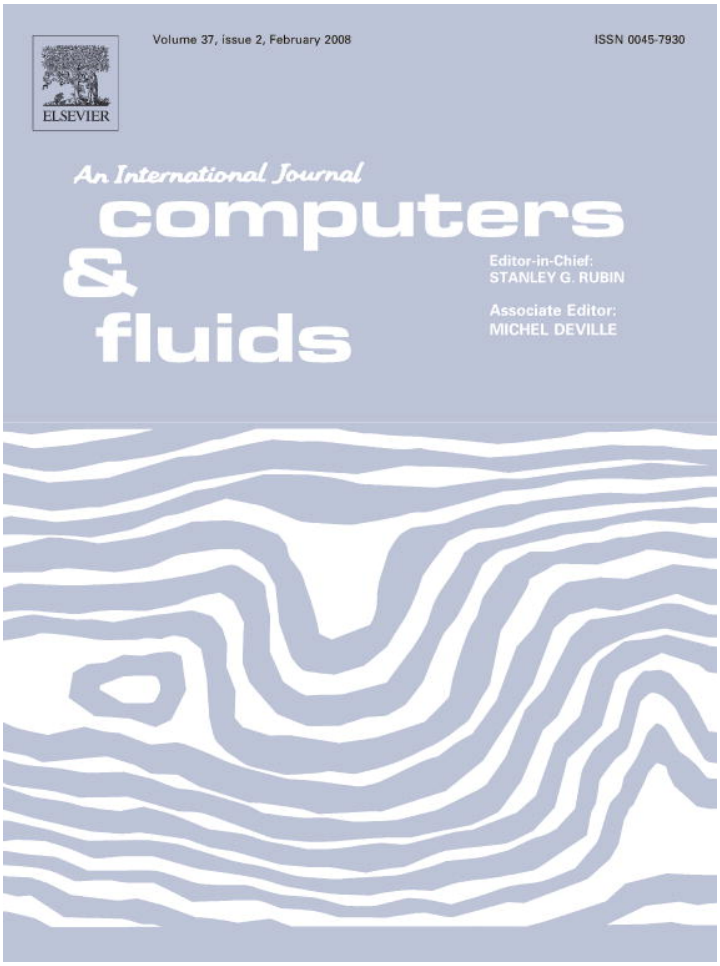

Available online at www.sciencedirect.com
ScienceDirect
Computers & Fluids 37 (2008) 107–120
computers & fluids
www.elsevier.com/locate/compfluid

Three-dimensional aerodynamic computations on unstructured grids using a Newton–Krylov approach

Peterson Wong, David W. Zingg *

University of Toronto Institute for Aerospace Studies, 4925 Dufferin Street, Toronto, Ontario, Canada M3H 5T6

Received 29 May 2006; received in revised form 24 October 2006; accepted 29 April 2007
Available online 21 June 2007

Abstract

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. Several approximate viscous operators that involve only the nearest neighboring terms are studied to reduce the cost of preconditioning. The performance of the algorithm is demonstrated through numerical studies of the ONERA M6 wing and the DLR-F6 wing-body configuration. A ten-order-of-magnitude residual reduction for the wing and wing-body configurations can be obtained with a computing cost equivalent to 5500 and 8000 function evaluations, respectively, on grids with a half million nodes.

1. Introduction

After many years of development, computational fluid dynamics has become an important aerodynamic design tool [1,2]. The current technology is capable of solving viscous compressible flows over complete aircraft configurations. Two Drag Prediction Workshops were organized to assess the accuracy of current solvers when applied to these flows [3–7]. Some issues arise in these studies, including inconclusive grid refinement results and variation among solutions obtained using different codes. These are expensive computations due to the use of fine grids, which are necessary to accurately resolve the flow features such as shock waves and boundary layers around the aircraft geometry. A typical computation on a grid with a few million nodes requires hours to obtain a solution on multiple processors. It is an area of research to improve the efficiency of algorithms to reduce computational time.

The Newton–Krylov method is an efficient method for solving the Navier–Stokes equations [8]. The method has a fast convergence rate when the solution is close to the final result. A fully converged solution is thus more obtainable, which is beneficial to numerical optimization in avoiding objective function uncertainties caused by inexact flow solves. Moreover, Newton's method is implicit; the convergence rate is less sensitive to high-aspect-ratio cells, which are typical in viscous applications. Newton–Krylov algorithms have been developed by many researchers [9–20]. Blanco and Zingg [14] compared approximate-Newton, standard Newton, and matrix-free Newton methods on triangular grids. A fast solver is developed using a matrix-free inexact-Newton approach together with an approximate-Newton startup strategy. Pueyo and Zingg [15] developed a preconditioned matrix-free Newton–Krylov algorithm, which converges faster and more reliably than an approximate Newton algorithm and an approximately-factored multigrid algorithm. Geuzaine et al. [16] studied mesh sequencing and multigrid preconditioning of the Newton–Krylov method. Nemec and Zingg [17] applied the Newton–Krylov method to numerical optimization using the discrete adjoint approach. Chisholm and Zingg [18] developed an efficient startup strategy for the Newton–Krylov algorithm using the Spalart–Allmaras

* Corresponding author. Tel.: +1 416 667 7709; fax: +1 416 667 7799.
E-mail address: dwz@oddjob.utias.utoronto.ca (D.W. Zingg).

0045-7930/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compfluid.2007.04.005

turbulence model. Smith et al. [19] compared several Jacobian formulations of the Newton–Krylov method on unstructured grids. Manzano et al. [20] applied the Newton–Krylov algorithm to three-dimensional inviscid flows using unstructured grids. A closely related Newton–Krylov Runge–Kutta algorithm has been successfully applied to implicit computations of unsteady flows [21].

The objective of this research is to extend the algorithm of Manzano et al. [20] to turbulent flows on hybrid unstructured grids. The use of unstructured grids is motivated by simpler grid generation about complex geometries and the possibility to use mesh adaptation to improve solution accuracy. For viscous computations, hybrid grids can be used to improve accuracy and to possibly reduce computational cost [22,23]. However, the use of hybrid grids increases the complexity of the spatial operator. In particular, the discretization of the viscous terms involves next-to-nearest neighbors. Several approximations to the viscous operator involving only nearest neighbors are compared here in the approximate Jacobian used to form the preconditioner. A detailed description of the algorithm is followed by studies identifying optimal values of various parameters. The algorithm is then applied to solve the flow over well-known wing and wing-body configurations using several different meshes. The algorithm presented here is based on the ideas presented in [14,15,17,18] and [20]. The key original contribution here is the extension of the algorithm to three-dimensional turbulent flows on unstructured meshes.

2. 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. They are written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega} Q\,\mathrm{d}V + \int_{\partial\Omega} \mathbf{F}\cdot\hat{\mathbf{n}}\,\mathrm{d}S = \int_{\partial\Omega} \mathbf{G}\cdot\hat{\mathbf{n}}\,\mathrm{d}S \tag{1}$$

where Ω is an arbitrary control volume, $\partial\Omega$ is the boundary of the control volume, and $\hat{\mathbf{n}}$ is the unit normal vector at the boundary pointing outward. Here, Q is the set of conservative flow variables (density ρ, momentum components $\rho u, \rho v, \rho w$, total energy ρE), $\mathbf{F}$ is the inviscid flux tensor, and $\mathbf{G}$ is the flux tensor associated with viscosity and heat conduction. They are written as

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

$$\mathbf{G} = \begin{bmatrix} \vec{0} \\ \tau_{xx}\hat{\mathbf{i}} + \tau_{yx}\hat{\mathbf{j}} + \tau_{zx}\hat{\mathbf{k}} \\ \tau_{xy}\hat{\mathbf{i}} + \tau_{yy}\hat{\mathbf{j}} + \tau_{zy}\hat{\mathbf{k}} \\ \tau_{xz}\hat{\mathbf{i}} + \tau_{yz}\hat{\mathbf{j}} + \tau_{zz}\hat{\mathbf{k}} \\ \boldsymbol{\tau}\cdot\vec{\mathbf{V}} + k\nabla T \end{bmatrix} \tag{3}$$

where $\vec{\mathbf{V}}$ is the velocity vector, and $\boldsymbol{\tau}\cdot\vec{\mathbf{V}}$ denotes the work done on the fluid due to viscosity. It is given by

$$\begin{aligned}\boldsymbol{\tau}\cdot\vec{\mathbf{V}} = {} & (u\tau_{xx} + v\tau_{xy} + w\tau_{xz})\hat{\mathbf{i}} + (u\tau_{yx} + v\tau_{yy} + w\tau_{yz})\hat{\mathbf{j}} \\ & + (u\tau_{zx} + v\tau_{zy} + w\tau_{zz})\hat{\mathbf{k}}\end{aligned} \tag{4}$$

The viscous stresses are given by the Stokes relation for a Newtonian fluid:

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu(\nabla\cdot\vec{\mathbf{V}})\delta_{ij} \tag{5}$$

where δ_{ij} is the Kronecker delta function. The heat flux vector is given by Fourier's law of heat conduction: $\mathbf{q} = -k\nabla T$. 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 μ. For turbulent flows the eddy viscosity, computed as described in the following section, is added to the viscosity. Pressure p for a perfect gas is written in terms of the conservative flow variables as

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho(u^2 + v^2 + w^2)\right) \tag{6}$$

3. Turbulence modeling

The Favre-averaged Navier–Stokes equations are solved for turbulent flows. The Reynolds-stress tensor that arises from time averaging is modeled with an eddy-viscosity term using the Boussinesq approximation. The eddy viscosity is modeled using the Spalart–Allmaras turbulence model [24]. In differential form, it is written as

$$\begin{aligned}\frac{\partial\tilde{\nu}}{\partial t} + (\vec{\mathbf{V}}\cdot\nabla)\tilde{\nu} = {} & c_{b1}(1 - f_{t2})\tilde{S}\tilde{\nu} \\ & + \frac{1}{\sigma}(\nabla\cdot((\nu + \tilde{\nu})\nabla\tilde{\nu}) + c_{b2}(\nabla\tilde{\nu})^2) \\ & - \left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right)\left(\frac{\tilde{\nu}}{d}\right)^2 + f_{t1}\Delta U^2\end{aligned} \tag{7}$$

where ΔU is the norm of the velocity difference between a field point and the trip. 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, diffusion, destruction, and trip terms respectively. Production of eddy viscosity is proportional to a vorticity-like term $\widetilde{S}$, 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 from 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 trip term models laminar-to-turbulent transition. Transition locations are specified by the user. The flow can be assumed to be fully turbulent by setting the trip functions f_{t1} and f_{t2} to zero. Closure coefficients σ, c_{b1}, c_{b2}, and c_{w1} are the same as those given by Spalart and Allmaras [25]. 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 [26] proposed a modification to $\widetilde{S}$ in the production term. The modification is found to produce better numerical properties [27] and is adopted in the current work.

4. Spatial discretization

The spatial discretization follows the finite-volume approach of Mavriplis and Venkatakrishnan [28] for hybrid unstructured grids. A cell-vertex approach is used; the flow variables are stored at grid vertices. Control volumes are constructed around the vertices using a median dual approach. The spatial terms in the governing equations are approximated by integrating fluxes over control volume boundaries. The inviscid flux at a face is given by

$$f_{ik} \simeq \frac{1}{2}(\mathbf{F}(\mathcal{Q}_i) + \mathbf{F}(\mathcal{Q}_k)) \cdot \vec{\mathbf{n}}_{ik} + D_{ik} \tag{8}$$

where f_{ik} is the inviscid numerical flux at face ik, which has neighboring cells i and k, and $\vec{\mathbf{n}}_{ik}$ is the area-weighted normal of face ik. Here, $\mathcal{Q}_i$ is the cell-averaged solution at cell i, D_{ik} is the numerical dissipation operator, and $\mathbf{F}(Q)$ is given by Eq. (2).

Artificial dissipation is added to the spatial discretization to resolve shocks and to provide numerical stability. The dissipation scheme follows the approach of Jameson and Mavriplis [29] using undivided Laplacian and biharmonic operators. The matrix dissipation scheme of Swanson and Turkel [30] is used to improve accuracy. The dissipation term is written as

$$D_{ik} = -\frac{1}{2}|A_{ik}|\left(\varepsilon^{(2)}_{ik}(\mathcal{Q}_k - \mathcal{Q}_i) - \varepsilon^{(4)}_{ik}(L_k - L_i)\right)$$
$$L_i = \sum_k (\mathcal{Q}_k - \mathcal{Q}_i)$$

where

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

and

$$\varepsilon^{(4)}_i = \max\left(0, \kappa_4 - \varepsilon^{(2)}_i\right) \tag{9}$$

where ε_{ik} is calculated by averaging from the two neighboring cells. 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 resulting scheme is second-order accurate except in the vicinity of shocks, where it becomes locally first-order accurate. The Laplacian operator is denoted as L, and A is the inviscid flux Jacobian: $A = (\partial\mathbf{F}/\partial Q) \cdot \vec{\mathbf{n}}$. The Jacobian can have small eigenvalues near stagnation points and sonic points, which can be avoided using two parameters V_l, V_n, as described by Swanson and Turkel [30]. A scalar dissipation scheme is recovered with $V_l = V_n = 1$, while $V_l = V_n = 0$ leads to a pure matrix-dissipation scheme. Values of $\kappa_2 = 2, \kappa_4 = 0.1, V_l = V_n = 0.25$ are used in the current study. The convective terms in the turbulence model are discretized using a first-order scheme, as suggested by Spalart and Allmaras [24].

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

4.1. Viscous discretization

The viscous flux at a face is written as

$$g_{ik} \simeq \mathbf{G}(\mathcal{Q}_{ik}, \nabla\mathcal{Q}_{ik}) \cdot \vec{\mathbf{n}}_{ik} \tag{10}$$

where g_{ik} is the viscous flux at face ik, ∇Q is the gradient of the flow variables, and $\mathbf{G}(Q, \nabla Q)$ is given by Eq. (3). The flow variables and gradient at a face can be obtained by averaging from neighboring cells. i.e.,

$$\mathcal{Q}_{ik} \simeq \frac{1}{2}(\mathcal{Q}_i + \mathcal{Q}_k) \tag{11}$$

$$\nabla\mathcal{Q}_{ik} \simeq \frac{1}{2}(\nabla\mathcal{Q}_i + \nabla\mathcal{Q}_k) \tag{12}$$

The gradient at a cell can be computed using

$$\nabla\mathcal{Q}_i \simeq \frac{1}{\Omega_i}\sum_{ik} \mathcal{Q}_{ik}\vec{\mathbf{n}}_{ik} \tag{13}$$

following Barth and Jespersen [31]. Here, the sum is performed over all faces of cell i, and Ω_i is the volume of the cell. The above viscous formulation has a non-compact stencil involving next-to-nearest neighboring terms. It has a 13-point stencil on regular quadrilateral grids. It is referred to as the simple averaging approach in this study.

Alternative approaches to calculate the face gradient $\nabla\mathcal{Q}_{ik}$ are studied to obtain viscous formulations with a smaller stencil. The first approach approximates the directional gradient normal to the face by finite differences from neighboring cells:

$$\nabla\mathcal{Q}_{ik} \cdot \hat{\mathbf{n}}_{ik} \simeq \left(\frac{\mathcal{Q}_k - \mathcal{Q}_i}{l_{ik}}\hat{\mathbf{l}}_{ik}\right) \cdot \hat{\mathbf{n}}_{ik} \tag{14}$$

This approach is used by Smith et al. [19] and Mavriplis [32]. Here, $\hat{\mathbf{l}}_{ik}$ is the unit vector from the centroid of cell i to the centroid of cell k, 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 has a compact stencil involving only neighboring cells. It has a five-point stencil on regular quadrilateral grids. However, this approximation is only accurate on regular grids. The method is inaccurate when the line joining the centroids of cells i and k is not in the normal direction of face ik. This occurs on irregular grids, especially on stretched triangular grids.

The second approach computes the face gradient by integration over a diamond shaped control volume, as studied by Coirier [33]. The gradient is given by

$$\nabla \mathcal{Q}_{ik} \simeq \frac{1}{\Omega_\psi} \sum_j \mathcal{Q}_j \vec{\mathbf{n}}_j \tag{15}$$

where Ω_ψ is the volume of the integrating region, and $j = 1, \ldots, N_j$ are the faces bounding the integrating region. The region is illustrated in Fig. 1a for a square grid. It is defined by the closed volume bounded by the centroids of cells i and k and the vertices of face ik. This extends to three dimensions and to hybrid grids. The flow variables at face j can be obtained by averaging from the face vertices (e.g. v_1, k in Fig. 1a). This requires knowledge of the flow variables at vertices of face ik (e.g. v_1), which can be approximated by averaging from neighboring cells (i.e. i, k, a_1, a_2).

The diamond path approach has a nine-point stencil on regular quadrilateral grids. On the other hand, it has the same stencil as the approximate difference approach on triangular grids.

The third approach computes the face gradient by integration over control volumes on the primal grid [33]. This is illustrated in Fig. 1b, again for a square grid. The gradient is obtained by averaging from face vertices:

$$\nabla \mathcal{Q}_{ik} \simeq \frac{1}{N_v} \sum_v \nabla \mathcal{Q}_v \tag{16}$$

where $v = 1, \ldots, N_v$ are the vertices of face ik. Each v has a corresponding cell on the primal grid. The gradient $\nabla \mathcal{Q}_v$ is obtained by integration over the corresponding primal-grid cell. The primal grid approach has the same stencil as the diamond path approach.

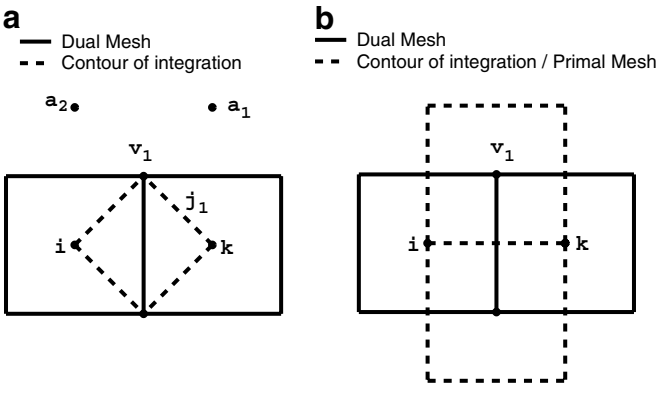

Fig. 1. Calculation of the face gradients by integration over (a) a diamond path and (b) a primal-grid cell.

5. Newton–Krylov algorithm

5.1. Newton iterations

After spatial discretization, the steady-state governing equations, including the turbulence model equations, become a system of nonlinear algebraic equations: $\mathscr{R}(\mathcal{Q}) = 0$. Newton's method, which has the potential for fast convergence, can be used to solve these equations. At each Newton iteration, a linear system is solved for the solution update $\Delta\mathcal{Q}$:

$$\left(\frac{\partial \mathscr{R}}{\partial \mathcal{Q}}\right)^n \Delta \mathcal{Q}^n = -\mathscr{R}(\mathcal{Q}^n) \tag{17}$$

$$\mathcal{Q}^{n+1} = \mathcal{Q}^n + \Delta \mathcal{Q}^n \tag{18}$$

Here, n denotes a Newton iteration, and $\partial\mathscr{R}/\partial\mathcal{Q}$ is the flow Jacobian. 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. This is because the linear system, Eq. (17), is a reasonable approximation to $\mathscr{R}^{n+1} = 0$ only for small $\Delta\mathcal{Q}^n$, when the solution is close to the final solution. Alternatively, the implicit-Euler method can be used, which is more robust. The matrix of the implicit-Euler method with local time linearization [34] is written as

$$\mathscr{A}(\mathcal{Q}^n) = \frac{\mathbf{\Omega}}{\Delta t} + \left(\frac{\partial \mathscr{R}}{\partial \mathcal{Q}}\right)^n \tag{19}$$

where $\mathbf{\Omega}/\Delta t$ represents a diagonal matrix of cell volumes divided by time steps. When the time step is increased towards infinity, Newton's method is approached from the implicit-Euler method. In the current study, the implicit-Euler approach is used to provide a robust startup strategy. Newton-type convergence is obtained using large time steps near convergence.

5.2. The linear system

The linear system that arises in the implicit Euler iterations can be written as $\mathscr{A}x = b$, where $\mathscr{A}$ is given in Eq. (19), $x = \Delta\mathcal{Q}^n$ and $b = -\mathscr{R}(\mathcal{Q}^n)$. The system is large and non-symmetric. In addition, the matrix is indefinite due to the hyperbolic nature of the Navier–Stokes equations. Krylov methods can be used to solve this class of problems. In particular, the generalized minimum residual method, GMRES, developed by Saad and Schultz [35] is found to be effective for aerodynamic applications. The method is implemented using the PETSc numerical code [36]. This method has the property of minimizing the L2-norm of the residual over all vectors in the Krylov subspace. The GMRES method computes a new search vector every iteration. The vector is added to the Krylov subspace to progressively improve the solution. However, more search directions incur higher memory and computational costs. For large problems, this limits the maximum number of iterations that can be used. The restarted version of the

algorithm can be used, where the algorithm is restarted from the most recent solution. However, for indefinite matrices, convergence of the restarted algorithm may stagnate. Alternatively, the number of iterations can be reduced in a more effective way by the use of preconditioning.

Complete solving of the linear system equation (17) is unnecessary to obtain fast convergence of Newton's method. An inexact Newton method can significantly reduce computational work by avoiding over-solving of the system [37]. The linear system can be solved until

$$\|\mathscr{R}(\mathscr{Q}^n) + \mathscr{A}(\mathscr{Q}^n)\Delta\mathscr{Q}^n\| \leqslant \eta_n\|\mathscr{R}(\mathscr{Q}^n)\| \tag{20}$$

with a tolerance parameter η_n.

A matrix-free approach can be used for Krylov methods, such as GMRES, that require Jacobian matrix–vector products but not explicit Jacobian matrices. The matrix–vector product can be approximated using a finite difference approximation to the Fréchet derivative:

$$\mathscr{A}v \simeq \frac{\mathscr{R}(\mathscr{Q}+\epsilon v) - \mathscr{R}(\mathscr{Q})}{\epsilon} + \frac{\mathbf{\Omega}}{\Delta t}v \tag{21}$$

where ϵ is a step size. The matrix-free approach 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. A stepsize of

$$\epsilon\|v\|_2 = \sqrt{10^{-10}} \tag{22}$$

is used following Chisholm and Zingg [18].

Scaling of the linear system as suggested by Chisholm [38] is found to improve convergence of the matrix-free inexact Newton method with a fully-coupled turbulence model. The scaled linear system is written as

$$\mathscr{D}_r\mathscr{A}\mathscr{D}_c^{-1}\mathscr{D}_c x = \mathscr{D}_r b \tag{23}$$

where $\mathscr{D}_r, \mathscr{D}_c$ are diagonal matrices containing scaling factors. The scaling factors can be chosen so that the coupled residual and solution vectors are better scaled, i.e. the entries are of similar magnitudes. Scaling the turbulence equation and variables by a value of 10^{-3} is considered in this work, following the approach of Chisholm [38]. Proper scaling of the equations ($\mathscr{D}_r$) ensures that both the turbulence and the mean-flow residuals are reduced in an inexact linear solve. On the other hand, scaling of the variables ($\mathscr{D}_c$) improves the accuracy of the matrix-free Jacobian approximation, which improves convergence of Newton's method.

5.3. Preconditioning

Preconditioning transforms the linear system to one which has the same solution, but is easier to solve by an iterative solver. It can dramatically reduce the number of GMRES iterations required to solve the system to a specified tolerance. Right preconditioning can be used:

$$\mathscr{A}\mathscr{M}^{-1}u = b, \quad u = \mathscr{M}x \tag{24}$$

where $\mathscr{M}$ is the preconditioner. This approach is selected since the residual is available every iteration, which is beneficial for checking convergence. The preconditioner $\mathscr{M}$ is chosen so that $\mathscr{A}\mathscr{M}^{-1}$ has a better condition number than $\mathscr{A}$, while $\mathscr{M}^{-1}$ is efficient to compute.

An incomplete lower-upper factorization preconditioner ILU(p) is used following Pueyo and Zingg [15]. This approach is found to be effective for aerodynamic flows. The factorization is based on an approximate Jacobian of the fully coupled discrete mean-flow and turbulence model residual equations after applying the reverse Cuthill–McKee reordering. This is written as

$$\mathscr{L}\mathscr{U} \simeq \widetilde{\mathscr{A}}, \quad \mathscr{M} = \mathscr{L}\mathscr{U} \tag{25}$$

where $\mathscr{L}$, $\mathscr{U}$ are the incomplete factors, and $\widetilde{\mathscr{A}}$ is the approximate Jacobian. The parameter p controls the amount of fill in the factors. Higher fill results in more effective preconditioning but with a higher memory usage and computational cost. The approximate Jacobian is constructed by a linearization of the flow equations with only second-difference dissipation. The resulting Jacobian has contributions from nearest neighbors only. This is found to be more effective than the complete Jacobian [15]. The dissipation coefficients in $\widetilde{\mathscr{A}}$ are given by

$$\varepsilon_{\mathrm{p}}^{(2)} = \varepsilon^{(2)} + \sigma\varepsilon^{(4)} \tag{26}$$

where σ is a parameter which controls the diagonal dominance of the matrix, $\varepsilon^{(2)}$, $\varepsilon^{(4)}$ are the dissipation coefficients as given in Eq. (9), and the subscript p denotes values used in the preconditioner. As a note, the same matrix dissipation parameters V_l, V_n are used in the preconditioner and on the right-hand side.

Approximate viscous calculations are studied for preconditioning. The simple averaging approach in Eq. (12), and the diamond-path and primal-grid approaches in Eqs. (15) and (16), have a non-compact stencil involving next-to-nearest neighboring terms. The inclusion of these terms leads to more non-zeros in $\widetilde{\mathscr{A}}$, which increases cost and memory usage of preconditioning, especially in three dimensions. Neglecting these terms is considered to maintain a compact stencil in $\widetilde{\mathscr{A}}$.

Besides the aforementioned preconditioner ILU(p) based on a level-of-fill strategy, an alternative approach based on a threshold strategy is also considered in this work. This approach provides more control over the number of nonzero entries in the preconditioner, which is important in three dimensions. The threshold approach ILUT(P, 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 P controls how many elements are kept.

5.4. Time-stepping strategy

This section describes the choice of the time step, Δt in Eq. (19), of the implicit Euler method. A time-stepping

strategy which combines the use of small time steps during startup and switches to large time steps near convergence is used to provide a robust and efficient algorithm with a Newton-type convergence. For the mean-flow equations, the local time step following Pulliam [39] is used:

$$\Delta t_{\text{flow},i} = \frac{\Delta t_{\text{ref}}}{1 + \sqrt{\Omega_i^{-1}}} \tag{27}$$

where i is the cell index, and Ω_i is the nondimensional local cell volume. A reference time step Δt_{ref} is used to control the time step globally. For the nth nonlinear iteration, the time step is written as

$$\Delta t_{\text{ref}}^n = \begin{cases} \Delta t_1 & n \leqslant n_1 \\ \alpha & n = n_1 + 1 \\ \beta \times \Delta t_{\text{ref}}^{n-1} & n \text{ is a multiple of } f \\ \Delta t_{\text{ref}}^{n-1} & \text{otherwise} \end{cases} \tag{28}$$

A small time step Δt_1 is used in the first n_1 iterations. After that, the time step is switched to a larger value α. The time step is multiplied by a factor β every f iterations. Suggested parameters are

$$n_1 = 2, \quad \Delta t_1 = 0.05, \quad \alpha = 2, \quad \beta = 2, \quad f = 5 \tag{29}$$

A safeguarding mechanism is included to prevent nonphysical solution updates caused by too large a time step. The solution update is checked every nonlinear iteration. If nonphysical flow quantities, such as negative pressure or density, are encountered, then the recent solution update is rejected and the time step of the next iteration is halved. A similar approach is used by Smith et al. [19].

The turbulence model requires small time steps during startup. Otherwise, large solution updates occur which cause the solution to become unstable. However, small time steps adversely affect convergence. Spalart and Allmaras [24] suggest the use of an M-type Jacobian matrix to prevent negative values of the turbulent solution. This improves robustness at a penalty on the convergence rate. Chisholm and Zingg [27] suggest an alternative approach. This approach attempts to prevent large updates by locally reducing the time step used in the turbulence model update only. Larger time steps are allowed elsewhere in the domain. Moreover, this approach can be used with the matrix-free method since it does not require a modification to the Jacobian matrix. The time step is written as

$$\Delta t_{\text{turb},i} = \begin{cases} \Delta t_{\text{flow},i} & \text{if} |\delta_{e,i}| < \delta_{m,i} \\ |\Delta t_{\text{limit},i}| & \text{otherwise} \end{cases} \tag{30}$$

where $\delta_{e,i}$ is an estimate of the local solution update, and $\delta_{m,i} = r\tilde{\nu}_i$ is the maximum allowable change in the turbulence solution specified by a parameter r. A typical value of r is 0.3, which allows a 30% change. When the estimate exceeds the allowable value, the time step is reduced to a small time step $\Delta t_{\text{limit},i}$. Otherwise, the mean-flow time step is used. The estimate is determined from the following scalar equation:

$$J_{\text{turb},i}\delta_{e,i} = -R_{\text{turb},i} \tag{31}$$

where $J_{\text{turb},i}$ is the diagonal entry in $\widetilde{\mathscr{A}}$ that corresponds to the local turbulence equation, and $R_{\text{turb},i}$ is the entry in the residual vector that corresponds to the local turbulence equation. The limiting time step is determined so that it reduces the estimate to the allowable value, i.e. $|\delta_{e,i}| = \delta_{m,i}$. This can be calculated by solving the following equation for $\Delta t_{\text{limit},i}$:

$$\left(\frac{\Omega_i}{\Delta t_{\text{limit},i}} + J_{\text{turb},i}\right)\delta_{m,i} = -R_{\text{turb},i} \tag{32}$$

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

6. Results

Two cases are presented. A summary of the flow conditions is given in Table 1. The grids are summarized in Table 2. Case 1 is a turbulent flow over the ONERA M6 wing [40]. The case is studied on three hybrid grids with increasing densities. The grids have prisms in the viscous region and tetrahedra elsewhere in the domain. Grid 1c is depicted in Fig. 2. Case 2 is a turbulent flow over the DLR F6 wing-

Table 1
Summary of test cases and flow conditions

Case	Geometry	M_∞	α°	Re
1	ONERA M6	0.8395	3.06	11.7×10^6
2	DLR F6	0.75	0.52	3.0×10^6

Table 2
Summary of grids

Grid	Geometry	Size	Type
1a	ONERA M6	179,758	Hybrid
1b	ONERA M6	480,775	Hybrid
1c	ONERA M6	1,124,270	Hybrid
2a	DLR F6	584,543	Hybrid
2b	DLR F6	1,293,250	Hybrid
2c	DLR F6	1,121,301	Tetrahedral

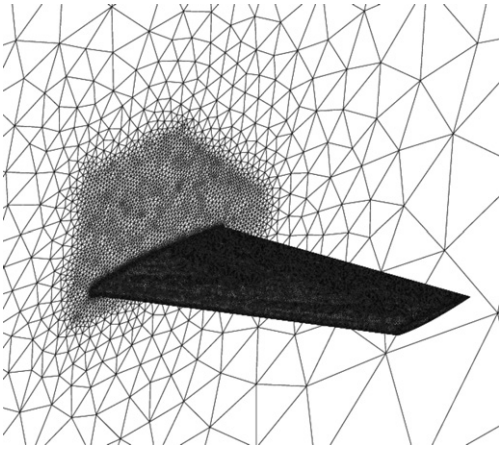

Fig. 2. Grid 1c over the ONERA M6 wing with 1,124,270 nodes (hybrid). Density volume region included above the wing surface.

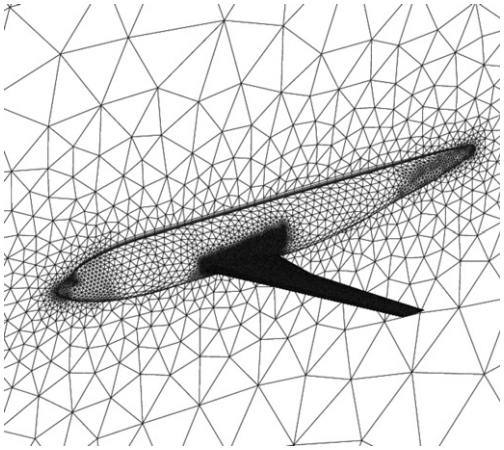

Fig. 3. Grid 2b over the DLR-F6 wing-body configuration with 1,293,250 nodes (hybrid). Density volume region included above the wing surface.

Table 3
Memory, CPU cost and effectiveness to reduce the inner residual by two orders of magnitude for different preconditioners

Preconditioner	nnzB/N	CPU-form	i-it	CPU-total
ILU(0)	10.9	10	33	236
ILU(1)	21.8	29	24	202
ILU(2)	41.4	96	12	197
ILU(3)	69.7	269	10	375
ILU(4)	106.8	638	9	759
D2-ILU(0)	45.4	139	16	350
D2-ILU(1)	134.0	1124	10	1362
D2-ILU(2)	294.5	5537	8	5856
ILUT$(20,10^{-3})$	14.2	548	42	918
ILUT$(80,10^{-3})$	25.1	1176	21	1395
ILUT$(160,10^{-3})$	36.0	2065	15	2262
ILUT$(20,10^{-5})$	15.3	2802	42	3199
ILUT$(80,10^{-5})$	31.6	7365	20	7616
ILUT$(160,10^{-5})$	50.1	14,971	14	15,201

CPU time units are in seconds.

body configuration [3,41]. Three grids are used for Case 2. Grids 2a and 2b are hybrid grids with a half million nodes and over one million nodes, respectively. Grid 2b is depicted in Fig. 3. Grid 2c is the tetrahedral grid with 1,121,301 nodes from the second Drag Prediction Workshop [3]. The cases are assumed to be fully-turbulent. The CPU times reported in this paper are based on a single 1.5 GHz Intel Itanium 2 processor.

6.1. Grid generation

The ICEM CFD grid generator is used to generate the hybrid grids 1a–1c, 2a, and 2b. For grids 1b, 1c, 2a, and 2b, a density volume region is included above the wing surface to provide a better resolution of the shock wave. This is depicted in Figs. 2 and 3. For grids 1a, 1b, and 2a, the prism layers are generated by extruding 15 layers of prism elements from the body surface mesh using a growth ratio of 1.5. For grids 1c and 2b, the prism layers are generated by extruding 26 layers of prism elements from the surface mesh. The first 15 layers from the body have a growth ratio of 1.2, and the remaining 11 layers have a growth ratio of 1.5. These parameters are comparable to those described by Lee-Rausch et al. [3]. The off-wall spacing is 2.5×10^{-6} mean aerodynamic chords for grids 1a–1c over the wing, and is 2×10^{-6} mean aerodynamic chords for grids 2a and 2b over the wing-body configuration. The far-field boundary is specified at 15 mean aerodynamic chords from the wing. It is located at 12 times the length of the fuselage from the wing-body configuration for grids 2a and 2b. These grids are not expected to be sufficiently fine to achieve a low numerical error in drag.

6.2. Preconditioning study

Table 3 tabulates the results of a study of various ILU preconditioners. The preconditioners are compared based on their effectiveness to reduce the linear residual by two orders of magnitude. The comparisons are performed for Case 1 on grid 1a (abbreviated as Case 1a in this paper) using the linear system when the nonlinear residual is 10^{-4}. The matrix $\widetilde{\mathscr{A}}$ for preconditioning is constructed as described in a previous section. The viscous terms in the matrix are calculated using the simple averaging approach given in Eq. (12) with and without the next-to-nearest neighboring contributions. The matrix neglecting these terms is denoted as a distance-1 (D1) matrix. This approach is used unless otherwise stated. The matrix including these terms is denoted as a distance-2 (D2) matrix. The viscous terms in the right-hand side $\mathscr{R}$ are computed using the simple averaging approach including the next-to-nearest neighboring terms.

The block version of the level-of-fill approach, ILU(p), is studied, where p is a parameter which controls the amount of fill. The ILU(p) approach in Table 3 is based on the distance-1 matrix. Higher fill improves the effectiveness of the preconditioner, requiring fewer GMRES iterations to solve the linear system, but with a higher computing cost and memory usage. As shown in Table 3, this approach becomes more effective when p is increased. For this case, the number of GMRES iterations, denoted as i-it in the table, is reduced from 33 to 9 when p is increased from 0 to 4. The computational work to factorize the matrix, denoted as CPU-form in the table, increases nonlinearly with p. The total cost, denoted as CPU-total, includes factorization cost and the cost of triangular back-solves $\mathscr{U}^{-1}\mathscr{L}^{-1}$. The former cost increases with p, while the latter cost increases with the number of GMRES iterations. Therefore, an optimal p can be found, in this case $p=2$. The storage of the factors, denoted as nnzB/N, also increases nonlinearly with p. This number is the average number of nonzero blocks per block row in the factors. In general, memory usage is unpredictable for the level-of-fill approach. Considering both computing cost and memory usage, the ILU(1) preconditioner is a better choice than ILU(2).

Preconditioning based on the distance-2 matrix is also examined in Table 3. This is denoted as D2-ILU(p), where the viscous terms in $\widetilde{\mathscr{A}}$ include next-to-nearest neighboring terms. For similar effectiveness, as indicated by i-it in Table 3, the distance-2 approach is more expensive and requires more memory usage than the distance-1 approach. An example is the comparison between ILU(3) and D2-ILU(1).

Finally, results for a scalar version of the threshold approach, ILUT(P,τ), are also shown in Table 3. The ILUT(P,τ) factorization is based on the distance-1 matrix. The parameter P is the number of scalar elements allowed to fill in each row in each of the factors $\mathscr{L},\mathscr{U}$, respectively. The elements in the factors that are smaller than τ in magnitude are dropped. Unlike the level-of-fill approach, memory usage of the threshold approach is predictable. For similar effectiveness in terms of inner iterations, the threshold approach has a much higher factorization cost than the level-of-fill approach, but with a similar memory usage. An example is the comparison between ILUT(80,10^{-3}) and ILU(1). The ILUT preconditioner becomes more effective when P is increased. On the other hand, the use of a smaller τ of 10^{-5} over 10^{-3} is only slightly more effective but with a much higher factorization cost. Based on these results, the level-of-fill approach ILU(p) with $p=1$ is selected.

6.3. Viscous preconditioning

Various viscous formulations are studied in the matrix $\widetilde{\mathscr{A}}$ for preconditioning. This includes the simple-averaging, approximate-difference, diamond-path, and primal-grid approaches, as described in a previous section. Except for the approximate-difference approach, these approaches have a non-compact stencil involving next-to-nearest neighboring terms. These terms can be dropped in the matrix $\widetilde{\mathscr{A}}$, as described in the previous section. This is referred to as a distance-1 (D1) approach.

Fig. 4 depicts the convergence histories using various viscous formulations in $\widetilde{\mathscr{A}}$. The study is performed for Case 1a. The simple-averaging approach is used on the right-hand side. Thus these cases converge to the same solution. Here, "CPU time in RHS evaluations" (denoted as CPU-f.e. in Table 4) is the CPU time in terms of equivalent right-hand side (RHS) evaluations, or function evaluations. It is the CPU time divided by the time required for one RHS evaluation. This unit provides a convenient measure to compare efficiency of algorithms on various platforms and grids. Convergence histories using the simple-averaging-D1, approximate-difference, diamond-path-D1, and primal-grid-D1 preconditioners are similar. These approaches have a faster convergence than the diamond-path and the primal-grid preconditioners when next-to-nearest neighbors are included. As a note, the simple-averaging approach with next-to-nearest neighbors is not included in the study because it is too expensive.

The number of outer (o-it) and inner (i-it) iterations is summarized in Table 4. The cost of the preconditioner can be measured in terms of the cost per inner iteration.

Table 4
Convergence statistics using different viscous calculations

Preconditioner	RHS	o-it	$\sum$i-it	CPU-f.e.	CPU-f.e./$\sum$i-it
Simple avg. D1	Simple avg.	92	1668	3777	2.3
Approx diff.	Simple avg.	92	1590	3591	2.3
Diamond path D1	Simple avg.	92	1592	3845	2.4
Primal grid D1	Simple avg.	95	1597	3851	2.4
Diamond path	Simple avg.	92	1504	5495	3.7
Primal grid	Simple avg.	97	1704	6017	3.5
Approx diff.	Approx Diff.	92	1593	3884	2.4
Diamond path D1	Diamond Path	92	1588	5285	3.3
Primal grid D1	Primal Grid	92	1582	5178	3.3

(CPU-f.e. denotes the CPU time normalized by the time for one RHS calculation using the simple averaging approach.)

This is denoted as CPU-f.e./$\sum$i-it in the table. The D1 preconditioners (simple-averaging-D1, diamond-path-D1, primal-grid-D1) and the approximate-difference approach have a cost of roughly 2.4 units. The diamond-path and the primal-grid approaches have a higher cost of 3.7 and 3.5 units, respectively. These two approaches have more nonzero elements in the matrix $\widetilde{\mathscr{A}}$, i.e. the next-to-nearest neighboring terms. It can be concluded from Fig. 4 and Table 4 that the D1 and the approximate-difference preconditioners have a lower cost.

The above results are computed using the various viscous formulations only in the matrix $\widetilde{\mathscr{A}}$ for preconditioning. Next we consider using these formulations on the right-hand side as well. This affects both accuracy and convergence. The next-to-nearest neighboring terms are kept on the right-hand side. These terms are dropped in the matrix $\widetilde{\mathscr{A}}$ using a D1 approach. The results are depicted in Fig. 5. The simple-averaging and the approximate-difference approaches have a faster convergence than the diamond-path and the primal-grid approaches. This is due to more expensive right-hand side calculations using the latter two approaches. The simple-averaging right-hand side formulation as given in Eq. (12) is inexpensive since the cell gradients $\nabla\mathscr{Q}_i$ are pre-calculated. These terms are

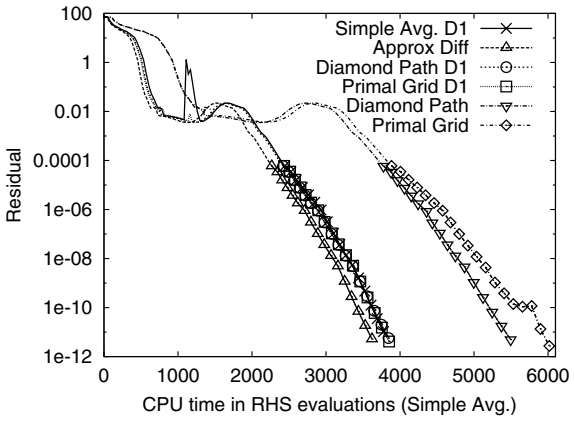

Fig. 4. A study of viscous preconditioning for Case 1a. The simple averaging approach is used on the right-hand side.

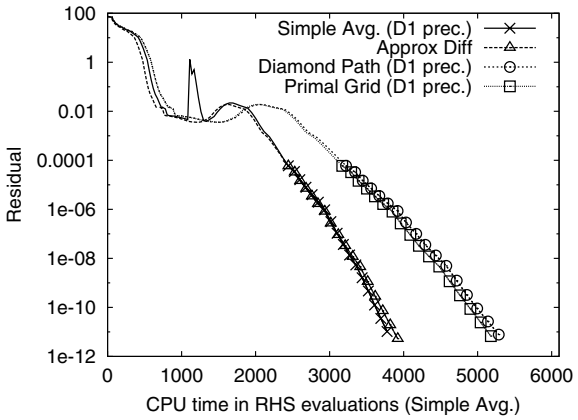

Fig. 5. A study of various viscous calculations in the preconditioner and on the right-hand side for Case 1a.

Table 5
Lift and drag coefficients using different viscous calculations on the right-hand side

RHS	C_L	C_D
Simple avg.	0.2640	0.01466
Approx diff.	0.2649	0.01462
Diamond path	0.2654	0.01466
Primal grid	0.2648	0.01461

Table 6
Convergence statistics to converge the mean-flow residual to 10^{-10}: effect of the linear system tolerance

Case	η	o-it	$\sum$i-it	CPU-f.e.
1b	10^{-1}	155	2152	5594
1b	10^{-2}	125	2546	5565
1b	10^{-3}	134	3522	7021
2a	10^{-1}	204	2126	6386
2a	10^{-2}	123	2698	5755
2a	10^{-3}	121	3138	6284

Table 7
Convergence statistics to converge the mean-flow residual to 10^{-10}: effect of scaling

Case	Scaling	o-it	$\sum$i-it	CPU-f.e.
1b	No scaling	210	2337	6755
1b	Scaling	125	2546	5565
2a	No scaling	557	3627	14,675
2a	Scaling	123	2698	5755

Table 8
Total number of GMRES iterations ($\sum$i-it) required to converge the mean-flow residual to 10^{-10} for different values of σ

σ	Case 1a	Case 1b	Case 2a
4	1928	3092	4752
6	1781	2649	3514
8	1684	2500	3043
10	1668	2546	2698
12	1689	2616	2613
14	1858	2884	3139

required to compute the source terms in the turbulence model.

The computed lift and drag coefficients are tabulated in Table 5 for the various viscous right-hand side formulations. For this particular case and mesh, all four approaches give very similar force coefficients. In principle, the diamond-path and the primal-grid approaches are more accurate than the simple-averaging approach. On the other hand, the approximate-difference approach can

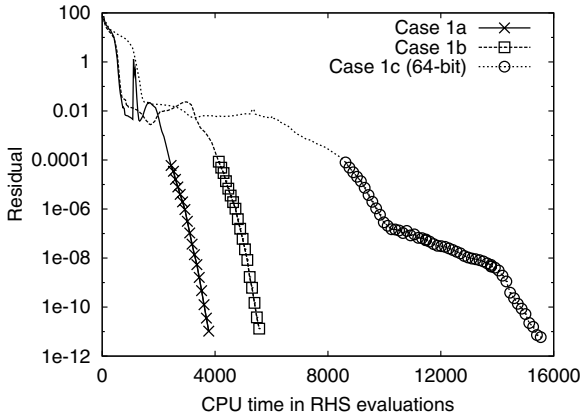

Fig. 6. Case 1 convergence histories.

Table 9
Statistics of the Newton–Krylov algorithm

Case	o-it	$\sum$i-it	$\sum$i-it/o-it	CPU-f.e.	CPU time (h)
1a	92	1668	18.1	3777	5.0
1b	125	2546	20.4	5565	20.9
1c[a]	187	4511	24.1	15,542	130.6
2a	154	4348	28.2	8568	38.1
2b[a,b]	99	2068	20.9	7812	72.4
2c[a]	298	9925	33.3	35,156	360.5

[a] Using 64-bit code.
[b] *Note*: Case 2b is not fully converged. For this case, results are shown for convergence to 10^{-6}.

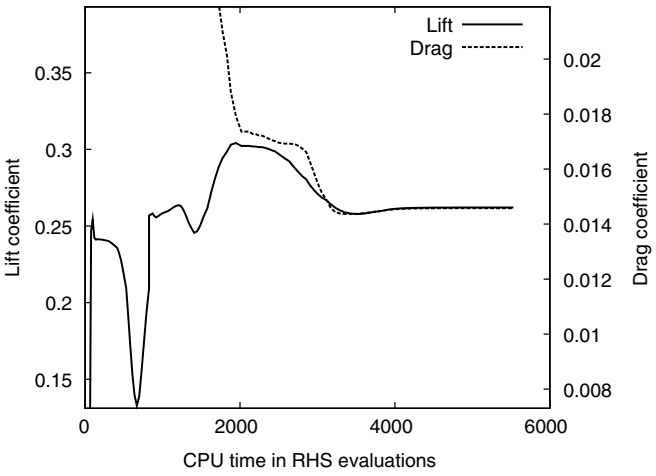

Fig. 7. Convergence of lift and drag coefficients for Case 1b.

be inaccurate on stretched triangular grids. For the remaining studies, the simple-averaging approach is used on the right-hand side. The simple averaging-D1 approach is used in $\widetilde{\mathscr{A}}$ for preconditioning.

Table 10
Convergence data for the lift and drag coefficients

Convergence criterion	CPU-f.e.
0.5% of C_L	3950
0.1% of C_L	4270
0.01% of C_L	4588
0.5% of C_D	3906
0.1% of C_D	4230
0.01% of C_D	4571
Fully converged	5565

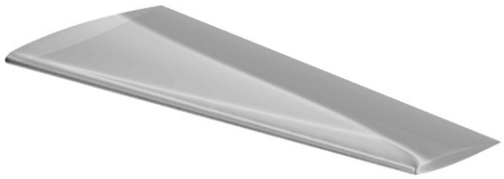

Fig. 8. Pressure contours over the ONERA M6 wing at $M_\infty = 0.8395$, $\alpha = 3.06^\circ$, and $Re = 11.7 \times 10^6$.

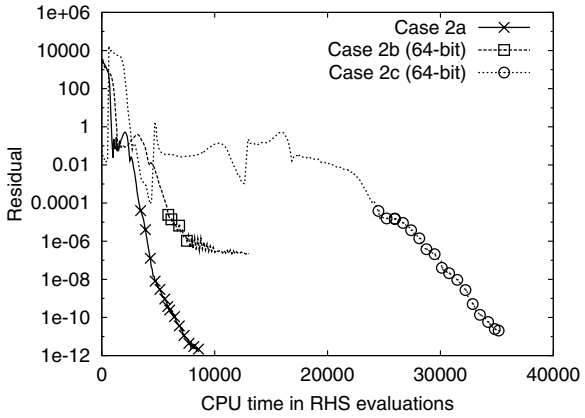

Fig. 9. Convergence for Case 2 over a wing-body configuration. Symbol spacing: 5 iterations.

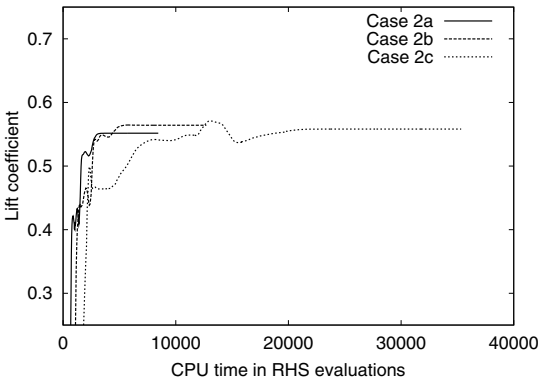

Fig. 10. Case 2 lift convergence.

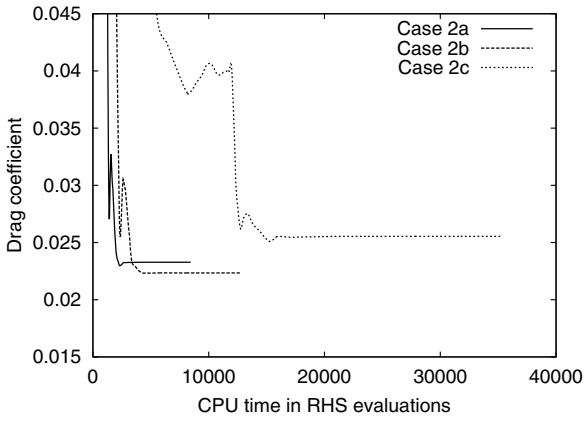

Fig. 11. Case 2 drag convergence.

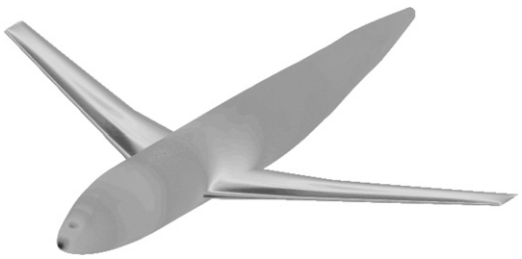

Fig. 12. Pressure contours over the DLR-F6 wing-body configuration at $M_\infty = 0.75$, $\alpha = 0.52^\circ$, and $Re = 3 \times 10^6$.

Table 11
Startup procedure for Case 2c

Stage	RHS	Prec.	Turb. model	Scaling	Max Δt_{ref}	Switching criteria
First-order	$\varepsilon^{(2)} = 1$, $\varepsilon^{(4)} = 0$	$\varepsilon_p^{(2)} = 1$	Off	Off	–	10 iter.
First-order	$\varepsilon^{(2)} = 1/4$, $\varepsilon^{(4)} = 0$	$\varepsilon_p^{(2)} = 1/2$	On	Off	–	$\mathscr{R}_f < 10^{-4}$
First-order	$\varepsilon^{(2)} = 1/4$, $\varepsilon^{(4)} = 0$	$\varepsilon_p^{(2)} = 1/2$	On	On	–	$\mathscr{R}_f < 10^{-4}$
Scalar diss.	$\kappa_2 = 2$, $\kappa_4 = 0.4$	$\sigma = 10$	On	On	16	50 iter.
Scalar diss.	$\kappa_2 = 2$, $\kappa_4 = 0.4$	$\sigma = 10$	On	On	–	30 iter.
Matrix diss.	$\kappa_2 = 2$, $\kappa_4 = 0.1$, $V_l = V_n = 0.25$	$\sigma = 10$, $V_l = V_n = 0.6$	On	On	256	70 iter.
Matrix diss.	$\kappa_2 = 2$, $\kappa_4 = 0.1$, $V_l = V_n = 0.25$	$\sigma = 10$, $V_l = V_n = 0.25$	On	On	–	–

$\mathscr{R}_f$ denotes L2-norm of the mean-flow residual.

6.4. Solver parameters

A summary of the solver parameters is given in this section. This set of parameters is used for all subsequent cases. These parameter values are effective for a broad range of problems without tuning for individual cases. A non-restarted version of GMRES with 50 Krylov vectors is used as a linear solver. An inexact-Newton approach with a linear system tolerance of $\eta = 10^{-2}$ is selected based on the results shown in Table 6. In the table, o-it denotes outer iterations, i-it denotes inner iterations, and CPU-f.e. is the CPU time in terms of equivalent function evaluations. Note that the CPU time is not proportional to the number of inner iterations. This is because the cost of preconditioning every outer iteration is a major contribution to the total cost in three dimensions. Thus, the CPU time also depends on the number of outer iterations.

Scaling is applied to the linear system as discussed in a previous section. The turbulence equation and variables are scaled by a value of 10^{-3}. This is found to improve convergence in terms of outer iterations, as shown in Table 7. Preconditioning is obtained using an incomplete factorization ILU(1) based on an approximate Jacobian matrix. A value of $\sigma = 10$ is used, where σ is the parameter defined in Eq. (26). A study of σ is summarized in Table 8. Note that the matrix dissipation parameters V_l and V_n in the

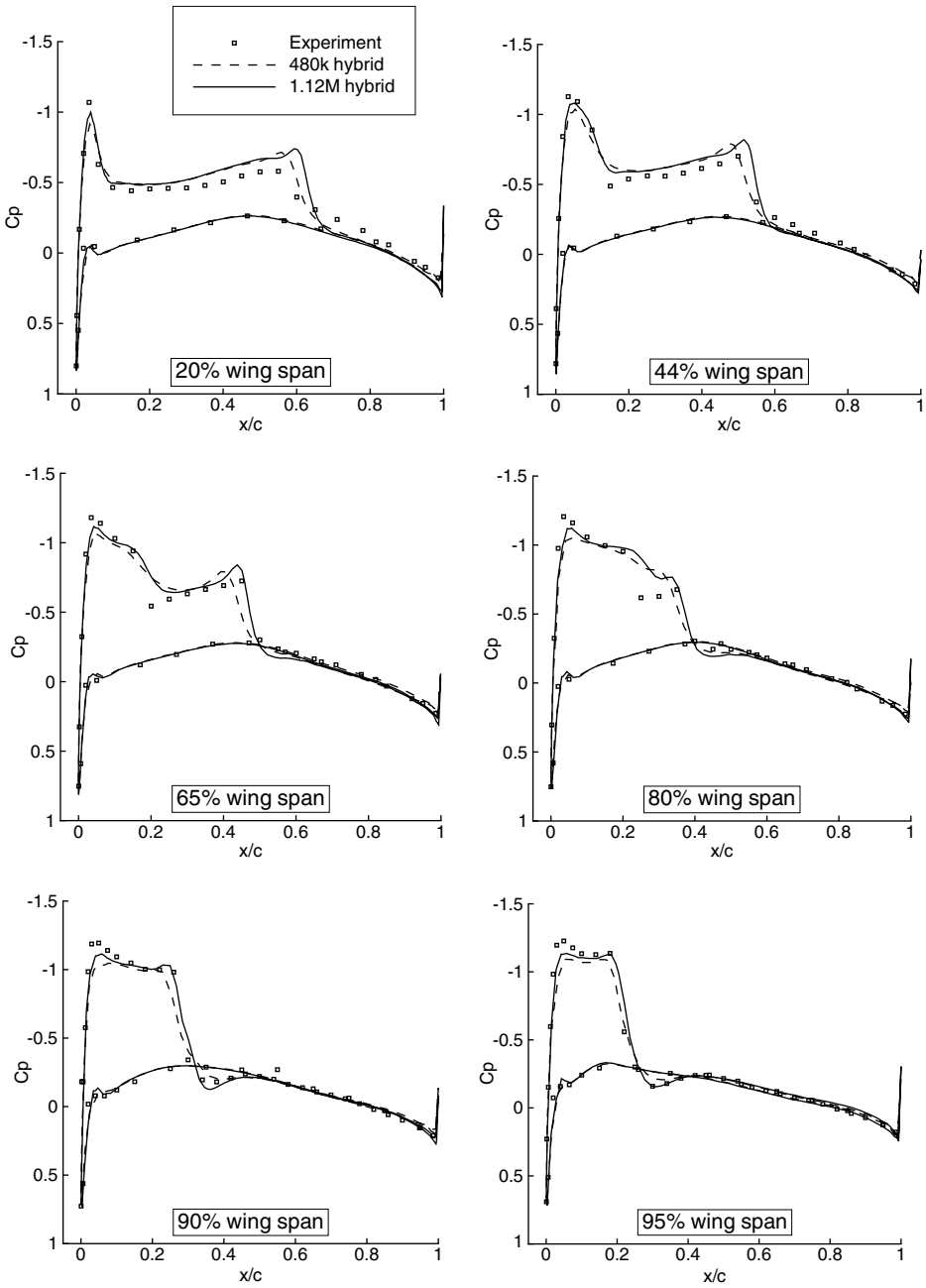

Fig. 13. Comparison with experimental pressure coefficients [40] at different wing span locations for Cases 1b and 1c.

approximate Jacobian are the same as those on the right-hand side. The simple-averaging approach is used to calculate the viscous terms on the right-hand side. The simple-averaging-D1 approach is used in the matrix $\widetilde{\mathscr{A}}$ for preconditioning. The Reverse Cuthill–McKee approach is used to reorder the matrix before factorization.

The time step sequence in Eq. (28) is used. The parameters are given in Eq. (29). A first-order discretization scheme is used before switching to matrix dissipation. This allows larger time steps to be used during startup and provides a better initial solution to Newton's method. The first-order scheme is defined with $\varepsilon^{(2)} = 1/4$, $\varepsilon^{(4)} = 0$, and $\varepsilon_p^{(2)} = \varepsilon^{(2)}$. When the mean-flow residual is reduced to 10^{-4}, the algorithm switches to the matrix-dissipation stage. The same time step sequence is used in both the first-order and the matrix-dissipation stages. A nonzero initial solution of $\tilde{\nu} = 10\nu_\infty$ is used for the turbulence model, as suggested by Chisholm and Zingg [27].

6.5. Convergence results

Fig. 6 depicts the convergence histories for Case 1 over a wing on three grids with increasing densities. The convergence statistics are given in Table 9. The solver parameters are as summarized in the previous section. For the half-million-node case (Case 1b), convergence to 10^{-10} is obtained

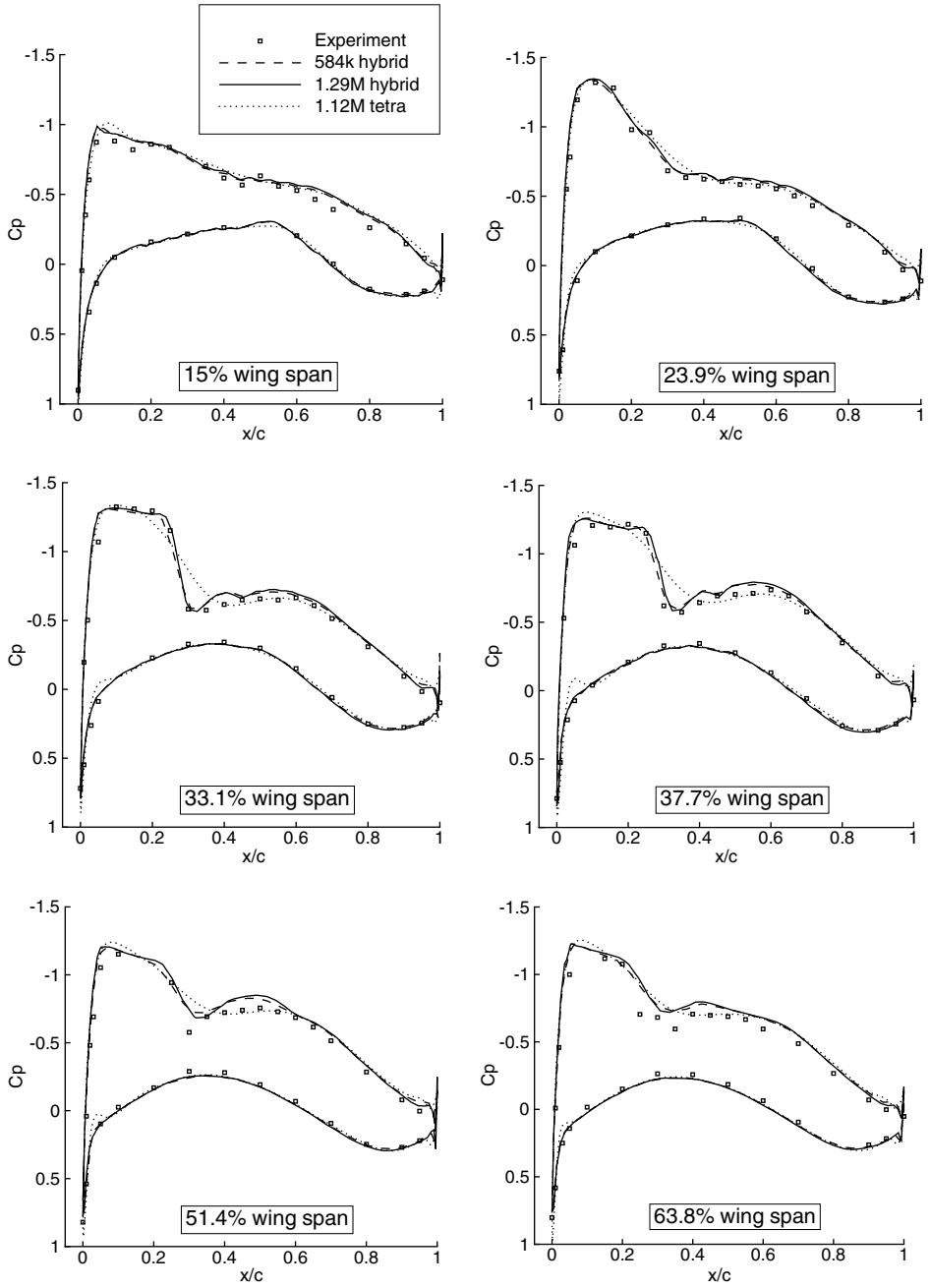

Fig. 14. Comparison with experimental pressure coefficients at different wing span locations for Cases 2a, 2b, and 2c.

in a CPU time equivalent to 5565 RHS evaluations (20.9 h). It requires 125 outer and 2546 inner iterations to converge. For the one-million-node case (Case 1c), the solution is converged using a different version of the code, which is referred to as the 64-bit code. This code uses 64-bit long integers to store memory locations, as supported by the PETSc numerical software [36]. This is necessary due to the large number of entries in the matrix for preconditioning. The 64-bit code is found to be 1.86 times slower than the original code in terms of both CPU time and CPU-f.e. This factor should be taken into account when interpreting the results.

Convergence of lift and drag coefficients for Case 1b is shown in Fig. 7. The force coefficients converge asymptotically to their final values as the residual converges to zero. The time required to converge the force coefficients to some specified tolerances is summarized in Table 10. It requires about 4000 RHS evaluations, which is about three quarters of the total time, to converge to within 0.5% of the converged lift and drag coefficients, which are 0.262 and 0.0146, respectively. Fig. 8 depicts the pressure contours over the wing on grid 1c. The pressure coefficients at different wingspan locations are compared to experimental data in Fig. 13. Reasonable agreement is obtained, but the solution is not grid converged. The solution on grid 1c has a stronger shock wave than the solution on grid 1b. The converged lift and drag coefficients for Case 1c are 0.282 and 0.0149, respectively.

Fig. 9 depicts the convergence histories for Case 2 over a wing-body configuration on different grids. The convergence statistics are given in Table 9. For the half-million-node case (Case 2a), convergence to 10^{-10} is obtained in a CPU time equivalent to 8568 RHS evaluations (38.1 h). Cases 2a and 2b are converged using the solver parameters as given in the previous section. Case 2c is converged using an alternative startup procedure as summarized in Table 11. This startup procedure allows larger time steps to be used for this case. Both Cases 2a and 2c are converged below 10^{-10} in the study. For Case 2b the residual convergence stalls just below 10^{-6}; however, the lift and drag coefficients are converged, as depicted in Figs. 10 and 11. This may be caused by poor mesh quality at the leading edge close to the wing tip. The experimental results are $C_L = 0.5$ and $C_D = 0.0295$ as given by Lee-Rausch et al. [3].

Fig. 12 depicts the pressure contours over the wing-body configuration on grid 2a. The pressure coefficients at different wingspan locations are compared to experimental data in Fig. 14. Note that the tetrahedral grid (grid 2c) does not have increased density in the shock wave region.

7. Conclusions

A Newton–Krylov algorithm is presented for the compressible Navier–Stokes equations on three-dimensional hybrid unstructured grids. Residual convergence to 10^{-10} for the ONERA M6 wing and the DLR F6 wing-body configuration can be obtained in a CPU time equivalent to 5500 and 8000 RHS evaluations, respectively, on grids with a half million nodes.

ILU preconditioning based on a distance-2 matrix including next-to-nearest neighboring terms is found to be computationally expensive in three dimensions. Distance-1 ILU(1) preconditioning is found to be relatively efficient with reasonable effectiveness and memory usage. Alternative distance-1 viscous formulations are considered for preconditioning and are found to be feasible alternatives with similar computational costs. Future work involves development of an efficient parallel implementation of the algorithm and investigation of multi-level preconditioners.

Acknowledgements

This research was supported by Bombardier Aerospace and the Government of Ontario. The authors thank J.V. Lassaline and T.T. Chisholm for many useful discussions.

References

[1] Johnson FT, Tinoco EN, Yu NJ. Thirty years of development and application of CFD at Boeing Commercial Airplanes, Seattle. AIAA Paper 2003-3439.
[2] Nelson TE, Zingg DW. Fifty years of aerodynamics: successes, challenges, and opportunities. CAS J 2004;50(1):61–84.
[3] Lee-Rausch EM, Frink NT, Mavriplis DJ, Rausch RD, Milholen WE. Transonic drag prediction on a DLR-F6 transport configuration using unstructured grid solvers. AIAA Paper 2004-0554.
[4] May G, van der Weide E, Jameson A, Sriram, Martinelli L. Drag prediction of the DLR-F6 configuration. AIAA Paper 2004-0396.
[5] Luo H, Baum JD, Löhner R. High-Reynolds number viscous flow computations using an unstructured-grid method. AIAA Paper 2004-1103.
[6] Mavriplis DJ, Levy DW. Transonic drag prediction using an unstructured multigrid solver. AIAA Paper 2002-0838.
[7] Mavriplis DJ. Grid resolution study of a drag prediction workshop configuration using the NSU3D unstructured mesh solver. AIAA Paper 2005-4729.
[8] Knoll DA, Keyes DE. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. J Comput Phys 2004;193:357–97.
[9] Venkatakrishnan V, Mavriplis DJ. Implicit solvers for unstructured meshes. J Comput Phys 1992;105:83–91.
[10] Barth TJ, Linton SW. An unstructured mesh newton solver for compressible fluid flow and its parallel implementation. AIAA Paper 95-0221.
[11] Nielsen EJ, Anderson WK, Walters RW, Keyes DE. Application of Newton–Krylov methodology to a three-dimensional unstructured Euler code. AIAA Paper 95-1733.
[12] Anderson WK, Rausch RD, Bonhaus DL. Implicit/multigrid algorithms for incompressible turbulent flows on unstructured grids. J Comput Phys 1996;128:391–408.
[13] Forsyth PA, Jiang H. Nonlinear iteration methods for high speed laminar compressible Navier–Stokes equations. Comp Fluids 1997;26(3):249–68.
[14] Blanco M, Zingg DW. Fast Newton–Krylov method for unstructured grids. AIAA J 1998;36(4):607–12.
[15] Pueyo A, Zingg DW. Efficient Newton–Krylov solver for aerodynamic computations. AIAA J 1998;36(11):1991–7.
[16] Geuzaine P, Lepot I, Meers F, Essers JA. Multilevel Newton–Krylov algorithms for computing compressible flows on unstructured meshes. AIAA Paper 99-3341.

[17] Nemec M, Zingg DW. Newton–Krylov algorithm for aerodynamic design using the Navier–Stokes equations. AIAA J 2002;40(6):1146–54.
[18] Chisholm T, Zingg DW. Start-up issues in a Newton–Krylov algorithm for turbulent aerodynamic flows. AIAA Paper 2003-3708.
[19] Smith TM, Hooper RW, Ober CC, Lorber AA, Shadid JN. Comparison of operators for Newton–Krylov method for solving compressible flows on unstructured meshes. AIAA Paper 2004-0743.
[20] Manzano LM, Lassaline JV, Wong P, Zingg DW. A Newton–Krylov algorithm for the euler equations using unstructured grids. AIAA Paper 2003-0274.
[21] Isono S, Zingg DW. A Runge–Kutta–Newton–Krylov algorithm for fourth-order implicit time marching applied to unsteady flows. AIAA Paper 2004-0433.
[22] Haselbacher A, McGuirk J, Page G. Finite volume discretization aspects for viscous flows on mixed unstructured grids. AIAA J 1999;37(2):177–84.
[23] Aftosmis M, Gaitonde D, Tavares T. Behavior of linear reconstruction techniques on unstructured meshes. AIAA J 1995;33(11):2038–49.
[24] Spalart PR, Allmaras SR. A one-equation turbulence model for aerodynamic flows. AIAA Paper 92-0439.
[25] Spalart PR, Allmaras SR. A one-equation turbulence model for aerodynamic flows. La Rech Aérospatiale 1994(1):5–21.
[26] Ashford GA. An unstructured grid generation and adaptive solution technique for high Reynolds number compressible flows. Ph.D. thesis, University of Michigan; 1996.
[27] Chisholm T, Zingg DW. A Newton–Krylov algorithm for turbulent aerodynamic flows. AIAA Paper 2003-0071.
[28] Mavriplis DJ, Venkatakrishnan V. A unified multigrid solver for the Navier–Stokes equations on mixed element meshes. AIAA Paper 95-1666.
[29] Jameson A, Mavriplis D. Finite volume solution of the two-dimensional Euler equations on a regular triangular mesh. AIAA J 1986;24(4):611–8.
[30] Swanson RC, Turkel E. On central-difference and upwind schemes. J Comput Phys 1992;101:292–306.
[31] Barth TJ, Jespersen DC. The design and application of upwind schemes on unstructured meshes. AIAA Paper 89-0366.
[32] Mavriplis DJ. On convergence acceleration techniques for unstructured meshes. AIAA Paper 98-2966.
[33] Coirier WJ. An adaptively-refined, cartesian, cell-based scheme for the Euler and Navier–Stokes equations. Ph.D. thesis, University of Michigan; 1994.
[34] Lomax H, Pulliam TH, Zingg DW. Fundamentals of computational fluid dynamics. Springer-Verlag; 2001.
[35] Saad Y, Schultz MH. GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comput 1986;7:856–69.
[36] Balay S, Buschelman K, Gropp W, Kaushik D, Knepley M, McInnes L, Smith B, Zhang H. PETSc Web page, <http://www.mcs.anl.gov/petsc>; 2001.
[37] Eisenstat SC, Walker HF. Choosing the forcing terms in an inexact Newton method. SIAM J Sci Comput 1996;17(1):16–32.
[38] Chisholm TT. A fully coupled Newton–Krylov solver with a one-equation turbulence model. Ph.D. thesis, University of Toronto; 2005.
[39] Pulliam TH. Efficient solution methods for the Navier–Stokes equations, Lecture Notes for the von Kármán Institute for fluid dynamics lecture series: numerical techniques for viscous flow computation in turbomachinery bladings. Brussels, Belgium; January 1986.
[40] Schmitt V, Charpin F. Pressure distributions on the ONERA-M6-Wing at transonic Mach numbers, experimental data base for computer program assessment. Report of the Fluid Dynamics Panel Working Group 04. AGARD AR 138; May 1979.
[41] Brodersen O, Stürmer A. Drag prediction of engine-airframe interference effects using unstructured Navier–Stokes calculations. AIAA Paper 2001-2414.