Chapter 11

The Design of Steady State Schemes for Computational Aerodynamics

F.D. Witherden*, A. Jameson* and D.W. Zingg†
*Stanford University, Stanford, CA, United States
†University of Toronto Institute for Aerospace Studies, Toronto, ON, Canada

Chapter Outline

ABSTRACT
The majority of industrial computational fluid dynamics simulations are steady. Such simulations therefore require numerical methods which can rapidly converge a boundary value problem, typically either the Euler or Reynolds-averaged Navier–Stokes (RANS) equations. In this chapter we outline two major approaches: time-marching methods built on top of Runge–Kutta time-stepping schemes and Newton–Krylov methods. A range of techniques for accelerating convergence are presented including local time-stepping, enthalpy damping, residual averaging, multigrid methods and preconditioning. More recent hybrid approaches such as the Runge–Kutta symmetric Gauss–Seidel are also discussed. The effectiveness of these methodologies, within the context of several Euler and RANS test cases, is also evaluated.

Keywords: Computational fluid dynamics, Gas dynamics, Finite volume methods, Newton–Krylov methods, Runge–Kutta methods

AMS Classification Codes: 65M08, 65M12, 65F08, 65N22

Handbook of Numerical Analysis, Vol. 18. http://dx.doi.org/10.1016/bs.hna.2016.11.006

1 INTRODUCTION

An important requirement for aeronautical applications of computational methods in aerodynamic design is the capability to predict the steady flow past a proposed configuration, so that key performance parameters such as the lift to drag ratio can be estimated. Even in manoeuvring flight the time scales of the motion are large compared with those of the flow, so the unsteady effects are secondary. Thus the aerodynamic design will normally be based on analysis of steady flow. In fact, no one would wish to be a passenger on an aircraft that could not sustain essentially steady flow. Unsteady flow due to buffet or wing flutter is not acceptable for normal operation, so the analysis of unsteady flow is required primarily for checking the structural integrity at the limits of the flight envelope, such as establishing that the minimum speed at which flutter can occur is greater than the maximum permissible speed in a dive.

We shall consider systems with the general form

$$\frac{dw}{dt} + R(w) = 0 \tag{1}$$

where $R(w)$ is the residual resulting from the spatial discretization of the Euler and Navier–Stokes equations for compressible flow.

One approach is to calculate the steady state solutions directly by an iterative method. In fact Newton's method was successfully used (Giles and Drela, 1987) to solve the two-dimensional Euler equations. In three dimensions the bandwidth of the Jacobian matrix is generally too large for direct inversion to be feasible, so the common practice is to use an iterative method to solve the linear system. In Newton–Krylov methods the subiterations are performed by a Krylov method such as GMRES. Newton methods have the disadvantage that they may not necessarily converge if the initial guess is not close enough to the solution.

The alternative approach is to use a time-marching method to advance the unsteady equation (1) to a steady state. Since the transient solution is immaterial, the time integration method may be modified by a variety of acceleration procedures to maximize the rate of convergence to a steady state, without maintaining any pretence of time accuracy. In the early days of CFD, it was commonly assumed that in order to obtain fast convergence to a steady state, it would be necessary to use an implicit scheme which allowed large time steps. Any implicit scheme, however, such as the backward Euler scheme

$$w^{n+1} = w^n - \Delta t R(w^{n+1}) \tag{2}$$

with the superscript n denoting the time levels, requires the solution of a large number of coupled nonlinear equations which have the same complexity as the steady state problem

$$R(w) = 0. \tag{3}$$

Accordingly a fast steady state solver can provide an essential building block for an implicit scheme to solve the unsteady equations. In fact this is the basis of dual time-stepping procedures (Jameson, 1991) for time accurate calculations using implicit schemes, which have proved very successful in practice (Jameson, 2015).

There is actually a close connection between direct iteration and time-marching methods. Denoting the change $w^{n+1} - w^n$ by Δw, and setting

$$R(w^{n+1}) = R(w^n) + \frac{\partial R}{\partial w}\Delta w + \mathcal{O}((\Delta w)^2)$$

the backward Euler scheme (2) may be linearized as

$$\left(I + \Delta t \frac{\partial R}{\partial w}\right)\Delta w + \Delta t R(w^n) = 0$$

and the Newton iteration is recovered in the limit $\Delta t \to \infty$. Accordingly it is a common practice with Newton methods to use the linearized backward Euler scheme in the early steps of the calculation. It may be noted, moreover, that an effective way to analyze classical relaxation methods for elliptic problems is to regards them as approximations to an artificial unsteady problem where each iterations represents a step in pseudo-time.

In this article we shall consider both time-marching and direct iterative schemes. Section 2 gives a brief review of the equations of gas dynamics and appropriate upwind biased spatial discretization schemes. Section 3 reviews steady state solvers based on pseudo-time-marching methods, while Section 4 reviews direct iterative methods such as Newton–Krylov procedures. Finally, Section 5 presents some conclusions and an overall assessment of the advantages and disadvantages of the various approaches.

2 EQUATIONS OF GAS DYNAMICS AND SPATIAL DISCRETIZATIONS

We shall consider steady state solvers for the Euler and Navier–Stokes equations for compressible flows. Let x_i, u_i, ρ, p, E and H denote the Cartesian coordinates, velocity, density, pressure, energy and enthalpy. The Euler equations for an inviscid gas are thus

$$\frac{\partial \boldsymbol{w}}{\partial t} + \frac{\partial}{\partial x_i}\boldsymbol{f}_i(\boldsymbol{w}) = 0 \tag{4}$$

where the state and flux vectors are

$$\boldsymbol{w} = \rho \begin{bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ E \end{bmatrix}, \quad \boldsymbol{f}_i = \rho u_i \begin{bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ H \end{bmatrix} + p \begin{bmatrix} 0 \\ \delta_{i1} \\ \delta_{i2} \\ \delta_{i3} \\ 0 \end{bmatrix}$$

Also,

$$p = (\gamma - 1)\rho\left(E - \frac{u^2}{2}\right), \quad H = E + \frac{p}{\rho} = \frac{c^2}{\gamma - 1} + \frac{u^2}{2}$$

where u is the speed, and c is the speed of sound

$$u^2 = u_i^2, \;\; c^2 = \frac{\gamma p}{\rho}$$

Let m_i and e denote the momentum components and total energy,

$$m_i = \rho u_i, \;\; e = \rho E = \frac{p}{\gamma - 1} + \frac{m_i^2}{2\rho}$$

Then w and f can be expressed as

$$\boldsymbol{w} = \begin{bmatrix} \rho \\ m_1 \\ m_2 \\ m_3 \\ e \end{bmatrix}, \; \boldsymbol{f}_i = u_i \begin{bmatrix} \rho \\ m_1 \\ m_2 \\ m_3 \\ e \end{bmatrix} + p \begin{bmatrix} 0 \\ \delta_{i1} \\ \delta_{i2} \\ \delta_{i3} \\ u_i \end{bmatrix} \tag{5}$$

Finite volume schemes directly approximate the integral form of the equations. The flux needs to be calculated across the interface between each pair of cells. Denoting the face normal and area by n_i and S, the flux is $\boldsymbol{f}S$ where

$$\boldsymbol{f} = n_i \boldsymbol{f}_i$$

This can be expressed in terms of the conservative variables w as

$$\boldsymbol{f} = u_n \begin{bmatrix} \rho \\ m_1 \\ m_2 \\ m_3 \\ e \end{bmatrix} + p \begin{bmatrix} 0 \\ n_1 \\ n_2 \\ n_3 \\ u_n \end{bmatrix} \tag{6}$$

where u_n is the normal velocity

$$u_n = n_i u_i = \frac{n_i m_i}{\rho}$$

Also

$$p = (\gamma - 1)\left(e - \frac{m_i^2}{2\rho}\right)$$

In smooth regions of the flow the equations can also be written in quasi-linear form as

$$\frac{\partial \boldsymbol{w}}{\partial t} + A_i \frac{\partial \boldsymbol{w}}{\partial x_i} = 0$$

where A_i are the Jacobian matrices

$$A_i = \frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{w}},$$

The composite Jacobian matrix at a face with normal vector $\vec{n}$ is

$$A = A_i n_i = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w}}$$

All the entries in $\boldsymbol{f}_i$ and $\boldsymbol{f}$ are homogenous of degree 1 in the conservative variables $\boldsymbol{w}$. It follows that $\boldsymbol{f}_i$ and $\boldsymbol{f}$ satisfy the identities

$$\boldsymbol{f}_i = A_i \boldsymbol{w}, \quad \boldsymbol{f} = A\boldsymbol{w}$$

The wave speeds normal to the interface are given by the eigenvalues of A: $u_n + c$, $u_n - c$, u_n, u_n and u_n.

In upwind discretizations the numerical flux across each interface is calculated from the left and right states w_L and w_R on either side of the interface, with an upwind bias depending on the wave speeds. Examples include the Roe flux (Roe, 1981), the Rusanov flux (Rusanov, 1961), HCUSP (Jameson, 1995b), AUSM (Liou and Steffen, 1993) and HLLC (Toro et al., 1994). First-order accurate schemes take w_L and w_R as the cell average values of the cell on either side of the interface. In second-order accurate schemes w_L and w_R are reconstructed from the values in a stencil of neighbouring cells by reconstruction methods such as MUSCL (van Leer, 1974) and SLIP (Jameson, 1995a), subject to limiters to prevent oscillations in the vicinity of discontinuities.

The construction of upwind biased fluxes generally depends on splitting the Jacobian matrix A as

$$A = \frac{1}{2}(A^+ + A^-)$$

where $A^\pm$ have positive and negative eigenvalues. For example if A is decomposed as $V \Lambda V^{-1}$ where the columns of V are the eigenvectors of A, and Λ is a diagonal matrix containing the eigenvalues, one may split Λ into $\Lambda^\pm$ containing and positive and negative eigenvalues, and define

$$A^\pm = V \Lambda^\pm V^{-1}$$

and also the absolute Jacobian matrix

$$|A| = V(\Lambda^+ - \Lambda^-)V^{-1} = V|\Lambda|V^{-1}$$

In the Navier–Stokes equations for viscous compressible flow Eq. (5) is augmented by the addition of viscous fluxes to the momentum and energy equations. In the momentum equations the viscous stresses area

$$\sigma_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \lambda \delta_{ij} \frac{\partial u_k}{\partial x_k}$$

where μ and λ are the coefficients of viscosity and bulk viscosity. Typically $\lambda = -\frac{2}{3}\mu$. The additional fluxes in the energy equation are

$$\sigma_{jk}u_k + \kappa\frac{\partial T}{\partial x_j}$$

where κ is the coefficient of heat conduction and T is the temperature. It is the usual practice to discretize the viscous terms with central differences.

Flows at high Reynolds numbers generally become turbulent, and one may then resort to solving the Reynolds-averaged (RANS) and Favre-averaged (FANS) equations for mean values of the flow quantities. Residual terms arising from the difference between the average of a product and the product of averages are then represented by turbulence models. In aerodynamic simulations the most popular turbulence models are the Spalart–Allmaras and Menter SST models. For a review of turbulence models the reader is referred to the book by Wilcox (1998). In turbulent flow simulations convergence to a steady state is generally impeded both by the presence of stiff source terms in the turbulence models, and the need to introduce meshes with a very fine mesh spacing in the normal direction in the vicinity of the wall. It is the usual practice to place the first mesh point adjacent to the wall at a distance of $y^+ \equiv u_* y/\nu = 1$ where u_* is the friction velocity at the wall, y is the distance to the wall, and ν is the kinematic viscosity of the fluid. This leads to mesh cells with a very high aspect ratio.

In the following sections it will be assumed that the spatial discretization is accomplished by a second-order accurate upwind biased scheme along these lines or the Jameson–Schmidt–Turkel (JST) scheme (Jameson et al., 1981). It may be noted that higher order reconstruction schemes such as WENO (Liu et al., 1994) may fail to converge to a completely steady state because they are prone to develop limit cycles in which the reconstruction stencils flip between different choices.

3 TIME-MARCHING METHODS

3.1 Model Problem for Stability Analysis of Convection Dominated Problems

The following model problem provides a useful tool for the stability analysis of time integration methods for convection dominated problems. Consider a semi-discretization of the linear advection equation

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \tag{7}$$

on a uniform grid with mesh interval Δx. Using central differences, augmented by an artificial diffusive term $\sim \Delta x^3 \dfrac{\partial^3 u}{\partial x^3}$, this can be written as

$$\Delta t\frac{dv_j}{dt} = \frac{\lambda}{2}(v_{j+1} - v_{j-1}) - \lambda\mu(v_{j+2} - 4v_{j+1} + 6v_j - 4v_{j-1} + v_{j-2}), \tag{8}$$

where Δt is the assumed time step, and λ is the CFL number

$$\lambda = a\frac{\Delta t}{\Delta x}. \tag{9}$$

For a Fourier mode

$$v_j = \hat{v}(t)e^{i\omega x_j}, \tag{10}$$

this reduces to

$$\Delta t\frac{d\hat{v}}{dt} = z\hat{v}, \tag{11}$$

where z is the Fourier symbol of the space discretization. With $\xi = \omega\Delta x$,

$$z(\xi) = -\lambda\left(i\sin\xi + 4\mu(1-\cos\xi)^2\right). \tag{12}$$

For stability of the fully discrete scheme, the locus of z, as ξ is varied, should be inside the stability region of the time integration method. The permissible CFL number thus depends on the stability interval along the imaginary axis as well as the negative real axis. Fig. 1 shows the stability region of the standard fourth order RK4 scheme. It also shows the locus of the Fourier symbol for a CFL number of 2.8, $\mu = 1/32$.

3.2 Multistage Schemes for Steady State Problems

The next section focuses on explicit time-stepping schemes tailored for steady state problems. For these one can use simplified multistage schemes designed purely to maximize the stability region with out regard to time accuracy. As a starting point we may use the low storage m stage scheme

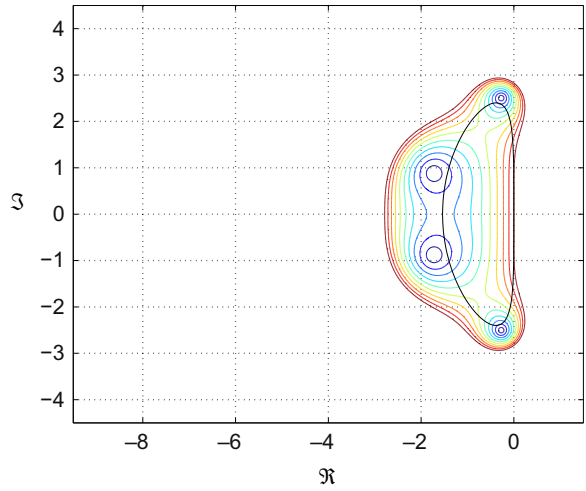

FIG. 1 Stability region for the standard four-stage RK scheme. *Black line*—Locus of Fourier symbol for linear advection with CFL number 2.4.

$$\begin{aligned}
\mathbf{w}^{(0)} &= \mathbf{w}^n \\
\mathbf{w}^{(1)} &= \mathbf{w}^{(0)} - \alpha_1 \Delta t \mathbf{R}\left(\mathbf{w}^{(0)}\right) \\
\mathbf{w}^{(2)} &= \mathbf{w}^{(0)} - \alpha_2 \Delta t \mathbf{R}\left(\mathbf{w}^{(1)}\right) \\
&\cdots
\end{aligned} \tag{13}$$

$$\begin{aligned}
\mathbf{w}^{(m)} &= \mathbf{w}^{(0)} - \Delta t \mathbf{R}(\mathbf{w}^{(m-1)}) \\
\mathbf{w}^{n+1} &= \mathbf{w}^{(m)}
\end{aligned} \tag{14}$$

The simplest choice is

$$\alpha_1 = \frac{1}{m}, \quad \alpha_2 = \frac{1}{m-1}, \ldots, \tag{15}$$

which gives mth-order accuracy for linear problems. It has been shown by Kinnmark (1984) how to chose the coefficients to maximize the stability interval along the imaginary axis, which should yield the largest possible time step for a pure convection problem.

With a diffusive or upwind-biased scheme the time-stepping scheme should also have a stability region which contains a substantial interval of the negative real axis. To achieve this it pays to treat the convective and dissipative terms in a distinct fashion. Accordingly the residual is split as

$$\mathbf{R}(\mathbf{w}) = \mathbf{Q}(\mathbf{w}) + \mathbf{D}(\mathbf{w}),$$

where $Q(w)$ is the convective part and $D(w)$ the dissipative part. Denote the time level $n\Delta t$ by a superscript n. Then the multistage time-stepping scheme is formulated as

$$\begin{aligned}
\mathbf{w}^{(\mathbf{0})} &= \mathbf{w}^{\mathbf{n}} \\
&\cdots \\
\mathbf{w}^{(\mathbf{k})} &= \mathbf{w}^{\mathbf{n}} - \alpha_k \Delta t \left(\mathbf{Q}^{(\mathbf{k-1})} + \mathbf{D}^{(\mathbf{k-1})}\right) \\
&\cdots \\
\mathbf{w}^{\mathbf{n+1}} &= \mathbf{w}^{(\mathbf{m})},
\end{aligned} \tag{16}$$

where the superscript k denotes the kth stage, $\alpha_m = 1$, and

$$\begin{aligned}
\mathbf{Q}^{(\mathbf{0})} &= \mathbf{Q}(\mathbf{w}^{\mathbf{n}}), \quad \mathbf{D}^{(\mathbf{0})} = \mathbf{D}(\mathbf{w}^{\mathbf{n}}) \\
&\cdots \\
\mathbf{Q}^{(\mathbf{k})} &= \mathbf{Q}\left(\mathbf{w}^{(\mathbf{k})}\right) \\
\mathbf{D}^{(\mathbf{k})} &= \beta_k \mathbf{D}\left(\mathbf{w}^{(\mathbf{k})}\right) + (1 - \beta_k)\mathbf{D}^{(k-1)}.
\end{aligned}$$

The coefficients α_k are chosen to maximize the stability interval along the imaginary axis, and the coefficients β_k are chosen to increase the stability interval along the negative real axis.

These schemes fall within the framework of additive Runge–Kutta schemes, and they have much larger stability regions (Jameson, 1985). Two schemes which have been found to be particularly effective are tabulated below. The first is a four-stage scheme with two evaluations of dissipation. Its coefficients are

$$\begin{aligned} \alpha_1 &= \frac{1}{3}, \quad \beta_1 = 1, \\ \alpha_2 &= \frac{4}{15}, \quad \beta_2 = \frac{1}{2}, \\ \alpha_3 &= \frac{5}{9}, \quad \beta_3 = 0, \\ \alpha_4 &= 1, \quad \beta_4 = 0. \end{aligned} \tag{17}$$

The second is a five-stage scheme with three evaluations of dissipation. Its coefficients are

$$\begin{aligned} \alpha_1 &= \frac{1}{4}, \quad \beta_1 = 1, \\ \alpha_2 &= \frac{1}{6}, \quad \beta_2 = 0, \\ \alpha_3 &= \frac{3}{8}, \quad \beta_3 = 0.56, \\ \alpha_4 &= \frac{1}{2}, \quad \beta_4 = 0, \\ \alpha_5 &= 1, \quad \beta_5 = 0.44. \end{aligned} \tag{18}$$

Figs. 2 and 3 display the stability regions for the 4-2 and 5-3 schemes defined by Eqs. (17) and (18).

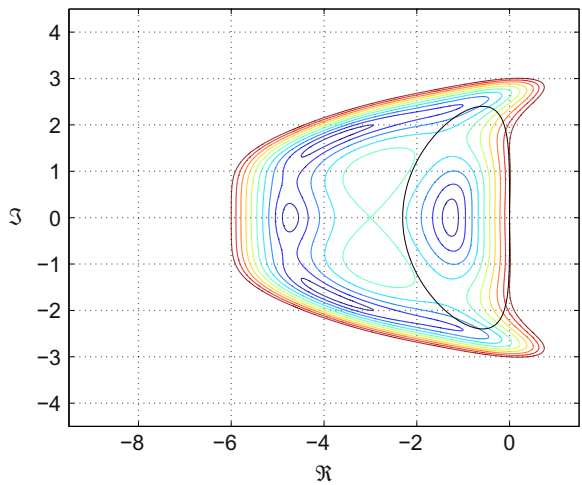

FIG. 2 Stability region for the 4-2 scheme (Eq. (17)). *Black line*—Locus of Fourier symbol for linear advection with CFL number 2.4.

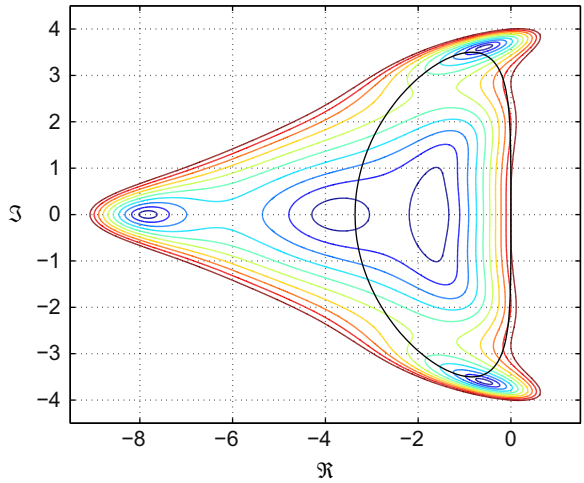

FIG. 3 Stability region for the 5-3 scheme (Eq. (18)). *Black line*—Locus of Fourier symbol for linear advection with CFL number 3.5.

The expansion of the stability region is apparent compared with the standard RK4 scheme as shown in Fig. 1. They also show the locus of the Fourier symbol for the model problem discussed in Section 3.1.

Typically calculations are performed on meshes with widely varying cell sizes, and the convergence to steady state can be significantly accelerated by using a variable local time step Δt corresponding to a fixed CFL number.

3.3 Implicit Schemes for Steady State Problems

A prototype implicit scheme suitable for steady state problems can be formulated by estimating $\dfrac{\partial \boldsymbol{w}}{\partial t}$ at $t + \epsilon\Delta t$ as a linear combination of $\mathbf{R}(\mathbf{w^n})$ and $\mathbf{R}(\mathbf{w^{n+1}})$. The resulting equation

$$\mathbf{w}^{n+1} = \mathbf{w}^n - \Delta t\left\{(1-\epsilon)\mathbf{R}(\mathbf{w^n}) + \epsilon\mathbf{R}\left(\mathbf{w}^{n+1}\right)\right\}, \tag{19}$$

represents the first-order accurate backward Euler scheme if $\epsilon = 1$, and the second-order accurate trapezoidal rule if $\epsilon = \dfrac{1}{2}$. In the two-dimensional case

$$\mathbf{R}(\mathbf{w}) = D_x\mathbf{f}(\mathbf{w}) + D_y\mathbf{g}(\mathbf{w}), \tag{20}$$

where $D_x\mathbf{f}$ and $D_y\mathbf{g}$ denote the discrete approximations to $\dfrac{\partial}{\partial x}\mathbf{f}$ and $\dfrac{\partial}{\partial y}\mathbf{g}$.

The implicit equation (19) can most readily be solved via linearization or by resorting to an iterative method. Defining the correction vector

$$\delta w = w^{n+1} - w^n,$$

Eq. (19) can be linearized by approximating the local fluxes as

$$f(w^{n+1}) \sim f(w^n) + A\delta w, \quad g(w^{n+1}) \sim g(w^n) + B\delta w,$$

where $A = \dfrac{\partial \mathbf{f}}{\partial \mathbf{w}}$ and $B = \dfrac{\partial \mathbf{g}}{\partial \mathbf{w}}$ are the Jacobian matrices, and the neglected terms are $O(\| \delta \mathbf{w} \|^2)$. This leads to a linearized implicit scheme which has the local form

$$\left\{I + \epsilon\Delta t(D_x A + D_y B)\right\}\delta w + \Delta t R(w) = 0. \tag{21}$$

Here we can recognize $D_x A + D_y B$ as $\dfrac{\partial R}{\partial w}$.

If one sets $\epsilon = 1$ and lets $\Delta t \to \infty$ this reduces to the Newton iteration for the steady state problem (20), which has been successfully used in two-dimensional calculations (Giles et al., 1985; Venkatakrishnan, 1988). In the three-dimensional case with, say, an $N \times N \times N$ mesh, the bandwidth of the matrix that must be inverted is of order N^2. Direct inversion requires a number of operations proportional to the number of unknowns multiplied by the square of the bandwidth, resulting in a complexity of the order of N^7. This is prohibitive, and forces recourse to either an approximate factorization method or an iterative solution method.

The main possibilities for approximate factorization are the alternating direction and LU decomposition methods. The alternating direction method, which may be traced back to the work of Gourlay and Mitchell (1966), was given an elegant formulation for nonlinear problems by Beam and Warming (1976). In a two-dimensional case equation (21) is replaced by

$$(I + \epsilon\Delta t D_x A)(I + \epsilon\Delta t D_y B)\delta w + \Delta t R(w) = 0, \tag{22}$$

where D_x and D_y are difference operators approximating $\partial/\partial x$ and $\partial/\partial y$, and A and B are the Jacobian matrices. This may be solved in two steps:

$$(I + \epsilon\Delta t D_x A)\delta w^* = -\Delta t R(w)$$
$$(I + \epsilon\Delta t D_y B)\delta w = \delta w^*.$$

Each step requires block tridiagonal matrix inversions and may be performed in $O(N^2)$ operations on an $N \times N$ mesh. The algorithm is amenable to vectorization by simultaneous solution of the tridiagonal system of equations along parallel coordinate lines. The method has been refined to a high level of efficiency by Pulliam and Steger (1985), and Yee (1985) has extended it to incorporate a TVD scheme. Its main disadvantage is that its extension to three dimensions is inherently unstable according to a von Neumann analysis.

The idea of the LU decomposition method (Jameson and Turkel, 1981) is to replace the operator in Eq. (13) by the product of lower and upper block triangular factors L and U,

$$LU\delta w + \Delta t R(w) = 0. \tag{23}$$

Two factors are used independent of the number of dimensions, and the inversion of each can be accomplished by inversion of its diagonal blocks. The method can be conveniently illustrated by considering a one-dimensional example. Let the Jacobian matrix $A = \partial \mathbf{f}/\partial \mathbf{w}$ be split as

$$A = A^+ + A^-, \tag{24}$$

where the eigenvalues of A^+ and A^- are positive and negative, respectively. Then we can take

$$L = I + \epsilon \Delta t D_x^- A^+, \quad U = I + \epsilon \Delta t D_x^+ A^-, \tag{25}$$

where D_x^+ and D_x^- denote forward and backward difference operators approximating $\partial/\partial x$. The reason for splitting A is to ensure the diagonal dominance of L and U, independent of Δt. Otherwise stable inversion of both factors will only be possible for a limited range of Δt. A crude choice is

$$A^{\pm} = \frac{1}{2}(A \pm \epsilon I), \tag{26}$$

where ϵ is at least equal to the spectral radius of A. If flux splitting is used in the calculation of the residual, it is natural to use the corresponding splitting for L and U. An interesting variation is to combine an alternating direction scheme with LU decomposition in the different coordinate directions (Obayashi and Kuwakara, 1984; Obayashi et al., 1986).

If one chooses to adopt the iterative solution technique, the principal alternatives are variants of the Jacobi and Gauss–Seidel methods. These may be applied to either the nonlinear equation (19) or the linearized equation (21). A Jacobi method of solving (19) can be formulated by regarding it as an equation

$$\mathbf{w} - \mathbf{w}^{(0)} + \epsilon \Delta t \mathbf{R}(\mathbf{w}) + (1 - \epsilon)\Delta t \mathbf{R}\left(\mathbf{w}^{(0)}\right) = 0, \tag{27}$$

to be solved for $\mathbf{w}$. Here $\mathbf{w}^{(0)}$ is a fixed value obtained as the result of the previous time step. Now using bracketed superscripts to denote the iterations, we have

$$\mathbf{w}^{(0)} = \mathbf{w}^{\mathbf{n}} \tag{28}$$

$$\mathbf{w}^{(1)} = \mathbf{w}^{(0)} + \Delta t \mathbf{R}\left(\mathbf{w}^{(0)}\right), \tag{29}$$

and for $k > 1$

$$\mathbf{w}^{(\mathbf{k}+1)} = \mathbf{w}^{(\mathbf{k})} + \sigma_{k+1}\left\{\left(\mathbf{w}^{(\mathbf{k})} - \mathbf{w}^{(0)} + \epsilon \Delta t \mathbf{R}\left(\mathbf{w}^{(\mathbf{k})}\right) + (1 - \epsilon)\Delta t \mathbf{R}\left(\mathbf{w}^{(0)}\right)\right)\right\}, \tag{30}$$

where the parameters σ_{k+l} can be chosen to optimize convergence. Finally, if we stop after m iterations,

$$\mathbf{w}^{\mathbf{n+1}} = \mathbf{w}^{(\mathbf{m})}. \tag{31}$$

We can express $\mathbf{w}^{(\mathbf{k+1})}$

$$\begin{aligned}\mathbf{w}^{(\mathbf{k+1})} = \mathbf{w}^{(\mathbf{0})} + (1+\sigma_{k+1})\left(\mathbf{w}^{(\mathbf{k})} - \mathbf{w}^{(\mathbf{0})}\right) \\ + \sigma_{k+1}\left\{\left(\epsilon\Delta t\mathbf{R}\left(\mathbf{w}^{(\mathbf{k})}\right) + (1-\epsilon)\Delta t\mathbf{R}\left(\mathbf{w}^{(\mathbf{0})}\right)\right)\right\}.\end{aligned} \tag{32}$$

Since

$$\mathbf{w}^{(\mathbf{1})} - \mathbf{w}^{(\mathbf{0})} = \sigma_1\Delta t\mathbf{R}\left(\mathbf{w}^{(\mathbf{0})}\right), \tag{33}$$

it follows that for all k we can express $\left(\mathbf{w}^{(\mathbf{k})} - \mathbf{w}^{(\mathbf{0})}\right)$ as a linear combination of $\mathbf{R}\left(\mathbf{w}^{(\mathbf{j})}\right)$, $j < k$. Thus this scheme is a variant of the multistage time-stepping scheme described by Eq. (13). It has the advantage that it permits simultaneous or overlapped calculation of the corrections at every mesh point and is readily amenable to parallel and vector processing.

Symmetric Gauss–Seidel schemes have proved to be particularly effective (Chakravarthy, 1984; Hemker and Spekreijse, 1984; MacCormack, 1985; Rieger and Jameson, 1988; Yoon and Jameson, 1987). Following the analysis of Jameson (1986a), consider the case of a flux split scheme in one dimension, for which

$$R(w) = D_x^+ f^-(w) + D_x^- f^+(w), \tag{34}$$

where the flux is split so that the Jacobian matrices

$$A^+ = \frac{\partial f^+}{\partial w} \quad \text{and} \quad A^- = \frac{\partial f^-}{\partial w}, \tag{35}$$

have positive and negative eigenvalues, respectively. Now Eq. (19) becomes

$$\left\{I + \epsilon\Delta t\left(D_x^+ A^- + D_x^- A^+\right)\right\}\delta w + \Delta t R(w) = 0. \tag{36}$$

At the j^{th} mesh point this is

$$\left\{I + \alpha\left(A_j^+ - A_j^-\right)\right\}\delta w_j + \alpha A_{j+1}^-\delta w_{j+1} - \alpha A_{j-1}^+\delta w_{j-1} + \Delta t R_j = 0, \tag{37}$$

where

$$\alpha = \epsilon\frac{\Delta t}{\Delta x}. \tag{38}$$

Set $\delta w_j^{(0)} = 0$. A two sweep symmetric Gauss–Seidel scheme is then

$$\begin{aligned}\left\{I + \alpha\left(A_j^+ - A_j^-\right)\right\}\delta w_j^{(1)} - \alpha A_{j-1}^+\delta w_{j-1}^{(1)} + \Delta t R_j = 0 \\ \left\{I + \alpha\left(A_j^+ - A_j^-\right)\right\}\delta w_j^{(2)} + \alpha A_{j+1}^-\delta w_{j+1}^{(2)} - \alpha A_{j-1}^+\delta w_{j-1}^{(1)} + \Delta t R_j = 0.\end{aligned}$$

Subtracting (1) from (2) we find that

$$\left\{I+\alpha\left(A_j^+-A_j^-\right)\right\}\delta w_j^{(2)}+\alpha A_{j+1}^-\delta w_{j+1}^{(2)}=\left\{I+\alpha\left(A_j^+-A_j^-\right)\right\}\delta w_j^{(1)}. \tag{39}$$

Define the lower triangular, upper triangular and diagonal operators L, U and D as

$$L=I-\alpha A^-+\epsilon t D_x^- A^+$$
$$U=I+\alpha A^++\epsilon t D_x^+ A^-$$
$$D=I+\alpha(A^+-A^-).$$

It follows that the scheme can be written as

$$LD^{-1}U\delta w=-\Delta t R(w). \tag{40}$$

Commonly the iteration is terminated after one double sweep. The scheme is then a variation of an LU implicit scheme.

If we use the simple choice (26) for $A^{\pm}$, D reduces to a scalar factor and the scheme requires no inversion. This is a significant advantage for the treatment of flows with chemical reaction with large numbers of species, and a correspondingly large numbers of equations.

The terms $\Delta t R_i-\alpha A_{i-1}^+\delta\tilde{v}_{i-1}$ of Eq. (37) are a linearization of $\Delta t R_i$ evaluated with $\tilde{v}_{i-1}=v_{i-1}+\delta\tilde{v}_{i-1}$. Following this line of reasoning, the *LU-SGS* scheme can be recast (Jameson and Caughey, 2001) as

$$\left\{I+\alpha\left(A_i^+-A_i^-\right)\right\}\delta\tilde{w}_i+\Delta t\tilde{R}_i=0; \tag{41}$$

$$\left\{I+\alpha\left(A_i^+-A_i^-\right)\right\}\delta\tilde{\tilde{w}}_i+\Delta t\tilde{\tilde{R}}_i=0, \tag{42}$$

where

$$\tilde{w}_i=w_i+\delta\tilde{w}_i;\quad \tilde{f}_i^{\pm}=f^{\pm}(\tilde{w}_i); \tag{43}$$

$$w_i^{n+1}=\tilde{\tilde{w}}_i=\tilde{w}_i+\delta\tilde{\tilde{w}}_i;\quad \tilde{\tilde{f}}_i^{\pm}=f^{\pm}(\tilde{\tilde{w}}_i); \tag{44}$$

and

$$\tilde{R}_i=\frac{1}{\Delta x}\left(f_{i+1}^--f_i^-+f_i^+-\tilde{f}_{i-1}^+\right), \tag{45}$$

$$\tilde{\tilde{R}}_i=\frac{1}{\Delta x}\left(\tilde{\tilde{f}}_{i+1}^--\tilde{f}_i^-+\tilde{f}_i^+-\tilde{f}_{i-1}^+\right). \tag{46}$$

With the definitions of Eq. (35), Eqs. (41) and (42) can be written as

$$\delta\tilde{w}_i=-\frac{\Delta t}{1+\mathcal{C}}\tilde{R}_i, \tag{47}$$

$$\delta\tilde{\tilde{w}}_i = -\frac{\Delta t}{1+\mathcal{C}}\tilde{\tilde{R}}_i, \tag{48}$$

where $\mathcal{C} = \rho\Delta t/\Delta x$ is the Courant number.

Alternatively, one may use the Jacobian splitting defined as

$$A^+ = \frac{1}{2}(A + |A|), \quad A^- = \frac{1}{2}(A - |A|), \tag{49}$$

where $|A| = V\,|\Lambda|V^{-1}$, and $|\Lambda|$ is the diagonal matrix whose entries are the absolute values of the eigenvalues of the Jacobian matrix A, while V and V^{-1} are the modal matrix of A and its inverse as defined in Section 2. Then Eqs. (41) and (42) can be written

$$\{I + \alpha|A|\}\delta\tilde{w}_i = -\Delta t\tilde{R}_i, \tag{50}$$

$$\{I + \alpha|A|\}\delta\tilde{\tilde{w}}_i = -\Delta t\tilde{\tilde{R}}_i, \tag{51}$$

and, in the limit as the time step Δt goes to infinity, these equations represent the *SGS* Newton iteration

$$|A|\delta\tilde{w}_i = -\Delta x\tilde{R}_i, \tag{52}$$

$$|A|\delta\tilde{\tilde{w}}_i = -\Delta x\tilde{\tilde{R}}_i. \tag{53}$$

The introduction of the splitting defined by Eq. (49) is motivated, in part, by the success of the similar preconditioner introduced by Allmaras (1993) and used by Pierce and Giles (1997) to accelerate the convergence of codes based on explicit Runge–Kutta time stepping. This preconditioner seems to have its roots in the diagonally dominant *ADI* scheme (Bardina and Lombard, 1987; MacCormack, 1997).

When the scheme corresponding to Eqs. (52) and (53) is implemented for the finite volume form (Jameson et al., 1981) of the equations, it can be represented (in two dimensions) as

$$\{|A| + |B|\}\delta\tilde{w}_{i,j} = -\sigma\tilde{R}_{i,j}, \tag{54}$$

$$\{|A| + |B|\}\delta\tilde{\tilde{w}}_{i,j} = -\sigma\tilde{\tilde{R}}_{i,j}, \tag{55}$$

where

$$\begin{aligned}\tilde{R}_{i,j} = {} & F^-_{i+1,j} - F^-_{i,j} + F^+_{i,j} - \tilde{F}^+_{i-1,j} + \\ & G^-_{i,j+1} - G^-_{i,j} + G^+_{i,j} - \tilde{G}^+_{i,j-1},\end{aligned} \tag{56}$$

$$\begin{aligned}\tilde{\tilde{R}}_{i,j} = {} & \tilde{\tilde{F}}^-_{i+1,j} - \tilde{F}^-_{i,j} + \tilde{F}^+_{i,j} - \tilde{F}^+_{i-1,j} + \\ & \tilde{\tilde{G}}^-_{i,j+1} - \tilde{G}^-_{i,j} + \tilde{G}^+_{i,j} - \tilde{G}^+_{i,j-1},\end{aligned} \tag{57}$$

and σ is a relaxation factor that can be used to optimize convergence rates. In these equations F^+, F^-, G^+ and G^- represent the split approximations of the flux vectors in the corresponding mesh coordinate directions.

Numerical experiments indicate that it can be beneficial to perform additional corrections in supersonic zones, when they are present in the solution. The CPU time required for these multiple sweeps is reduced by "freezing" the matrix coefficients $|A|$ and $|B|$ that appear in Eqs. (52) and (53). The additional memory required to store these coefficient matrices is minimized by storing only the symmetrized form of the Jacobians (which requires only seven additional quantities to be stored for each mesh cell).

3.4 Acceleration Methods

Acceleration methods fall into two main classes. The first class consists of modifications to the underlying differential equations which do not alter the steady state. Some ways of modifying the differential equations to accelerate the rate of convergence to a steady state are as follows:

1. to increase the speed at which disturbances are propagated through the domain,
2. to equalize the wave speeds of different types of disturbance, thereby enabling the use of larger time steps in the numerical scheme without violating the CFL condition,
3. to introduce terms which cause disturbances to be damped.

The second class of acceleration techniques consists of modifications to the numerical techniques. These range from modification to the time integration schemes to the adoption of general iterative methods for the solution of linear and nonlinear equations. The next sections explore both classes of acceleration methods.

3.4.1 Variable Local Time Stepping

An obvious way in which Eq. (1) can be modified without changing the steady state is to multiply the space derivatives by a preconditioning matrix P to produce

$$\frac{\partial w}{\partial t} + P\left(\frac{\partial}{\partial x} f(w) + \frac{\partial}{\partial y} g(w)\right) = 0 \tag{58}$$

in the two-dimensional case. The simplest choice of P is a diagonal matrix αI with more general choices of P being discussed in Section 3.4.3. Then, one can choose α locally so that the equations are advanced at the maximum CFL number permitted by the time integration scheme at every point in the domain. This is equivalent to using different local time steps at different points. Typically, one uses meshes with small cells adjacent to the body and increasingly large cells as the distance from the body increases, thus allowing increasingly large time steps in the far field.

As a model for this procedure, consider the wave equation in polar coordinates r and θ

$$\phi_{tt} = c^2 \left\{ \frac{1}{r} \frac{\partial}{\partial r} (r\phi_r) + \frac{1}{r^2} \phi_{\theta\theta} \right\}.$$

Suppose that the wave speed c is proportional to the radius, say $c = \alpha r$. Then,

$$\phi_{tt} = \alpha^2 \left\{ r \frac{\partial}{\partial r} (r\phi_r) + \phi_{\theta\theta} \right\}.$$

This has solutions of the form

$$\phi = \frac{1}{r^n} e^{-\alpha n t},$$

indicating the possibility of exponential decay.

In practice, there are often very large variations in the size of the cells in body fitted meshes, such as the O-meshes, typically used for airfoil calculations. Then, the use of variable local time steps with a fixed CFL number generally leads to an order of magnitude reduction in the number of time steps needed to reach a steady state.

3.4.2 Enthalpy Damping

Solutions of the wave equation in an infinite domain have constant energy. If, on the other hand, the wave equation is modified by the addition of a term $\alpha\phi_t$, the energy decays. The modified equation takes the form

$$\phi_{tt} + \alpha\phi_t = c^2(\phi_{xx} + \phi_{yy}).$$

Now, multiplying by ϕ_t and integrating the right-hand side by parts as before, we find that

$$\frac{d}{dt} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{1}{2} \left(\phi_t^2 + \phi_x^2 + \phi_y^2 \right) \; dx \; dy = - \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \alpha \phi_t^2 \; dx \; dy.$$

If $\alpha > 0$, the nonnegative integral on the left must continue to decrease as long as $\phi_t \neq 0$ anywhere in the domain. In fact, the over-relaxation method for solving Laplace's equation may be interpreted as a discretization of a damped wave equation, and the rate of convergence may be optimized by a proper choice of α.

Provided that the flow is irrotational, as can be expected in subsonic flow, the Euler equations are satisfied by solutions of the unsteady potential flow equation. This does not contain a term in ϕ_t. However, in unsteady potential flow

$$\frac{\partial \phi}{\partial t} + H - H_\infty = 0, \tag{59}$$

where H is the total enthalpy and H_∞ is the total enthalpy in the far field. This suggests that the difference $H - H_\infty$ can serve in the role of ϕ_t. Moreover, H is constant in steady solutions of the Euler equations, so such a term can be added without altering the steady state.

With enthalpy damping (Jameson, 1982) the Euler equations are modified by the addition of source terms proportional to $H - H_\infty$ as follows

$$\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial \mathbf{f}_1(\mathbf{w})}{\partial x_1} + \frac{\partial \mathbf{f}_2(w)}{\partial x_2} + \mathbf{s}(w) = 0,$$

where

$$\mathbf{w} = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho E \end{bmatrix}, \quad \mathbf{f}_1 = \begin{bmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ \rho u_1 H \end{bmatrix}, \quad \mathbf{f}_2 = \begin{bmatrix} \rho u_2 \\ \rho u_2 u_1 \\ \rho u_2^2 + p \\ \rho u_2 H \end{bmatrix}, \quad \mathbf{s} = \frac{\rho}{c^2}(H - H_\infty) \begin{bmatrix} 1 \\ u_1 \\ u_2 \\ H \end{bmatrix}.$$

The energy equation has a quadratic term in H, like a Riccati equation, which could be destabilizing. An alternative is to modify the energy equation to the form

$$\frac{\partial}{\partial t}\rho E + \frac{\partial}{\partial x_1}(\rho u_1 H) + \frac{\partial}{\partial x_2}(\rho u_2 H) + \beta(H - H_\infty) = 0,$$

which tends to drive H towards H_∞ if the coefficient β is positive.

3.4.3 Preconditioning

When the spatial derivatives are multiplied by a preconditioning matrix, as in Eq. (58), P is typically designed to equalize the eigenvalues, so that all the waves can be advanced with optimal time steps. van Leer et al. (1991) have proposed a symmetric preconditioner, which minimizes the ratio between the largest and smallest eigenvalues. When the equations are written in stream-aligned coordinates with the symmetrizing variables, which may be written in differential form as

$$d\tilde{w} = \left(\frac{dp}{c^2}, \frac{\rho}{c} du_1, \frac{\rho}{c} du_2, \frac{\rho}{c} du_3, \frac{dp}{c^2} - d\rho \right)^T,$$

it has the form

$$P = \begin{bmatrix} \frac{\tau}{\beta^2} M^2 & -\frac{\tau}{\beta} M & 0 & 0 & 0 \\ -\frac{\tau}{\beta} M & \frac{\tau}{\beta^2} + 1 & 0 & 0 & 0 \\ 0 & 0 & \tau & 0 & 0 \\ 0 & 0 & 0 & \tau & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

where

$$\beta = \tau = \sqrt{1 - M^2}, \quad \text{if} \quad M < 1,$$

or

$$\beta = \sqrt{M^2 - 1}, \quad \tau = \sqrt{1 - \frac{1}{M^2}}, \quad \text{if} \quad M \geq 1,$$

Turkel has proposed an asymmetric preconditioner designed to treat flow at low Mach numbers (Turkel, 1987). A special case of the Turkel preconditioner advocated by Smith and Weiss (1995) has the simple diagonal form

$$P = \begin{bmatrix} \epsilon^2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

when written in the symmetrizing variables. If ϵ^2 varies as M^2 as $M \to 0$, all the eigenvalues of PA depend only on the speed q. In order to improve the accuracy, the absolute Jacobian matrix $|A|$ appearing in the artificial diffusion should be replaced by $P^{-1}|PA|$. In effect, this makes the diffusion depends on the modified wave speeds. The use of preconditioners of this type can lead to instability at stagnation points, where there is a zero eigenvalue which cannot be equalized with the eigenvalues $\pm c$. With a judiciously chosen limit on ϵ^2 as $M \to 0$, they have been proved effective in treating low speed flows.

3.4.4 Residual Averaging

Another approach to increasing the time step is to replace the residual at each point by a weighted average of residuals at neighbouring points. Consider the multistage scheme described by Eq. (13). In the one-dimensional case, one might replace the residual R_j by the average

$$\bar{R}_j = \epsilon R_{j-1} + (1 - 2\epsilon) R_j + \epsilon R_{j+1} \tag{60}$$

at each stage of the scheme. This smooths the residuals and also increases the support of the scheme, thus relaxing the restriction on the time step imposed by the Courant–Friedrichs–Lewy condition. In the case of the model problem (7), this would modify the Fourier symbol (11) by the factor

$$A = 1 - 2\epsilon(1 - \cos\xi).$$

As long as $c < \frac{1}{4}$, the absolute value $|A|$ decreases with increasing wave numbers ξ in the range $0 \leq \xi \leq \pi$. If $\epsilon = \frac{1}{4}$, however, $\bar{R}_j = 0$ for the odd–even mode $R_j = (-1)^j$.

In order to avoid restriction on the smoothing coefficient, it is better to perform the averaging implicitly by setting

$$-\epsilon\bar{R}_{j-1}+(1-2\epsilon)\bar{R}_j-\epsilon\bar{R}_{j+1}=R_j. \tag{61}$$

This corresponds to a discretization of the inverse Helmholtz operator. For an infinite interval, this equation has the explicit solution

$$\bar{R}_j=\frac{1-r}{1+r}\sum_{q=-\infty}^{\infty}r^{|q|}R_{j+q}, \tag{62}$$

where

$$\epsilon=\frac{r}{(1-r)^2},\quad r<1.$$

Thus, the effect of the implicit smoothing is to collect information from residuals at all points in the field, with an influence coefficient which decays by a factor r at each additional mesh interval from the point of interest.

Consider the model problem (7). According to Eq. (61), the Fourier symbol (11) will be replaced by

$$Z=-\lambda\frac{i\sin\xi+4\mu(1-\cos\xi)^2}{1+2\epsilon(1-\cos\xi)}.$$

In the absence of dissipation,

$$|Z|=\lambda\left|\frac{\sin\xi}{1+2\epsilon(1-\cos\xi)}\right|.$$

Differentiating with respect to ξ, we find that $|Z|$ reaches a maximum when

$$\cos\xi=\frac{2\epsilon}{1+2\epsilon},\quad \sin\xi=\sqrt{1-\left(\frac{2\epsilon}{1+2\epsilon}\right)^2}.$$

Then, $|Z|$ attains the maximum value

$$|Z|_{\max}=\frac{1}{\sqrt{1+4\epsilon}}.$$

Accordingly, if λ^* is the stability limit of the scheme, the CFL number λ should satisfy

$$\lambda\le\lambda^*\sqrt{1+4\epsilon}.$$

It follows that we can perform stable calculations at any desired CFL number λ by using a large enough smoothing coefficient such that

$$\epsilon\ge\frac{1}{4}\left(\left(\frac{\lambda}{\lambda^*}\right)^2-1\right).$$

Implicit residual averaging was originally proposed by Jameson and Baker (1983). We note that this approach is similar to the simplified implicit scheme of Lerat et al. (1982). In practice, it has been found that the most rapid rate of convergence to a steady state is usually obtained with $\frac{\lambda}{\lambda^*}$ in the range of 2–3, or $\lambda \approx 8$ for a four-stage Runge–Kutta scheme.

3.5 Multigrid Methods

Radical improvements in the rate of convergence to a steady state can be realized by the multigrid time-stepping technique. The concept of acceleration by the introduction of multiple grids was first proposed by Fedorenko (1964). There is by now a fairly well-developed theory of multigrid methods for elliptic equations based on the concept that the updating scheme acts as a smoothing operator on each grid (Brandt, 1977; Hackbusch, 1978). This theory does not hold for hyperbolic systems. Nevertheless, it seems that it ought to be possible to accelerate the evolution of a hyperbolic system to a steady state by using large time steps on coarse grids so that disturbances will be more rapidly expelled through the outer boundary. Various multigrid time-stepping schemes designed to take advantage of this effect have been proposed (Anderson et al., 1986; Caughey, 1987; Hall, 1985; Hemker and Spekreijse, 1984; Jameson, 1983, 1986a,b; Ni, 1982).

One can devise a multigrid scheme using a sequence of independently generated coarser meshes by eliminating alternate points in each coordinate direction. In order to give a precise description of the multigrid scheme, subscripts may be used to indicate the grid. Several transfer operations need to be defined. First the solution vector on grid k must be initialized as

$$w_k^{(0)} = T_{k,k-1} w_{k-1},$$

where w_{k-1} is the current value on grid $k-1$, and $T_{k,k-1}$ is a transfer operator. Next it is necessary to transfer a residual forcing function such that the solution on grid k is driven by the residuals calculated on grid $k-1$. This can be accomplished by setting

$$P_k = Q_{k,k-1} R_{k-1}(w_{k-1}) - R_k\left[w_k^{(0)}\right],$$

where $Q_{k,k-1}$ is another transfer operator. Then $R_k(w_k)$ is replaced by $R_k(w_k) + P_k$ in the time-stepping scheme. Thus, the multistage scheme is reformulated as

$$w_k^{(1)} = w_k^{(0)} - \alpha_1 \Delta t_k \left[R_k^{(0)} + P_k\right] \qquad (63)$$

$$\cdots$$

$$w_k^{(q+1)} = w_k^{(0)} - \alpha_{q+1} \Delta t_k \left[R_k^{(q)} + P_k\right]. \qquad (64)$$

The result $w_k^{(m)}$ then provides the initial data for grid $k + 1$. Finally, the accumulated correction on grid k has to be transferred back to grid $k - 1$ with the aid of an interpolation operator $I_{k-1,k}$. With properly optimized coefficients multistage time-stepping schemes can be very efficient drivers of the multigrid process. A W-cycle of the type illustrated in Fig. 4 proves to be a particularly effective strategy for managing the work split between the meshes.

In a three-dimensional case the number of cells is reduced by a factor of eight on each coarser grid. On examination of the figure, it can therefore be seen that the work measured in units corresponding to a step on the fine grid is of the order of

$$1 + 2/8 + 4/64 + \cdots < 4/3,$$

and consequently the very large effective time step of the complete cycle costs only slightly more than a single time step in the fine grid.

This procedure has proved extremely successful for the solution of the Euler equations for inviscid flow. The most dramatic results to date have been

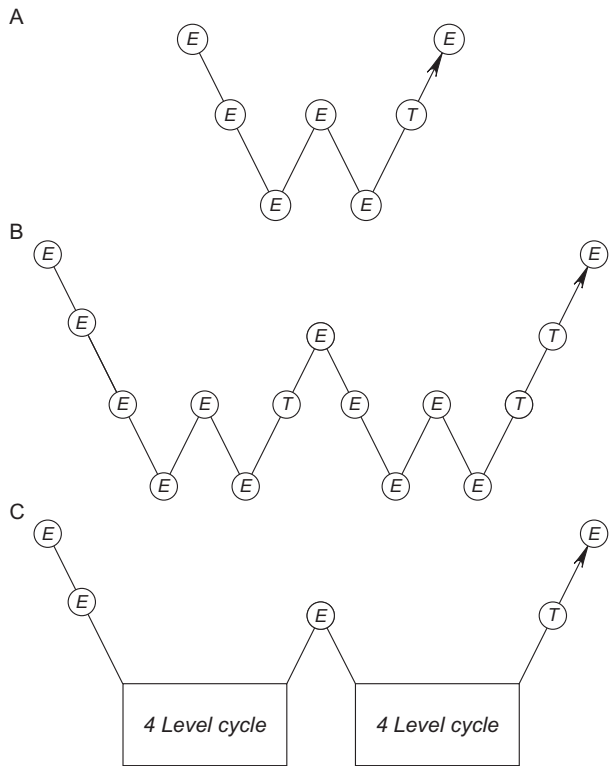

FIG. 4 Multigrid W-cycle for managing the grid calculation. E, evaluate the change in the flow for one step; T, transfer the data without updating the solution. (A) Three levels. (B) Four levels. (C) Five levels.

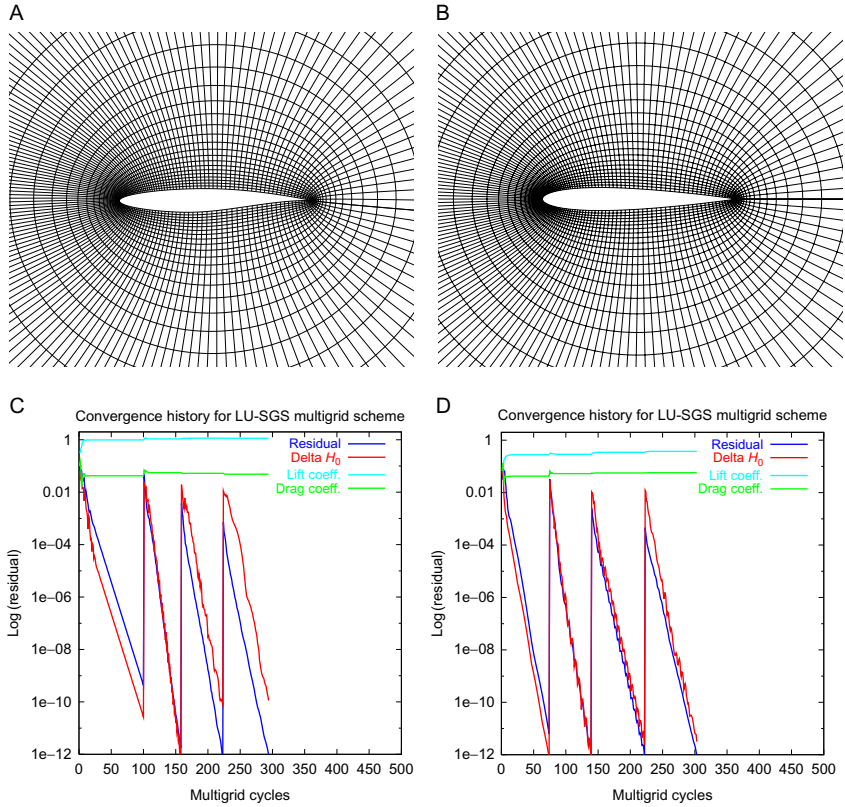

FIG. 5 Grid and convergence history of flow past the RAE 2822 at M_∞=0.75, α = 3 degrees and NACA 0012 airfoil at M_∞ = 0.8, α = 1.25 degrees. (A) and (C) RAE 2822. (B) and (D) NACA 0012.

achieved by using the nonlinear SGS scheme to drive a multigrid procedure using *W* cycles (Jameson and Caughey, 2001). Figs. 5, 6 and Table 1 illustrate the results for two-dimensional transonic flow calculations. In Fig. 6 the fully converged solution is shown by the solid lines, and it can be seen that the results are essentially fully converged after five cycles. This is an example of "text book" multigrid efficiency.

The multigrid method can be applied on unstructured meshes by interpolating between a sequence of separately generated meshes with progressively increasing cell sizes (Jameson and Mavriplis, 1987; Mavriplis and Jameson, 1990; Mavriplis and Martinelli, 1991; Peraire et al., 1992). It is not easy to generate very coarse meshes for complex configurations. An alternative approach, which removes this difficulty, is to automatically generate successively coarser meshes by agglomerating control volumes or by collapsing edges. This approach yields comparable rates of convergence and has proved

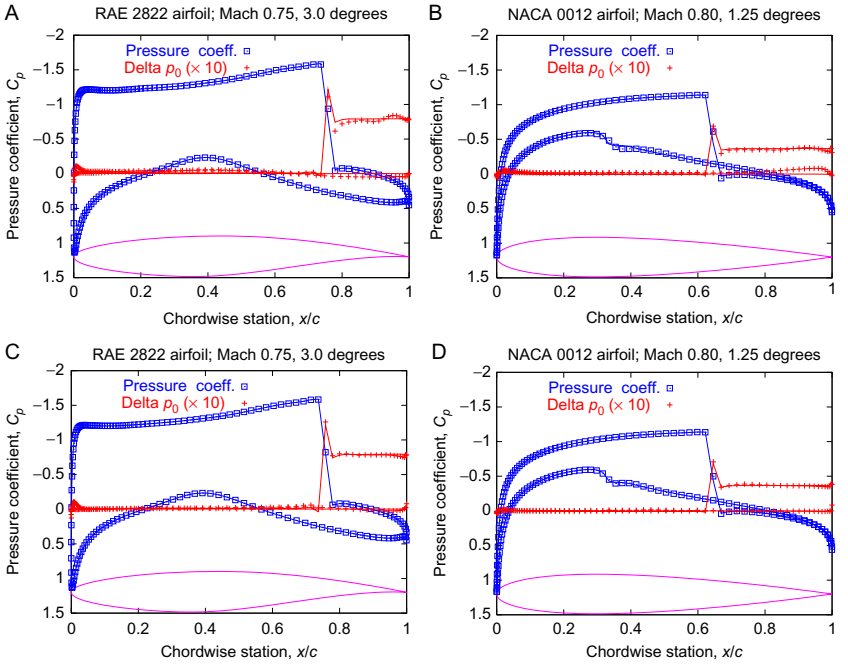

FIG. 6 Pressure distribution for flow past the RAE 2822 and NACA 0012 airfoil. *Solid lines* represent fully converged solution. (A) RAE 2822 after three cycles. (B) NACA 0012 after three cycles. (C) RAE 2822 after five cycles. (D) NACA 0012 after five cycles.

TABLE 1 Force Coefficients for the Fast, Preconditioned Multigrid Solutions Using CUSP Spatial Discretization

Case	Figures	MG Cycles	C_L	C_D
RAE 2822; $M_\infty = 0.75$; $\alpha = 3.00$	—	100	1.1417	0.04851
	Fig. 6C	5	1.1429	0.04851
	Fig. 6A	3	1.1451	0.04886
NACA 0012; $M_\infty = 0.80$; $\alpha = 1.25$	—	100	0.3725	0.02377
	Fig. 6D	5	0.3746	0.02391
	Fig. 6B	3	0.3770	0.02387

to be quite robust (Crumpton and Giles, 1995; Lallemand and Dervieux, 1987; Lallemand et al., 1992; Mavriplis and Venkatakrishnan, 1996).

3.6 RANS Equations

Multigrid methods have generally proved less effective in calculations of turbulent viscous flows using the Reynolds-averaged Navier–Stokes equations. These require highly anisotropic grids with very fine mesh intervals normal to the wall to resolve the boundary layers. While simple multigrid methods still yield fast initial convergence, they tend to slow down as the calculation proceeds to a low asymptotic rate. This has motivated the introduction of semicoarsening and directional coarsening methods (Allmaras, 1993, 1995, 1997; Mulder, 1989, 1992; Pierce and Giles, 1997; Pierce et al., 1997).

In 2007 Cord Rossow proposed using several iterations of an LUSGS scheme as a preconditioner at each stage of an explicit RK scheme (Rossow, 2007). This hybrid RKSGS scheme was further developed by Swanson et al. (2007), and it has been proven to be an effective driver of a full approximation multigrid scheme for steady state RANS calculations, yielding rates of convergence comparable to those that have been achieved for Euler calculations. Schemes of this type have also been extensively studied by the present authors and the following paragraphs describe an RKSGS scheme which has proved effective. The scheme is generally similar to the Swanson–Turkel–Rossow implementation but differs in some significant details.

In order to solve the equation

$$\frac{d\mathbf{w}}{dt} + \mathbf{R}(\mathbf{w}) = 0 \tag{65}$$

where $\mathbf{R}(\mathbf{w})$ represents the residual of the space discretization, an n stage RKSGS scheme is formulated as

$$\begin{aligned} \mathbf{w}^{(1)} &= \mathbf{w}^{(0)} - \alpha_1 \Delta t \;\; \mathbf{P}^{-1}\mathbf{R}^{(0)}, \\ \mathbf{w}^{(2)} &= \mathbf{w}^{(0)} - \alpha_2 \Delta t \;\; \mathbf{P}^{-1}\mathbf{R}^{(1)}, \\ &\cdots \\ \mathbf{w}^{(m)} &= \mathbf{w}^{(0)} - \Delta t \;\; \mathbf{P}^{-1}\mathbf{R}^{(m-1)}, \end{aligned} \tag{66}$$

where $\mathbf{P}$ denotes the LUSGS preconditioner. As in the case of the basis additive RK scheme the convective and dissipative parts of the residual are treated separately. Thus if $\mathbf{R}^{(k)}$ is split as

$$\mathbf{R}^{(k)} = \mathbf{Q}^{(k)} + \mathbf{D}^{(k)}, \tag{67}$$

then

$$\mathbf{Q}^{(0)} = \mathbf{Q}\left(\mathbf{w}^{(0)}\right), \;\; \mathbf{D}^{(0)} = \mathbf{D}\left(\mathbf{w}^{(0)}\right),$$

and for $k > 0$,

$$\begin{aligned}\mathbf{Q}^{(k)} &= \mathbf{Q}\left(\mathbf{w}^{(k)}\right),\\ \mathbf{D}^{(k)} &= \beta_k \mathbf{D}\left(\mathbf{w}^{(k)}\right) + (1-\beta_k)\mathbf{D}^{(k-1)},\end{aligned} \tag{68}$$

Two- and three-stage schemes which have proved effective are as follows. The coefficients of the two-stage scheme are

$$\alpha_1 = 0.24, \quad \beta_1 = 1, \alpha_2 = 1, \quad \beta_2 = 2/3,$$

while those of the three-stage scheme are

$$\begin{aligned}&\alpha_1 = 0.15, \quad \beta_1 = 1,\\ &\alpha_2 = 0.40, \quad \beta_2 = 0.5,\\ &\alpha_3 = 1, \quad \beta_3 = 0.5,\end{aligned}$$

While the residual is evaluated with a second-order accurate discretization, the preconditioner is based on a first-order upwind discretization. Without loss of generality we consider a quadrilateral cell, numbered zero, whose neighbours are numbered $k = 1$ to 4. The residual of a central difference scheme in this case is

$$\mathbf{R}_0^c = \frac{1}{S_0}\sum_{k=1}^{4} h_{k0} \tag{69}$$

where S_0 is the cell area and

$$h_{k0} = \frac{1}{2}[(\mathbf{f}_k + \mathbf{f}_0)\Delta y_{k0} - (\mathbf{g}_k + \mathbf{g}_0)\Delta x_{k0}]. \tag{70}$$

Introducing a Roe matrix $\mathbf{A}_{k0}$ satisfying

$$\mathbf{A}_{k0}(\mathbf{w}_k - \mathbf{w}_0) = (\mathbf{f}_k - \mathbf{f}_0)\Delta y_{k0} - (\mathbf{g}_k - \mathbf{g}_0)\Delta x_{k0}, \tag{71}$$

and noting that the sums

$$\sum_{k=1}^{4} \Delta y_{k0} = 0, \quad \sum_{k=1}^{4} \Delta x_{k0} = 0,$$

we can write

$$\mathbf{R}_0^c = \frac{1}{S_0}\sum_{k=1}^{4} \mathbf{A}_{k0}(\mathbf{w}_k - \mathbf{w}_0). \tag{72}$$

The residual of the first-order upwind scheme is obtained by subtracting $|\mathbf{A}_{k0}|(\mathbf{w}_k - \mathbf{w}_0)$ from h_{k0} to produce

$$\mathbf{R}_0 = \frac{1}{2S_0}\sum_{k=1}^{4} (\mathbf{A}_{k0} - |\mathbf{A}_{k0}|)(\mathbf{w}_k - \mathbf{w}_0).$$

Also noting that we can split $\mathbf{A}_{k0}$ as

$$\mathbf{A}_{k0} = \mathbf{A}_{k0}^{+} + \mathbf{A}_{k0}^{-},$$

where

$$\mathbf{A}_{k0}^{\pm} = \frac{1}{2}(\mathbf{A}_{k0} \pm |\mathbf{A}_{k0}|)$$

have positive and negative eigenvalues, respectively, we can write

$$\mathbf{R}_0 = \frac{1}{S_0}\sum_{k=1}^{4} \mathbf{A}_{k0}^{-}(\mathbf{w}_k - \mathbf{w}_0). \tag{73}$$

We now consider an implicit scheme of the form

$$\mathbf{w}^{n+1} = \mathbf{w}^{n} - \Delta t\big(\epsilon \mathbf{R}(\mathbf{w}^{n+1}) + (1-\epsilon)\mathbf{R}(\mathbf{w}^{n})\big)$$

and approximate $\mathbf{R}_0^{n+1}$

$$\mathbf{R}_0^{n+1} = \mathbf{R}_0^{n} + \frac{1}{S_0}\sum_{k=1}^{4} \mathbf{A}_{k0}^{-}(\delta\mathbf{w}_k - \delta\mathbf{w}_0),$$

where $\delta\mathbf{w}$ denotes $\mathbf{w}^{n+1} - \mathbf{w}^{n}$. This yields the approximate implicit scheme

$$\left(I - \epsilon\frac{\Delta t}{S_0}\sum_{k=1}^{4}\mathbf{A}_{k0}^{+}\right)\delta\mathbf{w}_0 + \epsilon\frac{\Delta t}{S_0}\sum_{k=1}^{4}\mathbf{A}_{k0}^{-}\delta\mathbf{w}_k = -\Delta t\mathbf{R}_0^{n}. \tag{74}$$

The LUSGS preconditioner uses symmetric Gauss–Seidel forward and backward sweeps to approximately solve the equation using the latest available values for $\delta\mathbf{w}_k$, and starting from $\delta\mathbf{w} = 0$. In practice it has been found that a single forward and backward sweep is generally sufficient, and very rapid convergence of the overall multigrid scheme can be obtained with both the two- and the three-stage schemes. Moreover, a choice of the coefficient $\epsilon < 1$ is effectively a way to over-relax the iterations, and it turns out that the fastest rate of convergence is obtained with ϵ around 0.65 for the two-stage scheme, and ϵ less than 0.5 for the three-stage scheme.

At each interface $\mathbf{A}^{-}$ is modified by the introduction of an entropy fix to bound the absolute values of the eigenvalues away from zero. Denoting the components of the unit normal to the edge by n_x and n_y, the normal velocity is

$$q_n = n_x u + n_y v$$

and the eigenvalues are

$$\lambda^{(1)} = q_n,$$

$$\lambda^{(2)} = q_n + c,$$

$$\lambda^{(3)} = q_n - c,$$

where c is the speed of sound. Velocity components, pressure p and density ρ at the interface may be calculated by arithmetic averaging and then $c = \sqrt{\frac{\gamma p}{\rho}}$.

The absolute eigenvalues $|\lambda^{(k)}|$ are then replaced by

$$e_k = \begin{cases} |\lambda^{(k)}|, & |\lambda^{(k)}| > a \\ \frac{1}{2}\left(a + \frac{{\lambda^{(k)}}^2}{a}\right), & |\lambda^{(k)}| \leq a \end{cases}$$

where a is set as a fraction of the speed of sound

$$a = d_1 c.$$

Without this modification the scheme diverges. In numerical experiments it has been found that the overall scheme converges reliably for transonic flow with $d_1 \approx 0.10$. The modified eigenvalues of $\mathbf{A}^-$ are then

$$\lambda^{(1)} = q_n - e_1,$$
$$\lambda^{(2)} = q_n + c - e_2,$$
$$\lambda^{(3)} = q_n - c - e_3.$$

It also proves helpful to further augment the diagonal coefficients by a term proportional to a fraction d_2 of the normal velocity q_n. The interface matrix is also modified to provide for contributions from the viscous Jacobian. The final interface matrix $\mathbf{A}^{\pm}$ can be conveniently represented via a transformation to the symmetrizing variables defined in differential form as

$$d\hat{\mathbf{w}}^T = \left[\frac{dp}{c^2}, \ \frac{\rho}{c}du, \ \frac{\rho}{c}dv, \ \frac{dp}{c^2} - d\rho\right]$$

Let s be the edge length. Then the interface matrix can be expressed as

$$\mathbf{A}^I = s\tilde{\mathbf{M}}\left(\tilde{\mathbf{A}}_c + \frac{s}{\rho S_0}\tilde{\mathbf{A}}_v - d_2 q_n I\right)\tilde{M}^{-1},$$

where $\tilde{\mathbf{A}}_c$ and $\tilde{\mathbf{A}}_v$ are the convective and viscous contributions, $d_2 q_n I$ is the diagonal augmentation term, and $\tilde{\mathbf{M}}$ is the transformation matrix from the conservative to the symmetrizing variables. Here

$$\tilde{\mathbf{M}} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ u & c & 0 & -u \\ v & 0 & c & -v \\ H & uc & vc & -\frac{q^2}{2} \end{bmatrix}$$

and

$$\tilde{\mathbf{M}}^{-1} = \begin{bmatrix} \tilde{\gamma}\dfrac{q^2}{2} & -\tilde{\gamma}u & -\tilde{\gamma}v & \tilde{\gamma} \\ -\dfrac{u}{c} & \dfrac{1}{c} & 0 & 0 \\ -\dfrac{v}{c} & 0 & \dfrac{1}{c} & 0 \\ \tilde{\gamma}\dfrac{q^2}{2} - 1 & -\tilde{\gamma}u & -\tilde{\gamma}v & \tilde{\gamma} \end{bmatrix}$$

where

$$q^2 = u^2 + v^2, \quad \tilde{\gamma} = \frac{\gamma - 1}{c^2}.$$

Define

$$r_1 = \frac{1}{2}(q_2 + q_3 - 1),$$
$$r_2 = \frac{1}{2}(q_2 - q_3).$$

Then the modified convective Jacobian is

$$\tilde{\mathbf{A}}_c = \begin{bmatrix} r_1 + q_1 & n_x r_2 & n_y r_2 & 0 \\ n_x r_2 & n_x^2 r_1 + q_1 & n_x n_y r_1 & 0 \\ n_y r_2 & n_x n_y r_1 + q_1 & n_y^2 r_1 + q_1 & 0 \\ 0 & 0 & 0 & q_1 \end{bmatrix}.$$

Also let u, λ and κ be the viscosity, bulk viscosity and conductivity coefficients. Then the viscous Jacobian is

$$\tilde{\mathbf{A}} = \begin{bmatrix} -(\gamma - 1)\kappa & 0 & 0 & -\kappa \\ 0 & -(\mu + n_x^2\mu^*) & -n_x n_y \mu^* & 0 \\ 0 & -n_x n_y \mu^* & -(\mu + n_y^2 \mu^*) & 0 \\ -(\gamma - 1)\kappa & 0 & 0 & -\mu \end{bmatrix}$$

where

$$\mu^* = \mu + \gamma,$$

and typically $\lambda = -\dfrac{2}{3}\mu$. Here it is assumed that the Reynolds stress is modelled by an eddy viscosity. Then if μ_m and μ_t are the molecular and eddy viscosity,

$$\mu = \mu_m + \mu_t,$$

while it is the common practice to take

$$\kappa = \frac{\mu_m}{Pr} + \frac{\mu_t}{Pr_t},$$

where Pr and Pr_t are the molecular and turbulent Prandtl numbers.

An alternative implementation can be derived from a first-order flux vector split scheme with the interface flux

$$h_{k0} = (\mathbf{f}_0^+ + \mathbf{f}_k^-)\Delta y_{k0} - (\mathbf{g}_0^+ + \mathbf{g}_k^-)\Delta x_{k0},$$

where $\mathbf{f}$ and $\mathbf{g}$ are split as

$$\mathbf{f} = \mathbf{f}^+ + \mathbf{f}^-, \quad \mathbf{g} = \mathbf{g}^+ + \mathbf{g}^-$$

and $\mathbf{f}^\pm$ and $\mathbf{g}^\pm$ have positive and negative eigenvalues, respectively. In this case the implicit equation (74) should be replaced by

$$\left(I - \frac{\epsilon\Delta t}{S_0}\sum_{k=1}^{4}\mathbf{A}_{k0}^+\right)\delta\mathbf{w}_0 + \frac{\epsilon\Delta t}{S_0}\sum_{k=1}^{4}\mathbf{A}_{k0}^-\delta\mathbf{w}_k = -\Delta t\mathbf{R}_0^n.$$

where

$$\mathbf{A}_{k0}^+ = \Delta y_{k0}\frac{\partial\mathbf{f}^+}{\partial\mathbf{w}} - \Delta x_{k0}\frac{\partial\mathbf{g}^+}{\partial\mathbf{w}}$$

and

$$\mathbf{A}_{k0}^- = \Delta y_{k0}\frac{\partial\mathbf{f}^-}{\partial\mathbf{w}} - \Delta x_{k0}\frac{\partial\mathbf{g}^-}{\partial\mathbf{w}}.$$

This formula is similar to the scheme proposed by Swanson et al. (2007), but the consistent procedure is then to evaluate $\mathbf{A}_{k0}^+$ at cell zero, and $\mathbf{A}_{k0}^-$ at cell k. Numerical tests indicate that the alternative formulations work about equally well. In any case these preconditioners may be applied with alternative second-order schemes for the residual $\mathbf{R}_0$ on the right-hand side. It turns out that the scheme works very well when $\mathbf{R}_0$ is evaluated by the JST scheme (Jameson et al., 1981) provided that the artificial viscosity coefficients are suitably tuned.

Fig. 7 shows the result for a standard test case, the RAE 2822, case 6, calculated with the JST scheme and a Baldwin–Lomax turbulence model.

4 NEWTON–KRYLOV METHODS

4.1 Background

The most popular direct iterative method is the Newton–Krylov method. As pointed out previously, an algorithm for obtaining steady solutions can also be used for time accurate computations of unsteady flows in combination with an implicit time-marching method. Hence Newton–Krylov methods are equally applicable to steady and unsteady flows; our emphasis here is on their application to the former.

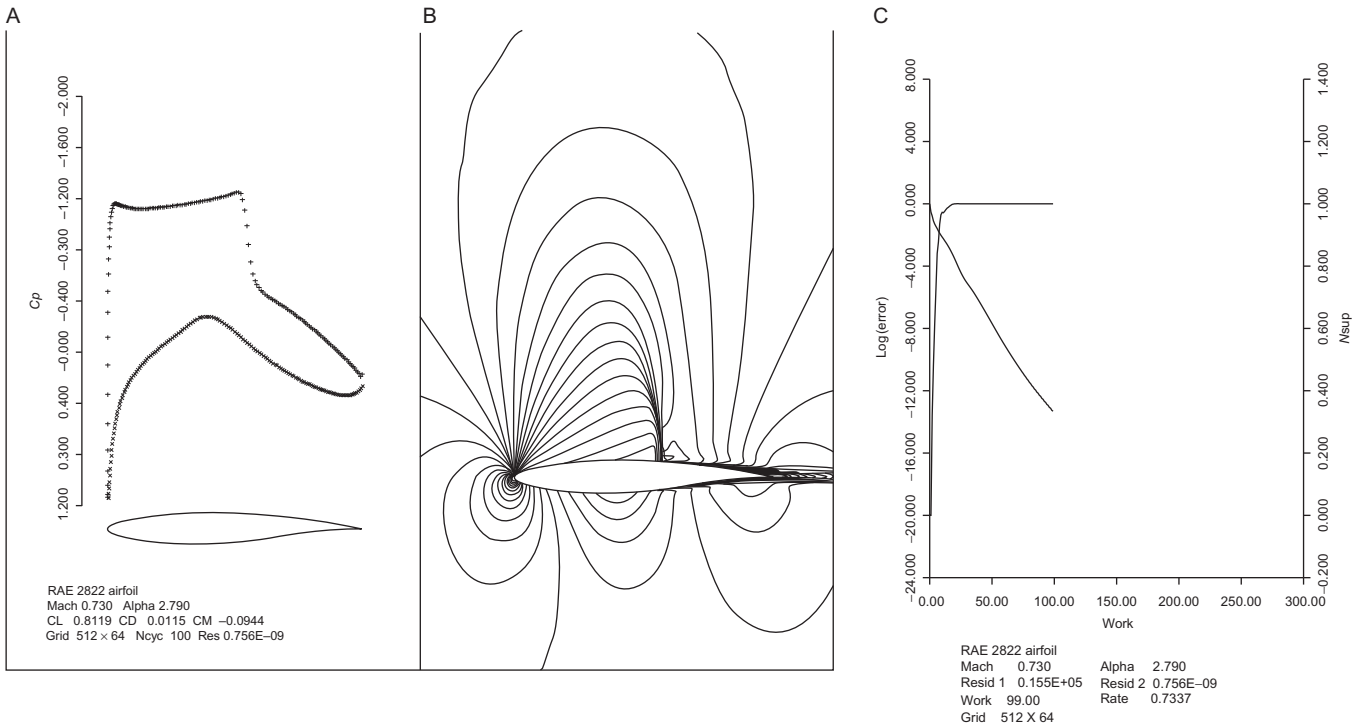

FIG. 7 RAE 2822 airfoil, $Ma = 0.730$, $\alpha = 2.790$. RKSGS scheme using the Baldwin–Lomax turbulence model and the JST scheme. (A) C_p distribution; (B) Mach contours; (C) the convergence history.

After discretization of the spatial derivatives in the Euler or Navier–Stokes equations and the turbulence model equations, if applicable, a system of ordinary differential equations is obtained (1). The steady state solution, if one exists, is the solution to the nonlinear algebraic system of equations given by

$$\mathbf{R}(\mathbf{w}) = 0. \tag{75}$$

The Newton method is a natural choice for the solution of nonlinear algebraic equations due to its quadratic convergence property if certain conditions are met. Application of the Newton method to large-scale problems in computational fluid dynamics has historically been limited by two issues. The first is that the method requires the solution of a large linear system at each iteration, and direct solution of these linear systems scales very badly with problem size and is generally infeasible for large-scale problems on current computing hardware. The second issue is that the Newton method will converge only if the initial guess is sufficiently close to the converged solution; in practical problems it is generally only possible to provide a suitable initial guess through a globalization procedure.

The first issue can be addressed through the use of iterative methods for linear algebraic systems, leading to the so-called *inexact* Newton method where the linear problem is solved iteratively to some tolerance at each Newton iteration. While this enables large-scale problems to be solved, it introduces the challenge of finding an iterative method that converges successfully for the linear systems encountered. Krylov methods for nonsymmetric linear systems, among which GMRES (Saad and Schultz, 1986) is most commonly used, have proven to be effective in this context and will be the focus of this section, hence the title *Newton–Krylov methods*.

The most common approach to globalization of the Newton method is pseudo-transient continuation in which a nontime-accurate time-dependent path is taken to partially converge the solution in order to provide a suitable initial solution for an inexact Newton method. This leads back to the time-marching methods described in Section 3. With this approach, the Newton–Krylov method is divided into two phases: a globalization phase followed by an inexact Newton phase. Recently, some success has been achieved through the use of homotopy continuation as an alternative to pseudo-transient continuation for the globalization phase (Brown and Zingg, 2016; Yu and Wang, 2016).

While it is natural to classify time-marching methods as explicit or implicit, in application to practical problems it is more accurate to view various methods as lying somewhere on a spectrum ranging from fully explicit to fully implicit, where implicitness is associated with the degree of coupling among solution variables as the solution is advanced iteratively (Pulliam and Zingg, 2014). In a purely explicit scheme, e.g., the explicit Euler time-marching method, the solution is advanced in time locally, without knowledge of the changes in the solution occurring at other nodes in the mesh.

With the Newton method, the solution is fully coupled, i.e., fully implicit, as all variables at all nodes, including boundary conditions, are advanced concurrently. The methods described in Section 3 lie somewhere between these extremes. For example, a multistage method uses intermediate solutions to increase the number of mesh nodes that are coupled as the solution advances. Similarly, implicit residual smoothing, multigrid and approximate factorization accomplish increased coupling in different ways.

In this section, we will discuss fully implicit methods, in particular Newton–Krylov methods that include a globalization phase. Our focus is on methods that are fully coupled, such that the mean-flow and turbulence model equations are coupled, and the boundary conditions and any interface conditions between mesh blocks are fully implicit. First, it is important to understand the motivation behind such methods and the problems for which they are likely to be an efficient choice. For this purpose, the concept of stiffness is relevant; with increasing stiffness an implicit approach becomes more efficient. Stiffness can arise through physics, for example, in chemically reacting flows, or through numerics, for example the high aspect ratio grid cells needed for efficient spatial resolution of turbulent flows at high Reynolds numbers in the solution of the RANS equations. Fully implicit methods can also be advantageous for equations with stiff source terms, for example, in a turbulence model. Finally, fully implicit methods have become increasingly popular when high-order spatial discretization is used, for example, through discontinuous Galerkin, flux reconstruction, or summation-by-parts methods, as such methods typically lead to increased stiffness (Persson and Peraire, 2008).

The Newton–Krylov approach to be described here is based on that developed in the third author's research group over the past 20 years, as described in the following four papers: Pueyo and Zingg (1998), Chisholm and Zingg (2009), Hicken and Zingg (2008) and Osusky and Zingg (2013). For a review of Newton–Krylov methods, the reader is directed to Knoll and Keyes (2004).

4.2 Methodology

4.2.1 Inexact Newton Method

Application of the implicit Euler method to (1) with a local time linearization gives

$$\left(\frac{\mathbf{I}}{\Delta t}+\mathbf{A}^{(n)}\right)\Delta\mathbf{w}^{(n)}=-\mathbf{R}(\mathbf{w}^{(n)}), \tag{76}$$

where

$$\mathbf{A}=\frac{\partial\mathbf{R}}{\partial\mathbf{w}} \tag{77}$$

is the Jacobian of the discrete residual vector. It is easy to see that the Newton method is obtained in the limit as Δt goes to infinity. This linear system must

be solved at each iteration. In the inexact Newton method, an iterative method is used for this purpose, and the system is solved to a specified finite relative tolerance, η_n, i.e.

$$\left|\left|\left(\mathbf{T}^{(n)}+\mathbf{A}^{(n)}\right)\Delta\mathbf{w}^{(n)}+\mathbf{R}(\mathbf{w}^{(n)})\right|\right|_2 \leq \eta_n||\mathbf{R}(\mathbf{w}^{(n)})||_2. \tag{78}$$

Here the term containing the time step has been modified to allow for a local time step as well as potentially a time step that varies between equations; for example the turbulence model equation can have a different time step from the mean-flow equations. Strictly speaking, this term is zero in an inexact Newton method, but we retain it here, as it is included in the pseudo-transient continuation approach to globalization. The sequence of relative tolerance values, η_n, affects both efficiency and robustness. If η_n is too small, the linear system will be over-solved, leading to an increase in computing time; if it is too large, the number of Newton iterations will increase, similarly leading to suboptimal performance. While an adaptive approach can be used that enables q-superlinear convergence (Eisenstat and Walker, 1996), this does not usually lead to the fastest performance in terms of CPU time. In their Newton–Krylov algorithm for the Euler equations, Hicken and Zingg (2008) use a sequence of values during the inexact Newton phase (which is preceded by a globalization phase) decreasing from an initial value of 0.5 to 0.01 based on the following formula (Eisenstat and Walker, 1996):

$$\eta_n = \max\left(0.01, \eta_{n-1}^{(1+\sqrt{5})/2}\right). \tag{79}$$

In contrast, Osusky and Zingg (2013) use $\eta_n = 0.01$ throughout the inexact Newton phase in solving the RANS equations. However, a maximum number of GMRES iterations is often reached before this tolerance is achieved.

4.2.2 Jacobian-Free Newton–Krylov Methods

One of the attributes of a Krylov method for solving linear systems in the form $\mathbf{Ax} = \mathbf{b}$, such as GMRES, is that the matrix $\mathbf{A}$ itself is never needed explicitly. What is needed is the product of $\mathbf{A}$ with an arbitrary vector $\mathbf{v}$. Therefore, explicit formation and storage of the Jacobian matrix can be avoided by approximating these matrix–vector products by a finite difference approximation to a Frechet derivative as follows:

$$\mathbf{A}^{(n)}\mathbf{v} \approx \frac{\mathbf{R}(\mathbf{w}^{(n)}+\epsilon\mathbf{v})-\mathbf{R}(\mathbf{w}^{(n)})}{\epsilon}. \tag{80}$$

The parameter ϵ must be chosen to balance truncation and round-off errors. Both Hicken and Zingg (2008) and Osusky and Zingg (2013) use the following formula from Nielsen et al. (1995):

$$\epsilon = \sqrt{\frac{N\delta}{\mathbf{v}^T\mathbf{v}}}, \tag{81}$$

where N is the number of unknowns, and δ is a small number that is further discussed below.

Although the use of (80) avoids the need to form and store the Jacobian matrix in the application of GMRES, it is generally necessary to precondition the linear system prior to the application of GMRES. Some of the most efficient preconditioners require the formation of a matrix that is an approximation to the Jacobian. With this approach, which is the subject of the next section, the method is not matrix free, but the approximate Jacobian used in the formation of the preconditioner generally requires less storage than the exact Jacobian.

4.2.3 Preconditioning and Parallelization

The Jacobian matrices arising from the discretization of the Euler and RANS equations are typically poorly conditioned such that GMRES will converge very slowly unless the system is first preconditioned. The choice of preconditioner has a major impact on the overall speed and robustness of the algorithm. Pueyo and Zingg (1998) studied numerous preconditioners and showed that an incomplete lower-upper (ILU) factorization with some fill based on a level of fill approach, ILU(p), is an effective option, where p is the level of fill (Meijerink and van der Vorst, 1977). The level of fill is an important parameter that is discussed further below. Moreover, a block implementation of ILU(p) is preferred over a scalar implementation (Hicken and Zingg, 2008).

The process of computing the incomplete factorization requires the formation and storage of a matrix. One option is to form the preconditioner based on an approximation of the Jacobian matrix rather than forming the complete Jacobian matrix. An approximation that includes nearest neighbour entries only reduces the size of the matrix considerably. In a typical second-order spatial discretization, next to nearest neighbour entries are associated with the numerical dissipation and cross-derivative viscous and diffusive terms. The cross-derivative terms can be neglected in forming the approximate Jacobian matrix upon which the ILU factorization is based. With an upwind spatial discretization, a first-order version can be used in the approximate Jacobian. Similarly, with a centred scheme with added second- and fourth-difference numerical dissipation, the approximate Jacobian can be formed by eliminating the fourth-difference dissipation terms and increasing the coefficient of the second-difference dissipation. This can be achieved by adding a factor σ times the fourth-difference coefficient to the second-difference coefficient in the approximate Jacobian. Values of σ between 4 and 6 are effective in the solution of the Euler equations (Hicken and Zingg, 2008), while higher values are preferred for the RANS equations, for example $\sigma = 5$ with scalar dissipation and $10 \leq \sigma \leq 12$ with matrix dissipation (Osusky and Zingg, 2013).

Pueyo and Zingg (1998) showed that an ILU(p) preconditioner based on such an approximate preconditioner is substantially more effective than an ILU(p) factorization of the full Jacobian. This arises from the off-diagonal

dominance of the full Jacobian matrix, which leads to poorly conditioned incomplete factors such that the long recurrences associated with the backward and forward solves can be unstable. The reduced off-diagonal dominance associated with the approximate Jacobian formed in the manner described above alleviates this behaviour. This conclusion was reached in the context of a typical second-order spatial discretization. Further study is needed in order to determine the optimal approach when high-order methods are used.

Two popular strategies for developing a parallel preconditioner are the additive-Schwarz (Saad, 2003) and approximate-Schur (Saad and Sosonkina, 1999) methods. Some form of domain decomposition is needed to assign blocks of data to unique processes. The incomplete factorization is then applied only to the local submatrices. The approximate-Schur approach necessitates the use of a flexible linear solver such as FGMRES (Saad, 1993). Hicken et al. (2010) found the approximate-Schur preconditioner to outperform the additive-Schwarz preconditioner when over 100 processors are used. Osusky and Zingg (2013) showed that the approximate-Schur approach scales well for over 6000 processors.

4.2.4 Globalization

A continuation or globalization phase provides a suitable solution to initiate the inexact Newton phase. Recently Yu and Wang (2016) and Brown and Zingg (2016) have shown promising results with homotopy continuation. However, the predominant approach to date is known as pseudo-transient continuation, in which a time-marching method is applied in a nontime-accurate manner to the system of ordinary differential equations (1). For a fully implicit solver, it is natural to use the implicit Euler method with a local time linearization and local time stepping:

$$\left(\mathbf{T}^{(n)} + \mathbf{A}^{(n)}\right)\Delta\mathbf{w}^{(n)} = -\mathbf{R}(\mathbf{w}^{(n)}), \tag{82}$$

where $\mathbf{T}^{(n)}$ is a diagonal matrix containing the reciprocal of the local time step. When an implicit algorithm is applied in this *delta* form, the converged steady solution is independent of the left-hand side matrix (Lomax et al., 2001). Consequently, since we seek neither time accuracy nor quadratic convergence during the globalization phase, in addition to the use of local time stepping, the Jacobian matrix can be approximated and a lagged Jacobian update can be used (Hicken and Zingg, 2008; Osusky and Zingg, 2013).

Both robustness and efficiency can be improved by substituting the approximate Jacobian matrix used in the formation of the preconditioner for $\mathbf{A}^{(n)}$ in (82). Hence this globalization is often described as the *approximate Newton* phase. This substantially reduces the number of iterations needed by the linear solver to achieve the specified convergence tolerance. Formation of the ILU factorization is one of the more expensive tasks of a nonlinear iteration. Consequently, it can be advantageous to reuse the approximate Jacobian

for three to five nonlinear iterations, thereby reducing the number of times the preconditioner must be formed during the approximate Newton phase.

In addition to a spatially varying local time step, the time step is steadily increased during the approximate Newton phase according to the formula

$$\Delta t_{\text{ref}}^{(n)} = ab^{m\left\lfloor \frac{n}{m} \right\rfloor}, \tag{83}$$

where the floor operator returns the largest integer less than or equal to its argument, $\Delta t_{\text{ref}}^{(n)}$ is a reference time step used in calculating the local time step, and m is the Jacobian update period. If the Jacobian is updated every iteration, then (83) simplifies to

$$\Delta t_{\text{ref}}^{(n)} = ab^{n}. \tag{84}$$

Suitable values of a and b are discussed in Section 4.2.5.

The switch from the approximate Newton phase to the inexact Newton phase is made when the normalized residual falls below a specified tolerance τ, i.e.

$$R_d^{(n)} = \frac{||\mathbf{R}^{(n)}||_2}{||\mathbf{R}^{(0)}||_2} < \tau. \tag{85}$$

If the parameter τ is too large, the inexact Newton phase will not converge. If it is too small, then the overall solution time will be increased. Suitable values of τ are presented in Section 4.2.5.

It is advantageous for robustness to include a local time step during the inexact Newton phase as well; this should increase rapidly as the residual decreases. An approach based on the successive evolution relaxation approach (Mulder and van Leer, 1985) is effective for this purpose. The reference time step used in computing the local time step increases according to the following formula:

$$\Delta t_{\text{ref}}^{(n)} = \max\left[\alpha\left(R_d^{(n)}\right)^{-\beta},\ \Delta t_{\text{ref}}^{(n-1)}\right], \tag{86}$$

where $3/2 \leq \beta \leq 2$. A smooth transition between the approximate and inexact Newton phases is obtained with

$$\alpha = ab^{m\left\lfloor \frac{n_d}{m} \right\rfloor}\left(R_d^{(n_d)}\right)^{\beta}, \tag{87}$$

where n_d is the index of the first inexact Newton iteration, while a and b are consistent with (83). Here m is the period of formation of the approximate Jacobian matrix and its factorization used for preconditioning. If this is updated at each Newton iteration, (87) becomes

$$\alpha = ab^{n_d}\left(R_d^{(n_d)}\right)^{\beta}. \tag{88}$$

4.2.5 Additional Considerations and Algorithm Parameters

The parallel Newton–Krylov algorithm described earlier, consisting of an approximate Newton pseudo-transient continuation phase followed by a Jacobian-free inexact Newton phase, both based on an approximate Schur parallel preconditioner using local block ILU(p) factorization, is an effective algorithm, but there are some additional considerations to ensure reliable performance. Moreover, the algorithm includes several user defined parameters that must be selected appropriately to provide the desired balance between robustness and fast convergence to steady state. Fortunately, sets of parameters can be found that provide excellent performance over a wide range of flow problems. Moreover, parameter tuning is possible to obtain convergence for particularly challenging problems. The parameters presented below have been used extensively in the context of gradient-based aerodynamic shape optimization, which provides a stiff challenge to a flow solver as a result of the many different geometries encountered and the need for reliable flow solution in order for the optimization algorithm to converge. Challenging aerodynamic shape optimization examples based on the application of the parallel Newton–Krylov algorithm for the Euler and RANS equations described here can be found in Hicken and Zingg (2010), Gagnon and Zingg (2016), Osusky et al. (2015) and Reist and Zingg (2016).

The accuracy of an incomplete lower-upper factorization is sensitive to the matrix structure, e.g., its bandwidth, and therefore to the ordering of the grid nodes. Pueyo and Zingg (1998) and Chisholm and Zingg (2009) have shown that the reverse Cuthill–McKee ordering (Liu and Sherman, 1976) can be very effective if a node on the downstream boundary is selected as the root node. With reverse Cuthill–McKee ordering, such a root node leads to an ordering from upstream to downstream, which improves the accuracy of the ILU factorization. Chisholm and Zingg (2009) also introduced a downstream bias into the tie-breaking strategy used in applying the reverse Cuthill–McKee ordering. The use of this ordering with a downstream root node can have a surprisingly large impact on the effectiveness of an ILU(p) preconditioner and hence the convergence of a Newton–Krylov algorithm. In a parallel implementation, the reverse Cuthill–Mckee ordering can be applied on each process independently.

The scaling of the solution variables and residual equations is another important consideration. Several difficulties can arise if either the solution variables or the residual values can vary greatly in magnitude either from equation to equation or from node to node. This can occur, for example, if the turbulence model variable takes on values much different from the mean-flow variables as a result of a particular nondimensionalization. If the residual equations are proportional to the cell volume, then their values will vary by several orders of magnitude. Such discrepancies can cause several problems. For example, they can make finding a value of ϵ in (80) that balances round-off and truncation error difficult or impossible to find. If the

entries in $\mathbf{v}$ vary greatly in magnitude, the scalar $\mathbf{v}^T\mathbf{v}$ will be representative only of the larger entries, leading to a value of ϵ that is inappropriately small for the smaller entries. Moreover, convergence tolerances are often defined with respect to residual norms, e.g., (78) and (85). If some residual entries dominate these norms, then tolerances will appear to be satisfied even if they are not met for the equations with smaller residual values that have little effect on the norm. A similar effect can arise in the time step formula used during the inexact Newton phase (86). If a large residual component is being reduced more rapidly than a smaller component, the time step will increase more rapidly than is desired for the smaller component.

In order to avoid the convergence difficulties associated with such issues, row and column scaling can be used to balance the components of both the solution variables and the residual. The precise approach taken will depend on the solver details, such as the nondimensionalization, in particular those associated with the turbulence model equations in a RANS solver. Examples for specific spatial discretizations of the RANS equations are given by Chisholm and Zingg (2009) and Osusky and Zingg (2013).

Convergence problems can also arise with turbulence models that require a quantity to be positive. Ideally, the turbulence model should be robust to small negative values, such as the negative Spalart–Allmaras one-equation turbulence model (Allmaras et al., 2012). This avoids the need to trim the values to nonnegative values. Nevertheless, a turbulence model can produce large destabilizing updates when time step values are used that are effective for the mean-flow equations. This can arise, for example, when the laminar-turbulent transition trip terms in the Spalart–Allmaras model are activated or when a local correlation-based transition model is solved. Under these circumstances, it can be advantageous to reduce the time step for the turbulence model equation to a value smaller than that used for the mean-flow equations. For example, Osusky and Zingg (2013) use a time step for the Spalart–Allmaras turbulence model one hundred times smaller than that used for the mean-flow equations in the presence of the trip terms, but found that this is not necessary when the trip terms are not used and fully turbulent flow is assumed.

User defined algorithm parameters are dependent on the nature of the particular equations, e.g., the turbulence model, the spatial discretization and how dissipative it is, and the range of flow problems of interest. We will present sample parameters from Hicken and Zingg (2008) for the Euler equations and Osusky and Zingg (2013) for the RANS equations, both applied to external aerodynamic flows. Both use multiblock structured meshes discretized using second-order summation-by-parts finite difference operators with block coupling and boundary conditions imposed weakly through a penalty approach. The solver also includes higher-order operators, but these have primarily been used for unsteady flows. Some of the algorithm parameters also depend on the specific nondimensionalization used in the solver, so the reader

is advised to establish suitable parameters for their specific solver; the parameters presented here provide only a starting point based not only on a particular solver but also on a particular balance between speed and robustness.

Beginning with the approximate Newton phase, Hicken and Zingg (2008) use the following parameters in (83): $a = 0.1$, $1.4 \leq b \leq 1.7$ and $3 \leq m \leq 5$, while Osusky and Zingg (2013) use $a = 0.001$, $b = 1.3$, and $m = 1$, as a much smaller initial time step is needed for the RANS equations. A level of fill of unity is used in the ILU(p) factorization for the Euler equations. For the RANS equations, $p = 2$ is used during the approximate Newton phase and $p = 3$ during the inexact Newton phase. Another major difference between the Euler and RANS equations is in the parameter used to switch from the approximate Newton to the inexact Newton phase, τ in (85). A value of $\tau = 0.1$ is effective for the Euler equations, but was found to be inadequate for the RANS equations, for which $\tau = 10^{-4}$ was found to provide a good balance between speed and robustness. Hicken and Zingg (2008) use a value of $\eta_n = 0.5$ for the linear solution tolerance during the approximate Newton phase for the Euler equations, whereas Osusky and Zingg (2013) use $\eta_n = 0.05$ for the RANS equations.

Parameters used for the inexact Newton phase are as follows. For the Euler equations, Hicken and Zingg (2008) use $\beta = 2$ in (86), an ILU(p) fill level of $p = 1$, $\delta = 10^{-13}$ in (80), and η_n determined according to (79). For the RANS equations, Osusky and Zingg (2013) use $3/2 \leq \beta \leq 2$, $p = 3$, $\delta = 10^{-10}$ and $\eta_n = 0.01$.

With these parameters, efficient convergence is achieved for a wide range of flow problems and grids. Sample convergence histories are presented in the next section.

4.2.6 Examples

In this section, convergence histories are presented for two popular flow problems governed by the RANS equations; parallel scaling performance is shown as well. The Spalart–Allmaras one-equation turbulence model is used in all cases. Convergence plots are shown with respect to the number of GMRES iterations. In addition, the residual is plotted vs CPU time in seconds and in equivalent residual evaluations. Intel Nehalem processors were used interconnected with nonblocking 4x- and 8x-DDR infiniband. The equivalent residual evaluations are determined by dividing the total CPU time by the CPU time required for a single complete residual evaluation. This enables comparisons that are independent of the particular hardware used.

The first test case is a transonic flow over the ONERA M6 wing at $M = 0.8395$, $Re = 11.72$ million, $\alpha = 3.06$ degrees. Experimental data are available in Schmitt and Charpin (1979). The flow is assumed fully turbulent. The grid used has 15.1 million nodes in 128 blocks with an off-wall spacing of 2.3×10^{-7} root chord units, giving an average y^+ value of 0.4. Load balancing is performed by hand during grid generation such that all blocks have the same number of nodes, and each block is assigned to a processor, i.e., 128 processors are used. Fig. 8 shows that the residual is reduced by twelve orders

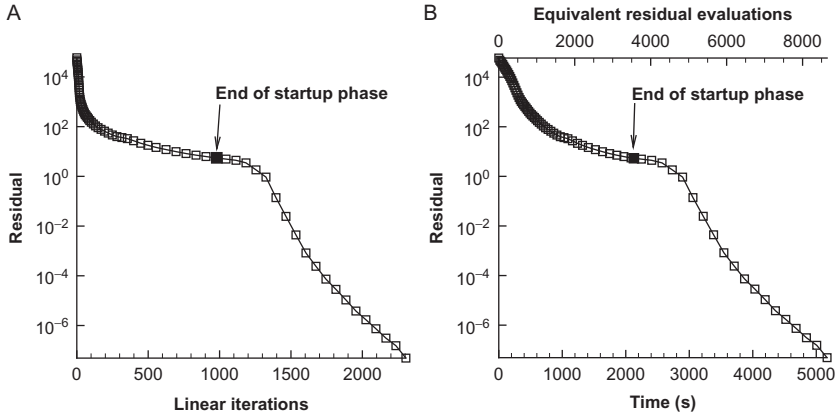

FIG. 8 Convergence history for computation of flow over the ONERA M6 wing at $M = 0.8395$, $Re = 11.72$ million, $\alpha = 3.06$ degrees on a grid with 15.1 million nodes using scalar dissipation and 128 processors. Each symbol represents a nonlinear or Newton iteration. (A) Residual norm vs number of GMRES iterations. (B) Residual norm vs CPU time in seconds and equivalent residual evaluations.

of magnitude in roughly 83 minutes, requiring just over 3000 GMRES iterations. The switch from the approximate Newton to the inexact Newton phase, which occurs once the residual has been reduced by four orders of magnitude, is indicated on the figure. The globalization phase takes just under half of the total CPU time. Roughly 8000 equivalent residual evaluations are required to achieve a 12-order reduction in the residual.

The next test case is a wing-body geometry known as the NASA Common Research Model (Vassberg, 2011). The flow conditions are $M = 0.85$, $Re = 5$ million, $C_L = 0.5$, where the Reynolds number is based on the mean aerodynamic chord. The convergence results shown in Fig. 9 were computed on a grid with 19 million nodes in an *O–O* topology on 704 processors. As a result of the original grid construction, the load balancing is not perfect; better performance is possible through a more formal load balancing approach. A 12-order residual reduction is achieved in roughly 80 min or 15,000 equivalent residual evaluations, requiring roughly 2300 GMRES iterations. In this case, the approximate Newton phase requires about one third of the total CPU time.

The same test case and flow conditions are used to examine the parallel scaling of the algorithm. Two grids are considered, one with 48 million nodes ("*X*" grid), the other with 154 million nodes ("*S*" grid). Both are subdivided into 6656 blocks, with the same number of grid nodes in each block, enabling perfect load balancing. Fig. 10 plots the CPU time required to reduce the residual by ten orders of magnitude for varying numbers of processors, with the number of blocks fixed at 6656, showing that the algorithm scales almost perfectly under these conditions. The number of GMRES iterations is nearly constant as the number of processors is varied, suggesting that the approximate-Schur preconditioning is performing very well.

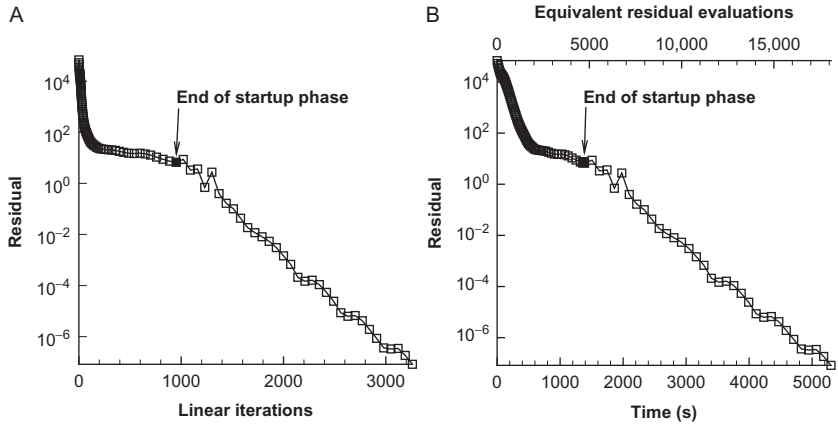

FIG. 9 Convergence history for computation of flow over the NASA Common Research Model wing-body configuration at $M = 0.85$, $Re = 5$ million, $C_L = 0.5$ on a grid with 19 million nodes using matrix dissipation and 704 processors. Each symbol represents a nonlinear or Newton iteration. (A) Residual norm vs number of GMRES iterations. (B) Residual norm vs CPU time in seconds and equivalent residual evaluations.

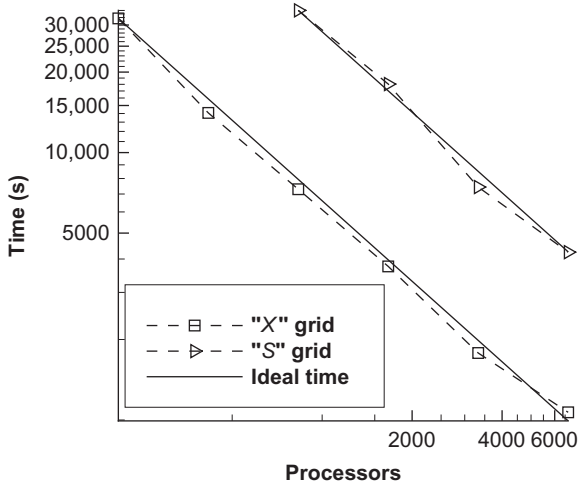

FIG. 10 Parallel scaling performance for computation of flow over NASA Common Research Model wing-body configuration at $M = 0.85$, $Re = 5$ million, $C_L = 0.5$ on grids with 48 million nodes ("X" grid) and 154 million nodes ("S" grid).

5 CONCLUSIONS

Time-marching methods, when paired with geometric multigrid, represent an extremely effective means by which to obtain steady state solutions to the Euler and RANS equations. A variety of schemes and acceleration methodologies have been presented along with results for several benchmark flow

problems. In particular, the hybrid RKSGS scheme is observed to converge a RANS problem in under 100 iterations.

Newton–Krylov methods provide an efficient option for problems with higher than usual stiffness. This can arise, for example, from the turbulence model, reacting flows, very high Reynolds numbers, or when high-order spatial discretization is used. The basic components of a Newton–Krylov algorithm have been described and convergence results presented for two well known flow problems.

REFERENCES

Allmaras, S., 1993. Analysis of a local matrix preconditioner for the 2-D Navier-Stokes equations. In: AIAA 11th Computational Fluid Dynamics Conference, AIAA Paper 93-3330, Orlando, FL.

Allmaras, S., 1995. Analysis of semi-implicit preconditioners for multigrid solution of the 2-D Navier-Stokes equations. In: AIAA 12th Computational Fluid Dynamics Conference, AIAA Paper 95-1651, San Diego, CA.

Allmaras, S., 1997. Algebraic smoothing analysis of multigrid methods for the 2-D compressible Navier-Stokes equations. In: AIAA 13th Computational Fluid Dynamics Conference, AIAA Paper 97-1954, Snowmass, CO.

Allmaras, S.R., Johnson, F.T., Spalart, P.R., 2012. Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. In: ICCFD7, Hawaii.

Anderson, W.K., Thomas, J.L., Whitfield, D.L., 1986. Multigrid acceleration of the flux split Euler equations. In: AIAA 24th Aerospace Sciences Meeting, AIAA Paper 86-0274, Reno, NV.

Bardina, J., Lombard, C.K., 1987. Three-dimensional hypersonic flow simulations with the CSCM implicit upwind Navier-Stokes method. In: AIAA 8th Computational Fluid Dynamics Conference, AIAA Paper 87-1114, Honolulu, HI.

Beam, R.W., Warming, R.F., 1976. An implicit finite difference algorithm for hyperbolic systems in conservation form. J. Comput. Phys. 23, 87–110.

Brandt, A., 1977. Multi-level adaptive solutions to boundary value problems. Math. Comput. 31, 333–390.

Brown, D.A., Zingg, D.W., 2016. A monolithic homotopy continuation algorithm with application to computational fluid dynamics. J. Comput. Phys. 321, 55–75.

Caughey, D.A., 1987. A diagonal implicit multigrid algorithm for the Euler equations. In: AIAA 25th Aerospace Sciences Meeting, AIAA Paper 87-453, Reno, NV.

Chakravarthy, S.R., 1984. Relaxation methods for unfactored implicit upwind schemes. In: AIAA 23rd Aerospace Sciences Meeting, AIAA Paper 84-0165, Reno.

Chisholm, T.T., Zingg, D.W., 2009. A Jacobian-free Newton-Krylov algorithm for compressible turbulent flows. J. Comput. Phys. 228, 3490–3507.

Crumpton, P.I., Giles, M.B., 1995. Implicit time accurate solutions on unstructured dynamic grids. In: AIAA 12th Computational Fluid Dynamics Conference, AIAA Paper 95-1671, San Diego, CA.

Eisenstat, S.C., Walker, F.H., 1996. Choosing the forcing terms in an inexact Newton method. SIAM J. Sci. Stat. Comput. 7, 16–32.

Fedorenko, R.P., 1964. The speed of convergence of one iterative process. USSR Comput. Math. Math. Phys. 4, 227–235.

Gagnon, H., Zingg, D.W., 2016. Euler-equation-based drag minimization of unconventional aircraft configurations. J. Aircr. 53 (4), 1361–1371.

Giles, M.B., Drela, M., 1987. Two-dimensional transonic aerodynamic design method. AIAA J. 25 (9), 1199–1206.

Giles, M., Drela, M., Thompkins, W.T., 1985. Newton solution of direct and inverse transonic Euler equations. In: AIAA 7th Computational Fluid Dynamics Conference, AIAA Paper 85-1530, Cincinnati, pp. 394–402.

Gourlay, A.R., Mitchell, A.R., 1966. A stable implicit difference scheme for hyperbolic systems in two space variables. Numer. Math. 8, 367–375.

Hackbusch, W., 1978. On the multi-grid method applied to difference equations. Computing 20, 291–306.

Hall, M.G., 1985. Cell vertex multigrid schemes for solution of the Euler equations. In: Morton, K.W., Baines, M.J. (Eds.), Conference on Numerical Methods for Fluid Dynamics, University Reading. Oxford University Press, Oxford, pp. 303–345.

Hemker, P.W., Spekreijse, S.P., 1984. Multigrid solution of the steady Euler equations. In: Proc. Oberwolfach Meeting on Multigrid Methods.

Hicken, J.E., Zingg, D.W., 2008. A parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms. AIAA J. 46 (11), 2773–2786.

Hicken, J.E., Zingg, D.W., 2010. Aerodynamic optimization algorithm with integrated geometry parameterization and mesh movement. AIAA J. 48 (2), 401–413.

Hicken, J.E., Osusky, M., Zingg, D.W., 2010. Comparison of parallel preconditioners for a Newton-Krylov solver. In: ICCFD6, St. Petersburg.

Jameson, A., 1982. Steady state solution of the Euler equations for transonic flow. In: Meyer, R.E. (Ed.), Proc. Symposium on Transonic, Shock, and Multidimensional Flows, Madison, 1980. Academic Press, Cambridge, MA, pp. 37–70.

Jameson, A., 1983. Solution of the Euler equations by a multigrid method. Appl. Math. Comput. 13, 327–356.

Jameson, A., 1985. Transonic flow calculations for aircraft. In: Brezzi, F. (Ed.), Numerical Methods in Fluid Dynamics, Lecture Notes in Mathematics. Springer-Verlag, Berlin, Heidelberg, pp. 156–242.

Jameson, A., 1986a. Multigrid algorithms for compressible flow calculations. In: Hackbusch, W., Trottenberg, U. (Eds.), Proceedings of the 2nd European Conference on Multigrid Methods, Lecture Notes in Mathematics, vol. 1228. Springer-Verlag, Berlin, pp. 166–201.

Jameson, A., 1986b. A vertex based multigrid algorithm for three-dimensional compressible flow calculations. In: Tezduar, T.E., Hughes, T.J.R. (Eds.), Numerical Methods for Compressible Flow—Finite Difference, Element and Volume Techniques, vol. AMD 78. ASME Publication, New York, NY, pp. 45–73.

Jameson, A., 1991. Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings.

Jameson, A., 1995a. Analysis and design of numerical schemes for gas dynamics 1, artificial diffusion, upwind biasing, limiters and their effect on multigrid convergence. Int. J. Comput. Fluid Dyn. 4, 171–218.

Jameson, A., 1995b. Analysis and design of numerical schemes for gas dynamics 2, artificial diffusion and discrete shock structure. Int. J. Comput. Fluid Dyn. 5, 1–38.

Jameson, A., 2015. Application of dual time stepping to fully implicit Runge-Kutta schemes for unsteady flow calculations. In: 22nd AIAA Computational Fluid Dynamics Conference, pp. 2753.

Jameson, A., Baker, T.J., 1983. Solution of the Euler equations for complex configurations. In: Proc. AIAA 6th Computational Fluid Dynamics Conference, Denvers, pp. 293–302.

Jameson, A., Caughey, D.A., 2001. How many steps are required to solve the Euler equations of steady compressible flow. In: 15th AIAA Computational Fluid Dynamics Conference, AIAA Paper 2001-2673, Anaheim, CA.

Jameson, A., Mavriplis, D.J., 1987. Multigrid solution of the Euler equations on unstructured and adaptive grids. In: McCormick, S. (Ed.), Multigrid Methods, Theory, Applications and Supercomputing, Lecture Notes in Pure and Applied Mathematics, vol. 110. Marcel Dekker, New York, NY, pp. 413–430.

Jameson, A., Turkel, E., 1981. Implicit schemes and LU decompositions. Math. Comput. 37, 385–397.

Jameson, A., Schmidt, W., Turkel, E., 1981. Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes. In: 4th Fluid Dynamics and Plasma Dynamics Conference, AIAA Paper 81-1259, Palo Alto, CA.

Kinnmark, I.P.E., 1984. One step integration methods with large stability limits for hyperbolic partial differential equations. In: Vichnevetsky, R., Stepleman, R.S. (Eds.), Advance in Computer Methods for Partial Differential Equations, vol. 5. Publ. IMACS, New Brunswick, NJ.

Knoll, D.A., Keyes, D.E., 2004. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. J. Comput. Phys. 193, 357–397.

Lallemand, M.H., Dervieux, A., 1987. A multigrid finite-element method for solving the two-dimensional Euler equations. In: McCormick, S.F. (Ed.), Proceedings of the Third Copper Mountain Conference on Multigrid Methods, Copper Mountain, Lecture Notes in Pure and Applied Mathematics. Marcel Dekker, New York, NY, pp. 337–363.

Lallemand, M.H., Steve, H., Dervieux, A., 1992. Unstructured multigridding by volume aggregation: current status. Comput. Fluids 21, 397–433.

Lerat, A., Sidès, J., Daru, V., 1982. An implicit finite-volume method for solving the Euler equations. In: Krause, E. (Ed.), Eighth International Conference on Numerical Methods in Fluid Dynamics: Proceedings of the Conference, Rheinisch-Westfälische Technische Hochschule Aachen, Germany, June 28–July 2, 1982. Springer, Berlin, Heidelberg, pp. 343–349.

Liou, M.-S., Steffen, C.J., 1993. A new flux splitting scheme. J. Comput. Phys. 107, 23–39.

Liu, W.H., Sherman, A.H., 1976. Comparative analysis of the Cuthill-McKee and reverse Cuthill-McKee ordering algorithms for sparse matrices. SIAM J. Numer. Anal. 13, 198–213.

Liu, X.D., Osher, S., Chan, T., 1994. Weighted essentially non-oscillatory schemes. J. Comput. Phys. 115, 200–212.

Lomax, H., Pulliam, T.H., Zingg, D.W., 2001. Fundamentals of Computational Fluid Dynamics. Springer, Germany.

MacCormack, R.W., 1985. Current status of numerical solutions of the Navier-Stokes equations. In: AIAA 23rd Aerospace Sciences Meeting, AIAA Paper 85-0032, Reno.

MacCormack, R.W., 1997. A new implicit algorithm for fluid flow. In: Proc. AIAA 13th CFD Conference, AIAA Paper, Snowmass, Colorado, pp. 112–119.

Mavriplis, D.J., Jameson, A., 1990. Multigrid solution of the Navier-Stokes equations on triangular meshes. AIAA J. 28 (8), 1415–1425.

Mavriplis, D.J., Martinelli, L., 1991. Multigrid solution of compressible turbulent flow on unstructured meshes using a two-equation model. In: AIAA 29th Aerospace Sciences Meeting, AIAA Paper 91-0237, Reno, NV.

Mavriplis, D.J., Venkatakrishnan, V., 1996. A 3D agglomeration multigrid solver for the Reynolds-averaged Navier-Stokes equations on unstructured meshes. Int. J. Numer. Methods Fluids 23, 1–18.

Meijerink, J.A., van der Vorst, H.A., 1977. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. Math. Comput. 31, 148–162.

Mulder, W.A., 1989. A new multigrid approach to convection problems. J. Comput. Phys. 83, 303–323.

Mulder, W.A., 1992. A high-resolution Euler solver based on multigrid, semi-coarsening, and defect correction. J. Comput. Phys. 100, 91–104.

Mulder, W.A., van Leer, B., 1985. Experiments with implicit upwind methods for the Euler equations. J. Comput. Phys. 59, 232–246.

Ni, R.H., 1982. A multiple grid scheme for solving the Euler equations. AIAA J. 20, 1565–1571.

Nielsen, E.J., Walters, R.W., Anderson, W.K., Keyes, D.E., 1995. Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code. In: AIAA 12th Computational Fluid Dynamics Conference, AIAA Paper 95-1733, San Diego, CA.

Obayashi, S., Kuwakara, K., 1984. LU factorization of an implicit scheme for the compressible Navier-Stokes equations. In: AIAA 17th Fluid Dynamics and Plasma Dynamics Conference, AIAA Paper 84-1670, Snowmass.

Obayashi, S., Matsukima, K., Fujii, K., Kuwakara, K., 1986. Improvements in efficiency and reliability for Navier-Stokes computations using the LU-ADI factorization algorithm. In: AIAA 24th Aerospace Sciences Meeting, AIAA Paper 86-0338, Reno.

Osusky, M., Zingg, D.W., 2013. A parallel Newton-Krylov-Schur flow solver for the Navier-Stokes equations discretized using summation-by-parts operators. AIAA J. 51 (12), 2773–2786.

Osusky, L., Buckley, H.P., Reist, T.A., Zingg, D.W., 2015. Drag minimization based on the Navier-Stokes equations using a Newton-Krylov approach. AIAA J. 53 (6), 1555–1577.

Peraire, J., Peirö, J., Morgan, K., 1992. A 3D finite-element multigrid solver for the Euler equations. In: AIAA 30th Aerospace Sciences Conference, AIAA Paper 92-0449, Reno, NV.

Persson, P.-O., Peraire, J., 2008. Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier-Stokes equations. SIAM J. Sci. Comput. 30 (6), 2709–2722.

Pierce, N.A., Giles, M.B., 1997. Preconditioning compressible flow calculations on stretched meshes. J. Comput. Phys. 136, 425–445.

Pierce, N.A., Giles, M.B., Jameson, A., Martinelli, L., 1997. Accelerating three-dimensional Navier-Stokes calculations. In: AIAA 13th Computational Fluid Dynamics Conference, AIAA Paper 97-1953, Snowmass, CO.

Pueyo, A., Zingg, D.W., 1998. Efficient Newton-Krylov solver for aerodynamic flows. AIAA J. 36 (11), 1991–1997.

Pulliam, T.H., Steger, J.L., 1985. Recent improvements in efficiency, accuracy and convergence for implicit approximate factorization algorithms. In: AIAA 23rd Aerospace Sciences Meeting, AIAA Paper 85-0360, Reno.

Pulliam, T.H., Zingg, D.W., 2014. Fundamental Algorithms in Computational Fluid Dynamics. Springer, Switzerland.

Reist, T.A., Zingg, D.W., 2016. High-fidelity aerodynamic shape optimization of a lifting fuselage concept for regional aircraft. J. Aircr. http://dx.doi.org/10.2514/1.C033798 (accepted for publication).

Rieger, H., Jameson, A., 1988. Solution of steady three-dimensional compressible Euler and Navier-Stokes equations by and implicit LU scheme. In: AIAA 26th Aerospace Sciences Meeting, AIAA Paper 88-0619, Reno, NV.

Roe, P.L., 1981. Approximate Riemann solvers, parameter vectors, and difference schemes. J. Comput. Phys. 43, 357–372.

Rossow, C.-C., 2007. Efficient computation of compressible and incompressible flows. J. Comput. Phys. 220 (2), 879–899.

Rusanov, V.V., 1961. Calculation of interaction of non-steady shock waves with obstacles. J. Comput. Math. Phys. USSR 267–279.

Saad, Y., 1993. A flexible inner-outer preconditioned GMRES algorithm. SIAM J. Sci. Stat. Comput. 14, 461–469.

Saad, Y., 2003. Iterative Methods for Sparse Linear Systems. SIAM, Philadelphia.

Saad, Y., Schultz, M.H., 1986. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. 7, 856–869.

Saad, Y., Sosonkina, M., 1999. Distributed Schur complement techniques for general sparse linear systems. SIAM J. Sci. Comput. 21, 1337–1357.

Schmitt, V., Charpin, F., 1979. Pressure distributions on the ONERA M6 wing at transonic Mach numbers. Tech. Rep., ONERA, France.

Smith, W.A., Weiss, J.M., 1995. Preconditioning applied to variable and constant density flows. AIAA J. 33 (11), 2050–2057.

Swanson, R., Turkel, E., Rossow, C.-C., 2007. Convergence acceleration of Runge-Kutta schemes for solving the Navier-Stokes equations. J. Comput. Phys. 224 (1), 365–388.

Toro, E.F., Spruce, M., Speares, W., 1994. Restoration of the contact surface in the HLL-Riemann solver. Shock Waves 4 (1), 25–34. ISSN 1432-2153.

Turkel, E., 1987. Preconditioned methods for solving the incompressible and low speed equations. J. Comput. Phys. 72, 277–298.

van Leer, B., 1974. Towards the ultimate conservative difference scheme. II, Monotonicity and conservation combined in a second order scheme. J. Comput. Phys. 14, 361–370.

van Leer, B., Lee, W.T., Roe, P.L., 1991. Characteristic time stepping or local preconditioning of the Euler equations. In: AIAA 10th Computational Fluid Dynamics Conference, AIAA Paper 91-1552, Honolulu, Hawaii.

Vassberg, J.C., 2011. A unified baseline grid about the common research model wing-body for the fifth AIAA CFD drag prediction workshop. In: 29th AIAA Applied Aerodynamics Conference, AIAA Paper 2011-3508, Honolulu, HA.

Venkatakrishnan, V., 1988. Newton solution of inviscid and viscous problems. In: AIAA 26th Aerospace Sciences Meeting, AIAA Paper 88-0413, Reno, NV.

Wilcox, D.C., 1998. Turbulence Modelling of CFD. DCW Industries, La Canada, CA.

Yee, H.C., 1985. On symmetric and upwind TVD schemes. In: Proc. 6th GAMM Conference on Numerical Methods in Fluid Mechanics, Gottingen.

Yoon, S., Jameson, A., 1987. Lower-upper Symmetric-Gauss-Seidel method for the Euler and Navier-Stokes equations. In: AIAA 25th Aerospace Sciences Meeting, AIAA Paper 87-0600, Reno, NV.

Yu, M.L., Wang, Z.J., 2016. Homotopy continuation of the high-order flux reconstruction/correction procedure via reconstruction (FR/CPR) methods for steady flow simulation. Comput. Fluids 131, 16–28.