AIAA Journal
Vol. 44, No. 2, February 2006

Convergence Acceleration for Solving the Compressible Navier–Stokes Equations

Cord-Christian Rossow*
Deutsches Zentrum für Luft- und Raumfahrt, D-38108 Brunswick, Germany

A fully implicit operator for implicit smoothing of residuals in the framework of explicit Runge–Kutta time stepping and multigrid acceleration is developed. To avoid memory overhead associated with storing the complete flux Jacobians, the Jacobians are expressed in terms of Mach number to enable economic computation of all flux Jacobians during iteration. The implicit operator allows increasing Courant–Friedrichs–Lewy numbers of the basic explicit scheme to the order of 100, and it properly addresses the stiffness in the discrete equations associated with highly stretched meshes. The proposed method was applied to different cases of viscous, turbulent airfoil flow, and convergence rates ranging from 0.8 to 0.87 were achieved. Comparing the present scheme to well-tuned state-of-the-art methods using common implicit residual smoothing techniques, CPU time is better than halved.

Introduction

FAST convergence of the iterative solution method is a prerequisite for efficient use of numerical flow simulation in an industrial environment. One of the most promising general solution strategies is multigrid, which was originally developed to solve elliptic equations.[1,2] Methods based on multigrid have since then proven very efficient for inviscid problems in solving the Euler equations,[3,4] and they are the most promising approach for viscous problems in solving the Navier–Stokes equations.[5] However, in contrast to the Euler equations, the solution of the Navier–Stokes equations still poses a formidable challenge. The main difficulty in computing viscous flow is tied to the fact that for economical resolution of steep gradients in boundary layers, highly stretched meshes are required, resulting in very high cell aspect ratios. These high cell aspect ratios lead to severe stiffness in the discrete system of equations, thus significantly reducing the efficiency of existing numerical algorithms.

Using multigrid, the main ingredient is an appropriate relaxation method to eliminate high-frequency errors on the current mesh level. Here, a general distinction can be made between explicit and implicit schemes, with both having advantages and disadvantages. Explicit schemes have a low operation count per iteration and low storage requirements, but they have only limited stability imposed by the Courant–Friedrichs–Lewy (CFL) condition, and they suffer severely on meshes with high-aspect-ratio cells. In contrast to that, implicit schemes theoretically offer unconditional stability, but they are computationally more expensive and incur a heavy memory overhead because of storing the flux Jacobian matrices, sometimes excluding their application to large-scale problems. Therefore, in practice approximate factorization such as alternate direction implicit (ADI) or incomplete lower–upper decomposition (LU) is employed. The factorization errors, however, prohibit the use of large time-steps and thus limit the potential for fast convergence.

One of the most widespread solution methods is the class of schemes devised by Jameson, Schmidt, and Turkel,[6] where an explicit Runge–Kutta time-integration scheme is augmented with implicit residual smoothing. In combination with multigrid, such methods represent a well-balanced compromise of simplicity in the basic explicit scheme and implicit consideration of the cell aspect ratio. Schemes based on this strategy proved to be well suited to inviscid flow problems, and they allowed successful solution of the Navier–Stokes equations. However, in solving for viscous, turbulent flows, convergence rates deteriorated to 0.98–0.99 from rates on the order of 0.9–0.95 typically observed for inviscid flow.

To overcome the geometrical stiffness in the discrete equations associated with high-aspect-ratio cells, several variants of the standard multigrid approach of Jameson were considered. One variant is the semicoarsening approach originally proposed by Mulder.[7] In this approach, coarsening of the computational mesh is not applied in an isotropic manner as in traditional full coarsening, but is applied separately in each curvilinear mesh coordinate direction to alleviate damping requirements on the relaxation scheme. The drawback of semicoarsening is the increase of operation count and programming complexity. To avoid the cost of complete semicoarsening, a j-line coarsening in only the wall-normal direction is used for coarsening.[8,9] Another variant in improving convergence of the standard multigrid approach is the augmentation of the implicit smoothing. In Ref. 10, the Jacobian matrices of the curvilinear coordinates were used for residual smoothing in each coordinate direction. The resulting smoothing closely resembled an ADI implicit scheme and suffered analogously from factorization error, thus preventing the use of larger time steps. To reduce complexity of this approach, similarly to j-line coarsening, j-line smoothing was developed, where the implicit solution is only applied in the normal direction to alleviate the time-step restriction from clustering grid lines in the boundary layer.[11]

Originally devised for structured meshes, the preceding derivatives of the standard multigrid approach did not gain much acceptance, due to programming complexity and limited applicability in general block-structured codes. It is, however, interesting to observe that for unstructured meshes, techniques similar to j-line coarsening and j-line smoothing are considered due to the higher flexibility provided by the unstructured database.[12] In combination with Jacobi preconditioning, here convergence rates close to 0.9 were observed for two-dimensional viscous, turbulent airfoil flow problems.

In all these approaches, directional information is fed into the solution method and the corresponding coarse-mesh generation algorithm. Furthermore, CFL numbers still remain on the order of 1–10, emphasizing the remaining explicit nature of the relaxation method. In the present work, instead of considering the geometrical stiffness by specifically addressing the directional dependence using directional coarsening and directional implicitness, a more general approach is proposed. To alleviate geometrical stiffness and significantly increase the admissible time step, the directional implicit smoothing is converted into a fully implicit operator with symmetric Gauss–Seidel iteration. The major challenge to be met in such an approach is the reduction of computational memory for storage of the Jacobians, which can be achieved by computing these values

Presented as Paper 2005-0094 at the AIAA 43rd Aerospace Sciences Meeting, Reno, NV, 10–13 January 2005; received 19 January 2005; revision received 16 September 2005; accepted for publication 19 September 2005. Copyright © 2005 by DLR. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 0001-1452/06 $10.00 in correspondence with the CCC.

*Professor, Institut für Aerodynamik und Strömungstechnik, Lilienthalplatz 7; cord.rossow@dlr.de. Senior Member AIAA.

at every smoothing step. This necessitates a significant reduction in the algebraic complexity of the flux Jacobians, and in the present work a formulation based on expressing the flux Jacobians in terms of Mach number is developed. The formulation in terms of Mach number was first presented in Ref. 13 to identify relevant terms of the Roe flux-difference splitting scheme[14] in the incompressible flow regime. Subsequently, also based on terms of Mach number, coefficients for implicit smoothing on unstructured meshes were derived to allow similar convergence rates, as obtained on structured meshes.[15] In the present work, by further exploiting the formulation of Jacobians in terms of Mach number in conjunction with transformation to primitive variables, comparably simple expressions for the Jacobians will be derived. Using these for economic evaluation of the flux Jacobians on cell faces, a fully implicit operator is constructed for implicitly smoothing residuals in the framework of a Runge–Kutta time-stepping scheme. The implications of this technique are then assessed for computation of viscous, turbulent airfoil flow. Comparison with the unsmoothed scheme and standard approaches to implicit smoothing in the same baseline code allows direct estimation of the potential of the present approach for convergence and efficiency improvement.

Governing Equations

We consider the two-dimensional Navier–Stokes equations for compressible flow. For a control volume fixed in time and space, the system of partial differential equations in integral form is given by

$$\iint_{\mathrm{Vol}} \frac{\partial \boldsymbol{W}}{\partial t}\,\mathrm{d}V + \int_S \boldsymbol{F}\cdot\boldsymbol{n}\,\mathrm{d}S = 0 \tag{1}$$

where $\boldsymbol{W}$ represents the vector of conservative variables, $\boldsymbol{F}$ is the flux-density tensor, and Vol, S, and $\boldsymbol{n}$ denote volume, surface, and outward facing normal of the control volume. The flux density tensor $\boldsymbol{F}$ may be split into an inviscid, convective part $\boldsymbol{F}_c$ and a viscous part $\boldsymbol{F}_v$:

$$\boldsymbol{F} = \boldsymbol{F}_c + \boldsymbol{F}_v \tag{2}$$

where $\boldsymbol{F}_c$ and $\boldsymbol{F}_v$ are given by

$$\boldsymbol{F}_c = \begin{bmatrix} \rho\boldsymbol{q} \\ \rho u\boldsymbol{q} + p\boldsymbol{i}_x \\ \rho v\boldsymbol{q} + p\boldsymbol{i}_y \\ \rho H\boldsymbol{q} \end{bmatrix}$$

$$\boldsymbol{F}_v = \begin{bmatrix} 0 \\ \tau_{xx}\boldsymbol{i}_x + \tau_{xy}\boldsymbol{i}_y \\ \tau_{xy}\boldsymbol{i}_x + \tau_{yy}\boldsymbol{i}_y \\ \left(u\tau_{xx} + v\tau_{xy} + k\dfrac{\partial T}{\partial x}\right)\boldsymbol{i}_x + \left(u\tau_{xy} + v\tau_{yy} + k\dfrac{\partial T}{\partial y}\right)\boldsymbol{i}_y \end{bmatrix} \tag{3}$$

where $\boldsymbol{q}$ is the velocity vector with Cartesian components u, v, and $\boldsymbol{i}_x$, $\boldsymbol{i}_y$ denote the unit vectors in the directions of the Cartesian coordinates x and y. ρ, p, H, and T represent density, pressure, total specific enthalpy, and temperature, k is the coefficient of thermal heat conductivity, and τ_{xx}, τ_{yy}, and τ_{xy} are the viscous stress tensor components.

To close the system given by Eq. (1), the equation of state

$$p/\rho = R\cdot T \tag{4}$$

is used with R as specific gas constant.

Basic Solution Scheme

The basic solution scheme is a cell-centered finite volume method as employed in Ref. 13. Using the finite volume technique for space discretization, a semidiscrete form of Eq. (1) may be written as

$$\mathrm{Vol}\frac{\partial \boldsymbol{W}}{\partial t} + \sum_{\text{all faces}} \boldsymbol{F}\cdot\boldsymbol{n}S = 0 \tag{5}$$

where Vol now represents the volume of a computational cell and$S$is the area of a cell face. Equation (5) can be rearranged to

$$\frac{\partial \boldsymbol{W}}{\partial t} + \frac{1}{\mathrm{Vol}}\sum_{\text{all faces}} \boldsymbol{F}_n S = 0 \tag{6}$$

where $\boldsymbol{F}_n$ is the flux density vector corresponding to the direction normal to the cell face. For discretization of the flux density vector on a cell face, in the present work the flux difference splitting (FDS) technique[14] is used. The convective part of the flux density vector $\boldsymbol{F}_c$ normal to a cell interface may then be written as

$$\boldsymbol{F}_c = \tfrac{1}{2}(\boldsymbol{F}^L + \boldsymbol{F}^R) - \tfrac{1}{2}|\boldsymbol{A}|\cdot\Delta\boldsymbol{W} \tag{7}$$

where $\boldsymbol{F}^L$ and $\boldsymbol{F}^R$ are the left and right states of the inviscid flux density vector normal to the cell interface, and $\boldsymbol{A}$ is the corresponding flux Jacobian. The expression $\Delta\boldsymbol{W}$ denotes differences in conservative variables on the left side L and right side R of a cell interface, giving $\Delta\boldsymbol{W} = \boldsymbol{W}^R - \boldsymbol{W}^L$, where the normal vector $\boldsymbol{n}$ at a cell interface is pointing from left to right when the cell boundary is traversed in a mathematically positive sense. Similarly to Ref. 13, in the present work an implementation is used where the expression $|\boldsymbol{A}|\cdot\Delta\boldsymbol{W}$ is formulated in terms of the interface Mach number M_0, with M_0 defined as[13]

$$M_o = \min(|M|, 1)\cdot\mathrm{sign}(M) \tag{8}$$

The resulting expressions for $|\boldsymbol{A}|\cdot\Delta\boldsymbol{W}$ are summarized in Table 1 as flux differences $\Delta\boldsymbol{F}$ for the continuity, momentum, and energy equations. In Table 1, q_n denotes the velocity normal to the cell interface, defined as $q_n = \boldsymbol{q}\cdot\boldsymbol{n}$, and the operator Δ indicates differences of variables between right and left states of the cell interface. For second-order accuracy, the variables ρ, ρu, ρv, ρH are reconstructed, and the symmetric limited positive limiter of Ref. 16 is used following the implementation outlined in Ref. 17. The FDS discretization given by Eq. (7) is implemented in the pattern of a central part plus artificial dissipation to allow for separate evaluation of these terms in the time-marching procedure.

Time integration of Eq. (6) is achieved with an explicit five-stage Runge–Kutta scheme, updating the conservative variables $\boldsymbol{W}$ as

$$\boldsymbol{W}^{(m)}_{i,j} = \boldsymbol{W}^{(0)}_{i,j} + \alpha^{(m)}\cdot\boldsymbol{R}_{i,j}\left(\boldsymbol{W}^{(m-1)}\right) \tag{9}$$

where the superscripts (m) denote the stage count, with m running from 1 to the maximum number of stages, the subscripts i, j correspond to the location at control volumes in the flowfield, $\alpha^{(m)}$ is the stage coefficient of the (m)-stage, and $\boldsymbol{R}_{i,j}(\boldsymbol{W}^{(m-1)})$ represent the conservative residuals of continuity, momentum, and energy equations evaluated using the variables from the previous $(m-1)$-stage:

$$\boldsymbol{R}_{i,j}\left(\boldsymbol{W}^{(m-1)}\right) = -\frac{\delta t_{i,j}}{\mathrm{Vol}_{i,j}}\sum_{\text{all faces}} \boldsymbol{F}_n(\boldsymbol{W}^{m-1})S \tag{10}$$

with $\delta t_{i,j}$ as the local time step of cell i, j. To accelerate convergence toward steady state, the explicit Runge–Kutta time stepping

Table 1 Flux difference splitting dissipation expanded in terms of Mach number[a]

Dissipative operator	Pressure upwinding	Pressure dissipation	Velocity dissipation	Velocity upwinding	Conservative variables dissipation
$\Delta F_\rho =$		$(1/c)(1-\lvert M_0\rvert)\Delta p$		$+\rho M_0\Delta q_n$	$+\lvert q_n\rvert\Delta\rho$
$\Delta F_{\rho u} =$	$n_x M_0\Delta p$	$+(1/c)u(1-\lvert M_0\rvert)\Delta p$	$+n_x\rho c(1-\lvert M_0\rvert)\Delta q_n$	$+\rho u M_0\Delta q_n$	$+\lvert q_n\rvert\Delta\rho u$
$\Delta F_{\rho v} =$	$n_y M_0\Delta p$	$+(1/c)v(1-\lvert M_0\rvert)\Delta p$	$+n_y\rho c(1-\lvert M_0\rvert)\Delta q_n$	$+\rho v M_0\Delta q_n$	$+\lvert q_n\rvert\Delta\rho v$
$\Delta F_{\rho E} =$		$(1/c)H(1-\lvert M_0\rvert)\Delta p + q_n M_0\Delta p$	$+q_n\rho c(1-\lvert M_0\rvert)\Delta q_n$	$+\rho H M_0\Delta q_n$	$+\lvert q_n\rvert\Delta\rho E$

[a] $M_0 = \min(|M|, 1)\cdot\mathrm{sign}(M)$.

is augmented by implicit residual smoothing, which allows an increase of the CFL number of the basic scheme by a factor of 2–3. Originally devised in Ref. 6, the explicitly evaluated residual $\boldsymbol{R}$ in Eq. (9) is replaced with a residual $\tilde{\boldsymbol{R}}$ obtained from successively solving a scalar tri-diagonal implicit system for each curvilinear coordinate direction. Martinelli[18] augmented the basic smoothing of Ref. 6 by introducing smoothing coefficients depending on the cell aspect ratio to efficiently solve the Navier–Stokes equations on highly stretched meshes. Time integration may be further enhanced by employing multigrid following the ideas of Jameson.[4] The influence of turbulence is modeled according to Baldwin and Lomax.[19] The basic outline of the framework for time integration used in this study may be found in Ref. 5.

Derivation of Present Approach

As outlined in the preceding, one of the major challenges in solving the Navier–Stokes equations for turbulent flows is the consideration of the stiffness of the discrete system of equations caused by high cell aspect ratios. Directional methods such as semicoarsening[8,9] or j-line preconditioning[11] did not gain widespread acceptance due to the complexity of programming and limited increase of CFL number. In principle, implicit methods bear the potential to overcome the stiffness problem. A fully implicit formulation of Eq. (6) yields

$$\frac{\partial \boldsymbol{W}}{\partial t} + \frac{1}{\mathrm{Vol}} \sum_{\text{all faces}} \boldsymbol{F}_n^{(n+1)} S = 0 \tag{11}$$

where $\boldsymbol{F}_n^{(n+1)}$ is evaluated at the new time level $(n+1)$. Linearizing $\boldsymbol{F}_n^{(n+1)}$ about the current time level (n), one obtains

$$\boldsymbol{F}_n^{(n+1)} + \boldsymbol{F}_n^{(n)} + \left(\frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{W}}\right)\delta \boldsymbol{W} = \boldsymbol{F}_n^{(n)} + \boldsymbol{A}_n \delta \boldsymbol{W} \tag{12}$$

with $\delta \boldsymbol{W}$ defining the time difference of conservative variables and $\boldsymbol{A}_n$ the flux Jacobian in the normal direction:

$$\delta \boldsymbol{W} = \boldsymbol{W}^{(n+1)} - \boldsymbol{W}^{(n)}, \qquad \boldsymbol{A}_n = \frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{W}} \tag{13}$$

Substituting Eq. (12) into Eq. (11) and replacing the time derivative in Eq. (11) with a finite difference approximation, one obtains

$$\frac{\delta \boldsymbol{W}}{\delta t} + \frac{1}{\mathrm{Vol}} \sum_{\text{all faces}} \boldsymbol{A}_n \delta \boldsymbol{W} S = -\frac{1}{\mathrm{Vol}} \sum_{\text{all faces}} \boldsymbol{F}_n^{(n)} S \tag{14}$$

Rearranging leads to

$$\left(\boldsymbol{I} + \frac{\delta t}{\mathrm{Vol}} \sum_{\text{all faces}} \boldsymbol{A}_n S\right)\delta \boldsymbol{W} = -\frac{\delta t}{\mathrm{Vol}} \sum_{\text{all faces}} \boldsymbol{F}_n^{(n)} S = \boldsymbol{R}^{(n)} \tag{15}$$

where $\boldsymbol{R}^{(n)}$ has been introduced to denote the residual at current time level (n), and $\boldsymbol{I}$ represents the identity matrix. The matrix $\boldsymbol{A}_n$ has real eigenvalues and may be split into two matrices $\boldsymbol{A}_n^+$ and $\boldsymbol{A}_n^-$, where

$$\boldsymbol{A}_n^+ = 0.5(\boldsymbol{A}_n + |\boldsymbol{A}_n|), \qquad \boldsymbol{A}_n^- = 0.5(\mathbf{A}_n - |\boldsymbol{A}_n|) \tag{16}$$

With definition (16), Eq. (15) may be rewritten as a point implicit scheme:

$$\left(\boldsymbol{I} + \frac{\delta t_{i,j}}{\mathrm{Vol}_{i,j}} \sum_{\text{all faces}} \mathbf{A}_n^+ S\right)\delta \boldsymbol{W}_{i,j} = \boldsymbol{R}_{i,j}^{(n)} - \frac{\delta t_{i,j}}{\mathrm{Vol}_{i,j}} \sum_{\text{all faces}} \boldsymbol{A}_n^- \delta \boldsymbol{W}_{\mathrm{NB}} S \tag{17}$$

where indices i, j denote the current cell and NB are all direct neighbors of cell i, j.

Note that the $\boldsymbol{A}_n^- \delta \boldsymbol{W}$ quantity represents the flux density change associated with waves having negative wave speed, that is, waves that enter cell i, j from outside. Only neighbor cells NB can contribute to these changes in flux density. Similarly, $\boldsymbol{A}_n^+ \delta \boldsymbol{W}$ represents flux density changes associated with positive wave speeds, that is, waves which leave cell i, j. These flux density changes are determined only by information from within cell i, j.

To solve Eq. (17) for the changes in conservative variables $\delta \boldsymbol{W}_{i,j}$, the 4×4 matrix (in three dimensions a 5×5 matrix) in the first term on the left-hand side has to be inverted, and symmetric Gauss–Seidel sweeps may be used for iterative solution.

The problem with the scheme of Eq. (17) is the memory requirement for storing the matrices $\boldsymbol{A}_n^+$ and $\boldsymbol{A}_n^-$. For solution of a two-dimensional (three-dimensional) problem, these are 4×4 (5×5) matrices, requiring storage of 32 (50) variables per cell face. Additionally, the necessary matrix-times-vector operations are associated with a high operation count, and as a consequence methods based on Eq. (17) are generally not applied to practical problems. Instead, LU schemes with simplified Jacobians as in Refs. 20, 21 or matrix-free methods[22] are frequently employed. However, these simplifications lead to deterioration of damping properties, resulting in degraded convergence especially in solving the Navier–Stokes equations, even though sometimes high CFL numbers can be used.

To avoid deterioration of damping properties, the present work attempts to solve Eq. (17) without simplifications of the Jacobians to properly address the stiffness problem on stretched meshes. To avoid storage overhead, the matrix components then have to be computed whenever needed. Such a strategy can only be competitive if the arithmetic operations necessary for computation are significantly reduced.

To efficiently evaluate the Jacobian matrices $\boldsymbol{A}_n^+$ and $\boldsymbol{A}_n^-$, the formalism of expressing the absolute Jacobian in terms of Mach number as given in Table 1 will be employed. For simplification, Eq. (14) is first transformed to the set of primitive variables $\boldsymbol{U} = [\rho, p, u, v]^T$:

$$\frac{\delta \boldsymbol{U}}{\delta t} + \frac{1}{\mathrm{Vol}} \sum_{\text{all faces}} \boldsymbol{P}_n \delta \boldsymbol{U} S = -\frac{\partial \boldsymbol{U}}{\partial \boldsymbol{W}} \frac{1}{\mathrm{Vol}} \sum_{\text{all faces}} \boldsymbol{F}_n^{(n)} S \tag{18}$$

with $\boldsymbol{P}_n$ as the analog of the normal flux Jacobian in primitive variables defined as

$$\boldsymbol{P}_n = \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{W}} \boldsymbol{A}_n \frac{\partial \boldsymbol{W}}{\partial \boldsymbol{U}} = \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{W}} \left(\boldsymbol{A}_n^+ + \boldsymbol{A}_n^-\right) \frac{\partial \boldsymbol{W}}{\partial \boldsymbol{U}} = \boldsymbol{P}_n^+ + \boldsymbol{P}_n^- \tag{19}$$

To ensure conservation, the multiplication with the Jacobian $\partial \boldsymbol{U}/\partial \boldsymbol{W}$ on the right-hand side of Eq. (18) has to be performed on the conservative flux balances, that is, outside the summation over the cell faces. Equations (18) and (19) may be combined to give the analog of Eq. (17) in primitive variables:

$$\left(\boldsymbol{I} + \frac{\delta t_{i,j}}{\mathrm{Vol}_{i,j}} \sum_{\text{all faces}} \boldsymbol{P}_n^+ S\right)\delta \boldsymbol{U}_{i,j} = \boldsymbol{Q}_{i,j}^{(n)} - \frac{\delta t_{i,j}}{\mathrm{Vol}_{i,j}} \sum_{\text{all faces}} \boldsymbol{P}_n^- \delta \boldsymbol{U}_{\mathrm{NB}} S \tag{20}$$

with

$$\boldsymbol{Q}_{i,j}^{(n)} = \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{W}} \boldsymbol{R}_{i,j}^{(n)} \tag{21}$$

denoting the residual vector in primitive variables at time level (n).

Using the definitions of Eq. (16) and the terms of Table 1 obtained from expressing the absolute Jacobians in terms of Mach number,[13] performing the multiplications with the transformation Jacobians given by Eq. (19), multiplying with time changes in primitive variables $\delta \boldsymbol{U}$, and sorting terms, the expressions $\boldsymbol{P}_n^+ \delta \boldsymbol{U}$ and $\boldsymbol{P}_n^- \delta \boldsymbol{U}$ in Eq. (20) may be written as

$$\boldsymbol{P}_n^+ \cdot \delta \boldsymbol{U}$$

$$= \begin{bmatrix} (q_n + |q_n|)\delta\rho + (1/c)(1 - |M_0|)\delta p + \rho(1 + M_0)\delta q_n \\ (q_n + |q_n|)\delta p + (\gamma - 1)(h/c)(1 - |M_0|)\delta p + \gamma p(1 + M_0)\delta q_n \\ (q_n + |q_n|)\delta u + n_x(1/\rho)(1 + M_0)\delta p + n_x c(1 - |M_0|)\delta q_n \\ (q_n + |q_n|)\delta v + n_y(1/\rho)(1 + M_0)\delta p + n_y c(1 - |M_0|)\delta q_n \end{bmatrix}$$

$$P_n^- \cdot \delta U = \begin{bmatrix} (q_n - |q_n|)\delta\rho - (1/c)(1-|M_0|)\delta p + \rho(1-M_0)\delta q_n \\ (q_n - |q_n|)\delta p - (\gamma-1)(h/c)(1-|M_0|)\delta p + \gamma p(1-M_0)\delta q_n \\ (q_n - |q_n|)\delta u + n_x(1/\rho)(1-M_0)\delta p - n_x c(1-|M_0|)\delta q_n \\ (q_n - |q_n|)\delta v + n_y(1/\rho)(1-M_0)\delta p - n_y c(1-|M_0|)\delta q_n \end{bmatrix} \tag{22}$$

where γ, h, c denote the ratio of specific heats, specific enthalpy, and speed of sound, respectively, and δq_n is defined as

$$\delta q_n = n_x \delta u + n_y \delta v \tag{23}$$

with n_x and n_y as the components of the outward-facing normal $\boldsymbol{n}$.

The definitions for $P^+\delta U$ and $P^-\delta U$ in Eq. (22) allow an implementation of the implicit scheme of Eq. (20) with a minimum of computational effort. Only the expressions $|q_n|$, M_0, $(1-|M_0|)$ are precomputed and stored for each cell face; all other components can be recomputed efficiently whenever necessary. The contributions of the viscous flux Jacobians can easily be incorporated with the definitions outlined in Ref. 23 for primitive variables.

The implicit scheme of Eq. (20) is implemented into the framework of the basic code analogously to implicit residual smoothing: in each Runge–Kutta stage, prior to updating conservative variables with Eq. (9), the explicitly evaluated residuals are transformed to residuals in primitive variables to form the right hand side of Eq. (18). Interpreting the changes in primitive variables δU now as smoothed residuals $\tilde{Q}$ then yields an equation for implicit residual smoothing:

$$\left(I + \frac{\delta t_{i,j}}{\mathrm{Vol}_{i,j}} \sum_{\text{all faces}} P_n^+ S\right)\tilde{Q}_{i,j} = Q_{i,j}^{(n)} - \frac{\delta t_{i,j}}{\mathrm{Vol}_{i,j}} \sum_{\text{all faces}} P_n^- \tilde{Q}_{\mathrm{NB}} S \tag{24}$$

Solution of Eq. (24) with three symmetric Gauss–Seidel (SGS) sweeps through the computational domain, yielding new, smoothed primitive residuals, which are subsequently transformed to new, smoothed conservative residuals $\tilde{R}$ by

$$\tilde{R}_{i,j} = \frac{\partial W}{\partial U}\tilde{Q}_{i,j} \tag{25}$$

The new, smoothed conservative residuals $\tilde{R}$ are then used to update conservative variables according to Eq. (9) in the Runge–Kutta framework by

$$W_{i,j}^{(m)} = W_{i,j}^{(0)} + \alpha^{(m)} \cdot \tilde{R}_{i,j} \tag{26}$$

Note that the only change with respect to the basic code is the replacement of the original routine for residual smoothing with a routine that performs the residual smoothing according to Eq. (24) with transformation to conservative residuals using Eq. (25).

Computational Results

The proposed method was used to compute viscous, turbulent flow around the RAE 2822 airfoil. As test cases, cases 1, 9, and 10 from the investigations of Cook et al.[24] were chosen. For case 1, the flowfield remains mainly subsonic and no shock wave occurs. case 9, one of the cases most frequently used in testing computational methods, exhibits a fairly strong shock on the upper surface of the airfoil. For case 10, the shock strength is increased, leading to substantial separation behind the shock. This case often severely degrades convergence of numerical methods. The different freestream conditions are listed in Table 2.

For all cases, a structured C-mesh was used with 64 cells in the normal direction and 320 cells in the circumferential direction, where 256 cells were located on the airfoil surface. For the multigrid algorithm, coarse meshes were created by successively omitting every second grid line, and a four-level W-cycle was employed. Computations were performed on an SGI Octane workstation using a single 360-MHz processor.

Table 2 Freestream conditions of test cases for RAE1822 airfoil

Case	M_∞	α [°]	Re	x_{tr}/l
1	0.676	1.93	5.7×10^6	0.11
9	0.730	2.80	6.5×10^6	0.03
10	0.750	2.80	6.2×10^6	0.03

Table 3 Computed total forces for RAE2822 airfoil

Case	c_l	c_{d_total}	c_{d_pressure}	c_{d_friction}
1	0.60875	0.00840	0.00246	0.00594
9	0.85260	0.01784	0.01223	0.00561
10	0.84503	0.02866	0.02316	0.00550

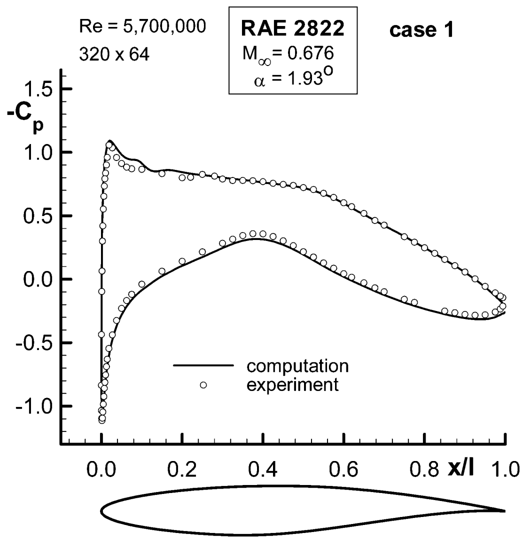

Fig. 1 Surface pressure distributions for flow around RAE 2822, case 1.

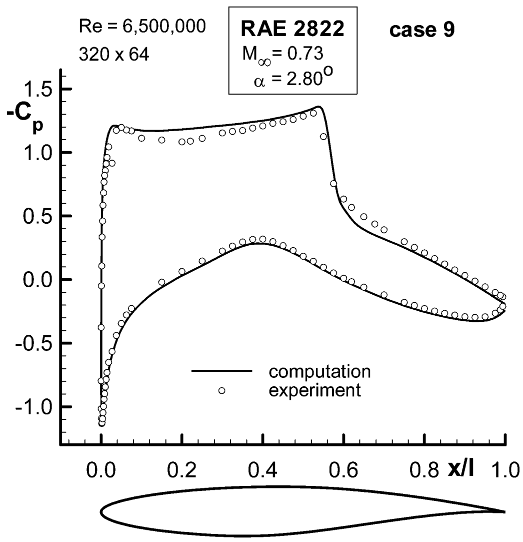

Fig. 2 Surface pressure distributions for flow around RAE 2822, case 9.

This investigation is focused on enhancement of time integration; therefore an identical space discretization is always employed. Following the method of lines in this study, space and time discretization are decoupled, and steady state results become independent from the time integration, provided a converged solution is obtained. For fair comparison, all computations were started from freestream on the finest mesh unless noted otherwise. Figures 1–3 display the pressure distributions obtained for the three test cases, and Table 3 summarizes the computed lift and drag coefficients.

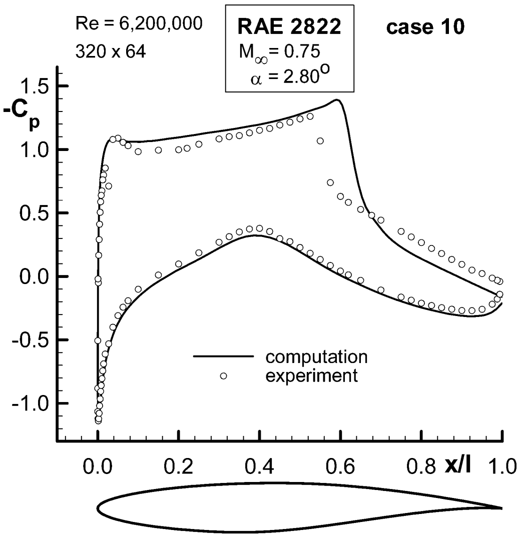

Fig. 3 Surface pressure distributions for flow around RAE 2822, case 10.

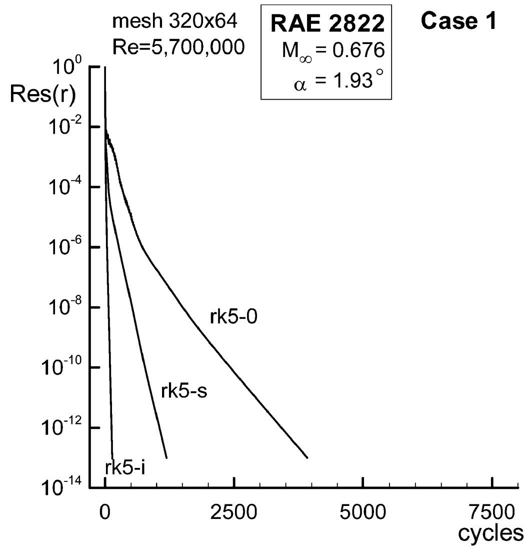

Fig. 4 Convergence history for computation of case 1 with four-level multigrid: rk5-i, five-stage Runge–Kutta and implicit SGS scheme; rk5-s, five-stage Runge–Kutta and standard smoothing; and rk5-0, five-stage Runge–Kutta, purely explicit.

Pressure distributions of cases 1 and 9 agree fairly well with experimental results. For case 10, the well-known deficiency of the Baldwin–Lomax turbulence model in predicting this case in accordance with measurements is evident. The results presented here are in good agreement with numerical results presented elsewhere in the literature.[5,17,25] All time integration schemes investigated here provided these results and the following discussion will therefore focus only on convergence properties and computational efficiency of the different schemes.

Figures 4–6 show convergence histories for three different time integration strategies. In all cases, a five-stage Runge–Kutta scheme was employed using the stage coefficients recommended in Ref. 26 for a second-order upwind discretization, with central flux and artificial dissipation evaluated in each stage. The first strategy, denoted by rk5-0, is purely explicit: the five-stage Runge–Kutta scheme is used with four-level multigrid without any further means of acceleration. The possible CFL number was 1.3, and to decrease the residual of the continuity equation by 13 orders of magnitude, this strategy required from about 4000 iterations for case 1 to about 7500 for case 10. Employing the implicit residual smoothing of Ref. 18, denoted by rk5-s, the admissible CFL number could be raised to 7.5, and convergence improved by a factor of 3–4. The required

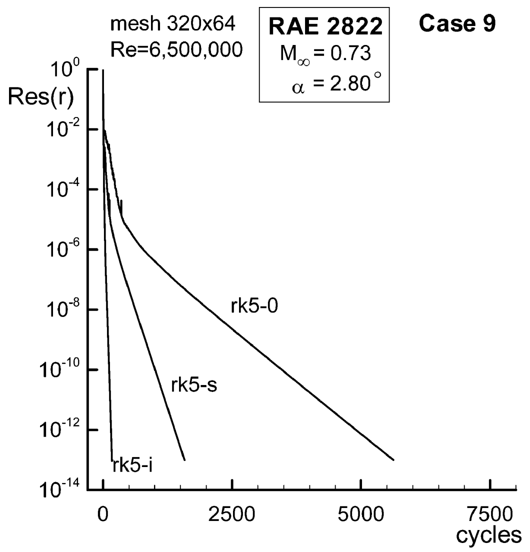

Fig. 5 Convergence history for computation of case 9 with four-level multigrid: rk5-i, five-stage Runge–Kutta and implicit SGS scheme; rk5-s, five-stage Runge–Kutta and standard implicit smoothing; and rk5-0, five-stage Runge–Kutta, purely explicit.

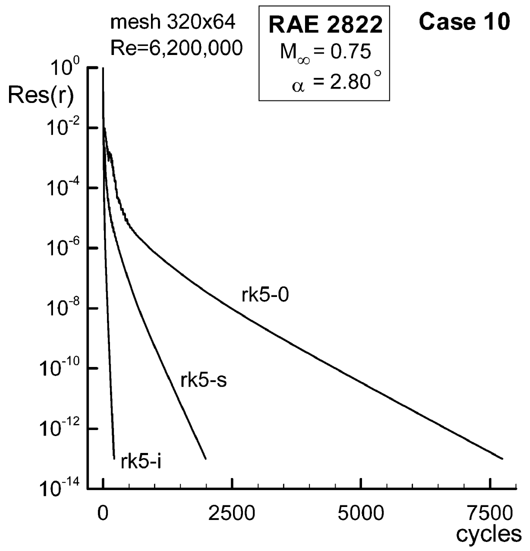

Fig. 6 Convergence history for computation of case 10 with four-level multigrid: rk5-i, five-stage Runge–Kutta and implicit SGS scheme; rk5-s, five-stage Runge–Kutta and standard implicit smoothing; and rk5-0, five-stage Runge–Kutta, purely explicit.

CPU time shows almost the same factors in reduction, as can be seen from Table 4. For the strategy proposed in this study, denoted by rk5-i, using the smoothing according to Eq. (24), the CFL number was set to 16 for the first eight multigrid cycles, and then it was fixed to 160. The scheme achieved a convergence improvement by a factor of 30–35 compared to the unsmoothed scheme. This is similar to the improvement shown in Ref. 9, where semicoarsening and Jacobi preconditioning were employed. Compared to the scheme with smoothing of Ref. 18, a factor of about 9 in convergence improvement was obtained. With respect to computational time, the improvement does not scale accordingly, due to the higher computational effort required for evaluation of flux Jacobians and solution of Eq. (24). However, a factor of about 6 in reduction of CPU time compared to the unsmoothed scheme could still be achieved, and a factor of about 2.5–2.8 with respect to the scheme with the smoothing of Ref. 18; see Table 4. The rate of convergence improved from about 0.995 of the unsmoothed scheme to roughly 0.98 for the smoothing of Ref. 18, and to about 0.85 for the present method, as can be seen in Table 4. It is interesting to note that the benefit resulting from application of the present method is

Table 4 Computational effort required for RAE 2822 test cases

Scheme	CPU time, s	No. of iterations	Reduction/cycle
		Case 1	
rk5-0	9,328	3,921	0.992
rk5-s	3,053	1,194	0.975
rk5/3-s	2,349	1,191	0.975
rk5-i	1,206	142	0.801
rk3-i	1,023	190	0.854
		Case 9	
rk5-0	13,580	5,627	0.995
rk5-s	4,048	1,579	0.981
rk5/3-s	3,129	1,585	0.981
rk5-i	1,495	176	0.843
rk3-i	1,337	248	0.886
		Case 10	
rk5-0	18,676	7,736	0.996
rk5-s	5,028	1,991	0.985
rk5/3-s	3,929	2,019	0.985
rk5-i	1,818	214	0.869
rk3-i	1,552	288	0.901

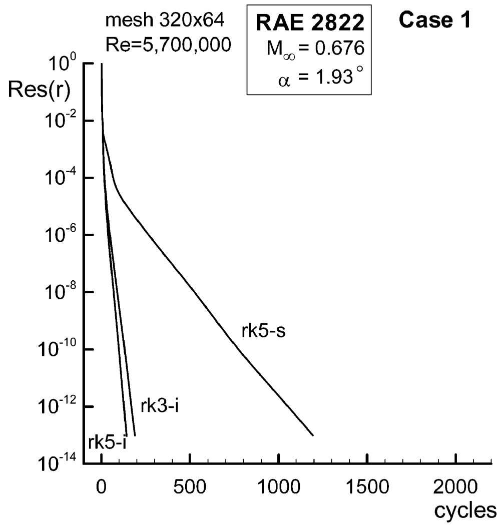

Fig. 7 Convergence history for computation of case 1 with four-level multigrid: rk5-i, five-stage Runge–Kutta and implicit SGS scheme; rk5-s, five-stage Runge–Kutta and standard implicit smoothing; and rk3-i, three-stage Runge–Kutta and implicit SGS scheme.

higher for the more difficult case 10 than for the relatively benign case 1.

Comparing the present method to well-established schemes using the smoothing of Ref. 18, one must take into account that in such codes hybrid Runge–Kutta schemes are usually employed, where the numerical dissipation is evaluated only in selected stages to reduce computational effort. Following Ref. 8, a different set of smoothing coefficients is employed, and artificial dissipation is evaluated only at odd stages of the five-stage Runge–Kutta scheme. CFL numbers remain at 7.5, and the number of iterations is almost not affected compared to the straightforward implementation of scheme rk5-s. However, the CPU time required is reduced by about 20–25%; see Table 4, where this scheme is denoted by rk5/3-s. This efficient implementation is the reason for the widespread use of such methods, and for a valid comparison of efficiency, these well tuned methods have to be used as benchmark. With respect to this basis for comparison, the computational benefit of method rk5-i is reduced to a factor of 2.0–2.2.

To increase computational efficiency of the present method, a three-stage Runge–Kutta scheme was employed with the implicit SGS scheme (rk3-i). The starting CFL number was set to 10 in the first 8 iterations, and raised to 100 for the remaining computation. Comparison of convergence histories is displayed in Figs. 7–9. Convergence rates are now on the order of 0.88, and CPU time with

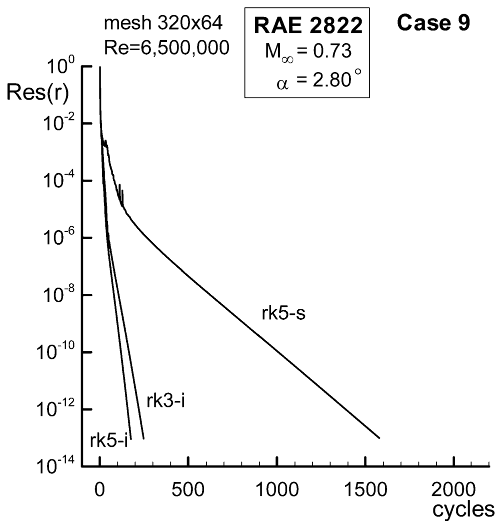

Fig. 8 Convergence history for computation of case 9 with four-level multigrid: rk5-i, five-stage Runge–Kutta and implicit SGS scheme; rk5-s, five-stage Runge–Kutta and standard implicit smoothing; and rk3-i, three-stage Runge–Kutta and implicit SGS scheme.

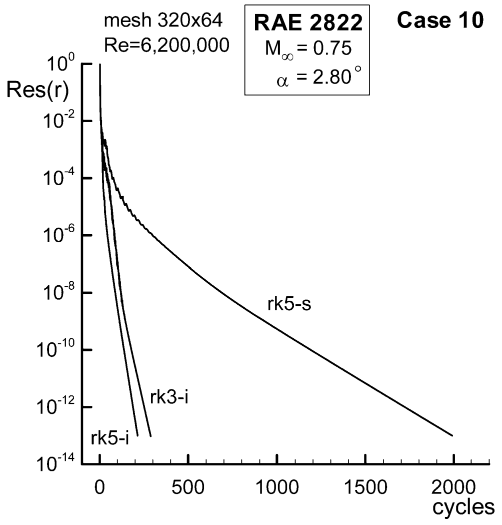

Fig. 9 Convergence history for computation of case 10 with four-level multigrid: rk5-i, five-stage Runge–Kutta and implicit SGS scheme; rk5-s, five-stage Runge–Kutta and standard implicit smoothing; and rk3-i, three-stage Runge–Kutta and implicit SGS scheme.

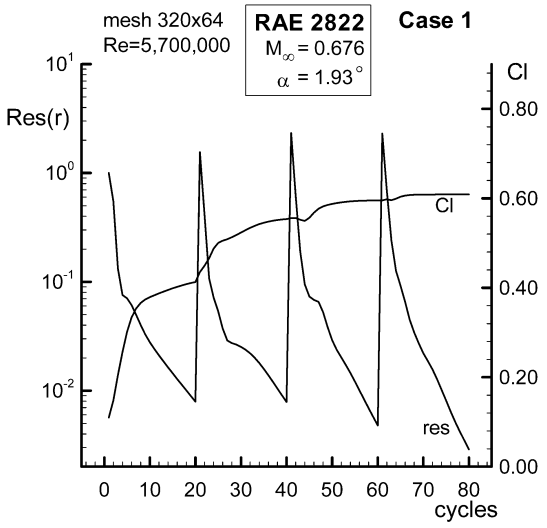

Fig. 10 Convergence history for computation of case 1 with four-level full multigrid using five-stage Runge–Kutta and implicit SGS scheme.

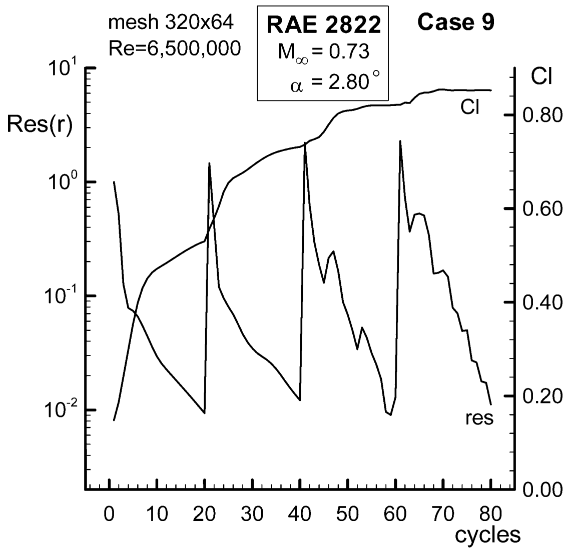

Fig. 11 Convergence history for computation of case 9 with four-level full multigrid using five-stage Runge–Kutta and implicit SGS scheme.

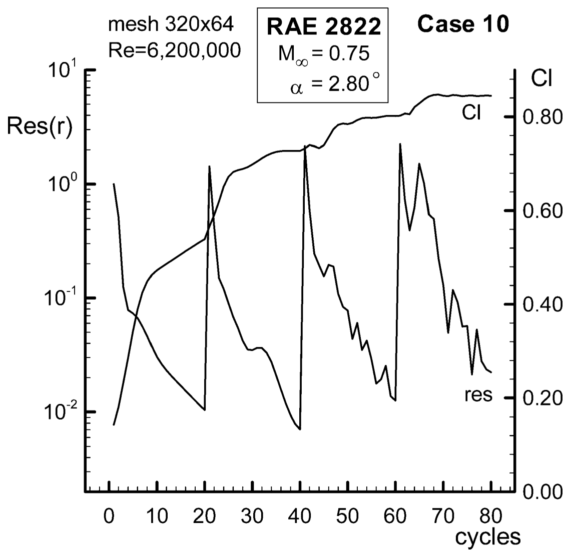

Fig. 12 Convergence history for computation of case 10 with four-level full multigrid using five-stage Runge–Kutta and implicit SGS scheme.

respect to the benchmark of the hybrid Runge–Kutta scheme is now reduced by a factor of roughly 2.3–2.5; see Table 4.

To illustrate the performance of the present method, Figs. 10–12 show convergence histories using the five-stage Runge–Kutta scheme, the fully implicit SGS smoothing, and full multigrid. Successively performing 20 multigrid cycles on each mesh level suffices for converging total forces to engineering accuracy. In these cases, the starting CFL number was set to 80 in the first two cycles on each mesh, and then raised to 160 for the following iterations.

Conclusions

In the present work, a computational approach was derived that addresses the stiffness problem associated with high-aspect-ratio cells for viscous flow computations. The proposed method is based on a fully implicit operator for implicit smoothing of residuals in the framework of Runge–Kutta time stepping. To avoid the memory overhead usually required for storage of the flux Jacobians, the components of the Jacobians are continuously recomputed during iteration. Efficient evaluation of the corresponding terms is achieved by using flux Jacobians expressed in terms of Mach number as outlined in Ref. 13. Transforming all terms to primitive variables leads to relatively simple expressions, and only three variables per cell face need to be stored, compared to 32 variables in two dimensions or 50 in three dimensions for a straightforward implementation. To solve the implicit equation, symmetric Gauss–Seidel iteration with three sweeps through the computational domain was employed, and in combination with Runge–Kutta time stepping, CFL numbers on the order of 100 could be used. The present method was applied for solving different cases of compressible, turbulent airfoil flow, and convergence rates ranging from 0.8 to 0.87 were achieved. In comparison to well-tuned common multigrid methods with Runge–Kutta time stepping and implicit residual smoothing, reductions in CPU time by more than a factor of 2 were realized. It is worth noting that in contrast to commonly employed implicit smoothing techniques, the method proposed in this study is not restricted to structured meshes, since the fully implicit operator does not rely on information obtained from curvilinear coordinates.

Acknowledgments

Parts of this study were conducted when the author was in residence at the former Institute for Computer Applications in Science and Engineering, ICASE, and the author expresses his thanks to Manuel Salas, former ICASE Director, for providing this opportunity.

References

[1]Brandt, A., "Multi-Level Adaptive Solutions to Boundary-Value Problems," *Mathematics of Computations*, Vol. 31, No. 138, 1977, pp. 333–390.

[2]Hackbusch, W., "On the Multigrid Method Applied to Differene Equations," *Computing*, Vol. 20, 1978, pp. 291–306.

[3]Jameson, A., "Solution of the Euler Equations for Two-Dimensional, Transonic Flow by a Multigrid Method," *Applied Mathematics and Computation*, Vol. 13, 1983, pp. 327–356.

[4]Jameson, A., "Multigrid Algorithms for Compressible Flow Calculations," *Second European Conference on Multigrid Methods, Cologne, 1985*, edited by W. Hackbusch and U. Trottenberg, Vol. 1228, Lecture Notes in Mathematics, Springer-Verlag, Berlin, 1986, pp. 166–201.

[5]Radespiel, R., Rossow, C.-C., and Swanson, R. C., "An Efficient Cell-Vertex Multigrid Scheme for the Three-Dimensional Navier–Stokes Equations," *AIAA Journal*, Vol. 28, No. 8, 1998, pp. 423–459.

[6]Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge–Kutta Time-Stepping Schemes," AIAA Paper 81-1259, June 1981.

[7]Mulder, W. A., "A New Multigrid Approach To Convection Problems," *Journal of Computational Physics*, Vol. 83, No. 2, 1989, pp. 303–323.

[8]Radespiel, R., and Swanson, R. C., "Progress with Multigrid Schemes for Hypersonic Flow Problems," *Journal of Computational Physics*, Vol. 116, No. 1, 1995, pp. 103–122.

[9]Pierce, N. A., and Giles, M. B., "Preconditioned Multigrid Methods for Compressible Flow Calculations on Stretched Meshes," *Journal of Computational Physics*, Vol. 136, No. 2, 1997, pp. 425–445.

[10]Blazek, J., "Verfahren zur Beschleunigung der Lösung der Euler- und Navier–Stokes-Gleichungen bei stationären Über- und Hyperschallströmungen," Ph.D. Dissertation, Institut fuer Stroemungsmechanik, Technical Univ., Brunswick, Germany, Aug. 1995.

[11]Turkel, E., Vatsa, V., and Venkatakrishnan, V., "Uni-Directional Implicit Acceleration Techniques for Compressible Navier–Stokes Solvers," AIAA Paper 99-3265, June 1999.

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

[13]Rossow, C.-C., "A Flux Splitting Scheme for Compressible and Incompressible Flows," *Journal of Computational Physics*, Vol. 164, No. 1, 2000, pp. 104–122.

[14]Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors and Difference Schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[15]Rossow, C.-C., "Convergence Acceleration on Unstructured Meshes," *Contributions to the 12th STAB/DGLR Symposium Stuttgart 2000*, Vol. 77, Notes on Numerical Fluid Mechanics, Springer, Berlin, 2002, pp. 304–311.

[16]Jameson, A., "Artificial Diffusion, Upwind Biasing, Limiters and their Effect on Accuracy and Multigrid Performance in Transonic and Hypersonic Flow," AIAA Paper 93-3359, July 1993.

[17]Swanson, R. C., Radespiel, R., and Turkel, E., "On Some Numerical Dissipation Schemes," *Journal of Computational Physics*, Vol. 147, No. 2, 1998, pp. 518–544.

[18]Martinelli, L., "Calculations of Viscous Flow with a Multigrid Method," Ph.D. Dissertation, Dept. of Mechanical and Aerospace Engineering, Princeton Univ., Princeton, NJ, Oct. 1987.

[19]Baldwin, B., and Lomax, H., "Thin Layer Approximation and Algebraic Turbulence Model for Separated Turbulent Flows," AIAA Paper 78-257, Jan. 1978.

[20]Jameson, A., and Turkel, E., "Implicit Schemes and LU Decompositions," *Mathematics of Computations*, Vol. 37, No. 156, 1981, pp. 385–397.

[21]Jameson, A., and Yoon, S., "Multigrid Solutions of the Euler Equations Using Implicit Schemes," *AIAA Journal*, Vol. 24, No. 11, 1986, pp. 1737–1743.

[22]Hong, L., Baum, J. D., and Löhner, R., "A Fast, Matrix-Free Implicit Method for Compressible Flows on Unstructured Grids," *Journal of Computational Physics*, Vol. 146, No. 2, 1998, pp. 664–690.

[23]Abarbanel, S., and Gottlieb, D., "Optimal Splitting for Two and Three Dimensional Navier–Stokes Equations with Mixed Derivatives," ICASE Rept. 80-6, Feb. 1980.

[24]Cook, P. H., McDonald, M. A., and Firmin, M. C. P., "Aerofoil RAE2822 Pressure Distributions and Boundary Layer and Wake Measurements," AGARD AR-138, Nov. 1979.

[25]Rudnik, R., "Untersuchung der Leistungsfähigkeit von Zweigleichungs-Turbulenzmodellen bei Profilumströmungen," Ph.D. Dissertation, Fachbereich Verkehrswesen und Angewandte Mathematik, Technical Univ., Berlin, Sept. 1997.

[26]Van Leer, B., Tai, C. H., and Powell, K. G., "Design of Optimally Smoothing Multi-Stage Schemes for the Euler Equations," AIAA Paper 89-1933, June 1989.

C. Kaplan
Associate Editor