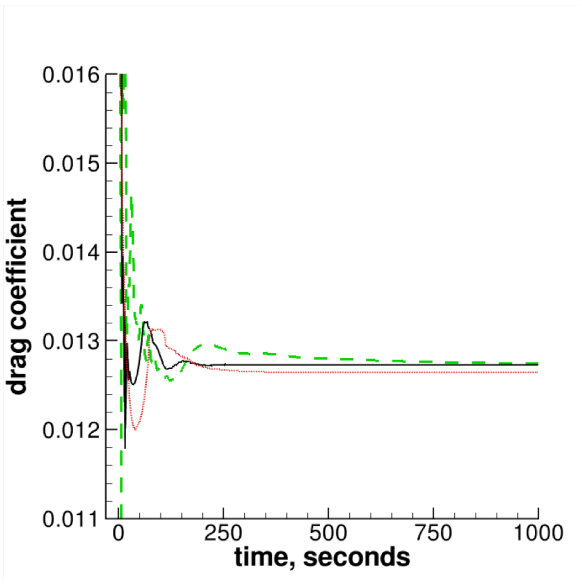


(b) CFL



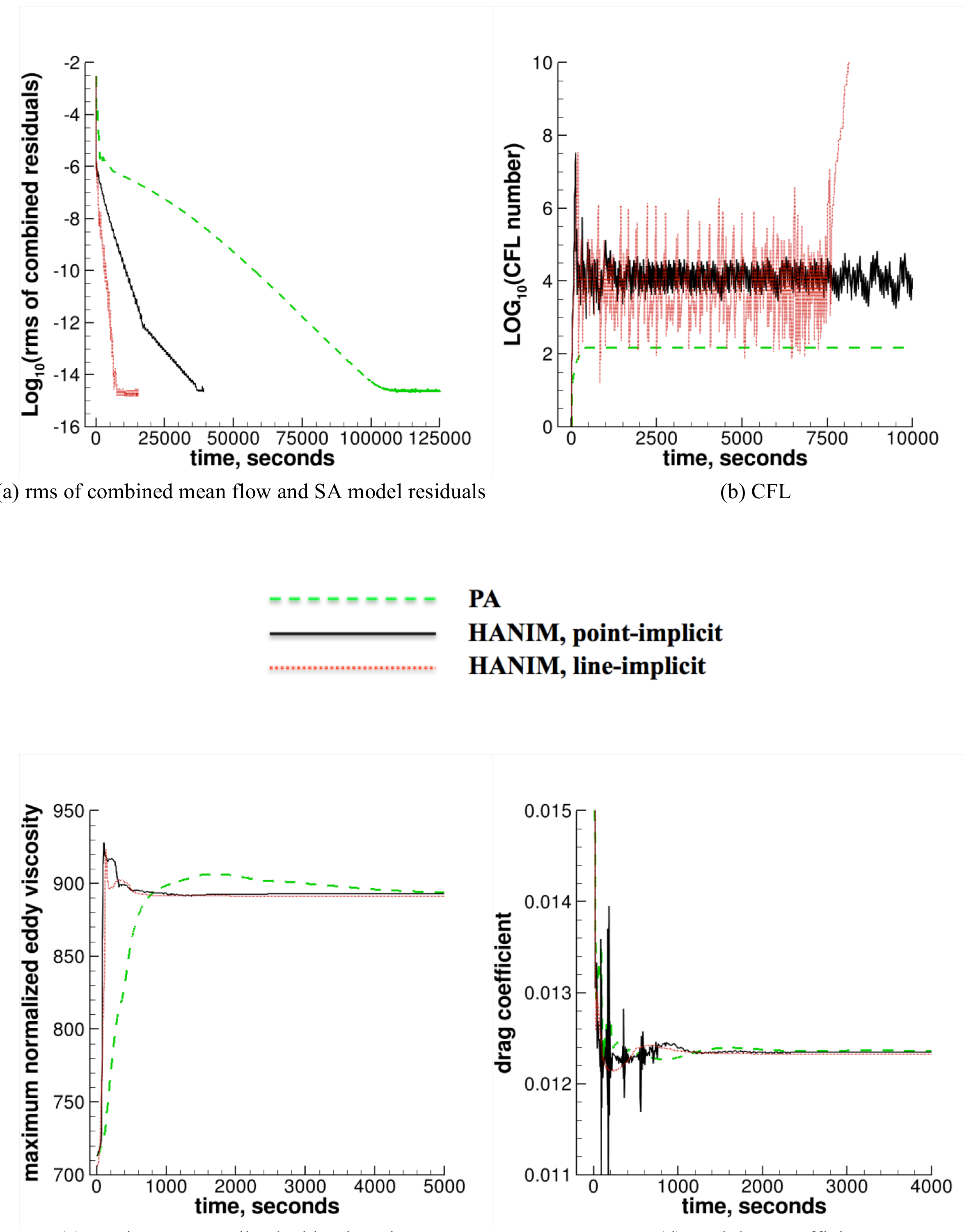


(c) maximum normalized eddy viscosity



(d) total drag coefficient

Figure 11. NACA 0012 solution convergence history using 2x449x129 hexahedral grid. M_∞ = 0.15, α = 10°, Re_c = 6x10^6.



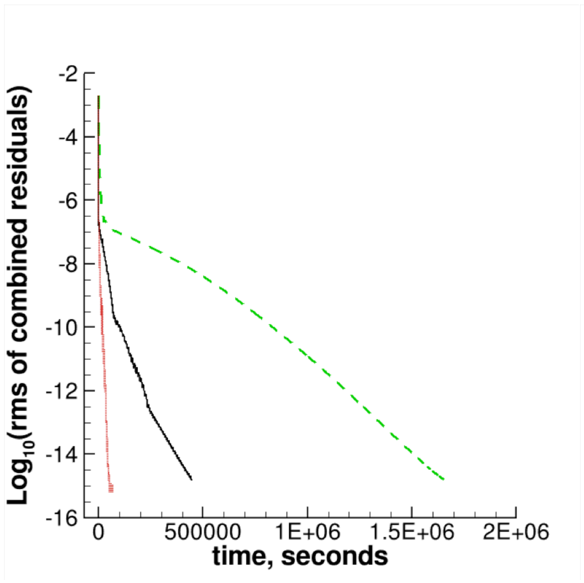
(a) rms of combined mean flow and SA model residuals

(b) CFL

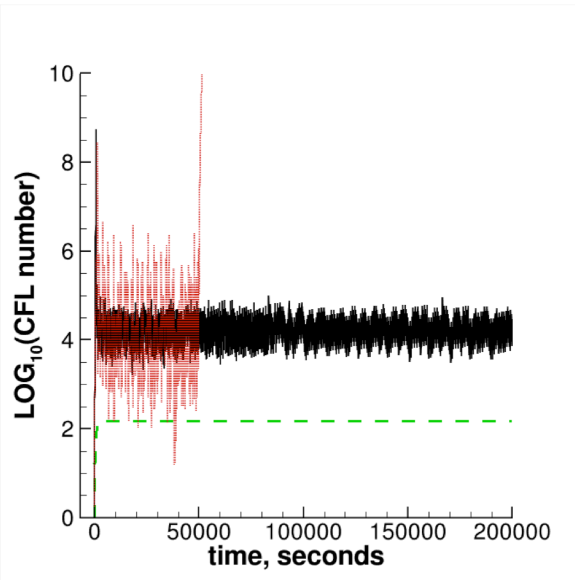
(c) maximum normalized eddy viscosity

(d) total drag coefficient

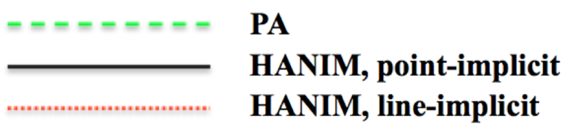
Figure 12. NACA 0012 solution convergence history using 2x897x257 hexahedral grid. M_∞ = 0.15, α = 10°, $Re_c = 6x10^6$.

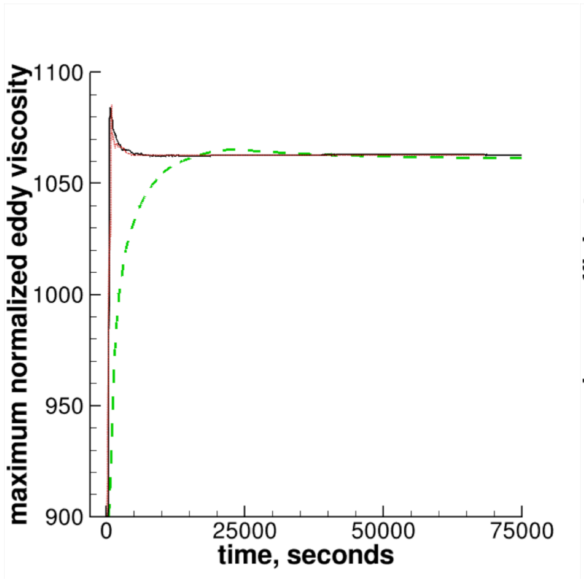


(a) rms of combined mean flow and SA model residuals

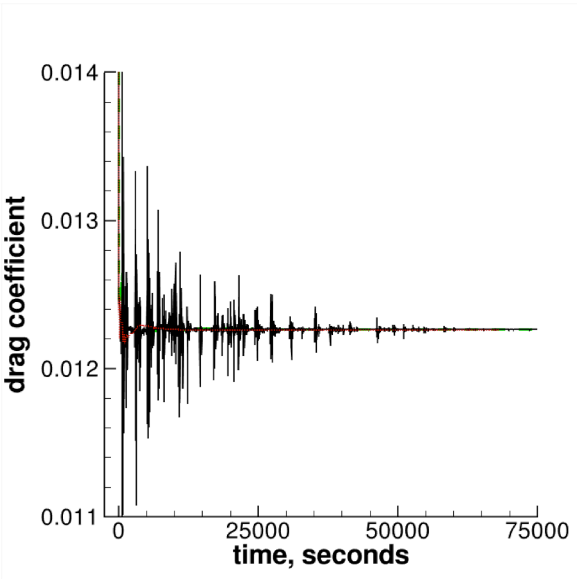


(b) CFL



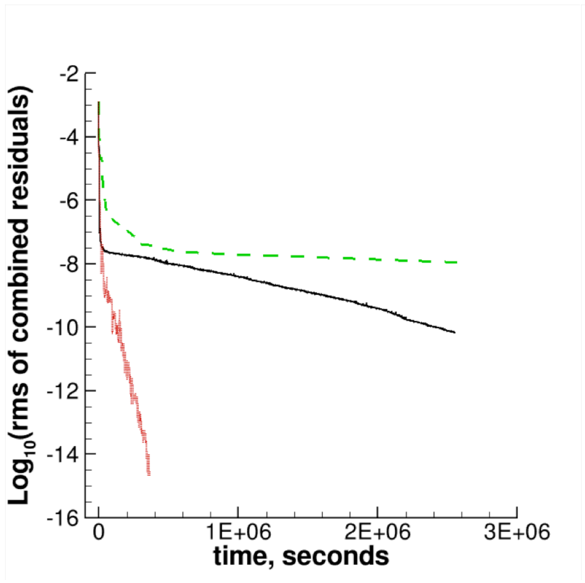


(c) maximum normalized eddy viscosity

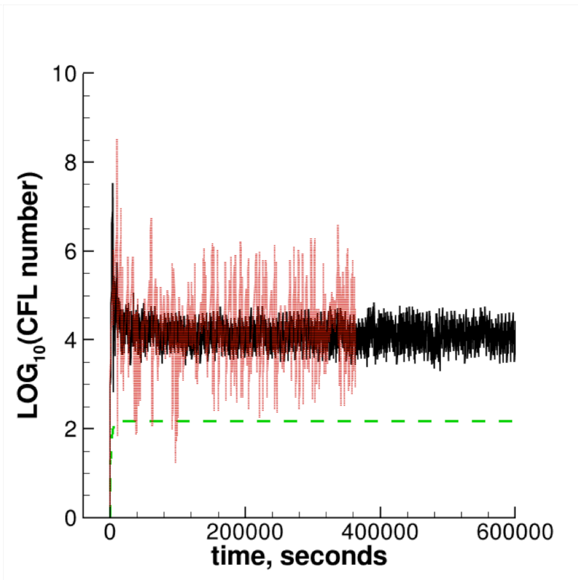


(d) total drag coefficient

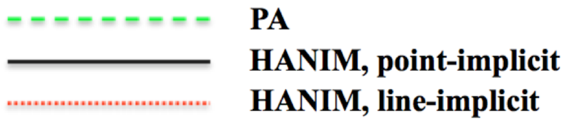
Figure 13. NACA 0012 solution convergence history using 2x1793x513 hexahedral grid. M_∞ = 0.15, α = 10°, $Re_c = 6x10^6$.

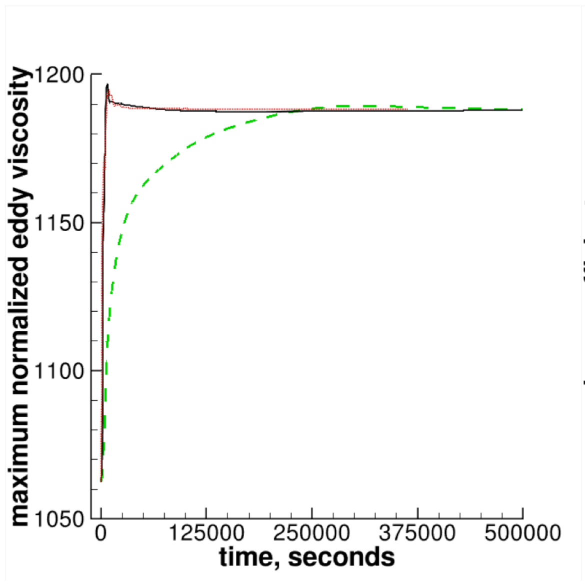


(a) rms of combined mean flow and SA model residuals

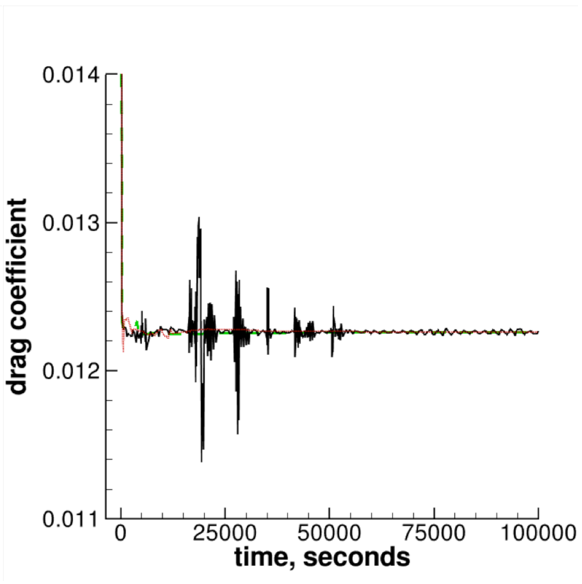


(b) CFL



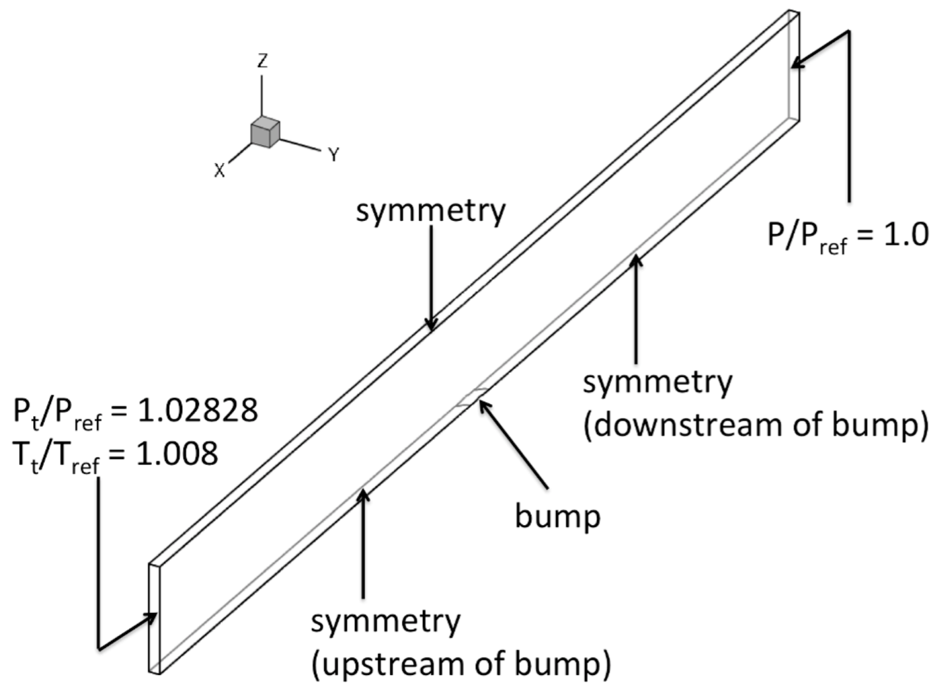


(c) maximum normalized eddy viscosity

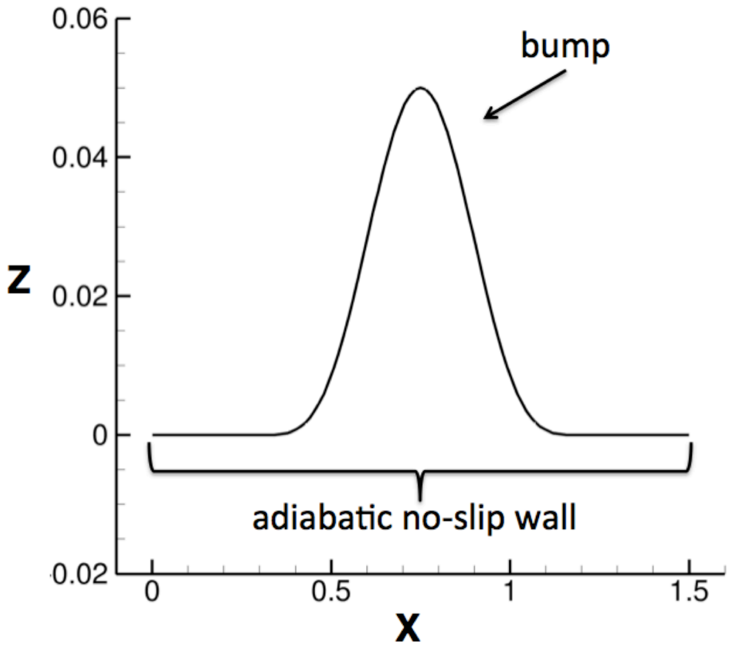


(d) total drag coefficient

Figure 14. NACA 0012 solution convergence history using 2x3585x1025 hexahedral grid. M_∞ = 0.15, α = 10°, Re_c = 6x10^6.

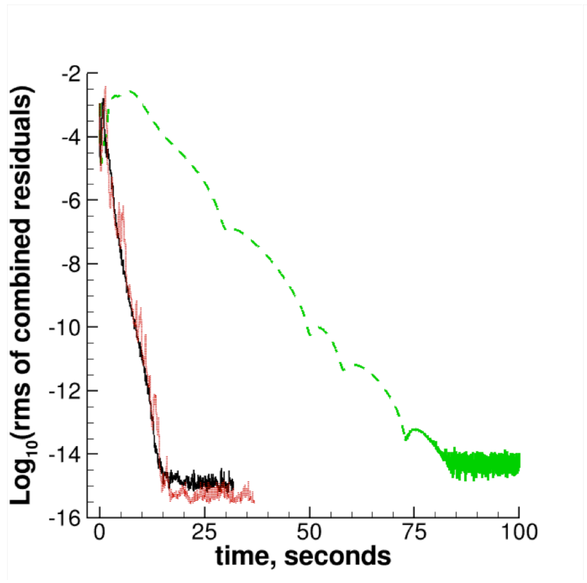


(a) far view

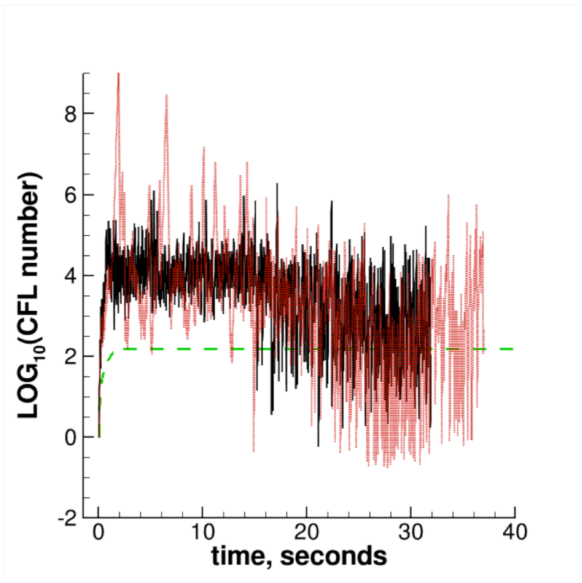


(b) near view

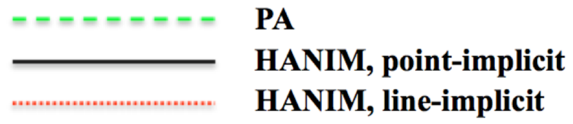
Figure 15. Schematic representation of the 3D bump-in-channel configuration and its computational domain with boundary conditions (BC).

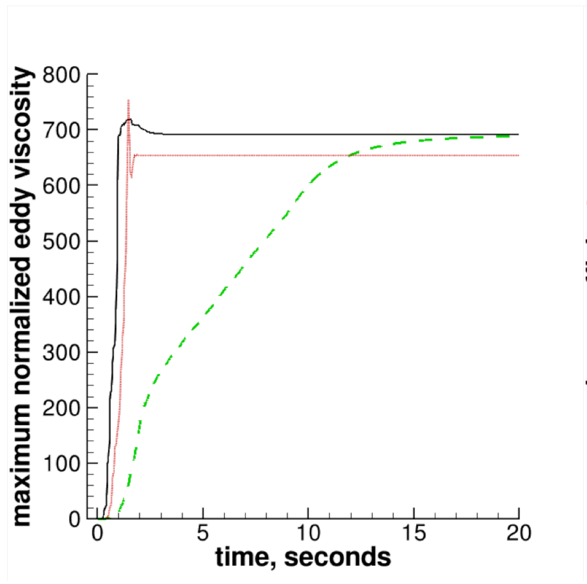


(a) rms of combined mean flow and SA model residuals

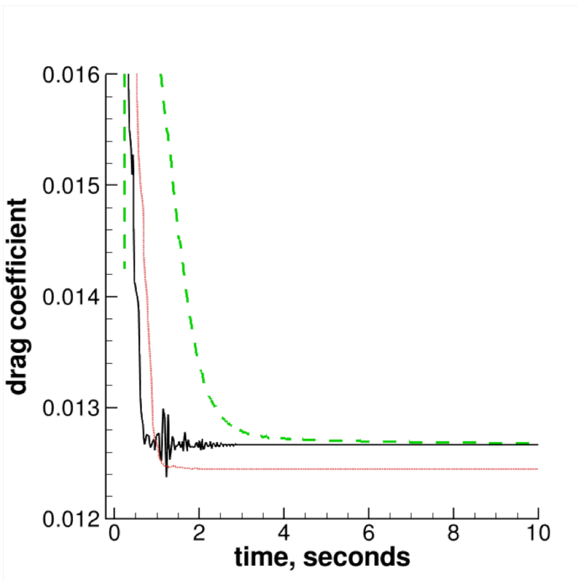


(b) CFL



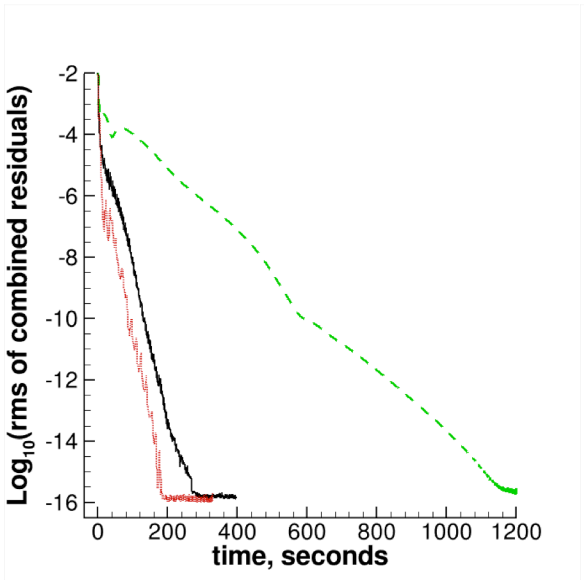


(c) maximum normalized eddy viscosity

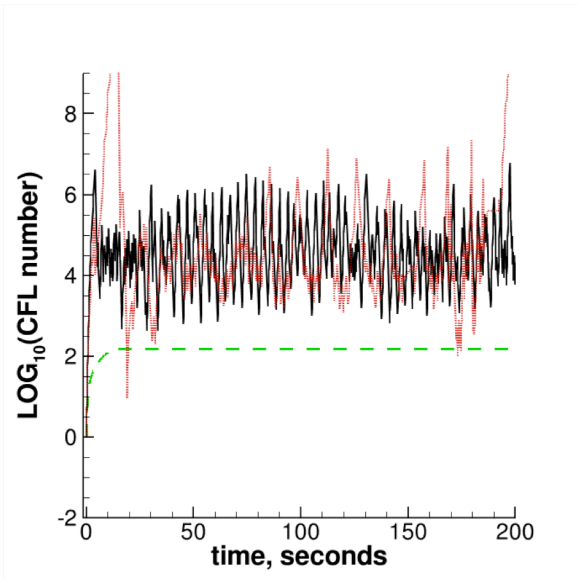


(d) total drag coefficient

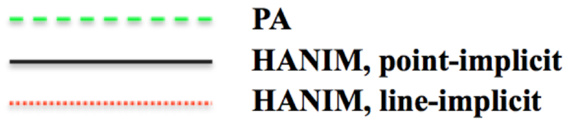
Figure 16. 3D Bump-in-channel solution convergence history using 3x45x21 hexahedral grid. M_∞ = 0.2, Re_L = 3x10^6.

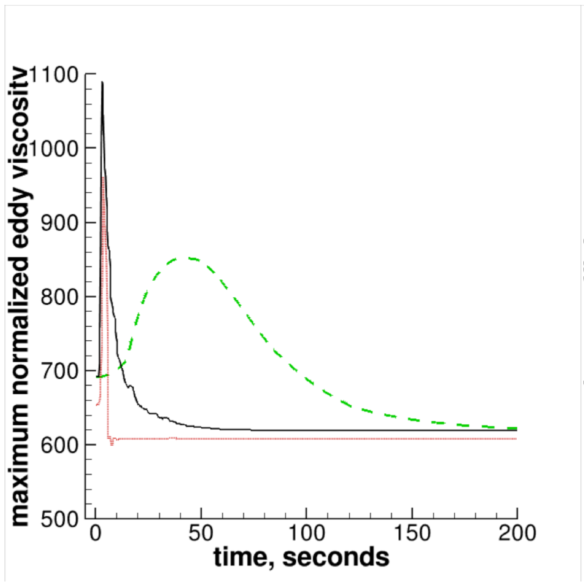


(a) rms of combined mean flow and SA model residuals

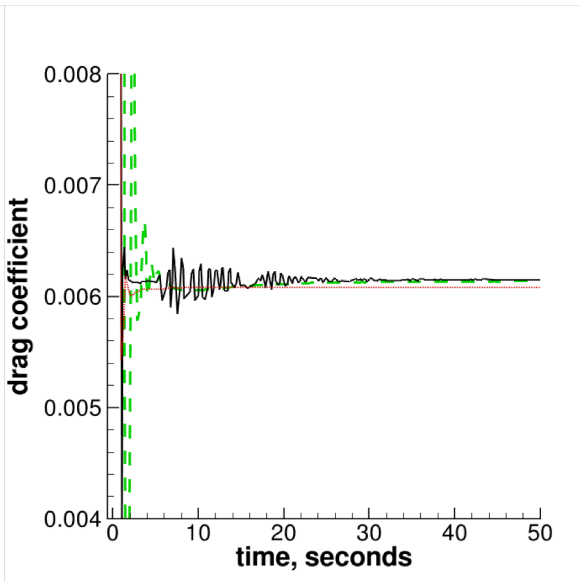


(b) CFL



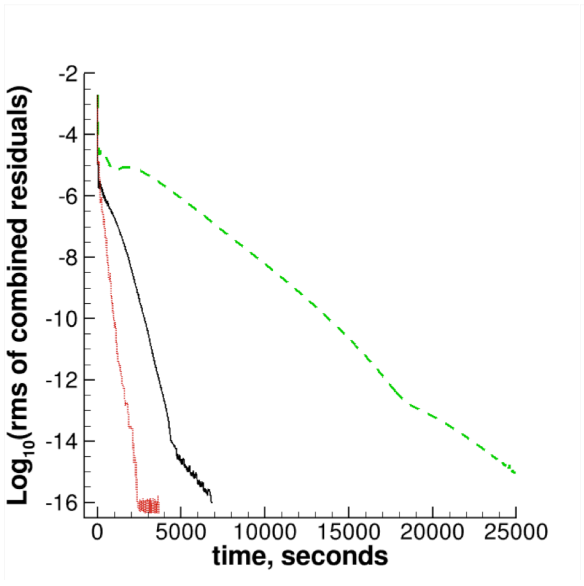


(c) maximum normalized eddy viscosity

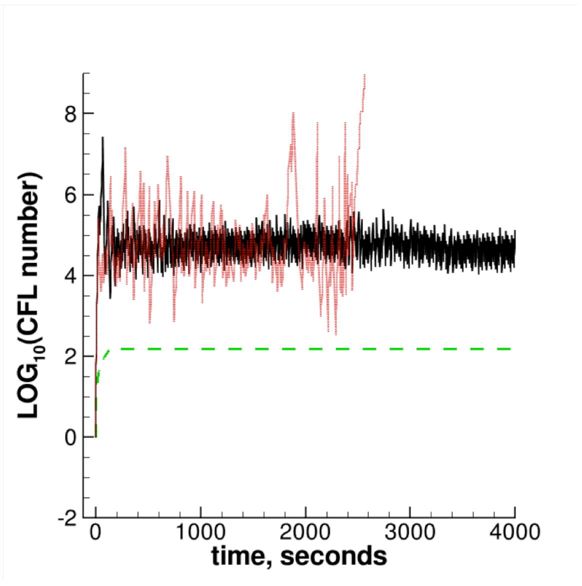


(d) total drag coefficient

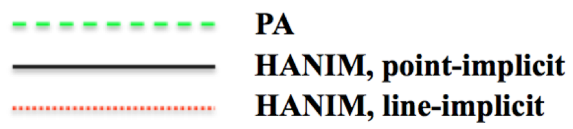
Figure 17. 3D Bump-in-channel solution convergence history using 5x89x41 hexahedral grid. $M_\infty = 0.2$, $Re_L = 3x10^6$.

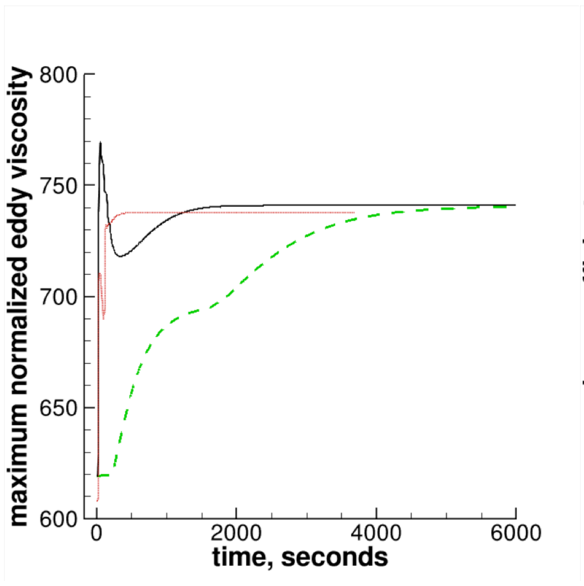


(a) rms of combined mean flow and SA model residuals

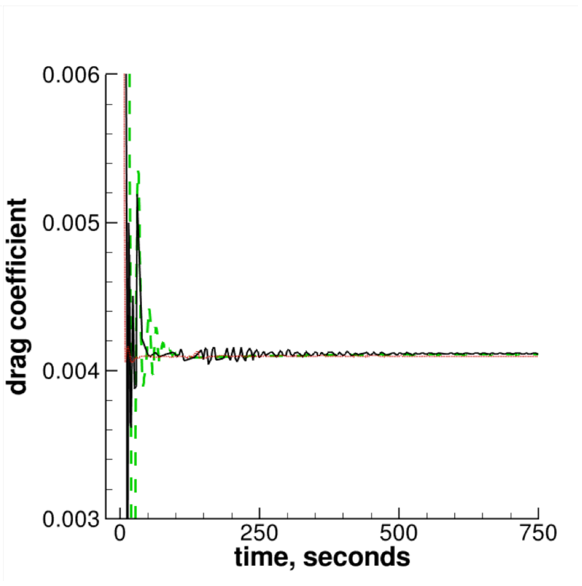


(b) CFL



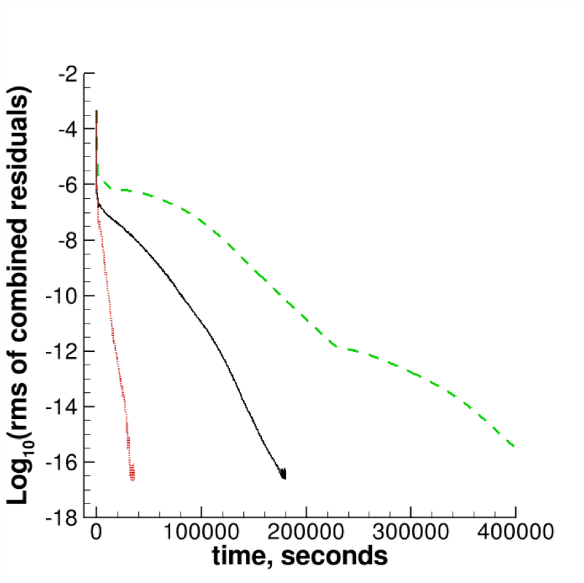


(c) maximum normalized eddy viscosity

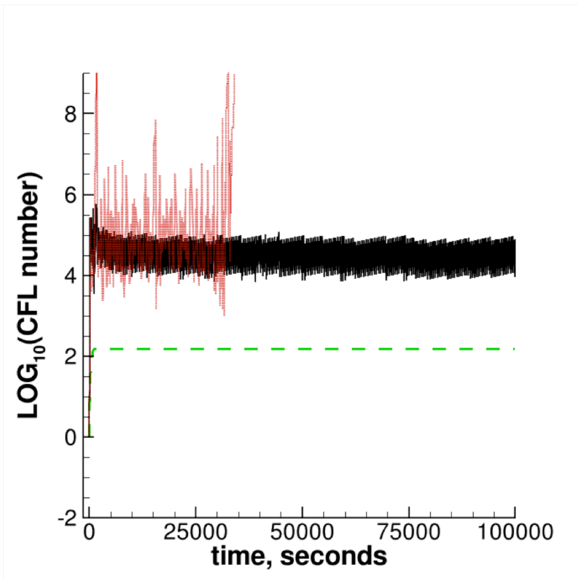


(d) total drag coefficient

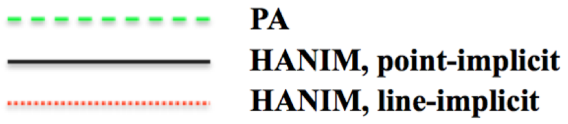
Figure 18. 3D Bump-in-channel solution convergence history using 9x177x81 hexahedral grid. $M_\infty = 0.2$, $Re_L = 3x10^6$.

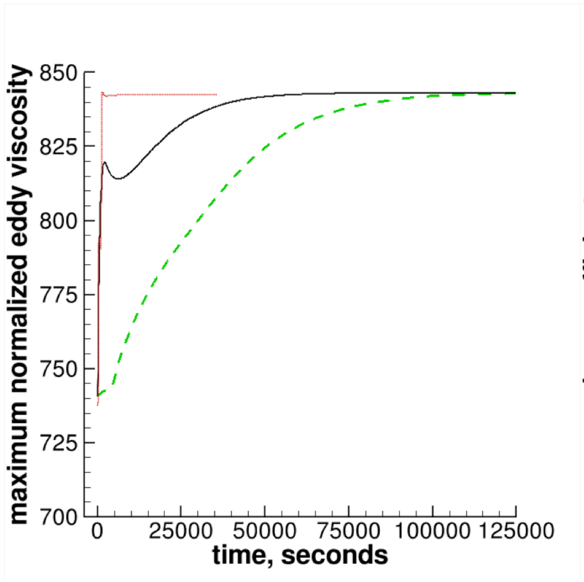


(a) rms of combined mean flow and SA model residuals

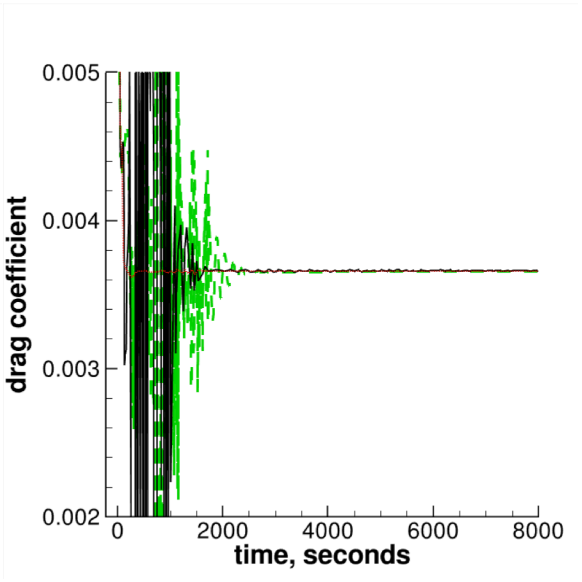


(b) CFL



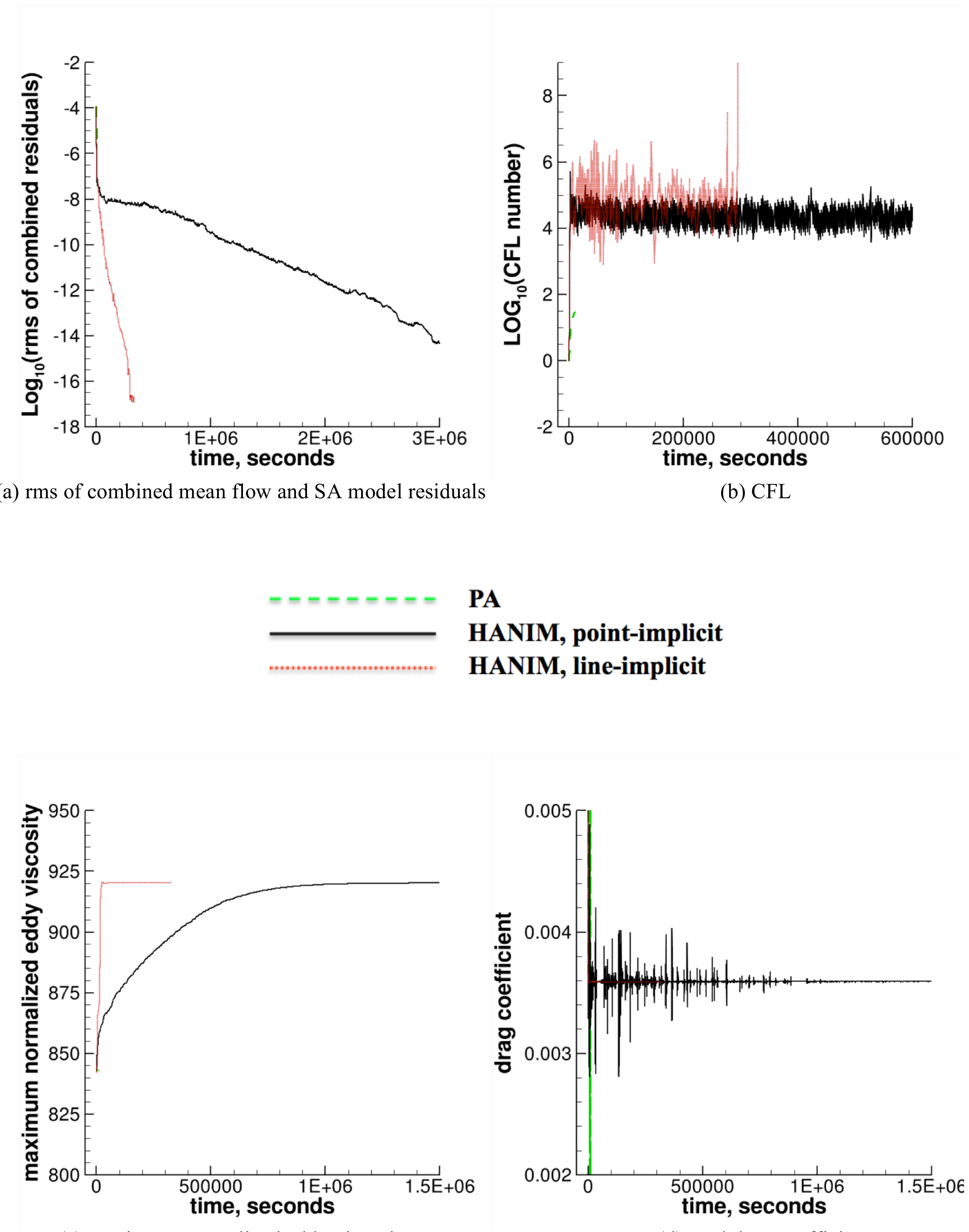


(c) maximum normalized eddy viscosity



(d) total drag coefficient

Figure 19. 3D Bump-in-channel solution convergence history using 17x353x161 hexahedral grid. M_∞ = 0.2, $Re_L = 3x10^6$.



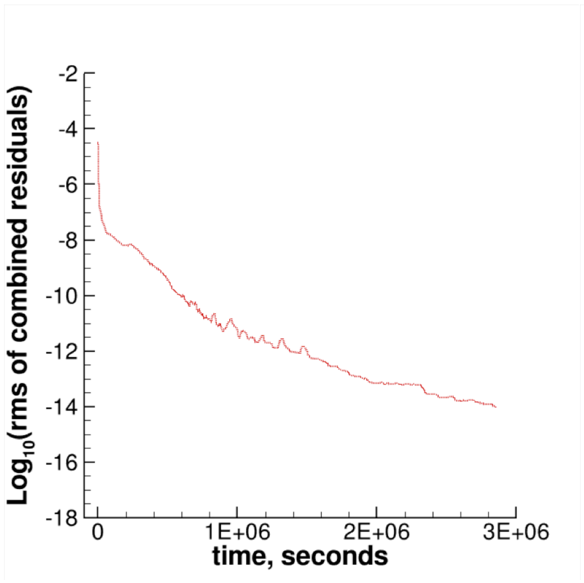
(a) rms of combined mean flow and SA model residuals

(b) CFL

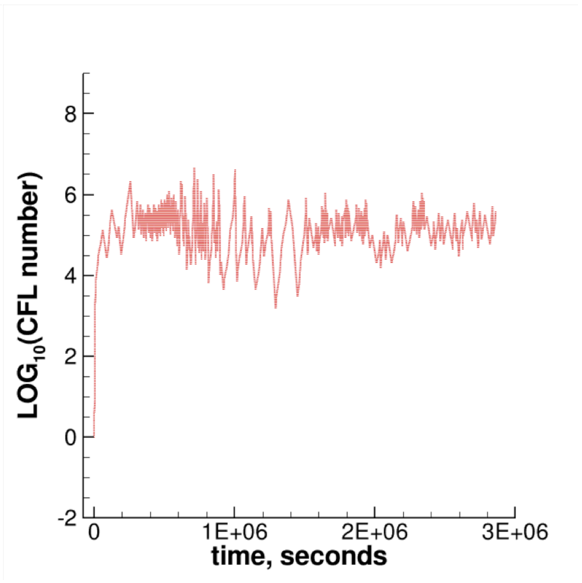
(c) maximum normalized eddy viscosity

(d) total drag coefficient

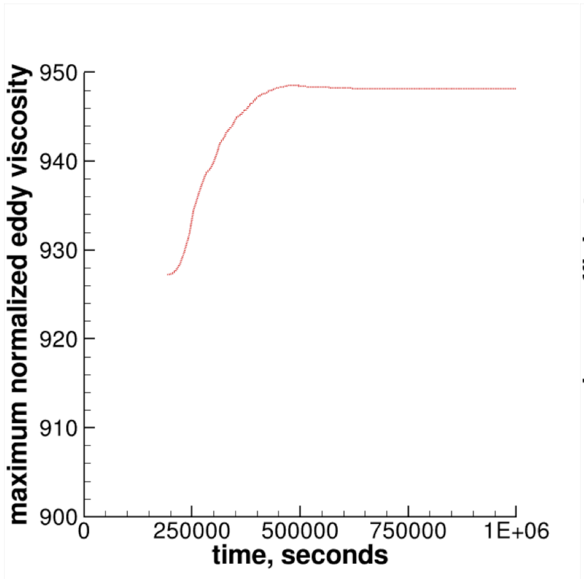
Figure 20. 3D Bump-in-channel solution convergence history using 33x705x321 hexahedral grid. M_∞ = 0.2, Re_L = 3x10^6.



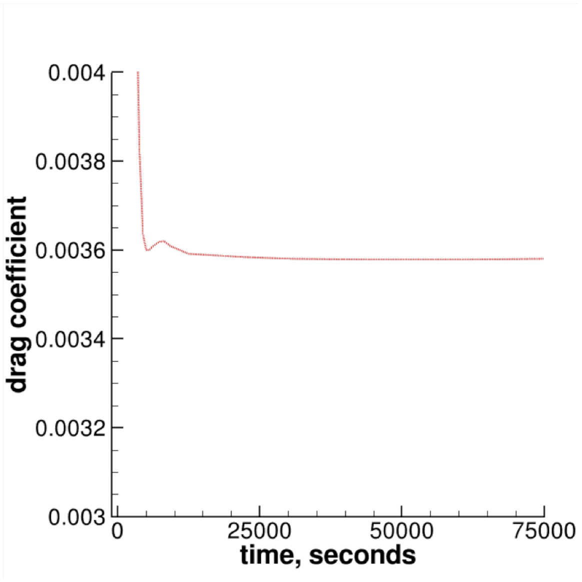
(a) rms of combined mean flow and SA model residuals



(b) CFL

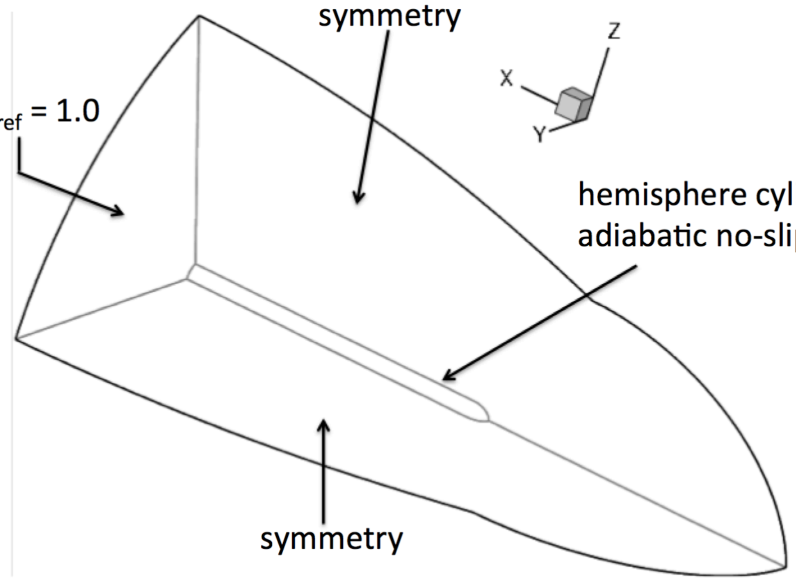


(c) maximum normalized eddy viscosity

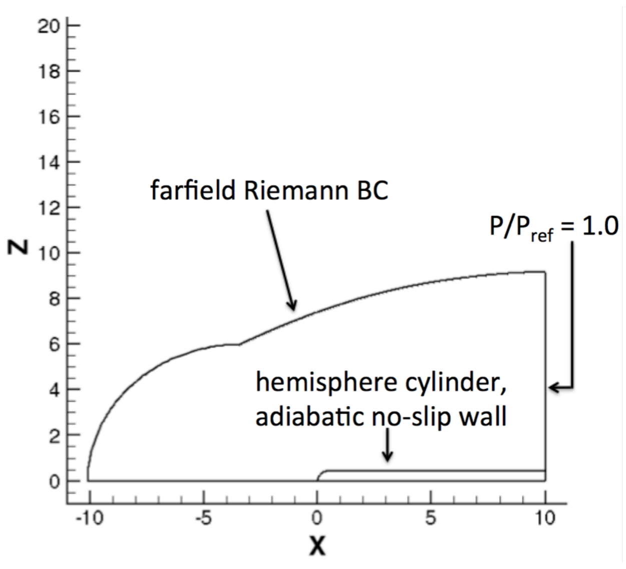


(d) total drag coefficient

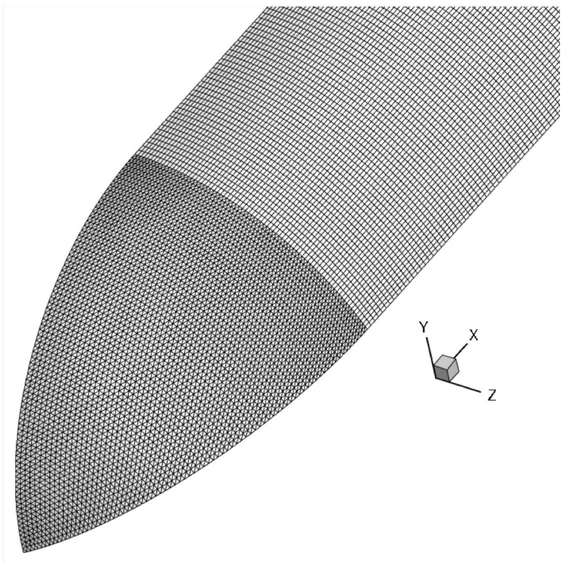
Figure 21. 3D Bump-in-channel solution convergence history using 65x1409x641 hexahedral grid. M_∞ = 0.2, $Re_L = 3x10^6$.



(a) isometric view

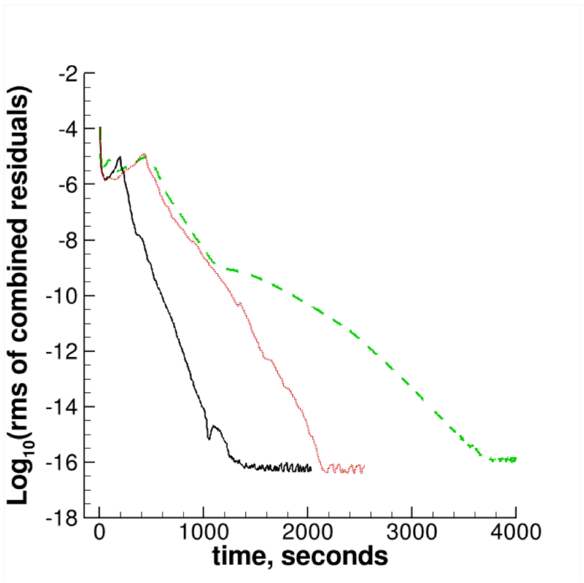


(b) sideview

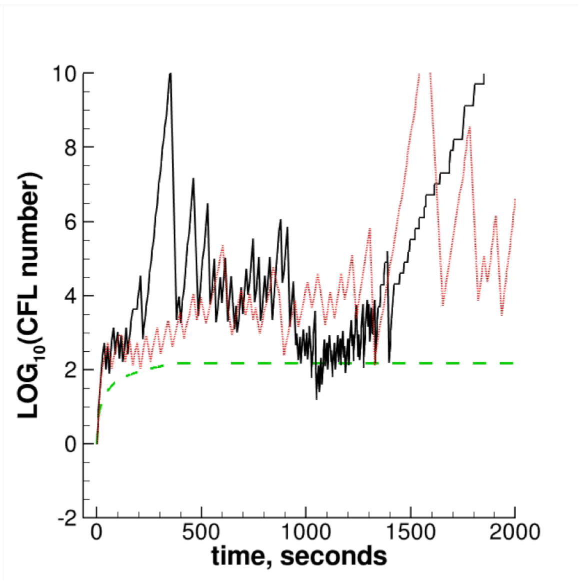


(c) closeup view of surface grid

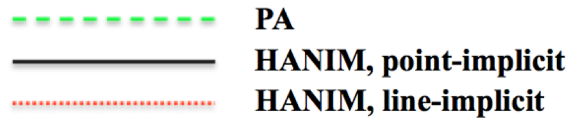
Figure 22. Schematic representation of the 3D hemisphere cylinder configuration and its computational domain with boundary conditions (BC) as well as a view of the grid near the interface of hemisphere and cylinder.

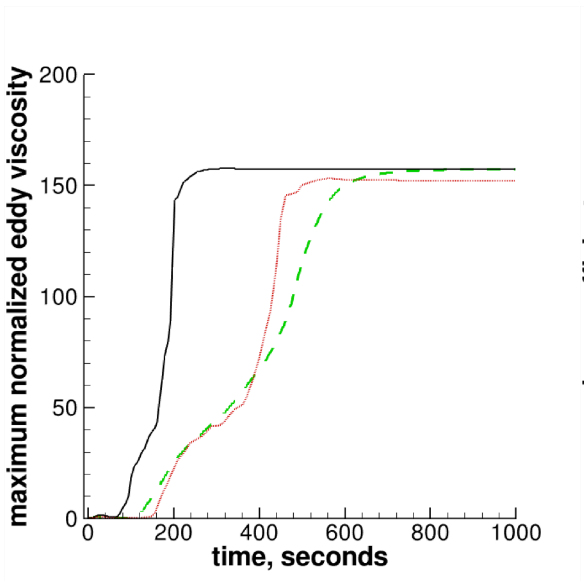


(a) rms of combined mean flow and SA model residuals

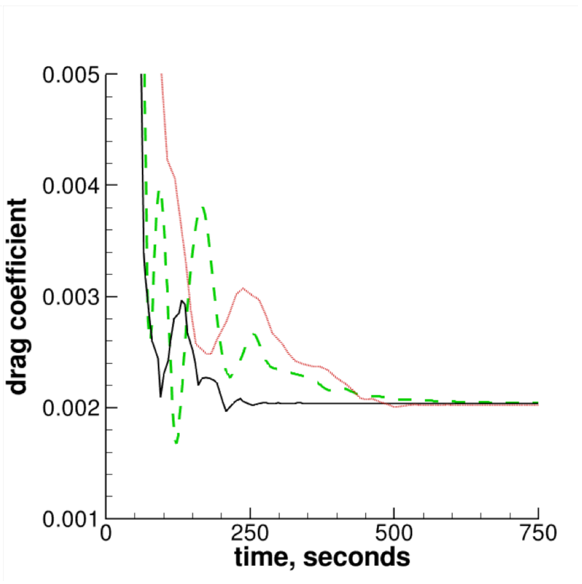


(b) CFL



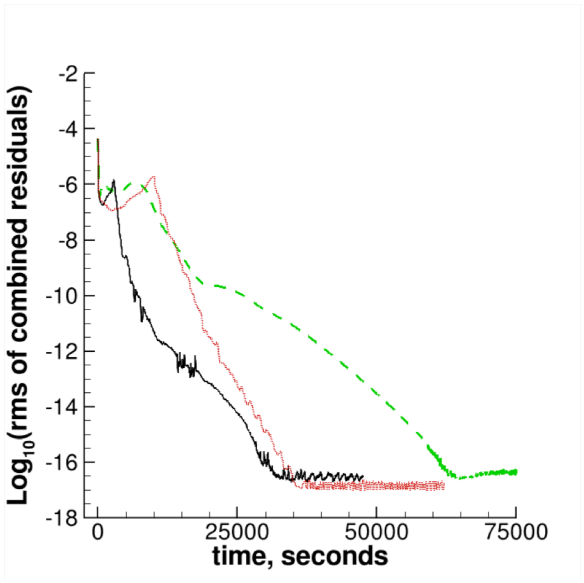


(c) maximum normalized eddy viscosity

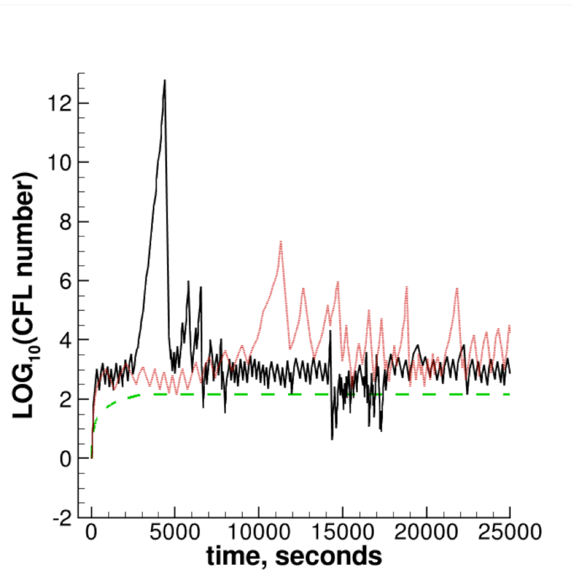


(d) total drag coefficient

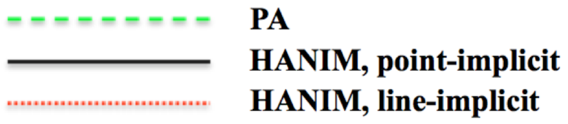
Figure 23. 3D Hemisphere Cylinder solution convergence history using a mixed prismatic, hexahedral coarse grid (Grid.3). $M_\infty = 0.6$, $\alpha = 0°$, $Re_L = 0.35x10^6$.

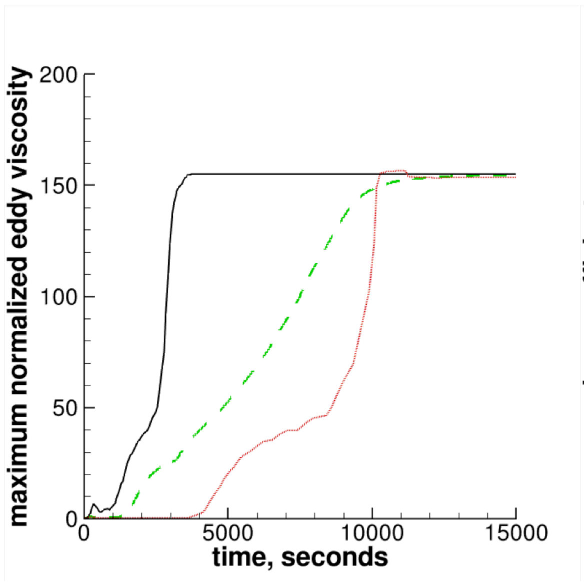


(a) rms of combined mean flow and SA model residuals

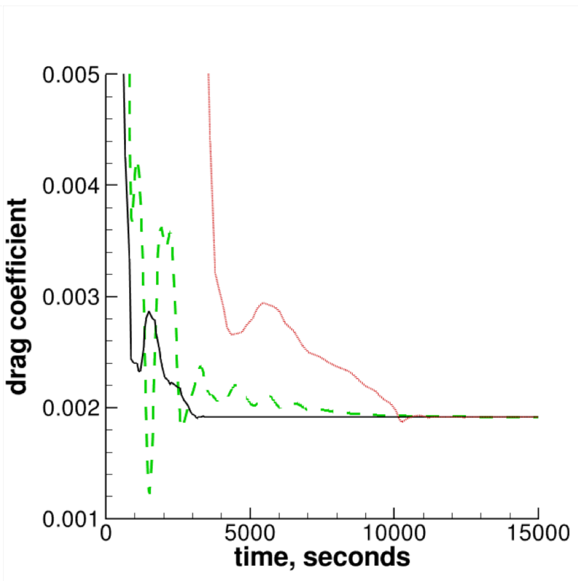


(b) CFL



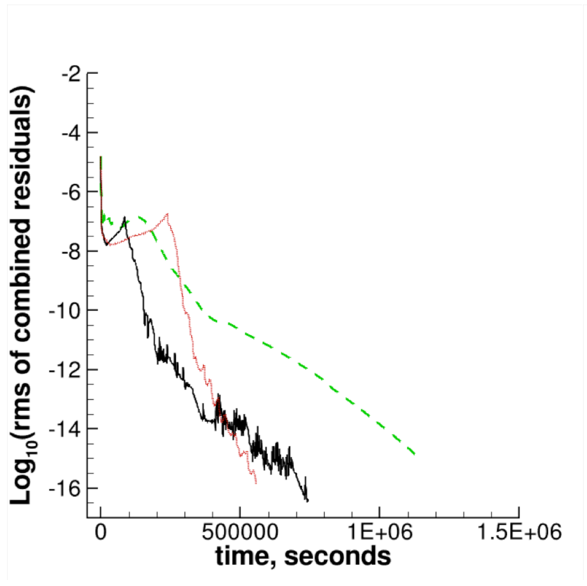


(c) maximum normalized eddy viscosity

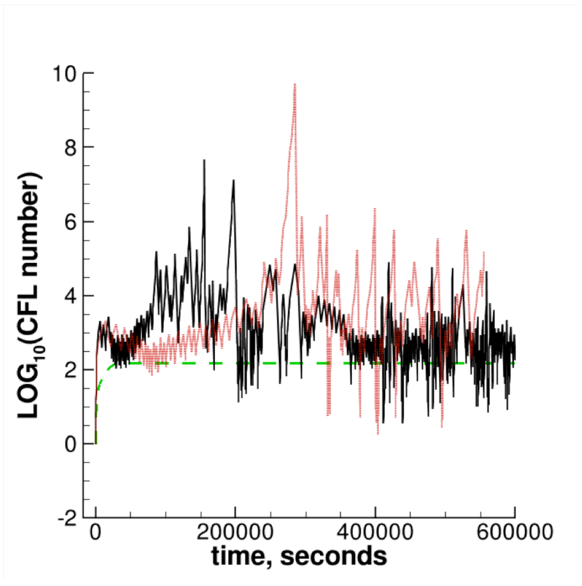


(d) total drag coefficient

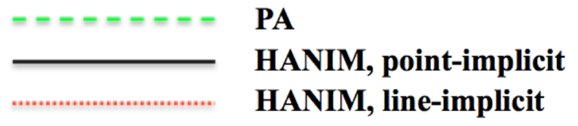
Figure 24. 3D Hemisphere Cylinder solution convergence history using a mixed prismatic, hexahedral medium grid (Grid.2). $M_\infty = 0.6$, $\alpha = 0°$, $Re_L = 0.35x10^6$.

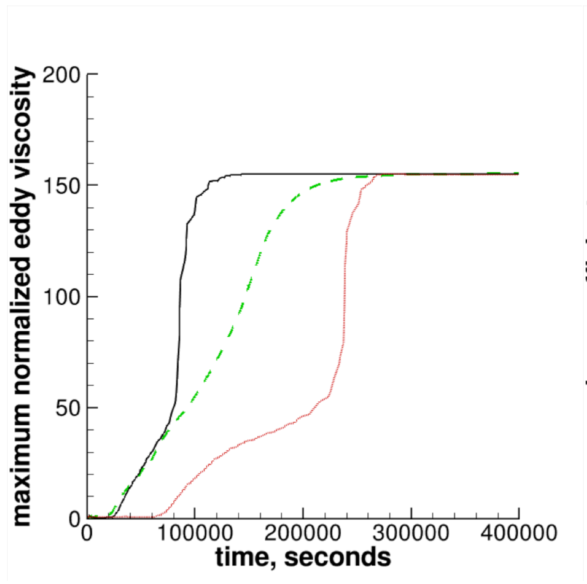


(a) rms of combined mean flow and SA model residuals

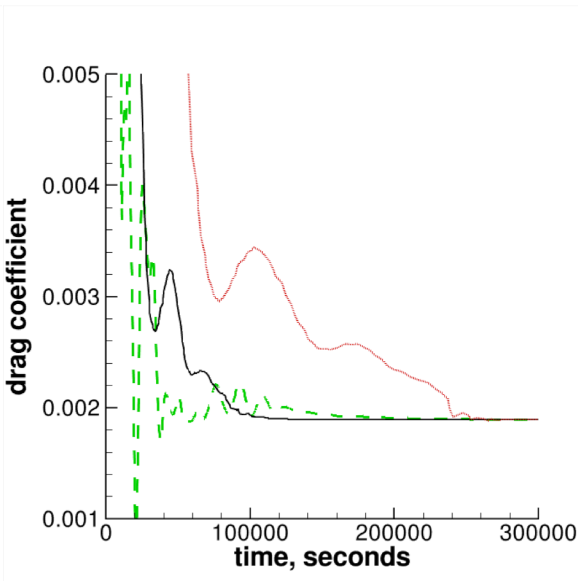


(b) CFL



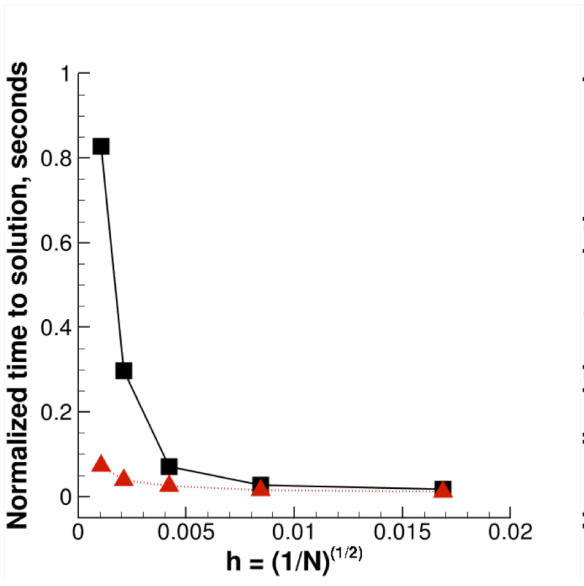


(c) maximum normalized eddy viscosity

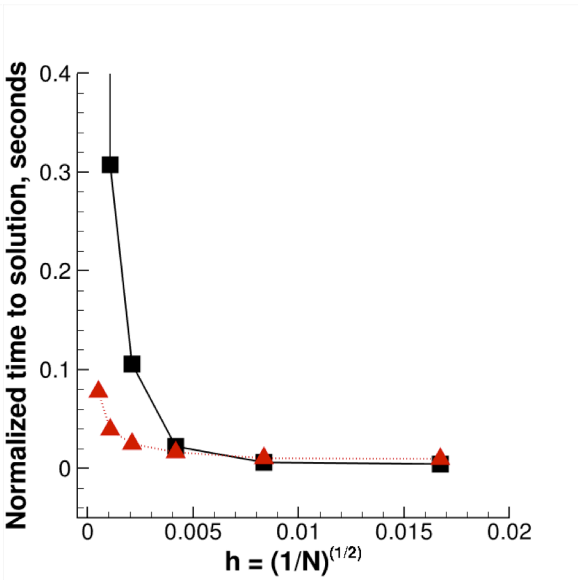


(d) total drag coefficient

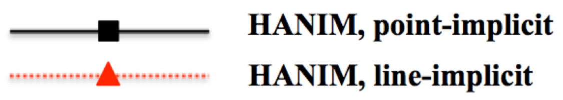
Figure 25. 3D Hemisphere Cylinder solution convergence history using a mixed prismatic, hexahedral fine grid (Grid.1). $M_\infty = 0.6$, $\alpha = 0°$, $Re_L = 0.35x10^6$.

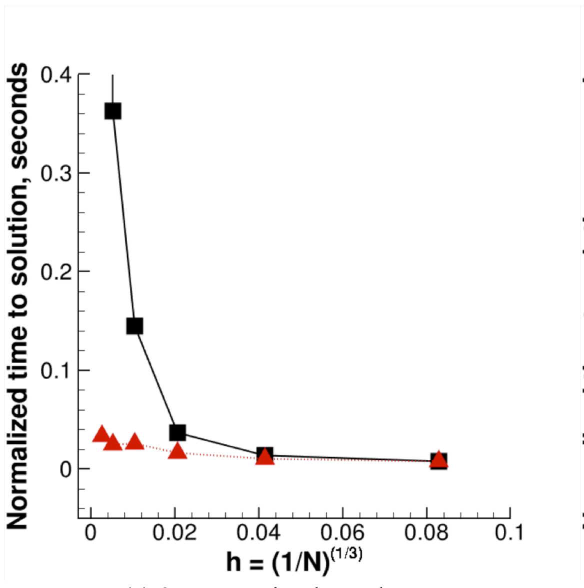


(a) 2D Bump-in-channel

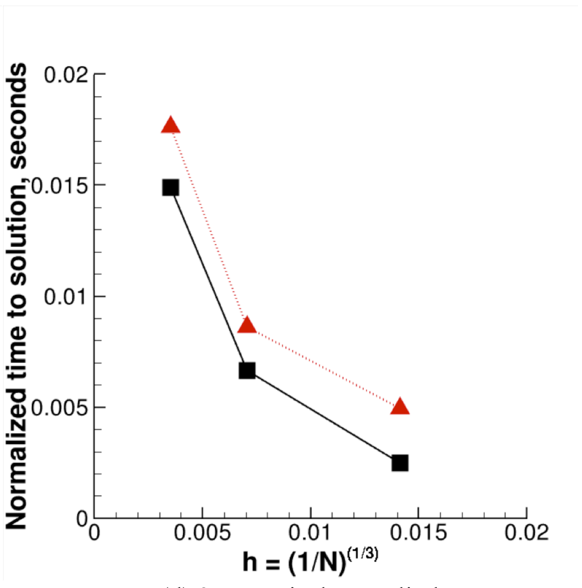


(b) 2D NACA 0012 airfoil



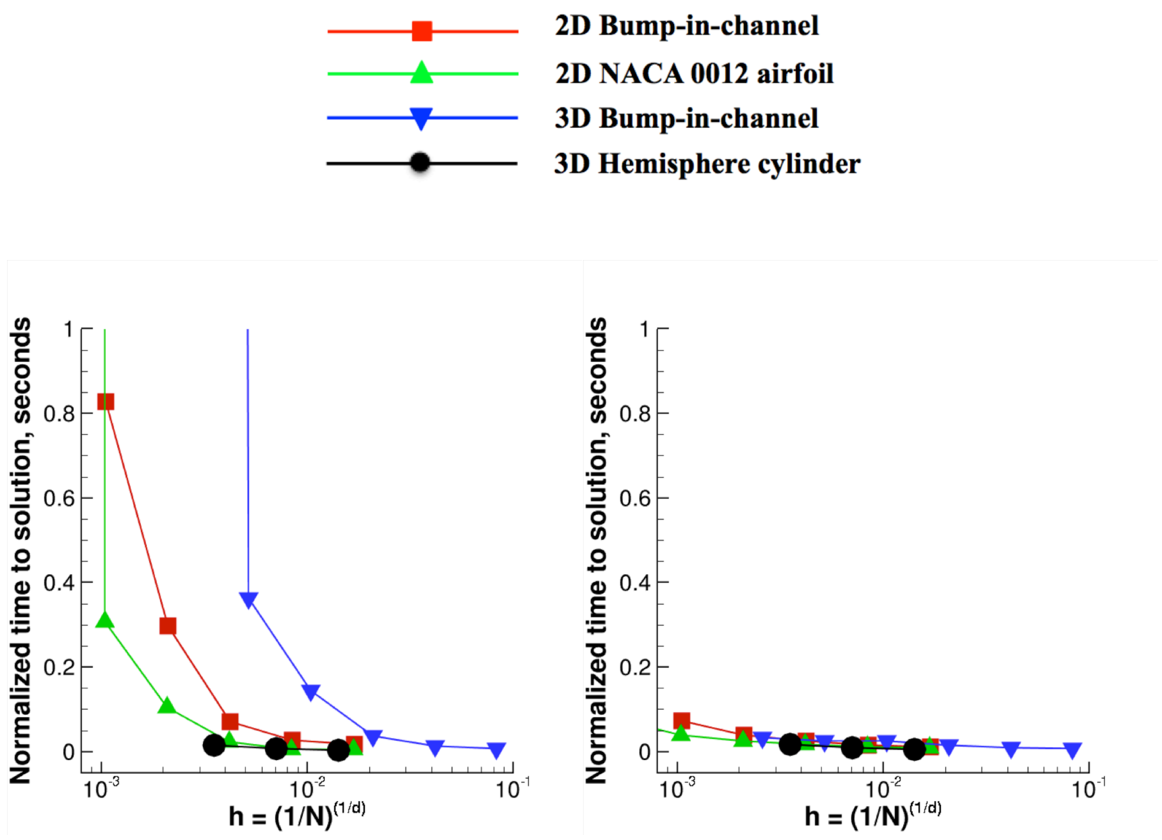


(c) 3D Bump-in-channel



(d) 3D Hemisphere cylinder

Figure 26. Variations of normalized CPU time to solution with respect to characteristic grid spacing (h) for various cases. The CPU time is normalized by degrees of freedom (N).



(a) point-implicit HANIM

(b) line-implicit HANIM

Figure 27. Variations of normalized CPU time to solution with respect to characteristic grid spacing (h) for point- and line-implicit HANIM. The CPU time is normalized by degrees of freedom (N). The parameter d is case dimension.