Three-Dimensional Aerodynamic Computations on Unstructured Grids Using a Newton-Krylov Approach

P. Wong[*] and D. W. Zingg[†]
University of Toronto Institute for Aerospace Studies
4925 Dufferin Street, Toronto, Ontario
Canada, M3H 5T6

A Newton-Krylov algorithm is presented for the compressible Navier-Stokes equations in three dimensions on unstructured grids. The algorithm uses a preconditioned matrix-free Krylov method to solve the linear system that arises in the Newton iterations. Incomplete factorization is used as the preconditioner, based on an approximate Jacobian matrix after the reverse Cuthill-McKee reordering of the unknowns. Approximate viscous operators that involve only the nearest neighboring terms are studied to construct an efficient and effective preconditioner. Two incomplete factorization techniques are examined: one based on a level-of-fill approach while the other utilizes a threshold strategy. The performance of the algorithm is demonstrated with numerical studies over the ONERA M6 wing and the DLR-F6 wing-body configuration. A ten-order-of-magnitude residual reduction can be obtained in 20-25 hours on a single processor for a mesh with roughly 500,000 nodes.

I. Introduction

After many years of development, computational fluid dynamics has become an important aerodynamic design tool.[1,2] The current technology is capable of analyzing viscous compressible flows over complete aircraft configurations. Two AIAA Drag Prediction Workshops were organized to assess the capability of current solvers when applied to such flows. Lee-Rausch et al.[3] summarized the results of three state-of-the-art unstructured grid solvers when applied to solve the the DLR-F6 transport configuration in the second Drag Prediction Workshop. May et al.[4] summarized the results for the same configuration from two well-known structured grid solvers. Other studies include the work of Luo et al.,[5] and Brodersen and Stürmer.[6] With parallelization, flow solutions can be obtained in several hours using a grid with three million nodes. However, the grid density is insufficient to achieve grid convergence. Moreover, variations are observed between solutions from different codes. In order to improve the accuracy of the technology, finer grids, mesh adaptation and higher-order methods have been suggested. Research also continues in the development of faster algorithms to reduce computational time.

The Newton-Krylov method is an efficient method to solve the Navier-Stokes equations.[7] Its implicit nature has good potential in turbulent applications when stretched meshes are used to calculate the viscous boundary layers. Some early work for aerodynamic applications can be found in the studies by Venkatakrishnan and Mavriplis,[8] Barth and Linton,[9] Nielsen et al.,[10] and Anderson et al.[11] Blanco and Zingg[12] performed a study comparing quasi-Newton, standard Newton, and matrix-free Newton methods. They developed a fast solver on triangular grids using a matrix-free inexact-Newton approach together with an approximate-Newton startup strategy. Pueyo and Zingg[13] developed a preconditioned matrix-free Newton-Krylov algorithm. Their optimized algorithm is found to converge faster and more reliably than an approximate Newton algorithm and an approximately-factored multigrid algorithm. Geuzaine et al.[14,15] studied mesh sequencing as well as multigrid preconditioning with the Newton-Krylov method. Nemec and Zingg[16] applied a Newton-Krylov method to numerical optimization. The same approach is applied to solve the flow equations

[*]Ph.D. Candidate, peterson@oddjob.utias.utoronto.ca
[†]Professor, Senior Member AIAA, Tier I Canada Research Chair in Computational Aerodynamics, 2004 Guggenheim Fellow, http://goldfinger.utias.utoronto.ca/dwz

as well as the adjoint equations to calculate the objective function gradients. Chisholm and Zingg[17,18] developed a strategy which provides effective and efficient startup for turbulent flows with the Newton-Krylov algorithm. Manzano et al.[19] applied the Newton-Krylov algorithm to three-dimensional inviscid flows using unstructured grids.

The purpose of this work is to extend the algorithm of Manzano et al. to turbulent flows on hybrid unstructured grids. The goal is to develop an efficient and robust algorithm for three-dimensional aerodynamic flows. Different aspects of the algorithm are studied and discussed in the paper, including preconditioning and startup strategy. The performance of the algorithm is demonstrated over a wing as well as a wing-body configuration.

II. Governing Equations

The governing equations are the compressible Navier-Stokes equations. These equations describe the conservation of mass, momentum and total energy for a viscous compressible flow. For an arbitrary control volume Ω, the integral from of the equations can be written as:

$$\frac{\partial}{\partial t}\int_{\Omega} Q dV + \int_{\partial\Omega} \mathbf{F}\cdot\hat{\mathbf{n}} dS = \int_{\partial\Omega} \mathbf{G}\cdot\hat{\mathbf{n}} dS \tag{1}$$

where Q is the set of conservative flow variables (density ρ, momentum components ρu, ρv, ρw, and total energy ρE), $\mathbf{F}$ is the inviscid flux tensor, and $\mathbf{G}$ is the flux tensor associated with viscosity and heat conduction. These quantities can be written as:

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho u w \\ u(\rho E + p) \end{bmatrix} \hat{\mathbf{i}} + \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ \rho v w \\ v(\rho E + p) \end{bmatrix} \hat{\mathbf{j}} + \begin{bmatrix} \rho w \\ \rho u w \\ \rho v w \\ \rho w^2 + p \\ w(\rho E + p) \end{bmatrix} \hat{\mathbf{k}} \tag{2}$$

$$\mathbf{G} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x \end{bmatrix} \hat{\mathbf{i}} + \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{yx} + v\tau_{yy} + w\tau_{yz} - q_y \end{bmatrix} \hat{\mathbf{j}} + \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u\tau_{zx} + v\tau_{zy} + w\tau_{zz} - q_z \end{bmatrix} \hat{\mathbf{k}} \tag{3}$$

For a Newtonian fluid in local thermodynamic equilibrium, Stokes relation is valid. The viscous stress tensor $\boldsymbol{\tau}$ can be related to the dynamic viscosity μ and the strain rate tensor using:

$$\boldsymbol{\tau} = \mu \begin{bmatrix} 2u_x & u_y + v_x & u_z + w_x \\ v_x + u_y & 2v_y & v_z + w_y \\ w_x + u_z & w_y + v_z & 2w_z \end{bmatrix} - \frac{2}{3}\mu\left(u_x + v_y + w_z\right)\mathbf{I} \tag{4}$$

where $\mathbf{I}$ is the unit tensor, and u_x denotes $\partial u/\partial x$ and so forth. The heat flux vector is given by Fourier's law $\mathbf{q} = -k\nabla T$. The thermal conductivity is related to the dynamic viscosity through the Prandtl number $Pr = c_p\mu/k$. Sutherland's law is used to calculate the dynamic viscosity. Assuming the fluid behaves as a perfect gas, the pressure p can be written in terms of the conservative variables to close the system:

$$p = (\gamma - 1)\left[\rho E - \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right)\right] \tag{5}$$

III. Turbulence Modeling

We solve the Reynolds-averaged Navier-Stokes equations for turbulent flows. The Reynolds-stress tensor is modeled using the Boussinesq approximation and introducing an eddy-viscosity term. The turbulent eddy viscosity is modeled with the one-equation Spalart and Allmaras turbulence model.[20] In differential form the model is written as:

$$
\begin{aligned}
\frac{\partial \tilde{\nu}}{\partial t} + (\mathbf{v} \cdot \nabla)\, \tilde{\nu} \;&=\; c_{b1}(1 - f_{t2})\tilde{S}\tilde{\nu} + \frac{1}{\sigma}\left[\nabla \cdot ((\nu + \tilde{\nu})\nabla\tilde{\nu}) + c_{b2}(\nabla\tilde{\nu})^2\right] \\
&\quad - \left[c_{w1} f_w - \frac{c_{b1}}{\kappa^2} f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2 + f_{t1}\Delta U^2
\end{aligned}
\tag{6}
$$

where $\mathbf{v}$ is the velocity vector, and ΔU is the norm of the velocity difference between a field point and the trip point. The model is solved in a form fully-coupled with the mean-flow equations. The eddy viscosity ν_t is calculated from the working variable $\tilde{\nu}$, and ν denotes the kinematic viscosity. The terms on the right-hand side of the equation are the production term, the diffusion term, the destruction term, and the trip term respectively. Production of eddy viscosity is proportional to a vorticity-like term $\tilde{S}$, which contains the magnitude of the vorticity in the mean flow. The destruction term governs the dissipation of the eddy viscosity due to blocking effects of the wall. The distance to the closest wall is denoted as d, κ is the von Kármán constant, and f_w is a function that models near-wall effects. The model includes a trip term that models laminar-to-turbulent flow transition. Transition locations are not predicted and are specified by the user. Alternatively, the flow can be assumed to be fully turbulent by setting the trip functions f_{t1} and f_{t2} to zero. This assumes transition occurs at the leading edge. Closure coefficients c_{b1}, σ, c_{b2}, and c_{w1} are the same as those given by Spalart and Allmaras.[21] The wall boundary condition is $\tilde{\nu} = 0$. A value of $\nu_\infty/10$ is used as the free-stream condition for $\tilde{\nu}$, where ν_∞ is the kinematic viscosity in the free stream.

Ashford[22] proposed a modification to $\tilde{S}$ in the production term. The modification is found to produce better numerical properties[17] and is adopted in the current work.

IV. Spatial Discretization

The spatial discretization follows that used by Mavriplis and Venkatakrishnan[23] for hybrid unstructured grids. A cell-vertex approach is utilized with centroidal-median-dual control volumes constructed around source-grid vertices. A finite-volume discretization is obtained by integrating the fluxes over the boundary of the control volume. The value of the flux at each control volume face is computed by averaging the fluxes in the two control volumes on either side of the face:

$$
f_{ik} \simeq \frac{1}{2}\left[\mathbf{F}(Q_i) + \mathbf{F}(Q_k)\right] \cdot \vec{\mathbf{n}}_{ik} + D_{ik} \tag{7}
$$

where f_{ik} is the inviscid numerical flux on the face ik with neighboring cells i and k, $\vec{\mathbf{n}}_{ik}$ is the area-weighted normal of the face ik, and D_{ik} is the dissipation operator.

Numerical dissipation is added for stability and resolving shocks. A matrix dissipation scheme is used. It is constructed from the undivided Laplacian and biharmonic operators:

$$
D_{ik} = -\frac{1}{2}|A_{ik}|\left[\varepsilon_{ik}^{(2)}(Q_k - Q_i) - \varepsilon_{ik}^{(4)}(L_k - L_i)\right]
$$

$$
L_i = \sum_k (Q_k - Q_i)
$$

where

$$
\varepsilon_i^{(2)} = \sum_k \kappa_2 \frac{|p_k - p_i|}{p_k + p_i}
$$

and

$$
\varepsilon_i^{(4)} = \max\left(0, \kappa_4 - \varepsilon_i^{(2)}\right) \tag{8}
$$

where ε_{ik} is calculated by averaging the values from the two neighboring cells i and k. Two parameters κ_2 and κ_4 control the addition of second- and fourth-difference dissipation. A pressure switch selects the second-difference operator in the presence of shocks, while the fourth-difference operator is used in areas of smooth flow. The Laplacian operator is denoted as L, and $|A|$ is the absolute value of the inviscid flux Jacobian. Small eigenvalues in the Jacobian may occur near stagnation points and sonic points using this approach. This affects convergence and can be avoided by introducing two parameters V_l and V_n, as described by Swanson and Turkel.[24] Values of $\kappa_2 = 2$, $\kappa_4 = 0.1$, $V_l = V_n = 0.25$ are used in the current work. A centered scheme is utilized for the diffusive-flux term. The convective terms in the turbulence model are discretized using a first-order scheme, as suggested by Spalart and Allmaras.[20]

Boundary conditions are enforced by extrapolating the solution to boundary faces and imposing the appropiate boundary conditions. They are handled in a fully-implicit manner in order to obtain fast convergence using Newton's method.

V. Newton-Krylov Algorithm

A. Newton iterations

After spatial discretization the steady-state governing equations become a system of nonlinear algebraic equations $\mathcal{R}(\mathcal{Q}) = 0$. We use Newton's method, which has the potential for fast convergence, to obtain a solution of these equations. At each Newton iteration, we need to solve a linear system for the solution update:

$$\left(\frac{\partial \mathcal{R}}{\partial \mathcal{Q}}\right)^n \Delta \mathcal{Q}^n = -\mathcal{R}(\mathcal{Q}^n) \tag{9}$$

$$\mathcal{Q}^{n+1} = \mathcal{Q}^n + \Delta \mathcal{Q}^n \tag{10}$$

This procedure is repeated until the solution satisfies some convergence tolerance.

Newton's method may not converge when the solution is far from the final result because (9) is a reasonable approximation to $\mathcal{R}(\mathcal{Q}^{n+1}) = 0$ only for small $\Delta\mathcal{Q}^n$. One alternative to improve robustness of the method is to include a time step and apply an implicit-Euler approach. The matrix of the linear system becomes:

$$\mathcal{A}(\mathcal{Q}^n) = \frac{\mathbf{\Omega}}{\Delta t^n} + \left(\frac{\partial \mathcal{R}}{\partial \mathcal{Q}}\right)^n \tag{11}$$

where $\mathbf{\Omega}/\Delta t^n$ represents a diagonal matrix of cell volumes divided by local time steps. When the time steps are increased towards infinity, Newton's method is approached. This provides some flexibility to control robustness versus efficiency during the solution process.

B. The linear system

The linear system that arises in the Newton iterations is large and sparse for practical problems. In addition, the matrix is non-symmetric due to the hyperbolic nature of the Navier-Stokes equations. Krylov subspace methods can be used to solve this class of problems. In particular, the generalized minimum residual method (GMRES) developed by Saad and Schultz[25] is found to be effective for aerodynamic applications. This method has the property of minimizing the 2-norm of the residual over all vectors in the Krylov subspace. A new search direction is constructed every iteration and is added to the subspace, thus progressively improving the solution. On the other hand, more search directions incur extra memory and computational costs. For large problems, this limits the maximum number of iterations that can be used. The restarted version of the algorithm is one alternative to reduce memory usage. However, the solution may stagnate for indefinite matrices. A more effective way to reduce the number of iterations is to use preconditioning.

Since (9) is only an approximation to zero the residual at the next iteration, complete solving of the system is found to be unnecessary to obtain quadratic convergence.[26] An inexact Newton method can be utilized which leads to efficient algorithms by avoiding oversolving of the linear system. The linear system is solved until the solution satisfies a tolerance specified by a parameter η_n:

$$||\mathcal{R}(\mathcal{Q}^n) + \mathcal{A}(\mathcal{Q}^n)\Delta\mathcal{Q}^n|| \leq \eta_n ||\mathcal{R}(\mathcal{Q}^n)|| \tag{12}$$

The choice of this parameter is a compromise between the accuracy of the update and efficiency. Choosing η_n too small will result in over-solving of the linear system, which slows down the algorithm.

The GMRES algorithm allows a matrix-free implementation; the matrix of the linear system is not required explicitly. The matrix-vector product can be calculated using finite differences:

$$\mathcal{A}v \simeq \frac{\mathcal{R}(\mathcal{Q} + \epsilon v) - \mathcal{R}(\mathcal{Q})}{\epsilon} + \frac{\mathbf{\Omega}}{\Delta t} v \tag{13}$$

This allows quadratic convergence of Newton's method because the matrix of the linear system is a complete linearization of the residual vector. Moreover, this approach reduces memory usage and avoids some difficulties during linearization. We use a matrix-free stepsize of:

$$\epsilon||v|| = \sqrt{10^{-10}} \tag{14}$$

following recent results from Chisholm and Zingg.[18]

C. Preconditioning

Preconditioning transforms the linear system (written as $\mathcal{A}x = b$) to one which has the same solution, but is easier to solve by an iterative solver. This reduces the number of inner iterations required to solve the system. The right-preconditioned GMRES algorithm is based on solving

$$\mathcal{A}\mathcal{M}^{-1}u = b,\; u = \mathcal{M}x \tag{15}$$

where $\mathcal{M}$ is the preconditioner. The matrix $\mathcal{A}\mathcal{M}^{-1}$ should have a better condition number than the original matrix $\mathcal{A}$. In practice, an iterative solver will perform efficiently if the eigenvalues of $\mathcal{A}\mathcal{M}^{-1}$ are clustered around unity. An effective preconditioner $\mathcal{M}$ is chosen so that $\mathcal{M}^{-1}$ approximates $\mathcal{A}^{-1}$, while $\mathcal{M}^{-1}$ is easy to compute. This operation is performed every outer iteration.

Pueyo and Zingg[13] have constructed a preconditioner which works well for many aerodynamic flows. It is based on an incomplete-lower-upper factorization (ILU(p)) of an approximate Jacobian after the reverse Cuthill-McKee reordering of the unknowns. The parameter p controls the amount of fill. Increasing its value results in more accurate factors with extra storage and computational costs. The approximate Jacobian is constructed by a linearization of the flow equations with only nearest-neighbor contributions. This improves the diagonal dominance of the matrix, and was found by Pueyo and Zingg to be more effective than the complete Jacobian. The coefficient of the dissipation term is calculated using

$$\varepsilon_p^{(2)} = \varepsilon^{(2)} + \sigma\varepsilon^{(4)} \tag{16}$$

with a parameter σ, where $\varepsilon^{(2)}$ and $\varepsilon^{(4)}$ are the coefficients of the dissipation term as defined in (8). The subscript p denotes the preconditioner. Chisholm and Zingg[17] have extended the approximate Jacobian to incorporate the matrix-dissipation scheme. They suggest two parameters $V_{l,p}$ and $V_{n,p}$ to avoid overly small diagonal elements in the matrix. Hence the blend of scalar and matrix dissipation can be altered in the approximate Jacobian used to form the preconditioner. Values typically used are $V_{l,p} = V_{n,p} = 0.6$.

Besides the aforementioned preconditioner based on a level-of-fill strategy, factorization with a threshold strategy (ILUT) is also considered in this work. Pueyo and Zingg compared the two preconditioners and found that the level-of-fill approach is more efficient for two-dimensional flow computations. The threshold strategy is reconsidered in this work since it provides more control over the number of nonzero entries in the preconditioner, which is important in three dimensions. The threshold strategy ILUT(l,τ) is based on dropping elements in the factorization according to their magnitude rather than their locations. The drop tolerance τ determines the elements to be neglected, while l controls how many elements are kept.

D. Preconditioning of the viscous term

The baseline viscous term is calculated by:

$$\left(\int_{\partial\Omega} \mathbf{G}\cdot\hat{\mathbf{n}}dS\right)_i \simeq \sum_k \mathbf{G}_{ik}\cdot\vec{\mathbf{n}}_{ik} \tag{17}$$

where $\mathbf{G}_{ik} = \mathbf{G}(Q_{ik},\nabla Q_{ik})$ is the viscous flux on a face ik, with neighboring cells i and k. ∇Q is the gradient of the flow variables. This is calculated using:

$$\nabla Q_{ik} = \frac{1}{2}\left(\nabla Q_i + \nabla Q_k\right) \tag{18}$$

where

$$\nabla Q_i \simeq \frac{1}{\Omega_i}\sum_k Q_{ik}\vec{\mathbf{n}}_{ik} \tag{19}$$

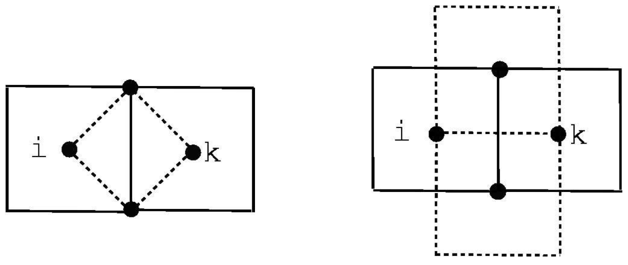

Figure 1. Calculation of the spatial derivatives by integration over (a) a diamond path, (b) a source-grid cell.

and

$$Q_{ik} = \frac{1}{2}(Q_i + Q_k) \tag{20}$$

where Ω_i is the volume of cell i. Based on this formulation, the viscous term produces a stencil involving the next-to-nearest neighboring terms. The inclusion of these terms leads to a significant increase of nonzero elements in the matrix for three-dimensional cases. This causes storage and factorization to become prohibitive. This phenomenon is not as detrimental in two dimensions.

A study of several viscous operators that lead to a reduced stencil is performed. The goal is to develop a preconditioner which is adequate for fast convergence at an affordable cost. The first approach constructs a matrix with a complete linearization but neglects contributions from next-to-nearest neighbors, i.e. by setting

$$\frac{\partial R_i}{\partial Q_{kk}} = 0 \tag{21}$$

where kk is a next-to-nearest neighbor of cell i. This approach only involves the nearest neighboring terms. It is referred as "distance-1 preconditioning" in the rest of the study.

The second approach approximates the gradient using an approximate-difference formula as suggested in references:[27,28]

$$\nabla Q_{ik} \cdot \hat{\mathbf{n}}_{ik} \simeq \frac{Q_k - Q_i}{l_{ik}} \tag{22}$$

where l_{ik} is the distance between the centroids of cells i and k, and $\hat{\mathbf{n}}_{ik}$ is the unit normal of face ik. This approach is efficient, and it has the same stencil as distance-1 preconditioning. However, the approximation is only exact on regular grids.

The third approach calculates the gradient on a face by integrating over a diamond-shaped control volume as developed by Coirier.[29] The path of integration is illustrated in Figure 1(a) for a square grid. This approach requires the knowledge of the variables at face vertices, which can be approximated by a simple averaging procedure from surrounding grid nodes, or a linearity-preserving weighting function as developed by Holmes and Connell.[30] The former is used in this work due to a lower cost, while the latter has a potential to obtain a more accurate viscous operator. This approach leads to the same stencil on triangular grids, but it has a larger stencil on structured grids when compared to the previous two methods.

The fourth approach calculates the gradient on a face by integrating over control volumes on the source grid.[29] This is illustrated in Figure 1(b), again for a square grid. This approach has the same stencil as the diamond-path approach. Extension of the above gradient calculations to hybrid unstructured grids in three dimensions is straightforward.

E. Time-stepping strategy

Our startup strategy utilizes an implicit-Euler approach by introducing a time step as given in (11). This improves both the stability of the nonlinear iterations and the conditioning of the linear system and thus results in a more robust procedure. On the other hand, the time step affects the convergence rate. Therefore, it is important to choose a time step that is both robust and efficient.

For the mean-flow equations, the local time step following Pulliam[31] is utilized:

$$\Delta t_{flow} = \frac{\Delta t_{ref}}{1 + \sqrt{\Omega^{-1}}} \tag{23}$$

Case	M_∞	$\alpha°$	Re	Geometry	Grid size
1	0.8395	3.06	11.7×10^6	ONERA M6	179,000
2	0.8395	3.06	11.7×10^6	ONERA M6	480,000
3	0.5	0.0	3.0×10^6	DLR-F6	431,000

Table 1. Flow conditions, geometry and grid size.

where Ω is the local cell volume. One way to calculate the reference time step Δt_{ref} is to follow the switched evolution relaxation approach from Mulder and van Leer:[32]

$$\Delta t_{ref} = \alpha ||\mathcal{R}||_2^{-\beta} \tag{24}$$

where $||\mathcal{R}||_2$ is the residual norm, and α and β are parameters. The idea is to increase the time step in inverse proportion to the residual norm, thus approaching Newton's method as the residual converges to zero. Other choices include the use of a constant value or a geometric series. These seem to be better choices for the startup stages due to their flexibility.

The turbulence model requires small time steps during the startup stage. Otherwise, unphysical negative values of $\tilde{\nu}$ will occur and cause the solution to become unstable. However, the use of small time steps slows down the convergence rate. Spalart and Allmaras[20] suggested the use of an M-type Jacobian matrix to prevent $\tilde{\nu}$ to become negative at a penalty on the convergence rate. Chisholm and Zingg[17] provide an alternative approach using a spatially varying time step during the start-up phase. This approach attempts to prevent negative $\tilde{\nu}$ by locally reducing the time step. It allows larger time steps to be used elsewhere in the domain. Moreover, a matrix-free approach can be utilized due to the lack of modification in the Jacobian matrix. This provides an effective startup strategy.

Chisholm and Zingg's time step can be summarized as follows:

$$\Delta t_{turb} = \begin{cases} \Delta t_{flow} & \text{if } |\delta_e| < \delta_m \\ \\ |\Delta t_{limit}| & \text{otherwise} \end{cases} \tag{25}$$

where δ_e is an estimate of the solution update, and $\delta_m = r\tilde{\nu}$ is the maximum allowable change specified by a parameter r. A typical value of r is 0.3. When the estimate exceeds the allowable value, the time step is reduced to Δt_{limit}. Otherwise, the mean-flow time step is utilized. The estimate is determined by applying Newton's method to the local uncoupled turbulence equation:

$$J_D \delta_e = -R \tag{26}$$

where J_D is the Jacobian entry on the diagonal in the turbulence model equation, and R is the right-hand side of the turbulence equation. The limiting time step is determined such that it reduces the estimate to the allowable value, i.e. $|\delta_e| = \delta_m$. This can be calculated solving the following equation for Δt_{limit}:

$$\left(\frac{\Omega}{\Delta t_{limit}} + J_D \right) \delta_m = -R \tag{27}$$

Further details about the local time step can be found in the original work by Chisholm and Zingg.[18]

VI. Results

Three turbulent cases are studied. The first two are transonic flows over a wing. The third case is a subsonic flow over a wing-body configuration. Flow conditions are summarized in Table 1. The cases are assumed to be fully turbulent. All cases are run on a single 1 GHz alpha EV68 processor at the high-performance advanced computing facility in the University of Toronto Institute for Aerospace Studies.

A. Grid generation

The ICEMCFD grid generator is utilized to generate the grids for the test cases. Prism layers are generated by extruding 15 layers of prism elements from the surface mesh using a growth ratio of 1.5. The off-wall

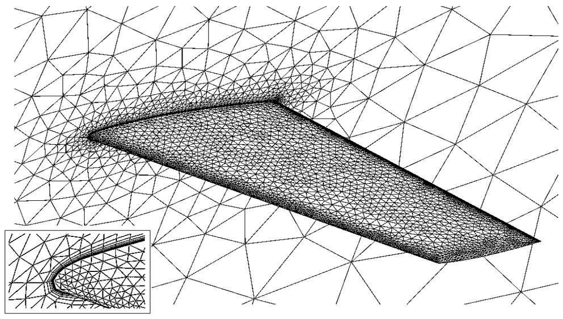

Figure 2. ONERA M6 wing grid with 179,000 nodes.

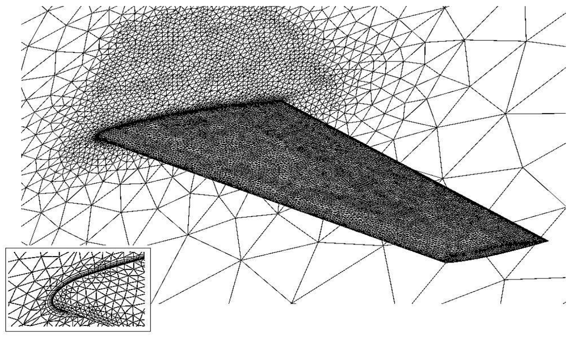

Figure 3. ONERA M6 wing grid with 480,000 nodes.

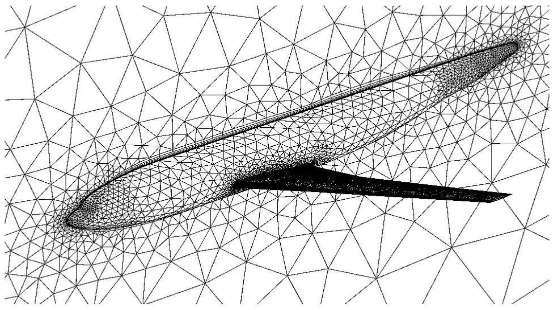

Figure 4. DLR-F6 wing-body grid with 431,000 nodes.

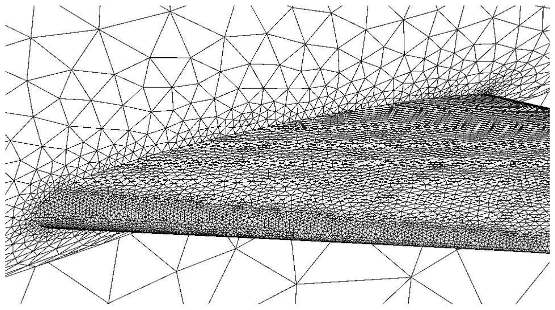

Figure 5. Close-up view at the wing-body junction.

spacing is 10^{-6} times the chord at the wing root. The far-field boundary is specified at 12 wing-root chords from the wing. It is located at 12 times the length of the fuselage from the wing-body configuration.

The geometry and grid size are summarized in Table 1. Figure 2 shows the grid for Case 1, with a close-up of the leading edge at the wing root. The grid has 179,000 nodes consisting of both tetrahedral and prismatic cells. Figure 3 shows the grid for Case 2. It is a finer grid with 480,000 nodes. The wing surface as well as the volume region above the wing are refined to provide better resolution of the shock wave. Figure 4 shows the grid with 431,000 nodes for Case 3. A close-up of the wing-body junction is shown in Figure 5. None of these grids is expected to be sufficiently fine to achieve a low numerical error in drag.

B. Solver parameters

The linear system is solved using a matrix-free non-restarted version of GMRES with 50 Krylov vectors. A linear system tolerance of $\eta = 10^{-2}$ is used in this work, based on a study given in a later part of the paper. The preconditioner is ILU(1) based on an approximate Jacobian matrix after the reverse Cuthill-McKee reordering of the unknowns. Values of $\sigma = 10$, $V_{l,p} = V_{n,p} = 0.6$ are utilized in the approximate Jacobian.

Startup is initiated using a first-order scalar scheme before switching to the matrix-dissipation scheme. Switching is triggered when the mean-flow residual converges to 10^{-4}. The first-order scheme is defined with $\varepsilon^{(2)} = 1/4$, $\varepsilon^{(4)} = 0$, and $V_l = V_n = 1$, where $\varepsilon^{(2)}$ and $\varepsilon^{(4)}$ are the coefficients of the dissipation term as given in (8).

One set of time step parameters is used for the three cases in this work. We use $\Delta t_{ref} = 1$ for the first three iterations. After that, Δt_{ref} is set to 20 and the value is doubled every 5 iterations. To prevent the solution from becoming unstable with too large a time step, the solution update is checked every Newton iteration. If nonphysical flow quantities are encountered, (i.e. negative pressure or density), then the recent solution update is rejected and Δt_{ref} for the next iteration is halved. A similar safeguarding mechanism is

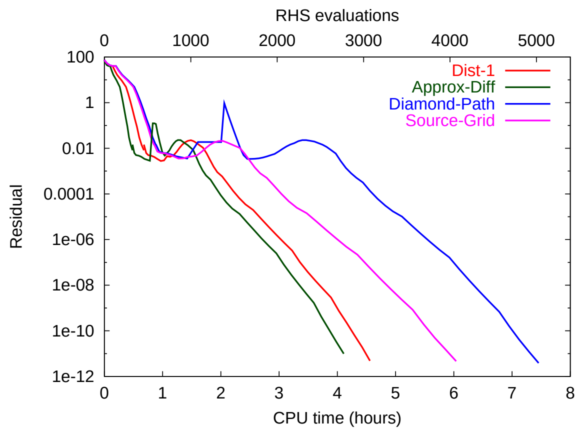

Figure 6. Case 1 convergence histories using different viscous calculations in the preconditioner. The baseline viscous calculation is used on the right-hand side.

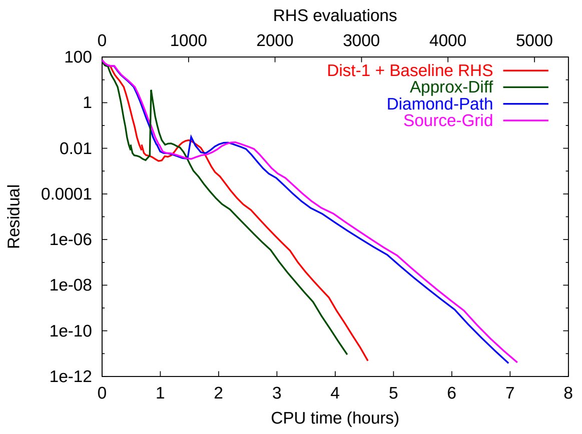

Figure 7. Case 1 convergence histories using various viscous calculations in the preconditioner and on the right-hand side. The approximate-difference, diamond-path, and source-grid approaches are used both in the preconditioner and on the right-hand side.

Case	Preconditioner	RHS	o-it	i-it	CPU time	CPU/i-it
1A	Dist-1	Baseline	65	1386	4.56	0.0033
1B	Approx-Diff	Baseline	69	1373	4.11	0.0030
1C	Diamond-Path	Baseline	83	1522	7.46	0.0049
1D	Source-Grid	Baseline	63	1265	6.04	0.0048
1E	Approx-Diff	Approx-Diff	64	1272	4.21	0.0033
1F	Diamond-Path	Diamond-Path	63	1266	6.97	0.0055
1G	Source-Grid	Source-Grid	63	1273	7.12	0.0056

Table 2. Case 1 convergence statistics using different viscous calculations. o-it denotes outer iterations. i-it denotes inner iterations. CPU time is shown in hours.

used in the work by Smith et al.[27] The same time step sequence is used for the first-order stage as well as the matrix-dissipation stage.

A nonzero initial solution of $\tilde{\nu} = 10\nu_\infty$ is used for the turbulence model, as suggested by Chisholm and Zingg.[17]

C. Preconditioning of the viscous term

Figure 6 depicts the convergence histories for Case 1 using different viscous preconditioning as described in a previous section. This includes distance-1 preconditioning, the approximate-difference formula, the diamond-path stencil, and the source-grid approach. These formulations are applied only to the preconditioner. The same baseline viscous calculation as given in (17) and (18) is used on the right-hand side; thus these cases converge to the same solution. The number of outer and inner iterations, computational cost, and cost per inner iteration are summarized in Table 2.

Convergence to 10^{-12} using distance-1 preconditioning is obtained in 4.5 hours or the equivalent of 3,000 residual evaluations. It requires 65 outer and 1,386 inner iterations in total. The approximate-difference preconditioner has a slightly lower cost per inner iteration. Convergence using this approach is slightly faster. On the other hand, the diamond-path and the source-grid preconditioners have higher costs per inner iteration. This is because the two preconditioners have more nonzero elements, which leads to slower convergence and higher memory usage. It can be concluded in Figure 6 that the four methods suggested are feasible alternatives, and the first two approaches are more efficient.

The above results were computed using the various approximations only in the approximate Jacobian ma-

RHS	C_l	C_d
Baseline	0.2647	0.01492
Approx-Diff	0.2654	0.01488
Diamond-Path	0.2659	0.01492
Source-Grid	0.2653	0.01487

Table 3. Lift and drag coefficients for Case 1 using different viscous-term calculations on the right-hand side.

Preconditioner	Storage	i-it	t_f	t_s	$t_f + t_s$
ILU(0)	1.0	33	10	226	236
ILU(1)	2.0	24	29	173	202
ILU(2)	3.8	12	96	101	197
ILU(3)	6.4	10	269	106	375
ILU(4)	9.8	9	638	121	759
ILUT(10^{-3},20)	1.3	42	548	370	918
ILUT(10^{-3},80)	2.3	21	1,176	219	1,395
ILUT(10^{-3},160)	3.3	15	2,065	197	2,262
ILUT(10^{-5},20)	1.4	42	2,802	397	3,199
ILUT(10^{-5},80)	2.9	20	7,365	251	7,616
ILUT(10^{-5},160)	4.6	14	14,971	230	15,201

Table 4. Memory, inner iterations, and CPU time in seconds to reduce the inner residual by two orders of magnitude for different preconditioners. t_f is the time to factorize the matrix, and t_s is the time to solve the system.

trix used to form the preconditioner. Next we consider using these approximations on the right-hand side as well. This affects both accuracy and convergence. The distance-1 approach cannot be used on the right-hand side. The convergence histories are shown in Figure 7. The number of iterations and computational costs are summarized in Table 2. Comparing the costs per inner iteration between Cases 1B and 1E in the table shows that the approximate-difference right-hand side is slightly more expensive than the baseline viscous calculation. The baseline viscous calculation is inexpensive as it only involves two face-based operations as given in (18) and (19). Similar comparisons between Cases 1C and 1F, as well as 1D and 1G, show the diamond-path and the source-grid right-hand side calculations are also more expensive. Convergence using distance-1 preconditioning with the baseline right-hand side is comparable to the other three approaches in Figure 7. It can be inferred that the effect of neglecting the next-to-nearest-neighboring terms in the preconditioner is small in this case.

The computed lift and drag coefficients are tabulated in Table 3 for the different viscous right-hand side calculations. For this particular case and mesh, all four techniques give very similar force coefficients. In principle, the diamond-path and source-grid approaches are more accurate than the baseline approach. The approximate-difference technique is prone to inaccuracy when the line joining the centroids of cells i and k is not perpendicular to the face ik. For the remaining studies in this paper, the baseline approach is used on the right-hand side with the distance-1 approach in the approximate Jacobian used for preconditioning.

D. Incomplete factorization

The drop-tolerance strategy ILUT is studied and compared to ILU(p) preconditioning. Table 4 tabulates memory, effectiveness and cost to reduce the linear residual by two orders-of-magnitude for several preconditioners. The study is performed on Case 1. The linear system that arises when the non-linear residual is 10^{-4} is studied. In the table, i-it is the number of inner iterations, t_f is the time to factorize the matrix, and t_s is the time to solve the system. Storage is the number of nonzeros in the factors divided by that in

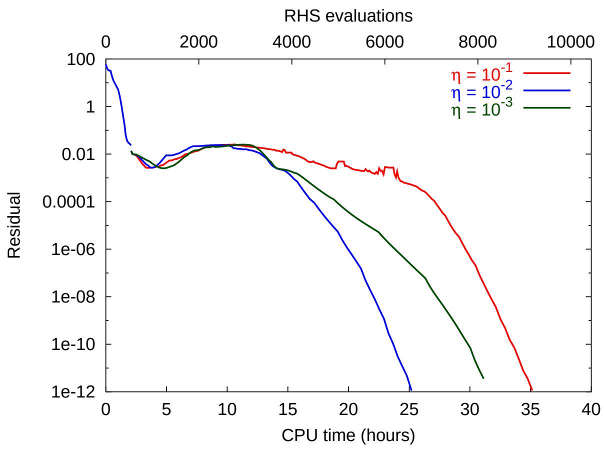

Figure 8. Case 2 convergence histories using different tolerances in the linear solver.

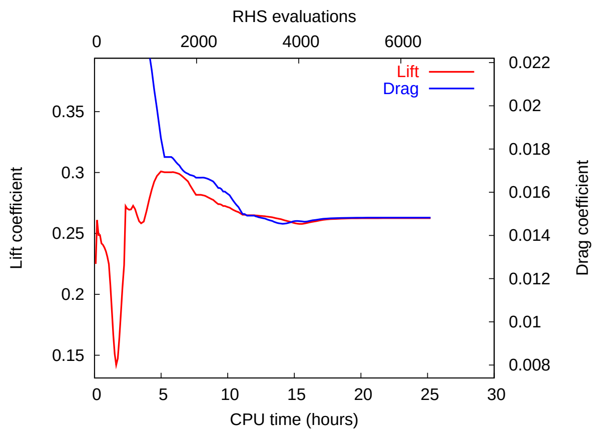

Figure 9. Convergence of lift and drag coefficients for Case 2.

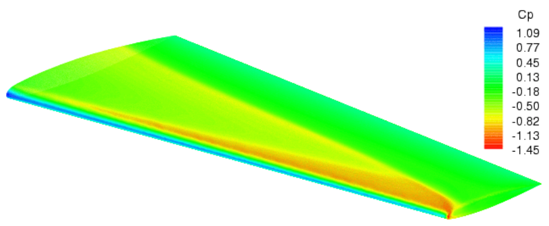

Figure 10. Pressure contours over the ONERA M6 wing at $M_\infty = 0.8395$, $\alpha = 3.06°$, and Re $= 11.7 \times 10^6$.

the approximate Jacobian matrix.

The ILU(p) preconditioner at different levels of fill p is studied. The number of inner iterations decreases as p is increased, showing that the preconditioner becomes more effective. The solving time t_s decreases as p is increased up to $p = 2$. It increases slightly beyond that. This is because the triangular back-solves become more expensive when p increases. At the same time, both storage and factorization cost increase with p. Considering the total cost as given by $t_f + t_s$, both $p = 1$ and $p = 2$ are good choices in this case, while $p = 1$ has a lower storage requirement. The storage of the factors can be a major contribution to the memory usage for three-dimensional applications. Hence ILU(1) is a good choice in three dimensions.

The ILUT preconditioner is studied at different fill parameters l and drop tolerances τ. It should be noted that the definition of l is different from p. While p indicates the level of fill, l limits the number of nonzero entries in a row. It represents the number of scalar nonzero entries in addition to the original matrix. Using a drop tolerance of 10^{-3}, the number of inner iterations and solving time decrease as l is increased, while storage and factorization cost increase with l. The use of a lower drop tolerance of 10^{-5} is found to produce a preconditioner with a similar level of effectiveness but with a significant increase in factorization cost. It can be concluded in the table that ILU(p) is much more efficient than ILUT in this case.

Convergence criterion	CPU time (hours)
0.5% of C_L	17.2
0.1% of C_L	18.8
0.01% of C_L	20.6
0.5% of C_D	16.9
0.1% of C_D	18.6
0.01% of C_D	20.5

Table 5. Convergence data for the lift and drag coefficients for Case 2.

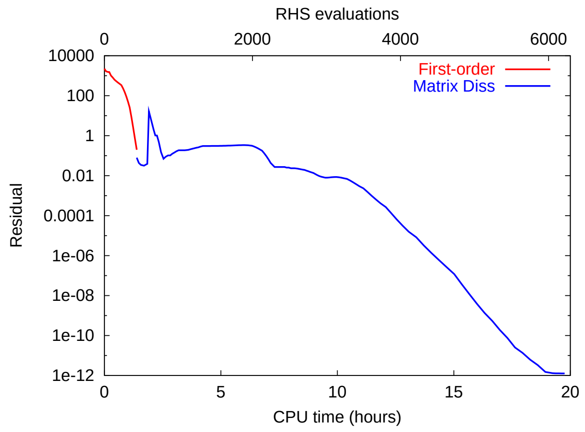

Figure 11. Case 3 convergence history.

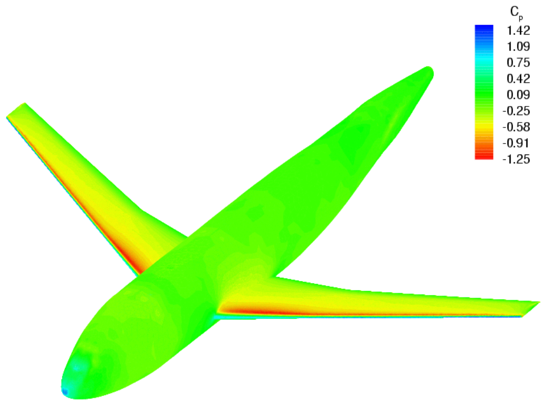

Figure 12. Pressure contours over the DLR-F6 wing-body configuration at M_∞ = 0.5, $\alpha = 0°$, and Re = 3 $\times$ 10^6.

E. Convergence results

Figure 8 shows the convergence for Case 2 using different linear system tolerances. ILU(1) is used as the preconditioner. Convergence to 10^{-12} is obtained in 25 hours or the equivalent of 6,000 residual evaluations for this half-million-node grid using a linear system tolerance of $\eta = 10^{-2}$. It requires 148 outer and 2,300 inner iterations. The use of a larger inner tolerance of 10^{-1} is found to produce a longer startup stage with an increased number of outer iterations, while a smaller inner tolerance of 10^{-3} leads to slower convergence with respect to CPU time resulting from an increased number of inner iterations.

Convergence of lift and drag coefficients with $\eta = 10^{-2}$ is given in Figure 9. The time required to converge the force coefficients to some specified tolerances is summarized in Table 5. It requires 17 hours to converge to within 0.5% of the converged lift and drag coefficients, which are 0.263 and 0.0148 respectively. Figure 10 shows the pressure contours over the wing. The pressure coefficients at different wingspan locations are compared to experimental data in Figure 13. Reasonable agreement is obtained, but the solution is not grid converged.

Figure 11 shows the convergence for the third case over the wing-body configuration with the same parameters and $\eta = 10^{-2}$. Convergence to 10^{-12} is obtained in 20 hours with the equivalent of 6,000 residual evaluations. It requires 165 outer and 2,000 inner iterations in total. The ratio of residual evaluations to inner iterations for this case is higher than the previous case. This is because more ILU factorizations are performed due to an increase in the number of outer iterations. The pressure contours over the wing-body configuration are shown in Figure 12.

VII. Conclusions

A Newton-Krylov algorithm is presented for turbulent aerodynamic flows on three-dimensional unstructured grids. Residual convergence to 10^{-12} for grids with a half-million nodes can be obtained in 20-25 hours on a single processor.

The inclusion of the next-to-nearest neighboring terms in the viscous operator causes preconditioning to become impractical for three-dimensional applications. Four approaches are suggested as alternatives and are found to be viable options. Distance-1 viscous preconditioning in conjunction with the baseline distance-2 viscous calculation on the right-hand side is selected based on both efficiency and accuracy considerations. The ILU(p) and ILUT preconditioners are studied; the former is found to be more efficient.

Current results have motivated further research to improve the efficiency of the current algorithm. Future work includes further investigation of startup strategies. The algorithm will also be extended to parallel computers to further reduce computational time. The improved algorithm will be applied to computations on finer grids to produce grid converged solutions.

VIII. Acknowledgments

This research was supported by Bombardier Aerospace and an OGS grant of the Government of Ontario. The authors would like to thank Prof. J. V. Lassaline for many useful discussions.

References

[1]Johnson, F. T., Tinoco, E. N., and Yu, N. J., "Thirty Years of Development and Application of CFD at Boeing Commercial Airplanes, Seattle," *AIAA Paper 2003-3439*, 2003.

[2]Nelson, T. E. and Zingg, D. W., "Fifty Years of Aerodynamics: Successes, Challenges, and Opportunities," *CAS Journal*, Vol. 50, No. 1, 2004, pp. 61–84.

[3]Lee-Rausch, E. M., Frink, N. T., Mavriplis, D. J., Rausch, R. D., and Milholen, W. E., "Transonic Drag Prediction on a DLR-F6 Transport Configuration Using Unstructured Grid Solvers," *AIAA Paper 2004-0554*, 2004.

[4]May, G., van der Weide, E., Jameson, A., Sriram, and Martinelli, L., "Drag Prediction of the DLR-F6 Configuration," *AIAA Paper 2004-0396*, 2004.

[5]Luo, H., Baum, J. D., and Löhner, R., "High-Reynolds Number Viscous Flow Computations Using an Unstructured-Grid Method," *AIAA Paper 2004-1103*, 2004.

[6]Brodersen, O. and Stürmer, A., "Drag Prediction of Engine-Airframe Interference Effects Using Unstructured Navier-Stokes Calculations," *AIAA Paper 2001-2414*, 2001.

[7]Knoll, D. A. and Keyes, D. E., "Jacobian-free Newton-Krylov methods: a Survey of Approaches and Applications," *Journal of Computational Physics*, Vol. 193, 2004, pp. 357–397.

[8]Venkatakrishnan, V. and Mavriplis, D. J., "Implicit Solvers for Unstructured Meshes," *Journal of Computational Physics*, Vol. 105, 1992, pp. 83–91.

[9]Barth, T. J. and Linton, S. W., "An Unstructured Mesh Newton Solver for Compressible Fluid Flow and its Parallel Implementation," *AIAA Paper 95-0221*, 1995.

[10]Nielsen, E. J., Anderson, W. K., Walters, R. W., and Keyes, D. E., "Application of Newton-Krylov Methodology to a Three-Dimensional Unstructured Euler Code," *AIAA Paper 95-1733*, 1995.

[11]Anderson, W. K., Rausch, R. D., and Bonhaus, D. L., "Implicit/Multigrid Algorithms for Incompressible Turbulent Flows on Unstructured Grids," *Journal of Computational Physics*, Vol. 128, 1996, pp. 391–408.

[12]Blanco, M. and Zingg, D. W., "Fast Newton-Krylov Method for Unstructured Grids," *AIAA Journal*, Vol. 36, No. 4, 1998, pp. 607–612.

[13]Pueyo, A. and Zingg, D. W., "Efficient Newton-Krylov Solver for Aerodynamic Computations," *AIAA Journal*, Vol. 36, No. 11, 1998, pp. 1991–1997.

[14]Geuzaine, P., Lepot, I., Meers, F., and Essers, J. A., "Multilevel Newton-Krylov Algorithms for Computing Compressible Flows on Unstructured Meshes," *AIAA Paper 99-3341*, 1999.

[15]Geuzaine, P., "Newton-Krylov Strategy for Compressible Turbulent Flows on Unstructured Meshes," *AIAA Journal*, Vol. 39, No. 3, 2000, pp. 528–531.

[16]Nemec, M. and Zingg, D. W., "Newton-Krylov Algorithm for Aerodynamic Design Using the Navier-Stokes Equations," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1146–1154.

[17]Chisholm, T. and Zingg, D. W., "A Newton-Krylov Algorithm for Turbulent Aerodynamic Flows," *AIAA Paper 2003-0071*, 2003.

[18]Chisholm, T. and Zingg, D. W., "Start-up Issues in a Newton-Krylov Algorithm for Turbulent Aerodynamic Flows," *AIAA Paper 2003-3708*, 2003.

[19]Manzano, L. M., Lassaline, J. V., Wong, P., and Zingg, D. W., "A Newton-Krylov Algorithm for the Euler Equations Using Unstructured Grids," *AIAA Paper 2003-0274*, 2003.

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

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

[22]Ashford, G. A., *An Unstructured Grid Generation and Adaptive Solution Technique for High Reynolds Number Compressible Flows*, Ph.D. thesis, University of Michigan, 1996.

[23]Mavriplis, D. J. and Venkatakrishnan, V., "A Unified Multigrid Solver for the Navier-Stokes Equations on Mixed Element Meshes," *AIAA Paper 95-1666*, 1995.

[24]Swanson, R. C. and Turkel, E., "On Central-Difference and Upwind Schemes," *J. Comp. Phys.*, Vol. 101, 1992, pp. 292–306.

[25]Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimum Residual Algorithm For Solving Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Computing*, Vol. 7, 1986, pp. 856–869.

[26]Eisenstat, S. C. and Walker, H. F., "Choosing the Forcing Terms in an Inexact Newton Method," *SIAM J. Sci. Comput.*, Vol. 17, No. 1, 1996, pp. 16–32.

[27]Smith, T. M., Hooper, R. W., Ober, C. C., Lorber, A. A., and Shadid, J. N., "Comparison of Operators for Newton-Krylov Method for Solving Compressible Flows on Unstructured Meshes," *AIAA Paper 2004-0743*, 2004.

[28]Mavriplis, D. J., "On Convergence Acceleration Techniques for Unstructured Meshes," *AIAA Paper 98-2966*, 1998.

[29]Coirier, W. J., *An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*, Ph.D. thesis, University of Michigan, 1994.

[30]Holmes, D. G. and Connell, S. D., "Solution of the 2D Navier-Stokes Equations on Unstructured Adaptive Grids," *AIAA Paper 89-1932*, 1989.

[31]Pulliam, T. H., "Efficient Solution Methods for the Navier-Stokes Equations," Tech. rep., Lecture Notes for the von Kármán Inst. for Fluid Dynamics Lecture Series: Numerical Techniques for Viscous Flow Computation in Turbomachinery Bladings, Brussels, Belgium, Jan. 1986.

[32]Mulder, W. A. and van Leer, B., "Experiments with Implicit Upwind Methods for the Euler Equations," *Journal of Computational Physics*, Vol. 59, 1985, pp. 232–246.

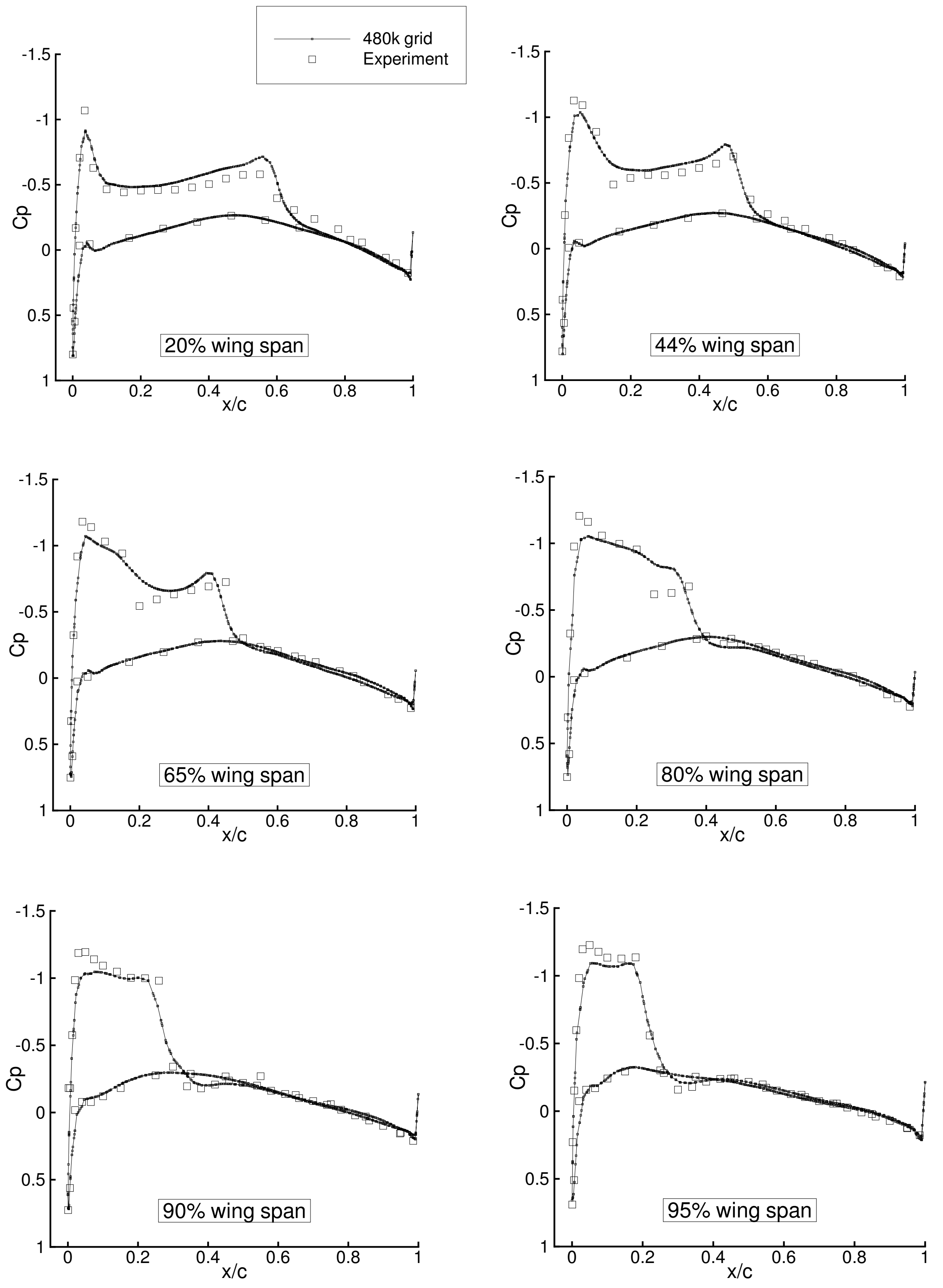

Figure 13. Comparison between experimental and computed pressure coefficients at different spanwise locations for the ONERA M6 wing at $M_\infty = 0.8395$, $\alpha = 3.06°$, and $Re = 11.7 \times 10^6$.