Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 227 (2008) 2582–2609

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

A high-order accurate unstructured finite volume Newton–Krylov algorithm for inviscid compressible flows

Amir Nejat, Carl Ollivier-Gooch [*]

Department of Mechanical Engineering, The University of British Columbia, 2054-6250 Applied Science Lane, Vancouver, BC, Canada V6T 1Z4

Received 27 March 2007; received in revised form 5 September 2007; accepted 4 November 2007
Available online 21 November 2007

Abstract

A fast implicit Newton–Krylov finite volume algorithm has been developed for high-order unstructured steady-state computation of inviscid compressible flows. The matrix-free generalized minimal residual (GMRES) algorithm is used for solving the linear system arising from implicit discretization of the governing equations, avoiding expensive and complex explicit computation of the high-order Jacobian matrix. The solution process has been divided into two phases: start-up and Newton iterations. In the start-up phase an approximate solution with the general characteristics of the steady-state flow is computed by using a defect correction procedure. At the end of the start-up phase, the linearization of the flow field is accurate enough for steady-state solution, and a quasi-Newton method is used, with an infinite time step and very rapid convergence. A proper limiter implementation for efficient convergence of the high-order discretization is discussed and a new formula for limiting the high-order terms of the reconstruction polynomial is introduced. The accuracy, fast convergence and robustness of the proposed high-order unstructured Newton–Krylov solver for different speed regimes is demonstrated for the second, third and fourth-order discretization. The possibility of reducing computational cost required for a given level of accuracy by using high-order discretization is examined.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Newton–Krylov; High-order discretization; Unstructured meshes; Compressible flow; Start-up; Limiter

1. Introduction

In computational aerodynamics, as in scientific computing generally, we wish to obtain an accurate solution as quickly as possible. When accuracy requirements are strict, high-order methods are preferable to second-order methods in the sense of providing a better solution on a given mesh, and therefore a comparable quality solution on a coarser mesh, when compared with second-order schemes. However, because of the higher computational cost per control volume and the possibility of degradation in convergence rates for high-order methods, the question of whether one can obtain an accurate solution more quickly with a high-order method

[*] Corresponding author. Tel.: +1 604 822 1854; fax: +1 604 822 2403.
E-mail addresses: nejat@mech.ubc.ca (A. Nejat), cfog@mech.ubc.ca (C. Ollivier-Gooch).

0021-9991/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2007.11.011

hinges on the ability to achieve rapid convergence with such schemes. In this paper, we will present evidence that a high-order finite-volume scheme can in fact reach a given level of accuracy with less computational effort than a second-order scheme. Specifically, we will present a high-order accurate unstructured mesh finite-volume solver for compressible, inviscid flow as well as the techniques we use to ensure rapid convergence of an implicit Newton-GMRES scheme for both third- and fourth-order discretizations.

1.1. High-order discretization methods

High-order finite-volume methods – by which we mean anything beyond second-order accurate – were first applied to aerodynamic flows by Rogers et al. [30], with a third-order discretization implemented in INS3D, and by Godfrey et al. [14], who applied ENO schemes in computational aerodynamics. More recently, De Rango and Zingg [8] dramatically reduced numerical error in drag using relatively coarse structured grids for steady turbulent flow over a 2D airfoil by applying a globally third-order accurate algorithm. Their results provide a convincing demonstration of the accuracy benefits of high-order methods compared with a second-order method for practical flows. Zingg et al. [36] compared different high-order accurate flux discretization techniques for laminar and turbulent flows (including transition) in both subsonic and transonic speed regimes, showing that high-order discretization produces comparably accurate solutions much more efficiently than a second-order method.

Our research in high-order unstructured solvers is motivated by the desire to combine the accuracy and efficiency benefits seen in the application of high-order methods on structured meshes with the geometric and adaptive flexibility of unstructured meshes. Also, to provide a natural upgrade path for the substantial base of existing unstructured finite-volume flow solvers, we focus on high-order finite-volume schemes rather than high-order discontinuous Galerkin methods.

The roots of high-order unstructured mesh finite-volume methods lie in the work of Barth and various co-workers, who published a series of papers describing techniques for using least-squares reconstruction to obtain high-order accuracy [5,3,4,6]. While the techniques described in these papers are applicable in principle to all orders of accuracy, the results presented in them do not extend beyond third-order accuracy (quadratic reconstruction). Ollivier-Gooch [24,25] experimented with solution-dependent weightings as a way to compute oscillation-free least-squares reconstructions for schemes of up to fourth-order accuracy. Delanaye and Essers [10] and Geuzaine et al. [13] proposed a quadratic reconstruction finite volume scheme, including a new approach to monotonicity enforcement. They computed the inviscid flux directly from their quadratic reconstruction; however, viscous terms were obtained through a linear interpolation and were therefore only second order. For monotonicity enforcement, they used a discontinuity detector to eliminate the high-order terms from the reconstruction in the vicinity of discontinuities.

1.2. Implicit methods for steady-state convergence

Achieving steady-state convergence by time marching, or pseudo-time marching, requires that all transient phenomena in the domain be damped or convected out of the domain. Implicit methods use large time steps to both smear and rapidly propagate errors; convergence rates are typically dictated by non-linear stability issues or by inexact linearization of a non-linear residual. In the limit of very large time steps, implicit time advance schemes reduce to Newton's method [34,32,2,6, are among the early uses in CFD]. In principle, Newton's method gives quadratic convergence rates for steady flow solution when starting from a good enough initial guess. In practice, however, forming and solving the linear system at each Newton iteration accurately enough to achieve quadratic convergence is difficult enough that most researchers choose a quasi-Newton method. Quasi-Newton methods sacrifice quadratic convergence in favor of lower memory usage and lower cost per iteration, with the net result of reducing overall CPU time [27]. Quasi-Newton methods are generally categorized as *approximate Newton* (in which an approximate linearization is used) and *inexact Newton* (in which the linear system is not solved exactly).

Regardless of the details of the space and time discretization techniques, implicit methods inevitably require the solution of a large linear system resulting from the linearization of the fluid flow equations in time. While approximate factorization techniques are very effective linear solvers for structured meshes, modern unstructured mesh implicit schemes in CFD generally rely on iterative linear solvers, especially the generalized minimal residual (GMRES) method [31], which was developed particularly for non-symmetric systems such as those arising from implicit time discretization on unstructured meshes. In the matrix-free variant of GMRES, the matrix-vector products required by the GMRES algorithm are computed by using directional derivatives, eliminating the problem of explicitly forming the Jacobian matrix, and thereby considerably reducing both memory usage and programming effort. This is especially helpful for high-order unstructured mesh solvers where the full (analytic) Jacobian calculation is costly and difficult.

The efficiency of GMRES in solving a linear system depends strongly on the conditioning of the linear system, and compressible flow problems on unstructured meshes do not typically have well-conditioned Jacobian matrices. This is especially true for high-order methods, as we shall see in Section 5. Effective preconditioning is therefore essential to rapid convergence of Newton-GMRES schemes.

To retain the low memory usage of a matrix-free GMRES method, we could use a matrix-free preconditioner, such as the lower–upper symmetric Gauss–Seidel (LU-SGS) preconditioner introduced by Luo et al. for 3D compressible flows [16]. They completely eliminated the storage of the preconditioning Jacobian matrix by approximating the Jacobian with numerical fluxes. However, they did not perform full Newton iterations, probably because of the stability considerations for their LU-SGS preconditioner, and their convergence rates remained nearly linear. We have experimented with LU-SGS preconditioning for a high-order matrix-free Newton-GMRES algorithm for compressible flow [20]. For supersonic flows, LU-SGS-GMRES was about as efficient for the third-order discretization as for the second-order one. However, for flows that are more difficult to converge, for fourth-order discretization, and for full Newton iterations (infinite time step), the LU-SGS preconditioner proved to be inadequate for our needs.

Currently, the most prevalent preconditioning approach for compressible flows is to use a low-order Jacobian as the preconditioner matrix and incomplete lower–upper (ILU) factorization to apply the preconditioner. For structured meshes, Pueyo and Zingg [28] presented an efficient matrix-free Newton–GMRES solver for steady-state aerodynamic flow computations using this approach. After a thorough parametric study for inviscid, laminar and turbulent 2D flows, they concluded that the approximate Newton method using matrix-free GMRES preconditioned with a first-order Jacobian and ILU(2) provides the best efficiency in terms of CPU time for most cases. Subsequently, Nichols and Zingg [23] developed a 3D multi-block Newton–Krylov solver for the Euler equations using the same approach, concluding that for this case, ILU(1) gives an appropriate balance between good preconditioning and low computational time per iteration.

For unstructured meshes, Venkatakrishnan and Mavriplis [35] developed an approximate Newton-GMRES implicit solver for computing compressible inviscid and turbulent flows around a multi-element airfoil. They compared different preconditioning strategies and found out that GMRES with ILU(0) preconditioning and a first-order Jacobian had the best performance. Manzano et al. [17] presented an efficient ILU preconditioned matrix-free Newton-GMRES algorithm for 3D unstructured meshes. They used different levels of fill (ILU(1–3)) depending on the case and the flux residual to achieve optimum performance. Delanaye et al. [12] presented the first-ever third-order accurate ILU preconditioned matrix-free Newton–GMRES solver for the Euler and Navier–Stokes equations. They showed that convergence can stall for stiff problems when using a high-order Jacobian and ILU(0) decomposition. Full convergence was achieved by allowing additional fill (ILU(1)) in the decomposition.

1.3. Scope of the present article

In this paper, we will present our recent work on third- and fourth-order accurate schemes for the Euler equations. In addition to verifying full-order accuracy in our solutions, we have also attained excellent efficiency in convergence to steady state, with the third- and fourth-order schemes comparable in efficiency to the second-order scheme. Our discretization scheme uses Barth and Frederickson's [5] reconstruction algorithm as a foundation for a fourth-order accurate solution of the compressible Euler equations. The details of our spatial discretization scheme are given in Section 3, including high-order boundary condition enforcement and a better-converging variant of Delanaye and Essers' [10] monotonicity enforcement scheme. Section 4 gives a comprehensive description of our time advance scheme, including formation of the preconditioning matrix and our strategies for both start-up and Newton iterations; the complex issue of preconditioning, which is only summarized in this section, is discussed fully in a companion paper [22]. The accuracy, efficiency and robustness of the current algorithm are demonstrated in Section 5 through several test cases.

2. Finite-volume discretization of the governing equations

Inviscid compressible flow simulations provide both a relatively accurate representation of aerodynamic flows and a stepping stone to viscous calculations, which depend critically on a robust, efficient inviscid solver as their basis. In integral form, we can write the Euler equations that describe inviscid compressible flow as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{CV} U\,\mathrm{d}V + \oint_{CS} F\,\mathrm{d}A = 0, \tag{1}$$

where the conserved solution vector U and flux vector F can be written as

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix} \quad F = \begin{bmatrix} \rho u_n \\ \rho u u_n + P\hat{n}_x \\ \rho v u_n + P\hat{n}_y \\ (E+P)u_n \end{bmatrix} \tag{2}$$

with $[\rho\ \rho u\ \rho v\ E]^{\mathrm{T}}$ are the densities of mass, x-momentum, y-momentum and energy, respectively, and $u_n = u\hat{n}_x + v\hat{n}_y$. The energy is related to the pressure by the perfect gas equation of state:

$$E = \rho c_v T + \frac{1}{2}\rho\left(u^2+v^2\right),$$
$$P = \rho R T.$$

Like many researchers, we choose to work with the equations in non-dimensional form, using the free stream density ρ_0 and sound speed c_0 as reference values. The form of the governing equations remains the same; the equation of state becomes

$$E^* = \frac{\rho^* T^*}{\gamma(\gamma-1)} + \frac{1}{2}\rho^*\left(u^{*^2}+v^{*^2}\right),$$
$$P^* = \frac{\rho^* T^*}{\gamma},$$

where γ is the ratio of specific heats for the gas. Hereafter, we will drop the $*$ superscript from the non-dimensional variables for simplicity. For a finite-volume formulation with fixed control volumes, we can use the definition

$$\overline{U}_i = \frac{1}{A_{CV_i}}\int_{CV_i} U\,\mathrm{d}A$$

to re-write Eq. (1) as

$$\frac{\mathrm{d}\overline{U}_i}{\mathrm{d}t} = -\frac{1}{A_{CV_i}}\oint_{CS_i} F\,\mathrm{d}A \equiv -R_i(\overline{U}), \tag{3}$$

where we have introduced the *residual* $R_i(\overline{U})$, which is a non-linear function of the average solutions for control volumes in the neighborhood of i.

3. Spatial discretization

Our solver is a cell-centered finite-volume solver, with spatial discretization accuracy as high as fourth-order. Computation of the residual, or flux integral, for all interior control volumes to high-order accuracy for smooth solutions is a three-step process. First, the control volume averages are reconstructed to produce a piecewise polynomial representation of the solution (see Section 3.1). Next, the reconstructed data is used to compute the numerical flux at flux integration points on the control volume interfaces (see Section 3.2).

Finally, flux is integrated numerically around the control volume (see Section 3.2). Boundary condition treatment will be discussed in Section 3.4, while Section 3.5 will describe our approach to enforcing monotonicity near discontinuities in the solution.

3.1. High-order reconstruction procedure

The goal of high-order reconstruction is to compute, based on a control-volume averaged solution, a piecewise polynomial representation of the underlying smooth function that is accurate to order $k+1$. Our reconstruction procedure [26] follows the approach of Barth and Frederickson [5]. Because we are concerned with enforcing monotonicity of density and pressure at solution discontinuities, we choose to reconstruct a vector of control volume averages of the primitive variables $\overline{V}_i \equiv [\bar{\rho}_i\ \bar{u}_i\ \bar{v}_i\ \overline{P}_i]^{\mathrm{T}}$ instead of the conserved variables $\overline{U}_i$. We write the reconstruction as a polynomial:

$$V_{\mathrm{R}}^{(k)}(x,y) = V(x_c,y_c) + \left.\frac{\partial V}{\partial x}\right|_C \Delta x + \left.\frac{\partial V}{\partial y}\right|_C \Delta y + \left.\frac{\partial^2 V}{\partial x^2}\right|_C \frac{\Delta x^2}{2} + \left.\frac{\partial^2 V}{\partial x \partial y}\right|_C \Delta x \Delta y + \left.\frac{\partial^2 V}{\partial y^2}\right|_C \frac{\Delta y^2}{2} + \left.\frac{\partial^3 V}{\partial x^3}\right|_C \frac{\Delta x^3}{6} + \left.\frac{\partial^3 V}{\partial x^2 \partial y}\right|_C \frac{\Delta x^2 \Delta y}{2} + \left.\frac{\partial^3 V}{\partial x \partial y^2}\right|_C \frac{\Delta x \Delta y^2}{2} + \left.\frac{\partial^3 V}{\partial y^3}\right|_C \frac{\Delta y^3}{6} + \cdots, \tag{4}$$

where $\Delta\vec{x} \equiv \vec{x} - \vec{x}_c$ and $\vec{x}_c$ is the control volume reference location, taken to be the centroid for cell-centered control volumes. The derivatives in Eq. (4) are pointwise values at the reference point C; we seek to compute all derivatives of degree $\leqslant k$ during reconstruction.

We write the mean constraint as

$$\begin{aligned}\overline{V}_i &= \frac{1}{A_i}\int_i V_{\mathrm{R}}^{(k)}(x,y)\,\mathrm{d}A \\ &= V(x_c,y_c) + \left.\frac{\partial V}{\partial x}\right|_C \overline{x} + \left.\frac{\partial V}{\partial y}\right|_C \overline{y} + \left.\frac{\partial^2 V}{\partial x^2}\right|_C \frac{\overline{x^2}}{2} + \left.\frac{\partial^2 V}{\partial x \partial y}\right|_C \overline{xy} + \left.\frac{\partial^2 V}{\partial y^2}\right|_C \frac{\overline{y^2}}{2} + \left.\frac{\partial^3 V}{\partial x^3}\right|_C \frac{\overline{x^3}}{6} + \left.\frac{\partial^3 V}{\partial x^2 \partial y}\right|_C \frac{\overline{x^2 y}}{2} \\ &\quad + \left.\frac{\partial^3 V}{\partial x \partial y^2}\right|_C \frac{\overline{xy^2}}{2} + \left.\frac{\partial^3 V}{\partial y^3}\right|_C \frac{\overline{y^3}}{6} + \cdots,\end{aligned} \tag{5}$$

where the integrated expression includes control volume moments of the form $\overline{x^n y^m} \equiv \frac{1}{A}\int x^n y^m\,\mathrm{d}A$. We compute these moments by applying Gauss's theorem

$$\int_{CV} x^n y^m\,\mathrm{d}A = \int_{CV} \nabla \cdot \left(\frac{1}{n+1} x^{n+1} y^m \hat{x}\right) \mathrm{d}A = \oint_{\partial CV} \frac{1}{n+1} x^{n+1} y^m n_x\,\mathrm{d}s \tag{6}$$

and evaluating the last integral exactly around each control volume by using Gauss quadrature with an appropriate number of quadrature points; as many as three are required for third moments, where $n+m=3$. While moments for triangular control volumes could be directly calculated with relative ease, the flux form on the right of Eq. (6) can be extended to curved boundaries, vertex-centered control volumes, and even to three dimensions much more easily than the area integral form.

If our reconstruction is accurate, it will closely approximate the underlying smooth solution, and the average of the reconstruction polynomial in control volume i, $V_{R;i}^{(k)}(x,y)$, over a neighboring control volume j, will nearly match the average in control volume j:

$$\overline{V}_j = \frac{1}{A_j}\int_j V_{R;i}^{(k)}(x,y)\,\mathrm{d}A \tag{7}$$

For each control volume i, we create a stencil of nearby control volumes and write Eq. (7) for each control volume in the stencil. To support calculation of all derivative terms, this stencil must have at least $(k+1)(k+2)/2$ control volumes, including i. We exceed that number by about 50%, using at least 4, 9, and 16 control volumes for second-, third-, and fourth-order accuracy; the additional data allows the

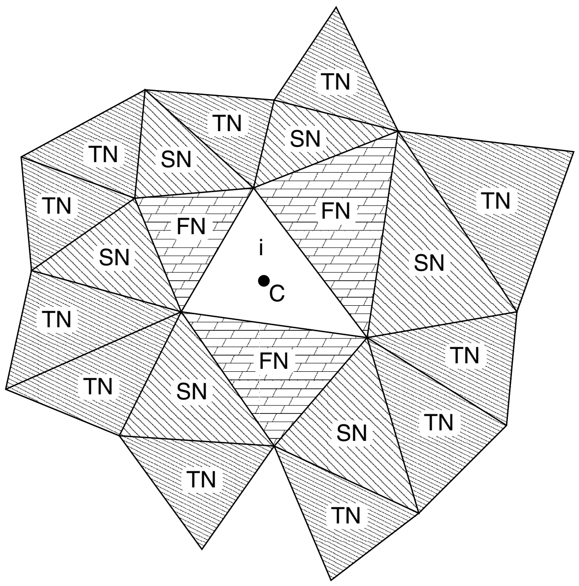


Fig. 1. A typical cell center control volume and its reconstruction stencil, including three layers of neighbors: FN, first neighbor; SN, second neighbor; TN, third neighbor.

least-squares reconstruction to filter out noise. We build stencils by successively adding layers of face neighbors, as shown in Fig. 1, until the desired stencil size is reached. Near boundaries, more layers of neighbors are typically required than in the interior, sometimes leading to geometrically large stencils.

Taken together, Eqs. (5) and (7) form a constrained least squares system, from which we typically eliminate the mean constraint analytically by Gauss elimination. After solving the remaining *un*constrained least squares system by singular value decomposition, we enforce the mean constraint by solving Eq. (5) for the point value at the reference location, V_C.

3.2. Numerical flux computation

We compute the numerical fluxes by using Roe's flux differencing scheme [29] with Harten's entropy fix [15]

$$F(U_L, U_R) = \frac{1}{2}[F(U_L) + F(U_R)] - \frac{1}{2}|\widetilde{A}|(U_R - U_L), \tag{8}$$

where U_L and U_R are computed by evaluating the reconstruction polynomials $V_R^{(k)}$ for the left and right control volumes at the Gauss point. $\widetilde{A}$ is the Jacobian matrix evaluated based on the Roe's average properties $\widetilde{U}(U_R, U_L)$, and $|\widetilde{A}|$ is written as:

$$|\widetilde{A}| = \widetilde{X}^{-1} \begin{vmatrix} f(\lambda_1) & & & \\ & f(\lambda_2) & & \\ & & f(\lambda_3) & \\ & & & f(\lambda_4) \end{vmatrix} \widetilde{X}, \tag{9}$$

where $\widetilde{X}$ are the right eigenvectors and Λ are the eigenvalues of the Jacobian matrix, and $f(\widetilde{\lambda}_i)$ incorporates the entropy fix:

$$f(\lambda) = \begin{cases} |\lambda|, & |\lambda| \geqslant \delta, \\ \frac{\lambda^2+\delta^2}{2\delta}, & |\lambda| < \delta. \end{cases}$$

In addition to eliminating expansion shocks, the entropy fix also makes Roe's flux function differentiable.

3.3. Flux integration

The accuracy of flux integration should be equal or higher than the accuracy of flow variable reconstruction to evaluate the residuals with high-order accuracy. This is achieved by standard Gauss quadrature. For the second-order (linear reconstruction) case, we use one quadrature point per face, whereas for the third-order (quadratic reconstruction) and fourth-order (cubic reconstruction) cases, we use two equally-weighted quadrature points per face.

3.4. High-order boundary condition enforcement

Boundary treatment is critical to achieving genuinely high-order accurate solutions for domains with curved boundaries. The shape of the domain must be known to at least the desired order of accuracy of the solution. In particular, the boundary Gauss quadrature information and moments of boundary control volumes must be computed properly.

Boundary Gauss points must fall on the curved boundary rather than on a piecewise linear approximation to the boundary, as shown in Fig. 2. Specifically, the boundary Gauss points must be spaced along the boundary according to arc length; in other words, for two Gauss points (third- and fourth-order accuracy), the Gauss points (1 and 2) must lie at $\frac{1}{2} \pm \frac{1}{2\sqrt{3}}$ of the way from one boundary vertex (A) to the next (B), as measured along the arc. The Gauss weights for such points must each be half of the arc length of the curve, while the normals used at boundary Gauss points must be perpendicular to the curved boundary, as shown. Gauss points that are "properly" located on the piecewise linear approximation to the boundary ($1'$ and $2'$) are $O(h^2)$ from the curved boundary, in general, and will cause second-order errors.

When computing moments for control volumes with curved boundaries, again integration points must be exactly on the curved boundary. Also, care must be taken to ensure that the integration is sufficiently accurate: the normal direction is no longer constant, and so the integrand in Eq. (6) is an arbitrary smooth function. We use three Gauss points along each curved boundary edge (such as AB) when computing moments to ensure that the moments are more accurate than the solution will be.

We employ two strategies to enforce boundary conditions to high-order accuracy. Many boundary conditions express a constraint on the pointwise value of the solution on the boundary. We enforce such boundary conditions by adding an additional constraint to the least-squares system. For example, for an isothermal wall condition, we would write (for a Gauss point $\vec{x}_{G_j}$ in control volume i and assuming a third-order accurate reconstruction):

$$T_w = T_i + \left.\frac{\partial T}{\partial x}\right|_i (x_{G_j} - x_i) + \left.\frac{\partial T}{\partial y}\right|_i (y_{G_j} - y_i) + \left.\frac{\partial^2 T}{\partial x^2}\right|_i \frac{(x_{G_j} - x_i)^2}{2} + \left.\frac{\partial^2 T}{\partial x \partial y}\right|_i (x_{G_j} - x_i)(y_{G_j} - y_i) + \left.\frac{\partial^2 T}{\partial y^2}\right|_i \frac{(y_{G_j} - y_i)^2}{2}.$$

In general, the boundary constraints can not be analytically eliminated from the least-squares system *a priori* (unlike the mean constraint), so these must be numerically eliminated by using Gauss elimination with pivoting. In certain cases, especially for vertex-centered control volumes, the boundary constraints may not be linearly independent, and so not all can be simultaneously satisfied by the reconstruction. Also, some boundary constraints couple variables in the reconstruction. For example, in inviscid flow, the wall boundary can be stated as a requirement for zero normal velocity, $u_n \equiv u n_x + v n_y = 0$. This boundary condition couples the reconstruction coefficients for u and v, making the reconstruction more expensive to compute. For boundary conditions enforced by solution constraints, we use the analytic flux at the boundary; that is, for the Euler equations we calculate $(\rho u_n \;\; \rho u u_n + P n_x \;\; \rho v u_n + P n_y \;\; u_n(E+P))^T$ using data from the reconstruction evaluated

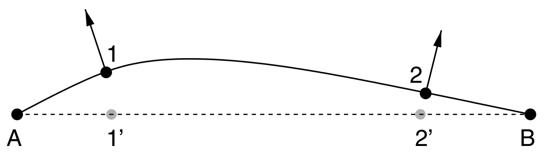


Fig. 2. A piece of a curved boundary, showing correct (1, 2) and incorrect ($1'$, $2'$) Gauss quadrature points.

at the Gauss point. For a full description of how to enforce boundary conditions using constraints on the reconstruction, see Ollivier-Gooch and Van Altena [26].

An alternative to boundary constraints is boundary condition enforcement by the use of special fluxes. For example, in the case of the inviscid wall boundary condition, we could compute an unconstrained reconstruction in boundary control volumes, then compute a flux that explicitly enforced $u_n = 0 : \ (0\ Pn_x\ Pn_y\ 0)$. Although we have not used this approach in the present work, we have applied this wall boundary condition with excellent results (both in solution quality and convergence rate) in other work [19].

In the present work, we also require a far-field boundary condition, for which we apply characteristic boundary conditions. That is, for subsonic inflow, we use three pieces of data from the far field (total pressure, total temperature, and flow angle) to compute three of the four characteristic invariants at the boundary, with the fourth coming from reconstructed data within the domain. Once a boundary state is established from the invariants, we calculate the analytic Euler flux at the boundary.[1] A similar approach is applied for subsonic outflow; supersonic inflow and outflow are trivial to apply, regardless of order of accuracy.

3.5. Enforcing monotonicity

Least-squares polynomial reconstruction of piecewise constant control volume averaged data produces overshoots in the reconstruction in the vicinity of sharp gradients and discontinuities, which can in turn produce stability problems. This occurs because the reconstruction stencil contains control volumes on opposite sides of the discontinuity – data that are not smoothly connected. In essence, when reconstructing non-smooth data near a discontinuity, the data are not *physically* compact even though the mesh stencil is *geometrically* compact. We enforce monotonicity by reducing the computed gradients (slope limiting), suppressing overshoots resulting from reconstruction over a non-physically compact stencil. Specifically, we require that the solution value $V_i(\vec{x}_\mathrm{G})$ that we compute at a Gauss point $\vec{x}_\mathrm{G}$ must fall between the highest and lowest control volume averages in the vicinity of control volume i:

$$\overline{V}_{\max} = \max(\overline{V}_i, \overline{V}_{\mathrm{FN}_j}), \tag{10}$$

$$\overline{V}_{\min} = \min(\overline{V}_i, \overline{V}_{\mathrm{FN}_j}), \tag{11}$$

$$\overline{V}_{\min} \leqslant V_i(\vec{x}_\mathrm{G}) \leqslant \overline{V}_{\max}, \tag{12}$$

where $\overline{V}_i$ is the control volume average and $\overline{V}_{\mathrm{FN}_j}$ are the control volume averages of the first neighbors. In theory, Eq. (12) should be valid for all points inside the control volume i, but in practice we check this condition only for Gauss points where the fluxes are actually computed. The unlimited reconstructed value at the Gauss point G is found by evaluating the reconstruction of Eq. (4)

$$V_i(\vec{x}_\mathrm{G}) = V_\mathrm{R}^{(k)}(\vec{x}_\mathrm{G}). \tag{13}$$

In the linear (second-order) case, the gradient ∇V_C computed from the reconstruction procedure is adjusted to satisfy the monotonicity condition, Eq. (12), by a scalar slope limiter ϕ

$$V_\mathrm{G} = V_\mathrm{C} + \phi_i \nabla V|_C \overrightarrow{r}_\mathrm{G}. \tag{14}$$

In general, an ideal limiter is differentiable and acts firmly in the shock region suppressing possible over/undershoots. A limiter also should not be active in smooth regions despite the existence of non-monotone solutions. These issues are even more severe for high-order methods. Venkatakrishnan's limiter [33], though not fully differentiable, addresses most of these issues. For each Gauss point, Venkatakrishnan's limiter computes ϕ_G from

$$\begin{aligned} \Delta_{1,\max} &= \overline{V}_{\max} - \overline{V}_i, \\ \Delta_{1,\min} &= \overline{V}_{\min} - \overline{V}_i, \\ \Delta_2 &= \mathrm{sign}(V_\mathrm{G} - \overline{V}_i)(|V_\mathrm{G} - \overline{V}_i| + \omega), \end{aligned} \tag{15}$$

[1] We recognize that the simplicity of this boundary condition has implications for how far away our far field boundary must be for external aerodynamic flows; while we intend to address this issue in the future, it is beyond the scope of the present research.

$$\varepsilon^2 = (K\Delta x)^3,$$

$$\phi_G = \begin{cases} \frac{1}{\Delta_2}\left[\frac{(\Delta_{1,\max}^2+\varepsilon^2)\Delta_2+2\Delta_2^2\Delta_{1,\max}}{\Delta_{1,\max}^2+2\Delta_2^2+\Delta_{1,\max}\Delta_2+\varepsilon^2}\right] & \text{if } \Delta_2 > 0 \\ \frac{1}{\Delta_2}\left[\frac{(\Delta_{1,\min}^2+\varepsilon^2)\Delta_2+2\Delta_2^2\Delta_{1,\min}}{\Delta_{1,\min}^2+2\Delta_2^2+\Delta_{1,\min}\Delta_2+\varepsilon^2}\right] & \text{if } \Delta_2 < 0 \end{cases} \tag{16}$$

where ω is chosen to be 10^{-12} for 64-bit arithmetic computations to prevent division by zero. Δx in Eq. (15) is the mesh length scale and can be picked as an average mesh length scale or a local mesh length scale. We use a local mesh length scale equal to the diameter of the circle inscribed in an equilateral triangle with the same area as the control volume

$$\Delta x = 2\sqrt{\frac{\text{Area}}{3\sqrt{3}}}. \tag{17}$$

Once a limiter value ϕ_i has been calculated for a control volume, the obvious analog to limited linear reconstruction suggests that we write the limited high-order reconstruction as

$$V_G^{(k)}(x_G, y_G) = \overline{V}_i + \phi_i[\{\text{Linear Part}\} + \{\text{High-Order Part}\}].$$

Because $\phi = 1 + \mathcal{O}(\Delta x^2)$ in smooth regions for Venkatakrishnan's limiter, this application of the limiter degrades solution accuracy from third- or fourth-order to second-order. We have found that the addition of a discontinuity detector, similar to that described by Delanaye and his co-workers [10,11], makes the limiter less diffusive. With this discontinuity detector, we write the limited reconstruction polynomial as:

$$V_G^{(k)}(x_G, y_G) = \overline{V}_i + [(1-\sigma)\phi_i + \sigma]\{\text{Linear Part}\} + \sigma\{\text{High-Order Part}\}, \tag{18}$$

where σ is a limiter for high-order terms. In smooth regions, the full high-order reconstruction is applied by choosing $\sigma = 1$. Near discontinuities we switch from the high-order to the limited linear polynomial to prevent possible oscillatory behavior of second and third derivatives. We compute the limiter value, ϕ, by using the Venkatakrishnan limiter, Eq. (16), with V_G computed using all terms. We then compute $\sigma(\phi)$, which is designed to be a differentiable switch for convergence reasons, with σ nearly one for ϕ larger than a critical value ϕ_0 and nearly zero for small values of ϕ [21]:

$$\sigma = \frac{1 - \tanh(S(\phi_0 - \phi))}{2}, \tag{19}$$

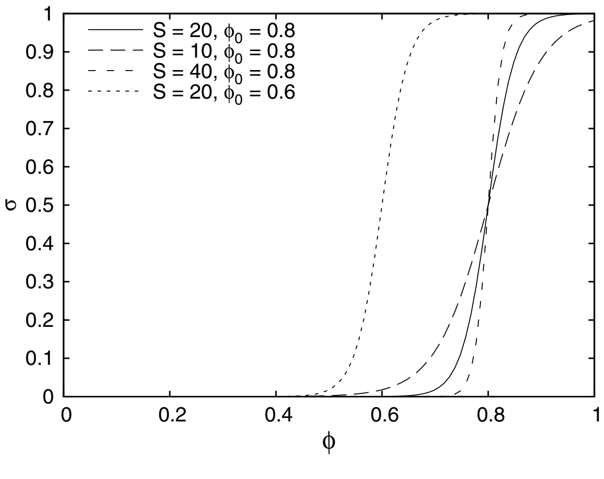


Fig. 3. Defining the discontinuity detector, σ, as a function of the limiter value, ϕ.

where S determines the sharpness of the step function, adjusting the rapidity of the transition between zero and one, while ϕ_0 defines the limiter value that activates the switch. Fig. 3 shows $\sigma(\phi)$ for several values of ϕ_0 and S. We have found that choosing $\phi_0 = 0.8$ and $S = 20$ provides a reasonable switch function whose good behavior is relatively case independent. The shock resolution and location are not sensitive to these values, which mainly affect convergence and the oscillations around the shock; the value of S has a greater effect on the convergence and oscillations around shock, and too small a value for S can also reduce accuracy in smooth regions. Too small a value for ϕ_0 can produce oscillations because the unlimited high-order reconstruction is sometimes used for cases where limiting is required for monotonicity. Also, too large a value for ϕ_0 can decrease accuracy in smooth regions by invoking limiting unnecessarily.

4. Convergence to steady state

We seek steady-state solutions, which implies that we wish to drive the residual, or flux integral, in Eq. (3) to zero as quickly as possible. We apply backward Euler time differencing to the semi-discrete form, which results in a time advance scheme of the form

$$\begin{aligned} &\left(\frac{I}{\Delta t_i} + \frac{\partial R}{\partial \overline{U}}\right)\delta \overline{U}_i = -R(\overline{U}_i^n), \\ &\overline{U}_i^{n+1} = \overline{U}_i^n + \delta \overline{U}_i \end{aligned} \tag{20}$$

where I is an identity matrix, $\frac{\partial R}{\partial \overline{U}}$ is the Jacobian matrix resulting from residual linearization, and $\delta \overline{U}_i$ is the solution update. Δt_i is a local time step computed based on a global CFL number and a local, per control volume, characteristic time [3].

Because we are interested only in the steady state solution, we are unconcerned about the first-order time accuracy of this time discretization, and in fact prefer to use the largest possible time steps. In the limit of infinite time step, Eq. (20) reduces to Newton iteration

$$\left(\frac{\partial R}{\partial \overline{U}}\right)\delta \overline{U}_i = -R(\overline{U}_i^n) \tag{21}$$

As is well-known, however, Newton's method converges only if the initial guess at the solution is sufficiently close to the actual solution and the linearization $\frac{\partial R}{\partial \overline{U}}$ is therefore sufficiently accurate. Therefore, like numerous other researchers, we divide our solution procedure into two phases: a start-up phase, during which we seek to efficiently obtain an approximation of the steady-state solution by advancing Eq. (20) in (pseudo-)time; and a Newton phase, during which we drive the residual rapidly to zero with a few iterations of Eq. (21).

Regardless of which phase we are in, we must solve a large, sparse system of linear equations to compute the solution update for each iteration. We use right-preconditioned GMRES [31] without restart as an iterative solver throughout; right preconditioning is used because this leaves the residual of the linear system unchanged. We apply the preconditioner by using ILU; we have found this to be both more robust and about an order of magnitude more efficient in CPU time compared with our previous approach of using LU-SGS [20]. The main distinctions between our start-up phase (described in Section 4.2) and our Newton phase (see Section 4.3) lie in how the Jacobian matrix $\frac{\partial R}{\partial \overline{U}}$ is approximated within GMRES and in how we compute and apply a preconditioning matrix. Computation of the preconditioning matrix used in GMRES is a complex enough subject to deserve separate treatment, which is found in Section 4.1.

4.1. Computing the preconditioning matrix

As will be discussed below in Sections 4.2 and 4.3, we never explicitly compute the high-order Jacobian; nevertheless, we must still explicitly compute and factor some approximation of the Jacobian for preconditioning, without which the convergence of GMRES for the linear system is quite poor. The optimal preconditioning matrix is not unique, depending both on the problem and on how the preconditioner is applied. A low-order Jacobian matrix captures the essential physical information about the flow, is narrowly banded after reordering and is much better conditioned than the high-order Jacobian. As such, constructing and applying such a matrix as a preconditioner for the high-order linear system will be relatively cheap compared with the cost of solving the original linear system.

We consider two distinct approaches to computing the first-order Jacobian. First, we derive an approximate analytic Jacobian of the first-order flux integral. Second, we consider a finite difference approximation to the first-order flux Jacobian.

4.1.1. Approximate analytic Jacobian

In this approach to Jacobian calculation, we consider a simplified (first-order) flux integral, and compute its Jacobian. In the case of the two-dimensional Euler equations discretized over a cell-centered unstructured mesh, each control volume has three direct neighbors, as shown in Fig. 4. The first-order flux integral or residual function of the control volume i only depends on these three neighbors and the control volume i itself:

$$R_i = \sum_{m=1,2,3} (F \cdot \hat{n}\,\mathrm{d}s)_m = F(U_i, U_{N_1}) \cdot \hat{n}_1 l_1 + F(U_i, U_{N_2}) \cdot \hat{n}_2 l_2 + F(U_i, U_{N_3}) \cdot \hat{n}_3 l_3, \tag{22}$$

where $\hat{n}_m$ and l_m are the outward unit normal and the length of the face m of the control volume i respectively. Next we take the derivative of the residual R_i with respect to the solution vector U in control volume i and its neighbors:

$$\frac{\partial R_i}{\partial U_{N_1}} = \frac{\partial F(U_i, U_{N_1})}{\partial U_{N_1}} \hat{n}_1 l_1, \tag{23}$$

$$\frac{\partial R_i}{\partial U_{N_2}} = \frac{\partial F(U_i, U_{N_2})}{\partial U_{N_2}} \hat{n}_2 l_2, \tag{24}$$

$$\frac{\partial R_i}{\partial U_{N_3}} = \frac{\partial F(U_i, U_{N_3})}{\partial U_{N_3}} \hat{n}_3 l_3, \tag{25}$$

$$\frac{\partial R_i}{\partial U_i} = \frac{\partial F(U_i, U_{N_1})}{\partial U_i} \hat{n}_1 l_1 + \frac{\partial F(U_i, U_{N_2})}{\partial U_i} \hat{n}_2 l_2 + \frac{\partial F(U_i, U_{N_3})}{U_i} \hat{n}_3 l_3. \tag{26}$$

Since both the flux F and solution U are four-component vectors, each entry in the Jacobian matrix is a 4×4 matrix. Although the total size of the block matrix is $n \times n$, where n is the total number of control volumes, there are never more than four non-zero blocks per row, and the resulting Jacobian matrix is very sparse.

The flux differencing term in Eq. (8), $|\widetilde{A}|(U_{N_1} - U_i)$, can be recast in the form of $|\widetilde{A}|\Delta U$ and the full derivative of this term with respect to the solution vector (in general form) is:

$$\frac{\partial(|\widetilde{A}|\Delta U)}{\partial U_j} = \overbrace{\frac{\partial|\widetilde{A}|}{\partial U_j}}^{1} \Delta U + \overbrace{|\widetilde{A}| \frac{\partial(\Delta U)}{\partial U_j}}^{2}, \tag{27}$$

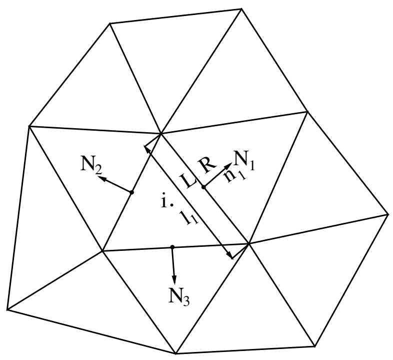


Fig. 4. Schematic of direct neighbors.

where the index j represents either control volume i or its neighbor. We neglect term 1 on the right side of Eq. (27) to avoid the complexity of third-rank tensors. Although this introduces an error in the Jacobian, the Jacobian is still accurate enough to serve as a preconditioning matrix or for the start-up linearization.This simplification of the Jacobian of Roe's flux can be written in the following form for cells i and N_1:

$$\frac{\partial F(U_i, U_{N_1})}{\partial U_{N_1}} = \frac{1}{2}\left[\frac{\partial F(U_{N_1})}{\partial U_{N_1}} - \left|\widetilde{A}\right|\right], \tag{28}$$

$$\frac{\partial F(U_i, U_{N_1})}{\partial U_i} = \frac{1}{2}\left[\frac{\partial F(U_i)}{\partial U_i} + |\widetilde{A}|\right]. \tag{29}$$

The other Jacobian terms in Eqs. (24)–(26) can be derived similarly.

We note that the data used to compute $\widetilde{A}$ and $\frac{\partial F}{\partial U}$ would ordinarily be the control volume averages $\overline{U}_i$ and $\overline{U}_{N_1}$. However, we find that in practice using the reconstructed values $V_{\mathrm{R}}^{(k)}(\vec{x}_{\mathrm{G}})$ in each control volume at the Gauss point produces a more effective preconditioning matrix. Because the reconstruction has already been computed, the additional cost of using this more accurate data is negligible.

With this approach we include the effects of boundary conditions in the approximate Jacobian by computing the Jacobian of the boundary flux with respect to the solution in the interior of the domain. In this case, only the $\frac{\partial F}{\partial U_i}$ term in the Jacobian need be computed.

The cost of one approximate analytic Jacobian evaluation is 0.6–0.7 times the cost of a second-order residual evaluation; the cost of reconstruction is included in the residual evaluation cost, but not the Jacobian cost, because the reconstruction is re-used.

4.1.2. Finite difference Jacobian

The second approach that we employ to find an approximate Jacobian for preconditioning is finite differencing. The finite difference Jacobian, though easier to code, is more expensive than the approximate analytic Jacobian, so we use it only when the need for a more accurate Jacobian justifies it. We compute the finite difference Jacobian by perturbing each element of the solution vector U at each Gauss point and recomputing the flux function; the difference between the perturbed and unperturbed flux functions yields one column of each of two blocks of the global Jacobian matrix. Boundary conditions can be treated implicitly by recomputing boundary fluxes with perturbed solution data.

The cost of one Jacobian evaluation is 1.3–1.5 times the computation cost of the same residual evaluation if the finite difference Jacobian is employed. Again, the reconstruction is re-used from the residual evaluation.

4.2. Start-up phase

During the start-up phase, we are far from the steady-state solution. Any linearization we make for the non-linear problem will be inadequate to drive Newton's method, or even to use a large time step in a pseudo-time-stepping scheme. In general, finding a good initial guess or reasonable approximate solution requires knowing the physics of the problem. Like preconditioning, start-up is problem dependent and is more an art than an exact science. Various techniques such as mesh sequencing, multigrid, mixed explicit/implicit iterations, exact solutions of a simplified problem, potential flow solutions, and so on can be helpful here. We base our start-up process on the defect correction procedure, *a la* Mavriplis [18].

Because we expect the start-up process to require multiple iterations to achieve a good enough initial guess to switch to Newton iteration, we use an implicit iterative process where the linearization is based on the inexpensive approximate analytic first-order Jacobian and the flux calculation remains high-order:

$$\left(\frac{I}{\Delta t_i} + \left.\frac{\partial R}{\partial U}\right|_{1\mathrm{st}}^{n}\right)\Delta U^{n+1} = -R_{\mathrm{High}}(U^n), \tag{30}$$

where Δt_i is a local time step based on a global CFL number times the maximum allowable explicit time step for control volume i [3]. We use the low-order Jacobian on the left-hand side to reduce per-iteration costs, as we are far from the exact numerical solution, and any linearization poorly predicts what that solution will be.

Under these circumstances, the inexact linearization on the left-hand side of Eq. (30) is not a hindrance to convergence, especially as time steps are relatively small to prevent us from going too far wrong in any one time step.

Because we are using an approximate linearization, it does not make sense to solve the approximate linear system exactly during these defect correction pre-iterations. Instead, we solve the linear system up to some fraction of the non-linear residual, *i.e.*, tolerance $= C \cdot \|\mathrm{Res}(U)\|_2$ with $C \approx 0.05$–0.1. The right-preconditioned GMRES algorithm is used with a limited subspace size; no restart is allowed. Because we are explicitly forming the low-order Jacobian during the start-up phase, we use matrix-explicit GMRES. The preconditioner matrix is the same approximate analytic Jacobian as the linear system matrix and the preconditioner is applied by using a low fill-level incomplete factorization, ILU(1), which is accurate and efficient enough for this first-order linear system.

In general, we decrease the L_2 norm of the non-linear residual by a given fraction before switching to Newton iterations. At that point, the linearization is deemed accurate enough to take a very large time step at each iteration. The value used for this criterion varies from problem to problem. However, it is possible to find reasonable values for different categories of compressible flows. More detail is provided in that regard in discussion of results in Section 5.

4.3. Newton phase

When the Newton phase of the solution procedure begins, the major transient behaviors of the flow field have been damped out, and the main steady features have appeared. As a result, the linearization of the non-linear residual is accurate enough for seeking the steady-state solution. To take advantage of this state of the solution, we use Newton iteration:

$$\left.\frac{\partial R}{\partial U}\right|_{\mathrm{High}}^{n} \Delta U^{n+1} = -R_{\mathrm{High}}(U^n). \tag{31}$$

In contrast to the start-up phase, our goal in the Newton phase is to perform very few iterations with updates that dramatically reduce the non-linear residual. As such, we require a precise linearization, and use of the high-order Jacobian is crucial. Rather than calculate the high-order Jacobian, we use matrix-free GMRES, eliminating the need to *explicitly* form this large, complex matrix. Instead, we compute the matrix vector products (Az) by a matrix-free approach using directional derivatives, at the cost of a high-order residual evaluation per GMRES inner iteration:

$$\frac{\partial R}{\partial U} \cdot z \approx \frac{R(U + \varepsilon z) - R(U)}{\varepsilon}, \quad \varepsilon = \frac{\varepsilon_0}{\|z\|_2}. \tag{32}$$

ε_0 is typically taken as the square root of machine accuracy to balance truncation and round-off effects. However, for fine meshes where the mesh length scale is very small, we must choose ε_0 larger than this so that perturbations in fourth-order terms are not lost in machine round-off errors. Again, a fixed number of search directions is employed in GMRES.

The linear system is right preconditioned by the low-order Jacobian, which indeed is essential due to bad conditioning of the high-order Jacobian. The first-order finite difference Jacobian, which is more accurate than the approximate first-order analytical Jacobian, is used as the preconditioner matrix. ILU(p) is again used for preconditioning, with a larger fill level (typically $p = 2$–4). Our experience indicates that for high-order computation (especially for fourth-order and transonic flows), increasing the fill level is highly beneficial; for a detailed discussion of preconditioning issues, including cost comparisons of the various alternatives, we refer interested readers to a companion article [22]. The linear system is solved approximately with a tighter tolerance ($10^{-2} \cdot \|R(U)\|_2$), as a more accurate update is needed for the Newton phase. No restart is allowed. Approximately solving the linear system can increase the number of non-linear outer iterations, but still reduces the overall CPU time as considerable computation time is saved by not solving the linear system to machine zero [7,28,17,21].

5. Results

5.1. Supersonic vortex, $M = 2.0$

To study the correctness, basic performance and solution accuracy of the proposed high-order unstructured Newton–Krylov solver, we have investigated a smooth (isentropic) supersonic vortex in an annulus geometry, whose exact solution is given in non-dimensional form by

$$\begin{aligned}
\rho &= \rho_{\mathrm{i}}\left(1+\frac{\gamma-1}{2}M_i^2\left(1-\frac{R_{\mathrm{i}}^2}{r^2}\right)\right)^{\frac{1}{\gamma-1}},\\
U_{\mathrm{i}} &= M_i\rho_{\mathrm{i}}^{\frac{\gamma-1}{2}},\\
U &= \frac{U_{\mathrm{i}}R_{\mathrm{i}}}{r},\\
u &= \frac{yU}{r},\\
v &= \frac{-xU}{r},\\
P &= \frac{\rho^{\gamma}}{\gamma}.
\end{aligned}$$

The unusual-looking form of U_i is a direct consequence of how the sound speed is computed with this non-dimensionalization. Having the exact solution provides us a direct option for accuracy measurement of the numerical solution. We choose an inner radius R_{i} of 2, an outer radius R_{o} of 3, a Mach number at the inner boundary M_{i} of 2, and a density at the inner boundary of $\rho_{\mathrm{i}} = 1$.

Five different meshes (Mesh 1: 108 CVs to Mesh 5: 27389 CVs) are employed in this test case. Fig. 5 shows the sequence of meshes. Each mesh has almost exactly four times more control volumes than the immediate

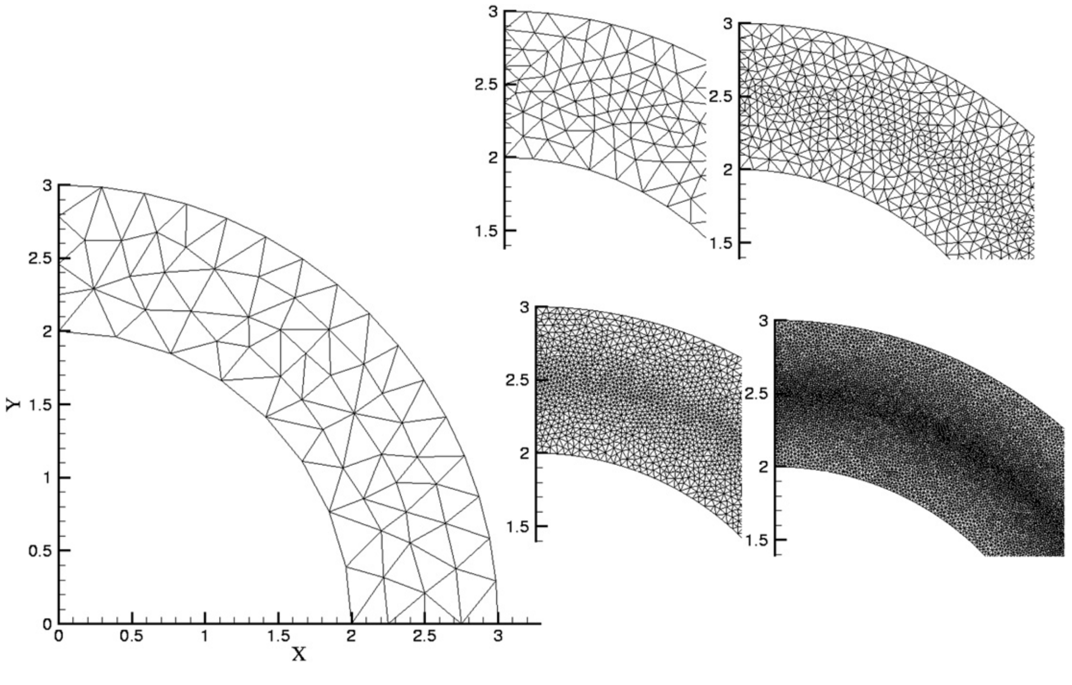


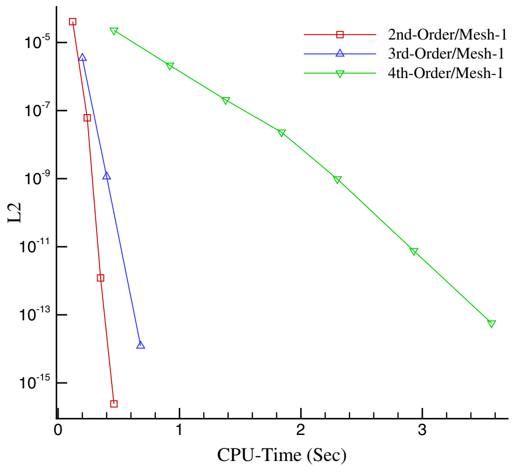
Fig. 5. Meshes used for the supersonic vortex test case.

coarser level and uniform refinement has been applied in the mesh generation. All meshes are irregular and have been created independently of each other.

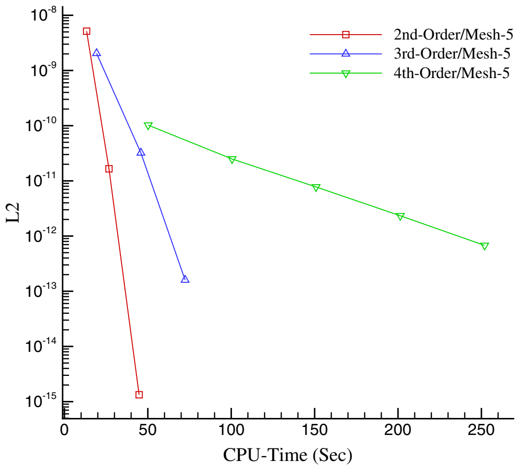
As an initial solution, we use the control volume averages of the exact solution, which differs from the converged numerical solution because of truncation error. While such a starting solution is clearly unrealistic in general, our goal here is to study the efficiency of our Newton–Krylov solver without confounding effects from the start-up process. In this way, we can first test the correctness of the Newton–GMRES solver for a given subspace size, preconditioning strategy and preconditioner matrix. Also, we can show how the efficiency of our matrix-free approach is affected purely by discretization order when the best possible starting solution is used. Newton iteration (infinite time step) is performed for all cases. An approximate analytic Jacobian with ILU(4) is used for preconditioning. The convergence criterion is $\|\text{Res}(U)\|_2 = 1 \times 10^{-12}$; $K = 30$ is used as the subspace size. Fig. 6 shows the convergence history for Mesh 1 and Mesh 5 in terms of CPU time. Since the solution is started from a good initial guess, super-linear convergence is achieved from the very first iteration both for the second and third-order discretizations. Full convergence to machine accuracy is achieved for the fourth-order case, but the CPU time is much larger than the other two orders of accuracy. The fourth-order discretization was of course expected to require the most CPU time, because more operations are performed, especially in the reconstruction, which has a cost that rises quadratically with increasing discretization order. However, a considerable part of the difference in computing cost is due to a noticeable increase in the number of GMRES outer iterations. The linear system arising from the fourth-order discretization is ill-conditioned and difficult to solve with the fixed subspace size which results in relatively poor solution updates for the non-linear problem, requiring more outer iterations for convergence. On the other hand, both the second and third-order cases quickly converge displaying the effectiveness of the preconditioning.

The accuracy of the current solver can be verified through comparison of the numerical and exact solutions over the sequence of meshes. Fig. 7 shows the L_1 and L_∞ norms of the error in density for this problem. The L_1 norm clearly reaches the nominal order of accuracy in all cases, while the L_∞ norm shows roughly a one order deterioration of convergence for all orders of accuracy. This is almost certainly due to an interaction between boundary conditions, which may be imposed in a way that is subtly incompatible in the corners of the domain.

Here, some accuracy–efficiency analysis can be carried out. The solution CPU time versus the error norm is plotted in Fig. 8. For a fixed CPU time, for example (CPU time = 10 s), the third-order performance characteristic line generates the minimum error. For a fixed error level, $L_1 = 1 \times 10^{-5}$, the fourth-order scheme has the minimum computation time, although on that error level the third and fourth-order lines are very close to each other. In all cases the second-order scheme is out-performed by its high-order counterparts. Though not shown here, a similar performance characteristic graph with the same trend, but different CPU time, can be achieved when the solution starts from a constant Mach number as an initial guess with a start-up process.



(a) Mesh 1, 108 CVs



(b) Mesh 5, 27389 CVs

Fig. 6. Convergence history for supersonic vortex.

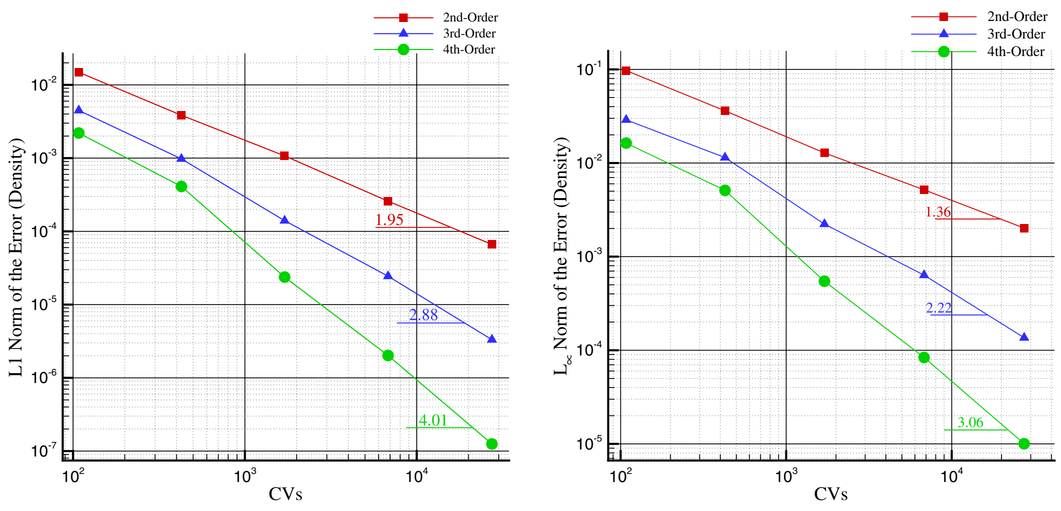


Fig. 7. Convergence of error in density with mesh refinement.

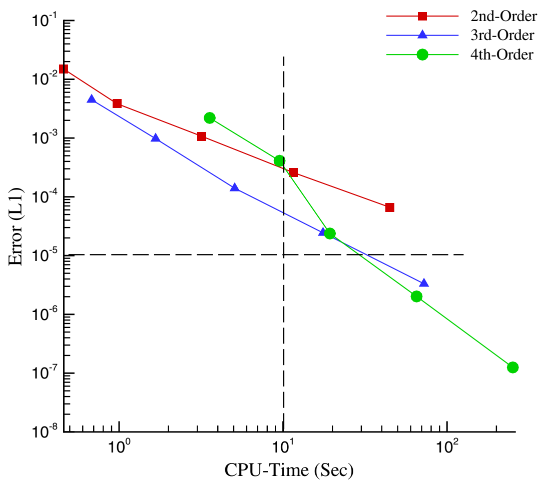


Fig. 8. Error versus CPU time for the supersonic vortex test case.

5.2. Subsonic airfoil, NACA 0012, $M = 0.63$, $\alpha = 2.0^\circ$

Here we present a subsonic case that illustrates most of the features of the solver's performance. The convergence characteristics are investigated for a series of meshes. Five different meshes from coarse to relatively fine have been used; the meshes are shown in Fig. 9. All meshes have proper refinement at the leading and trailing edges. The far field is located at 25 chords and characteristic boundary conditions are implemented implicitly.

The tolerance in solving the linear system for the start-up phase is 5×10^{-2} and for the Newton part is 1×10^{-2}. For all test cases a subspace of 30 has been set and no restart is allowed. The preconditioning for the start-up pre-iterations is performed by employing the approximate analytical Jacobian matrix with ILU(1) factorization and for the Newton iteration the first-order finite difference Jacobian matrix with ILU(4) factorization is used. The Newton iteration is matrix-free and $\varepsilon = \frac{\varepsilon_0}{\|z\|}$ with $\varepsilon_0 = 10^{-6}$ is used for directional differencing. The initial condition is the free stream flow.

Experiments for subsonic flow have shown that a reasonable starting point for Newton iteration can be easily achieved by a relatively small number of pre-iterations, and there is no need to decrease the residual

0.4
0.2
Y 0
0.4
0.2
Y 0

Fig. 9. Meshes used for the NACA 0012 airfoil cases.

by a significant factor. Only a rough physical solution over the airfoil is good enough for starting Newton iterations. The approach described below works well, but could almost certainly be optimized to be simpler and more efficient.

The solution starts with 30 pre-iterations in the start-up process to reach a good initial solution before switching to Newton iterations. As shown in Table 1, the CFL number starts at 2.0 and is increased gradually

Table 1
Variation of CFL number with iteration for subsonic airfoil case

Iteration	1–5	6–10	11–30	Newton
CFL	2	20	100	10^9

to CFL = 100 for the first 15 pre-iterations which are performed with first-order accurate flux evaluation. The remaining 15 pre-iterations are performed in the form of the defect correction with constant CFL of 100, where the first-order Jacobian is used both for constructing the left hand side and for preconditioning the linear system. The right hand side of Eq. (30) is evaluated to second-order accuracy. The cost of each pre-iteration includes one first order Jacobian evaluation and its incomplete factorization, one flux evaluation, and one linear system solve using GMRES, which is not matrix-free since the Jacobian matrix is available explicitly.

To justify our decision to use second-order accurate flux evaluation for defect correction, let us examine the effect on the solution at the end of the start-up procedure of using high-order accurate flux evaluations during defect correction. Fig. 10 displays the pressure coefficient over the upper surface of the airfoil at the end of the start-up process for Mesh 5, which is the finest mesh. While the general trend of the pressure coefficient over the airfoil is recovered, the suction peak has not been properly resolved yet. Because of the mesh volume, the diversity of mesh length scales, and the local time stepping approach, solution time evolution is far from complete, especially in the small cells near the leading edge; therefore, the suction peak is still growing. The key point to note here is the violent oscillations of the fourth-order discretization around the suction region in the course of the solution evolution. Such oscillations around extrema are not unusual during the transient part of the solution, especially in places where the size of boundary cells changes abruptly. The second and third-order discretizations show mild oscillations, but for the fourth-order scheme, with its low dissipation, the oscillations are quite vigorous. These oscillations can be a source of instability in the start-up process and indeed the fourth-order case is sensitive in the early period of solution process. Since the cost of the high-order start-up is more than the second-order one, and it could generate a noisy solution state (before switching to Newton iteration), we choose for the sake of robustness to use the second-order residual evaluation throughout the defect correction part of the solution process for all cases shown hereafter.

After start-up, the solution process is switched to Newton iteration and an infinite CFL is employed. To compare the computing cost for different meshes, a work unit has been used, which is equal to the cost of one residual evaluation for the corresponding order of accuracy; different meshes and different orders of accuracy therefore obviously require different amounts of CPU time per work unit. Convergence is reached when the L_2 norm of the non-linear density residual falls below 1×10^{-12}.

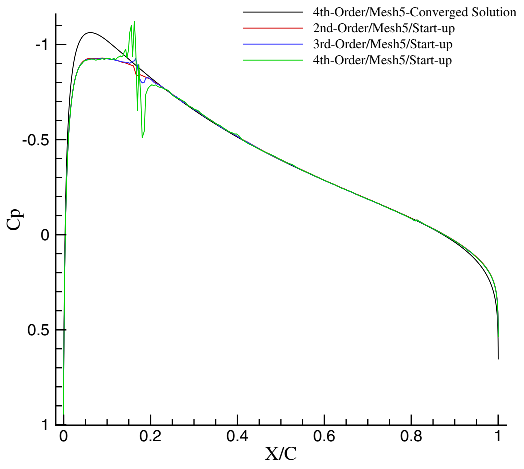


Fig. 10. Pressure coefficient over the upper surface after start-up, Mesh 5 (19957 CVs).

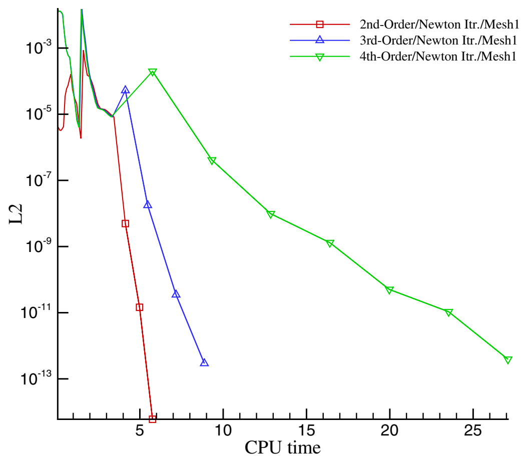
Table 2
Convergence summary for NACA 0012 airfoil, $M = 0.63, \alpha = 2^\circ$

Order	Resid. Eval.	Time (sec)	Work units	Newton iterations	Newton phase	
					Work Units	% of Total
Mesh 1						
2nd	100	5.76	240.0	3	96.7	40.3
3rd	132	8.87	197.1	4	122.4	62.1
4th	244	27.09	258.0	7	226.8	87.9
Mesh 2						
2nd	121	12.88	280.0	3	118.3	42.3
3rd	136	17.71	213.4	4	128.2	60.0
4th	283	58.15	312.6	8	274.8	87.9
Mesh 3						
2nd	126	26.88	349.1	3	136.1	39.0
3rd	147	36.03	248.5	4	141.2	56.8
4th	247	90.54	289.3	7	239.2	82.7
Mesh 4						
2nd	158	60.39	399.9	4	182.8	45.7
3rd	158	73.98	276.0	4	159.0	57.6
4th	318	217.60	371.4	9	317.8	85.6
Mesh 5						
2nd	254	164.10	562.0	7	325.3	57.9
3rd	286	225.87	456.3	8	321.8	70.5
4th	542	682.0	639.8	16	577.1	90.2

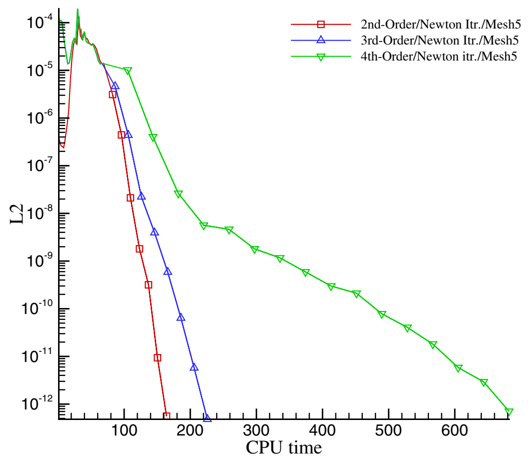
Table 2 summarizes the convergence behavior for the second-, third-, and fourth-order discretizations. For all meshes, after 30 pre-iterations, only a few Newton iterations are required to reach full convergence. The total number of work units required for convergence for moderate-sized meshes (meshes 3 and 4) range from 250 to 400, which is comparable to other results in the literature for second-order schemes. Also, the number of work units required varies little with order of accuracy, with third-order being (relatively) the fastest, and fourth-order the slowest of the three. The total number of work units does increase somewhat as finer meshes are used. The linear system arising from a denser mesh is more difficult to solve than a similar system arising from a coarser mesh, so using a constant subspace size without restart becomes less effective in solving the linear system as the size and complexity of the system increases. Consequently more outer Newton iterations are needed for convergence. Also, the solution state after the start-up for the fine mesh is not as close to the steady-state solution as for the coarse mesh; this also contributes to slowing the Newton convergence on the fine mesh. Notice that the total number of work units has increased only by about a factor of 2.5 while the mesh size has increased by a factor of 16, so while not fully mesh-size independent, the CPU time required for convergence is growing only slightly faster than the mesh size.

The convergence history for the coarse mesh (Mesh 1) and the fine mesh (Mesh 5) are shown in Fig. 11. Although the start up process is the same for all orders of discretizations, the starting residual is different because the residual (r_0) which initializes the implicit start-up is based on the correct corresponding order of residual evaluation. However, at the end of the start-up process, the solution state is almost the same for all discretization orders. After start-up, full convergence is achieved within a few Newton iterations in second and third order cases. For the fourth-order case convergence is about two times slower (in terms of outer iterations). The rapid Newton convergence for the second and third-order cases is evident.

Fig. 12a examines the correlation between the non-linear residual after a Newton iteration and the final residual of the linear system from that iteration. The correlation between these two is nearly linear, indicating that the linear system is solved progressively more accurately as the non-linear problem converges. Fig. 12b shows the quality of the linear system solution for each Newton iteration. The linear residual reduction within the GMRES linear solver is plotted versus the final residual of that system for all outer iterations. In other words this shows how much the linear system residual is reduced through inner iterations. For the

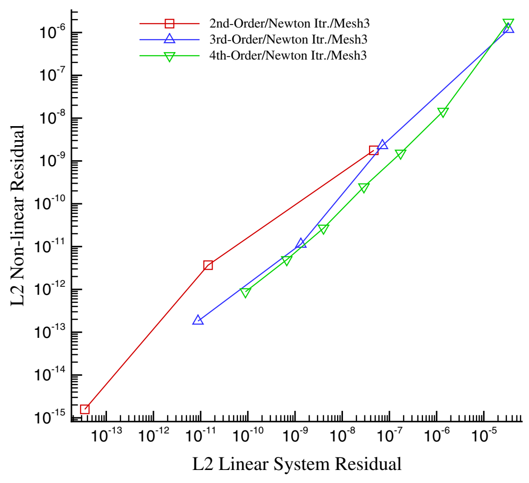


(a) Mesh 1, 1245 CVs

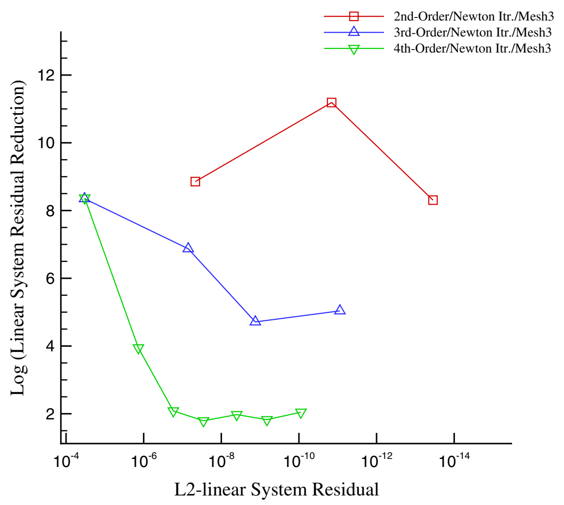


(b) Mesh 5, 19957 CVs

Fig. 11. Convergence history, NACA 0012, $M = 0.63, \alpha = 2^\circ$.



(a) Non-linear residual versus linear system residual



(b) Linear system residual drop

Fig. 12. Non-linear/linear residual detail, NACA 0012, Mesh 3, $M = 0.63, \alpha = 2^\circ$.

second-order case the linear system is solved quite effectively and more than 8 orders of residual reduction are achieved given the subspace size of 30 with no restart. For the third-order and fourth-order discretization this residual reduction is about 5 and 2 orders respectively for the same subspace size. This reduction in quality in solving the linear system eventually increases the number of Newton iterations for high-order discretizations. However, we have shown elsewhere that by using multiple restarts and solving the linear system up to machine accuracy for each Newton iteration semi quadratic convergence is attainable even for the fourth-order discretization with a dramatic penalty in computation cost [22].

Lift and drag coefficients for all meshes and discretization orders are tabulated in Table 3. In the finest mesh case, lift coefficients up to third decimal place and the drag coefficients up to the fourth decimal place have converged to the same value. However, the drag coefficient still is far from zero. This is mainly due to lack of pressure recovery at the trailing edge singular point. The third-order drag coefficient is consistently larger than the second-order drag coefficient, which is surprising. However, the fourth-order drag is consistently the smallest, as was expected.

Table 3
Lift and drag coefficients for all meshes and discretization orders, NACA 0012, $M = 0.63, \alpha = 2^\circ$, far field size of 25 chords

Test case	Lift coefficient	Drag coefficient
Mesh1		
2nd	0.322318	0.00097664
3rd	0.317393	0.00184889
4th	0.322223	0.00072576
Mesh2		
2nd	0.322302	0.00057225
3rd	0.321846	0.00103438
4th	0.322448	0.00040369
Mesh3		
2nd	0.324905	0.00040197
3rd	0.325214	0.00049820
4th	0.325588	0.00034757
Mesh4		
2nd	0.325159	0.000336973
3rd	0.324740	0.00037595
4th	0.325323	0.00032525
Mesh5		
2nd	0.324801	0.00032136
3rd	0.324568	0.00032711
4th	0.324474	0.00030828

Table 4
Effect of the far field distance on lift and drag coefficients, NACA 0012, $M = 0.63, \alpha = 2^\circ$

Mesh information	Order	Lift coefficient	Drag coefficient
Mesh 4-1	2nd	0.332506	0.00010622
10,604 CVs	3rd	0.332882	0.00012667
128 Chords	4th	0.332699	7.09651e-5
DeZeeuw and Powell [9]	2nd	0.3289	0.0004
10,694 CVs (Cartesian)			
128 chords			
Mesh 5-1	2nd	0.333575	4.88588e-5
22,421 CV	3rd	0.333530	5.98949e-5
200 Chords	4th	0.334306	4.04234e-5
Mesh 5-2	2nd	0.332908	1.82357e-5
24,514 CV	3rd	0.333715	2.35558e-5
1600 Chords	4th	0.333374	5.66076e-6

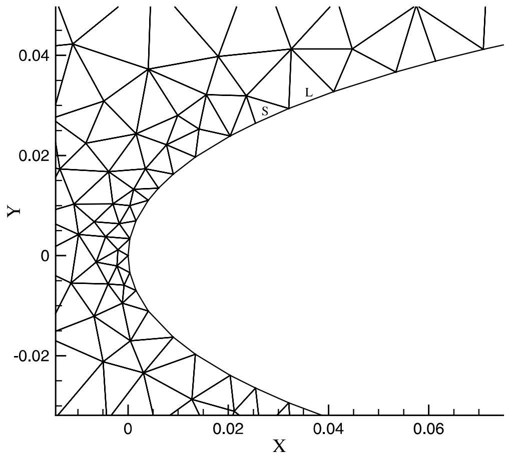
All of the test cases so far are performed with a far field size of 25 chords. To study the effect of the far field size on lift and drag, and to show that computing smaller drag is possible with the same mesh resolution, the far field size is increased considerably while the mesh resolution, the number of points on the airfoil and refining factors remain the same. With this approach, the new meshes are equivalent to the meshes of the previous test cases up to 25 chords. The results are tabulated in Table 4. In all cases the drag coefficient has been reduced dramatically by increasing the far field distance. For instance, the fourth-order computed drag over Mesh 5-2 has been reduced by more than 50 times compared to the fourth-order computed drag over Mesh 5. The lift coefficient is also affected by the far field distance: in the most cases the lift coefficient was slightly increased by extending the far field size.[2] Included in the table is a comparison result from DeZeeuw and

[2] It should be mentioned that in this research no far field correction is used, and high-order far field correction was beyond of the scope of this research.

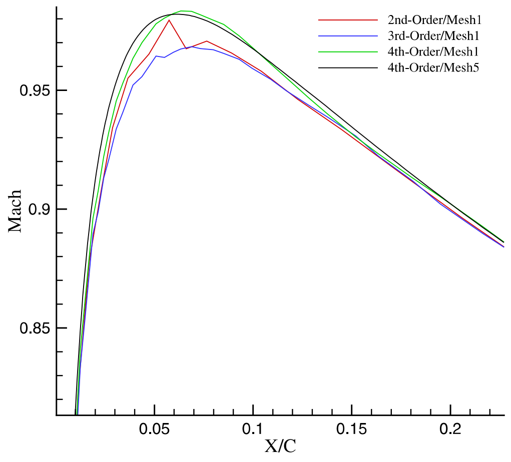
Powell [9]; their mesh is roughly comparable to our Mesh 4-1, and their scheme is second-order accurate. Lift coefficients vary by only about 1% between our Mesh 4-1 results and the DeZeeuw and Powell result, and our drag coefficients on that mesh are three to six times smaller.

While general conclusions about drag accuracy would be premature without more careful study, we can say with confidence that both far field boundary conditions and the behavior of the discretization at the trailing edge of the airfoil are crucial factors. Also, based on the comparison of second- and third-order results, we speculate that discretization methods with even orders of accuracy may benefit from a cancellation effect of some sort.

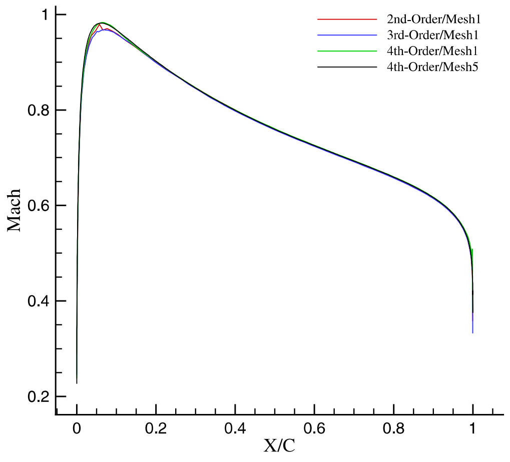
The Mach profiles (computed at Gauss points) along the chord of the airfoil for Mesh 1 (the coarsest mesh) are shown in Fig. 13. We see the effect of a sudden change in the area of boundary cells in the acceleration region. In this converged solution, the effect on the Mach profile is most pronounced for the second-order scheme. However, the high-order solution has reduced the amplitude of this jump so that in the fourth-order case, the jump has disappeared. Also, the Mach profile over the lower surface of the airfoil shows the second-order Mach profile is slightly lower than its high-order counterparts. We conclude that the flow acceleration



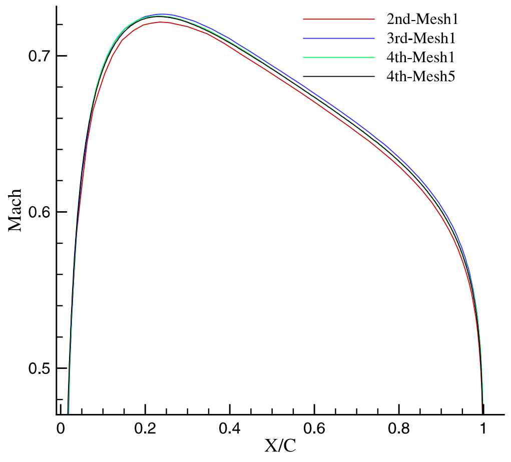
(a) Mesh 1, near the leading edge.



(b) Close up of upper surface Mach number near the suction peak.



(c) Upper surface Mach number.



(d) Closeup of the lower surface Mach number.

Fig. 13. Mach distribution on the airfoil surface, NACA 0012 airfoil, Mesh 1, $M = 0.63, \alpha = 2^\circ$.

around the leading edge of the airfoil is better recovered for high-order discretizations when the mesh resolution is coarse.

5.3. Transonic airfoil, NACA 0012, $M = 0.8, \alpha = 1.25^\circ$

For a transonic flow, in general, it is more difficult to get fast convergence. This is because of the mixed subsonic/supersonic nature of the flow and the existence of discontinuities (shocks) in the solution. The methodology for handling discontinuities can increase the complexity of the problem. This is especially true for implicit schemes, where the solution – and perhaps more sensitively the limiter values – can change dramatically between iterations. In the case of the matrix-free approach in which matrix-vector multiplication is computed through flux perturbation, any oscillatory behavior in the limiter could severely degrade the accuracy of the Frechet derivatives and therefore the solution convergence. All these factors increase the complexity and difficulty in solving transonic flows.

The transonic flow around the NACA 0012 at $M = 0.8, \alpha = 1.25^\circ$ is studied. Mesh 3 with 4958 CVs is employed for this test case.[3] Flow is solved for all orders of accuracy using Venkatakrishnan's limiter with proper high-order modification. For the second and third-order cases $K = 10$ is used in the limiter, and for the fourth-order discretization $K = 1$ is employed. The limiter values are allowed to change through all iterations and no freezing is considered. The tolerance of solving the linear system, like previous test cases, for the start-up phase is 5×10^{-2} and for the Newton phase is 1×10^{-2}. For all test cases a subspace of 30 has been set and no restart is allowed. The preconditioning for the start-up pre-iterations is performed using the approximate analytical Jacobian matrix with ILU(1) factorization and for the Newton iteration the finite difference Jacobian matrix with ILU(4) factorization is applied. The Newton iteration is matrix-free and $\varepsilon = \frac{\varepsilon_0}{\|z\|}$ with $\varepsilon_0 = 10^{-7}$ is used for directional differencing. The initial condition is free stream flow. Convergence is reached when the L_2 norm of the non-linear density residual falls below 10^{-12}.

For transonic flow, the shock locations in the flow field and their strengths need to be captured relatively accurately before switching to Newton iteration; otherwise Newton iterations will not decrease the residual of the non-linear problem effectively. This normally is achieved by reducing the non-linear residual by some 1.5–2 orders of magnitude with respect to the initial residual in the course of the start-up process.

Multiple implicit pre-iterations are performed in the form of defect correction, before switching to Newton iteration. For the second and third-order start-up phases, pre-iterations in the form of defect correction continue until the residual of the non-linear problem drops 1.5 order below the residual of the initial condition. As shown in Table 5, in the defect correction phase the starting CFL number is 2 and it is increased gradually to 200 after 50 iterations. The CFL is not increased further because increasing CFL does not help convergence when the linearization is inaccurate, as in the start-up phase. The second and third-order discretizations require, respectively, 69 and 81 pre-iterations in the start-up phase. In the Newton phase, the second- and third-order cases use a very large, but not infinite time step. The start-up phase for the fourth-order discretization includes 200 pre-iterations with a similar CFL trend. Although the target residual reduction of 1.5 orders was not achieved through pre-iterations, the solution after 200 iterations was good enough for starting the Newton phase. For the fourth-order transonic case, using too large a time step in the Newton phase does not accelerate convergence because of inaccurate linearization and limiter oscillation; therefore, CFL = 10,000 has been set for the Newton phase for fourth order.

Fig. 14 displays the Mach profile at the end of the start-up phase for all orders of accuracy. There are some oscillations, especially near the strong upper surface shock; however, the shock locations and their strengths are captured reasonably accurately before switching to Newton iteration.

The convergence summary is shown in Table 6. Similar to the subsonic case, the number of Newton iterations for the fourth-order discretizations is twice as large as for the second and third-order discretizations. The major cost of the solution in all cases is the start-up cost. Therefore there is a reasonable potential to reduce the total solution work dramatically by employing an optimized start-up technique which is able to

[3] We show results for only a single mesh here because mesh refinement illustrates only the well-known fact that the shock location is more sharply defined on finer meshes.

Table 5
Variation of CFL number with iteration for the transonic airfoil case

	CFL at iteration...					Total number of pre-iterations
	1–10	11–30	31–50	51+	Newton	
2nd	2	20	100	500	10^6	69
3rd	2	20	100	200	10^6	81
4th	2	20	100	200	$5@5 \times 10^3$, then 10^4	200

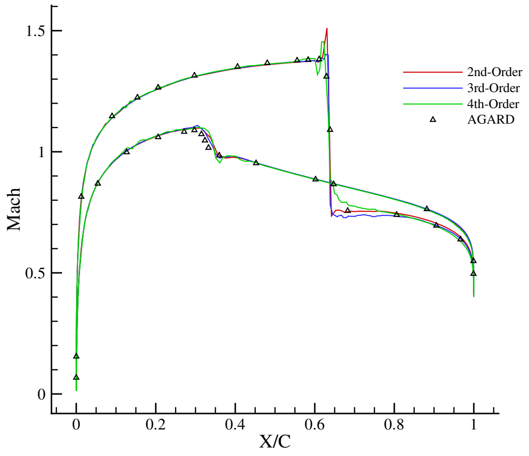


Fig. 14. Mach profile at the end of the start-up process, $M = 0.8, \alpha = 1.25°$.

Table 6
Convergence summary for NACA 0012 airfoil, $M = 0.8, \alpha = 1.25°$

Order	Residual evaluations	Time (s)	Work Units	Newton iterations	Work units, Newton phase
2nd	197	65.6	279	4	91
3rd	241	106.7	281	5	119
4th	450	311.4	590	10	221

capture shocks and establish the major flow features efficiently. Fig. 15 shows the convergence history graph. This time, reduction in convergence rate for the fourth-order case is not only due to poor convergence of the solution of the linear system in each Newton iteration, but also to the limiter firing which changes the linearization in each iteration.

Manzano et al. [17] presented a very fast Newton-GMRES algorithm based on a second-order artificial dissipation discretization. For the same case on a similar unstructured mesh, their total computation cost in terms of work units is just under 400. Our second-order scheme is about 30% faster than this; even allowing for differences in speed caused by differences in implementation, the current solver is very competitive.

Table 7 summarizes the lift and drag coefficients for all discretization orders, which are in good agreement with the AGARD reference data [1]. Both C_L and C_D in transonic flow are effectively determined by location and strength of the shocks; our good agreement with this reference solution confirms that our scheme captures shocks accurately.

The Mach profile along the surface is shown in Fig. 16. Both the location and strength of the shocks are in good agreement with the AGARD data [1]. The second-order discretization has an overshoot right before the upper shock. The third-order discretization produces less noise in this shock capturing case. The fourth-order discretization shows some oscillations inside the shock region; this behavior is clearly related to the high-order discretization, although not directly to a Gibbs-like phenomenon in the reconstruction within control volumes in the shock, because the oscillations span multiple control volumes.

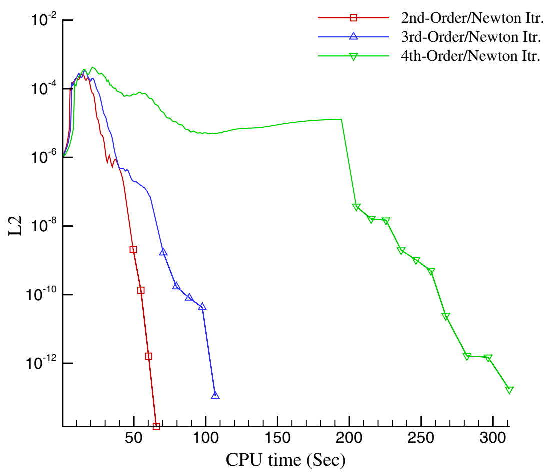


Fig. 15. Convergence history for NACA 0012, $M = 0.8, \alpha = 1.25^\circ$.

Table 7
Lift and drag coefficients, NACA 0012, $M = 0.8, \alpha = 1.25^\circ$

Test case	C_L	C_D
2nd	0.3375	0.02205
3rd	0.3393	0.02226
4th	0.3451	0.02247
AGARD-211 structured (192 × 39)	0.3474	0.0221

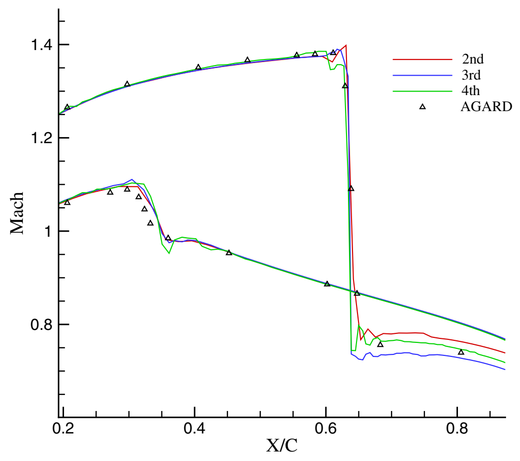


Fig. 16. Closeup of the Mach number distribution on the airfoil surface, NACA 0012, $M = 0.8, \alpha = 1.25^\circ$.

6. Concluding remarks

While, in general, computing cost remains one of the main concerns for the high-order computation of fluid flow problems, this article demonstrates that fast convergence – and a reasonable computing cost – for high-order unstructured solutions is indeed possible. Results for the implicit flow solution algorithm described in this research show that the second- and third-order schemes both display semi-quadratic or super-linear convergence for all test cases if started from a good initial guess or approximate solution. The fourth-order scheme still converges quickly although it is slower than the other discretization orders, requiring nearly two times the number of outer iterations as the second-order scheme started from a similar approximate flow solution. Our work shows that an efficient start-up technique, a good preconditioner matrix, and an effective preconditioning strategy are key issues for robustness and fast convergence of a Newton–GMRES solver, not only for second-order discretizations but for high order as well. The current second-order solver algorithm, even without multigrid augmentation, is among the world's fastest solver algorithms for unstructured meshes. The third and fourth-order schemes are comparable in efficiency to the second-order scheme as measured by residual evaluations.

The start-up procedure we describe is very effective in providing a good approximate solution as an initial condition for the Newton phase with a reasonable cost. We use a defect correction procedure, with a high-order residual evaluation and a first-order approximate analytic Jacobian; we form the Jacobian explicitly, and use it both in matrix-vector multiplies in GMRES and as a preconditioner matrix. For subsonic cases, a small fixed number of pre-iterations with a second-order residual is sufficient to reach a good initial solution for all orders. For the transonic case, we use the full-order residual evaluation and the number of pre-iterations increases, since some residual reduction target must be met before switching to Newton iterations; for the fourth-order scheme, an upper limit on the number of pre-iterations was reached, but with no ill effects on the robustness of the Newton solver.

Preconditioning and the accuracy of the linear system solution are vital factors in the Newton phase of the Newton–GMRES solver. We have shown that the use of a first-order finite difference Jacobian for preconditioning is sufficient, provided that the fill level in the ILU factorization is high enough to take advantage of all the information in the preconditioner matrix. We have found that ILU(4) factorization provides excellent performance for the second- and third-order schemes, and very good performance for the fourth-order scheme. For the fourth-order scheme, convergence of the GMRES inner iterations is slow, because the preconditioned matrix is still too ill-conditioned.

The performance of the current flow solver in terms of CPU time scales only slightly faster than linearly with the mesh size for all orders of discretization. As an overall performance assessment (including the start-up phase), the third-order solution is about 1.3–1.5 times, and the fourth-order solution is about 3.5–5 times, more expensive than the second-order solution with the current solver technology. With increasing mesh size, eventually the fixed subspace size in GMRES is insufficient to solve the linear system to our pre-specified tolerance at each iteration. A multilevel approach is likely to alleviate this problem, whether applied as an algebraic multigrid preconditioner within GMRES or as a non-linear full-approximation multigrid scheme with GMRES acting as a smoother. A similar approach should also prove useful during start-up.

Extension of the current high-order 2D solver algorithm to a 3D version, considering the availability of the 3D reconstruction procedure, is reasonably straightforward given high-order boundary data and control volume moments for a 3D mesh. Since the 2D algorithm is developed for unstructured meshes, there is no limitation in such an extension, and the performance of the flow solver in terms of the convergence could even be improved by introducing the third dimension (3D relieving effect). Extension of the current solution procedure to viscous flow computation will require, at a minimum, viscous residual function evaluation and a proper anisotropic unstructured/hybrid mesh. In addition, there are likely to be convergence issues related to both geometrical and physical stiffness, and to interaction between the mean flow and a turbulence model. These issues will require careful attention to proper preconditioning.

Acknowledgment

This work was supported by the Canadian Natural Sciences and Engineering Research Council under Grant OPG-0194467.

We thank the reviewers of this paper for many helpful comments which improved the presentation significantly.

References

[1] AGARD Fluid Dynamics Panel. Test Cases for Inviscid Flow Field Methods. AGARD Advisory Report AR-211, AGARD, May 1985.

[2] H.E. Bailey, R.M. Beam, Newton's method applied to finite difference approximations for steady state compressible Navier–Stokes equations, Journal of Computational Physics 93 (1991) 108–127.

[3] Timothy J. Barth, Aspects of unstructured grids and finite-volume solvers for the Euler and Navier–Stokes equations, in: Unstructured Grid Methods for Advection-Dominated Flows, pp. 6-1–6-61. AGARD, Neuilly sur Seine, France, 1992. AGARD-R-787.

[4] Timothy J. Barth, Recent developments in high order *k*-exact reconstruction on unstructured meshes, AIAA paper 93-0668, January 1993.

[5] Timothy J. Barth, Paul O. Frederickson, Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction, AIAA paper 90-0013, January 1990.

[6] Timothy J. Barth, Samuel W. Linton, An unstructured mesh Newton solver for compressible fluid flow and its parallel implementation, AIAA paper 95-0221, January 1995.

[7] Max Blanco, David W. Zingg, Fast Newton–Krylov method for unstructured grids, American Institute of Aeronautics and Astronautics Journal 36 (4) (1998) 607–612.

[8] S. De Rango, D.W. Zingg, Higher-order spatial discretization for turbulent aerodynamic computations, American Institute of Aeronautics and Astronautics Journal 39 (7) (2001) 1296–1304.

[9] Darren De Zeeuw, Kenneth G. Powell, An adaptively refined Cartesian mesh solver for the Euler equations, Journal of Computational Physics 104 (1) (1992) 56–68.

[10] M. Delanaye, J.A. Essers, Quadratic-reconstruction finite volume scheme for compressible flows on unstructured adaptive grids, American Institute of Aeronautics and Astronautics Journal 35 (4) (1997) 631–639.

[11] Michel Delanaye, Polynomial Reconstruction Finite Volume Schemes for the Compressible Euler and Navier–Stokes Equations on Unstructured Adaptative Grids. PhD thesis, Universite de Liege, Faculte des Sciences Appliquees, 1998.

[12] Michel Delanaye, Phillippe Geuzaine, J.A. Essers, Compressible flows on unstructured adaptive grids, in: Proceedings of the Thirteenth AIAA Computational Fluid Dynamics Conference, 1997.

[13] P. Geuzaine, M. Delanaye, J.-A. Essers, Computation of high Reynolds number flows with an implicit quadratic reconstruction scheme on unstructured grids, in: Proceedings of the Thirteenth AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, 1997, pp. 610–619.

[14] Andrew G. Godfrey, Curtis R. Mitchell, Robert W. Walters, Practical aspects of spatially high-order accurate methods, American Institute of Aeronautics and Astronautics Journal 31 (9) (1993) 1634–1642.

[15] Ami Harten, High-resolution schemes for hyperbolic conservation-laws, Journal of Computational Physics 49 (3) (1983) 357–393.

[16] H. Luo, J. Baum, R. Lohner, A fast matrix-free implicit method for compressible flows on unstructured grids, Journal of Computational Physics 146 (1998) 664–690.

[17] L.M. Manzano, J.V. Lassaline, P. Wong, D.W. Zingg, A Newton–Krylov Algorithm for the Euler Equations Using Unstructured Grids, AIAA Paper 2003-0274, in: 41th AIAA Aerospace Sciences Meeting and Exhibit, 2003.

[18] Dimitri J. Mavriplis, On convergence acceleration techniques for unstructured meshes, Technical Report ICASE No. 98-44, Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, NASA Langley Research Center, Hampton VA 23681-2199, 1998.

[19] Krzysztof Michalak Carl, Ollivier-Gooch, Matrix-explicit GMRES for a higher-order accurate inviscid compressible flow solver, in: Proceedings of the Eighteenth AIAA Computational Fluid Dynamics Conference, 2007.

[20] Amir Nejat, Carl Ollivier-Gooch, A high-order accurate unstructured GMRES algorithm for inviscid compressible flows, in: Proceedings of the Seventeenth AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, June 2005.

[21] Amir Nejat, Carl Ollivier-Gooch, A high-order accurate unstructured Newton–Krylov solver for inviscid compressible flows, in: 36th AIAA Fluid Dynamics Conference, 2006. AIAA 2006-3711.

[22] Amir Nejat, Carl Ollivier-Gooch, Effect of discretization order on preconditioning and convergence of a high-order unstructured Newton–GMRES solver for the Euler equations, Journal of Computational Physics, in press, doi:10.1016/j.jcp.2007.10.024.

[23] J. Nichols, D.W. Zingg, A three-dimensional multi-block Newton–Krylov flow solver for the Euler equations, in: Proceedings of the Seventeenth AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, 2005.

[24] Carl F. Ollivier-Gooch, High-order ENO schemes for unstructured meshes based on least-squares reconstruction, AIAA Paper 97-0540, January 1997.

[25] Carl F. Ollivier-Gooch, Quasi-ENO schemes for unstructured meshes based on unlimited data-dependent least-squares reconstruction, Journal of Computational Physics 133 (1) (1997) 6–17.

[26] Carl F. Ollivier-Gooch, Michael Van Altena, A high-order accurate unstructured mesh finite-volume scheme for the advection–diffusion equation, Journal of Computational Physics 181 (2) (2002) 729–752.

[27] P.D. Orkwis, Comparison of Newton's and Quasi-Newton's method solvers for the Navier–Stokes equations, American Institute of Aeronautics and Astronautics Journal 31 (1993) 832–836.

[28] A. Pueyo, D.W. Zingg, Efficient Newton–Krylov solver for aerodynamic computations, American Institute of Aeronautics and Astronautics Journal 36 (11) (1998) 1991–1997.

[29] P.L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (1981) 357–372.

[30] S.E. Rogers, D. Kwak, C. Kiris, Steady and unsteady solutions of the incompressible Navier–Stokes equations, American Institute of Aeronautics and Astronautics Journal 29 (4) (1991) 603–610.

[31] Youcef Saad, Martin H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal of Scientific and Statistical Computing 7 (3) (1986) 856–869.
[32] V. Venkatakrishnan, Newton solution of inviscid and viscous problems, American Institute of Aeronautics and Astronautics Journal 27 (7) (1989) 885–891.
[33] V. Venkatakrishnan, Convergence to steady-state solutions of the Euler equations on unstructured grids with limiters, Journal of Computational Physics 118 (1995) 120–130.
[34] V. Venkatakrishnan, T.J. Barth, Application of direct solvers to unstructured meshes for the Euler and Navier–Stokes equations using upwind schemes, in: Twenty-seventh Aerospace Sciences Meeting, 1989. AIAA Paper 89-0364.
[35] V. Venkatakrishnan, D. Mavriplis, Implicit solvers for unstructured meshes, Journal of Computational Physics 105 (1993) 83–91.
[36] D. Zingg, S. De Rango, M. Nemec, T. Pulliam, Comparison of several spatial discretizations for the Navier–Stokes equations, Journal of Computational Physics 160 (2000) 683–704.