Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 227 (2008) 2366–2386
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Effect of discretization order on preconditioning and convergence of a high-order unstructured Newton-GMRES solver for the Euler equations

Amir Nejat, Carl Ollivier-Gooch *

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

Received 27 February 2007; received in revised form 5 September 2007; accepted 19 October 2007
Available online 7 November 2007

Abstract

This article studies the effect of discretization order on preconditioning and convergence of a high-order Newton–Krylov unstructured flow solver. The generalized minimal residual (GMRES) algorithm is used for inexactly solving the linear system arising from implicit time discretization of the governing equations. A first-order Jacobian is used as the preconditioning matrix. The complete lower–upper factorization (LU) and an incomplete lower–upper factorization (ILU(4)) techniques are employed for preconditioning of the resultant linear system. The solver performance and the conditioning of the preconditioned linear system have been compared in detail for second, third, and fourth-order accuracy. The conditioning and eigenvalue spectrum of the preconditioned system are examined to investigate the quality of preconditioning.

Keywords: Preconditioning; Newton-GMRES; High-order discretization; Unstructured grids; Euler equations

1. Introduction

From the origins of computational fluid dynamics, the long-term goal of the discipline has been the accurate, efficient solution of problems of practical engineering interest. In recent years, techniques for unstructured meshes have simplified computations for complex real-world geometries, while high-order methods are emerging as a leading tool for computing accurate solutions. Much work remains, however, both to improve the robustness and efficiency of these schemes and to promote their wide-spread use.

Our recent work on high-order methods has led us to a reasonably complete understanding of the important issues that must be addressed to create a truly high-order finite-volume solver for unstructured meshes. The best understood part of this problem is reconstruction of the control volume averages to produce a high-order accurate approximation to the true solution. Numerous researchers, beginning with Barth and

* 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.2007.10.024

Frederickson [8], have explored this topic, both from the k-exact least-squares perspective [6,16,36,17,37] and from the (weighted) essentially non-oscillatory point of view [1,2,20,24]; we choose to follow the k-exact approach. While accurate reconstruction – by whatever means – is of course the foundation of a high-order finite-volume method, accurate flux integration, accurate boundary treatment and robust treatment of discontinuities are equally important, and receive little or no treatment in the literature. High-order accurate flux integration, for instance, requires Gauss quadrature along all control volume interfaces [35]. While "everyone knows" that curved boundaries must be used to obtain genuinely high-order solutions, there is essentially no discussion in the literature on what level of accuracy is required, or how to get it. We have shown [37,33,35] that the common knowledge is correct: the order of accuracy of the boundary shape must equal the order of accuracy of the solver. In addition, we have shown that Gauss point locations along the boundary must be spaced by arc length rather than being projections of Gauss points from a polygonal boundary representation onto the curved boundary [35]. Finally, treatment of shocks and other solution discontinuities requires particular attention for high-order schemes, both to eliminate overshoots at discontinuities and to avoid ruining accuracy in smooth regions of the flow while retaining good convergence properties. Here, we have experimented with two approaches based on Venkatakrishnan's limiter [44]: a smooth version of the approach of Delanaye and his co-workers [16,18,17], which reduces to a limited linear reconstruction near discontinuities, and a variant of our own that limits but does not eliminate all derivatives [27,28]; in this paper, we will use the former exclusively. We will describe our flow solver in more detail in Section 2.

The result that a high-order scheme gives a more accurate solution on a given mesh than a second-order scheme is no surprise, as the accuracy analysis guarantees that this will happen asymptotically. The question, historically, has always been whether a high-order scheme can be made to converge rapidly enough to be competitive with second-order schemes on a computational cost basis. While there are some results that suggest this is the case for structured meshes [41,21,46,14], the situation for unstructured meshes is less clear, because little work has been done on optimizing convergence rates for high-order unstructured mesh solvers. The main contribution of much of our recent research [30,31,29,33] has been to fill this gap by developing an efficient implicit time advance scheme for our high-order accurate solver. In doing so, we have applied and combined a number of existing techniques already used with success for other CFD problems, including using GMRES for linear system solution, ILU(p) preconditioning, and Newton's method for rapid convergence near steady state.

The use of GMRES [43] as a linear system solver in CFD is wide-spread. In particular, the matrix-free variants of GMRES (see [7,10,18] for early usage examples in CFD) are popular, because the matrix appears only in matrix–vector products, which can be replaced by Frechet derivatives. This approach reduces memory usage considerably and removes the problem of explicitly forming the high-order Jacobian matrix, greatly simplifying the overall implicit algorithm. However, the Jacobian matrix is typically ill-conditioned, so effective preconditioning is crucial for good convergence of GMRES and therefore of the outer, non-linear iterations. Preconditioning consists of two parts: computing a preconditioning matrix and the method used to apply the preconditioner. A good preconditioning matrix should be an easy-to-compute approximation to the global flux Jacobian and usually needs to be available explicitly. Common practice for second-order methods is to use the first-order Jacobian for preconditioning, sacrificing precise linearization in the preconditioner for cheaper computation [38,39,45,18,34]. For high-order schemes, the cost and complexity of computing the full high-order Jacobian matrix is even higher than for second order, and so we also choose a simplified first-order Jacobian for simplicity.

For effective preconditioning, in addition to applying a good preconditioner matrix, we need to employ a good preconditioning technique [42]. For structured meshes, Pueyo and Zingg [38,39] presented an efficient matrix-free Newton-GMRES solver for steady-state aerodynamic flow computations. They investigated the efficiency of different quasi-Newton methods, incomplete lower–upper factorization (ILU) preconditioning strategies, and reordering techniques for a variety of compressible inviscid, laminar and turbulent flows using a GMRES iterative solver. They showed that the approximate Newton method using matrix-free GMRES-ILU(2) with the first-order Jacobian preconditioner and reverse Cuthill–McKee (RCM) reordering produces the best overall efficiency for most cases. Later Nichols and Zingg [34] developed a 3D multi-block Newton–Krylov solver for the Euler equations using the same approach and showed that ILU(1) gives the best performance.

For unstructured meshes, Venkatakrishnan and Mavriplis [45] 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 that GMRES with ILU preconditioning had the best performance. In their case the graphs of the linearized Jacobian and the unstructured mesh were the same, as the Jacobian was approximated based on the direct neighbors, and ILU(0) gave satisfactory performance. Delanaye et al. [18] presented an ILU preconditioned matrix-free Newton-GMRES solver for the Euler and Navier–Stokes equations on unstructured adaptive grids using quadratic reconstruction. This study shows that ILU(0) preconditioning is sometimes insufficient for reaching full convergence of stiff problems when the Jacobian is high-order. By permitting more fill in the ILU decomposition (ILU(1)), full convergence was achieved. We also have used ILU to apply our preconditioner, and have found that high-order accurate schemes require a higher fill level for optimal convergence than second-order schemes, as we will discuss below.

This article will focus on the critical issue of how best to compute and apply a preconditioning matrix for a matrix-free Newton-GMRES scheme for the Euler equations with third- and fourth-order discretizations. We give an overview of our discretization scheme in Section 2. Our approach for achieving steady-state convergence is laid out in full detail in Section 3, including not only preconditioning-specific issues, but also a discussion of how we divide computational effort between a start-up phase and a full Newton phase. In Section 4, we present our numerical results, focusing on analysis of preconditioning and convergence. Both LU and ILU are employed in the Newton phase to study the effect of factorization accuracy on the quality of preconditioning and convergence rate. The restarted version of the GMRES algorithm is also used to explore the possibility of reducing the number of Newton (outer) iterations (especially for the fourth-order discretization). The conditioning and eigenvalue spectrum of the preconditioned operator are examined for subsonic and transonic cases. Finally, Section 5 summarizes the paper and presents our conclusions about how best to precondition a high-order accurate unstructured mesh discretization for the Euler equations.

2. Spatial discretization

Our discretization scheme is a cell-centered finite-volume scheme, using k-exact reconstruction to obtain a high-order (up to fourth-order) approximation to the solution and Gauss quadrature to integrate fluxes accurately. This section will provide an overview of the scheme; we refer interested readers to [33] for full details.

2.1. Governing equations

The finite-volume formulation of the unsteady 2D Euler equations for an arbitrary control volume can be written in the form of a volume and a surface integral

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

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

where U is the solution vector in conservative variables, and F is the flux vector. The total energy per unit volume, E, and the pressure, P, are related by the ideal gas equation of state

$$E = \frac{P}{\gamma - 1} + \frac{1}{2}\rho\left(u^2 + v^2\right) \tag{3}$$

2.2. High-order reconstruction procedure

Finite-volume schemes, by their nature, compute the average value $\overline{U}_i$ of the solution within each control volume. As these averages differ by $\mathcal{O}(h)$, where h is a characteristic control volume size, flux integrals com-

puted using the averages directly will be only first-order accurate. To obtain higher accuracy, we compute a k-exact piecewise polynomial representation $U^{(k)}_{R;i}(x,y)$ of the solution, as described first by Barth and Frederickson [8]. In two dimensions, such a reconstruction polynomial can be written as

$$U^{(k)}_{R;i}(x,y) = U_i + \left.\frac{\partial U}{\partial x}\right|_i \Delta x + \left.\frac{\partial U}{\partial y}\right|_i \Delta y + \left.\frac{\partial^2 U}{\partial x^2}\right|_i \frac{\Delta x^2}{2} + \left.\frac{\partial^2 U}{\partial x\,\partial y}\right|_i \Delta x \Delta y + \left.\frac{\partial^2 U}{\partial y^2}\right|_i \frac{\Delta y^2}{2} + \left.\frac{\partial^3 U}{\partial x^3}\right|_i \frac{\Delta x^3}{6} + \left.\frac{\partial^3 U}{\partial x^2\,\partial y}\right|_i \frac{\Delta x^2 \Delta y}{2} + \left.\frac{\partial^3 U}{\partial x\,\partial y^2}\right|_i \frac{\Delta x \Delta y^2}{2} + \left.\frac{\partial^3 U}{\partial y^3}\right|_i \frac{\Delta y^3}{6} + \cdots \tag{4}$$

up to the kth derivatives, where $\overrightarrow{\Delta x} \equiv \vec{x} - \vec{x_i}$ and $\vec{x_i}$ is the reference point for control volume i, which we take to be the triangle centroid for cell-centered meshes. The derivatives on the right-hand side are the unknowns we seek. To ensure conservation of the mean of each variable within each control volume, we require that

$$\frac{1}{A_i}\int_i U^{(k)}_{\mathrm{R}}(x,y)\,\mathrm{d}A = \overline{U}_i \tag{5}$$

In addition, for accuracy, we require that the reconstruction polynomial $U^{(k)}_{R;i}$ in control volume i predict well the mean value in a stencil of nearby control volumes j:

$$\overline{U}_j = \frac{1}{A_j}\int U^{(k)}_{R;i}(x,y)\,\mathrm{d}A_j \tag{6}$$

As shown in detail elsewhere [37], simplification of this integral leads to

$$\overline{U}_j = U|_i + \left.\frac{\partial U}{\partial x}\right|_i \hat{x}_{ij} + \left.\frac{\partial U}{\partial y}\right|_i \hat{y}_{ij} + \left.\frac{\partial^2 U}{\partial x^2}\right|_i \frac{\widehat{x^2}_{ij}}{2} + \left.\frac{\partial^2 U}{\partial x \partial y}\right|_i \widehat{xy}_{ij} + \left.\frac{\partial^2 U}{\partial y^2}\right|_i \frac{\widehat{y^2}_{ij}}{2} + \left.\frac{\partial^3 U}{\partial x^3}\right|_i \frac{\widehat{x^3}_{ij}}{6} + \left.\frac{\partial^3 U}{\partial x^2 \partial y}\right|_i \frac{\widehat{x^2 y}_{ij}}{2} + \left.\frac{\partial^3 U}{\partial x \partial y^2}\right|_i \frac{\widehat{xy^2}_{ij}}{2} + \left.\frac{\partial^3 U}{\partial y^3}\right|_i \frac{\widehat{y^3}_{ij}}{6} + \cdots \tag{7}$$

where the geometric terms are moments of control volume j about the reference point $\vec{x_i}$ of control volume i:

$$\widehat{x^n y^m}_{ij} \equiv \frac{1}{A_j}\int_{V_j} ((x - x_j) + (x_j - x_i))^n \cdot ((y - y_j) + (y_j - y_i))^m \,\mathrm{d}A = \sum_{l=0}^{m}\sum_{k=0}^{n} \frac{n!(x_j - x_i)^k}{k!(n-k)!} \cdot \frac{m!(y_j - y_i)^l}{l!(m-l)!} \cdot \overline{x^{n-k} y^{m-l}}_j$$

We write Eq. (7) for all control volumes j in the stencil, which, at a minimum, must include as many control volumes as the number of derivatives we seek in $U_{R;i}$. In general, this requires $\frac{(k+1)(k+2)}{2}$ control volumes; that is, control volume i plus 2, 5, and 9 neighbors for second-, third-, and fourth-order accuracy, respectively. We choose to use about a 50% excess of control volumes so that resulting least-squares solution can filter out noise in the discrete solution (3, 9, and 15 neighbors). Stencils are constructed – both in the interior and near boundaries – by adding layers of face neighbors until the required number of total neighbors is reached; Fig. 1 illustrates several layers of neighbors. The mean constraint, Eq. (5) can be eliminated analytically, leaving an unconstrained least-squares problem, which we solve by QR factorization (Golub and Van Loan, 1983 [22]).

2.3. Residual calculation

The high-order accurate least-squares reconstruction scheme of Section 2.2 enables us to compute all the flow variables in the interior and at the boundaries up to fourth-order accuracy. Fluxes at control volume boundaries are computed by Roe's flux differencing [40]

$$F(U_{\mathrm{L}}, U_{\mathrm{R}}) = \frac{1}{2}[F(U_{\mathrm{L}}) + F(U_{\mathrm{R}})] - \frac{1}{2}\left|\widetilde{A}\right|(U_{\mathrm{R}} - U_{\mathrm{L}}) \tag{8}$$

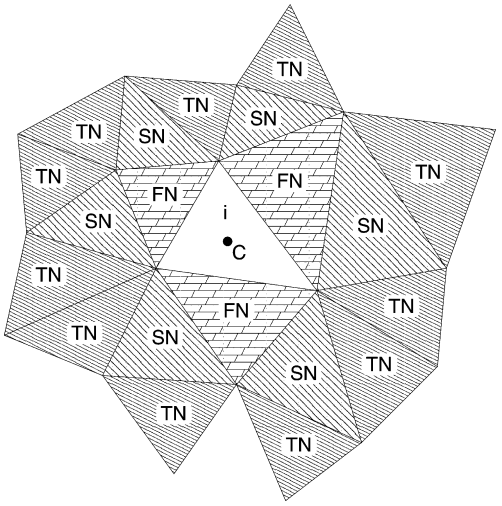


Fig. 1. A typical cell center control volume and its reconstruction stencil, including three layers of neighbors. A second-order reconstruction would use the first neighbors (FN); for third-order, the second neighbors (SN) would be added; and for fourth-order, the third neighbors (TN) are also required.

where U_L and U_R are computed from the reconstruction polynomials 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 in diagonalized form in practice as

$$\left|\widetilde{A}\right| = \widetilde{X}^{-1}\left|\widetilde{\Lambda}\right|\widetilde{X}, \qquad \left|\widetilde{\Lambda}\right| = \mathrm{Diag}(f(\widetilde{\lambda}_i)) \tag{9}$$

where $\widetilde{X}$ are the right eigenvectors and Λ are the eigenvalues of the Jacobian matrix, and $f(\widetilde{\lambda}_i)$ incorporates Harten's entropy fix [23]. Having computed the fluxes, we use Gauss quadrature to integrate the fluxes to the same order of accuracy as the reconstruction; this implies the use of two Gauss points for third- and fourth-order accuracy.

2.4. Monotonicity enforcement

Enforcing monotonicity is an important issue both for the second-order and high-order upwind schemes. Limiters are often needed to suppress oscillations around discontinuities and to avoid reconstructing a non-physical solution (such as a negative density) at Gauss points located close to such locations. However, limiters cause two major problems. First, they hamper convergence, as small changes in the solution near a shock may cause disproportionate changes in limiter behavior. This is especially true for non-differentiable limiters such as the Barth–Jespersen limiter [9], but even using a differentiable limiter does not guarantee good convergence behavior. Second, limiters change the reconstruction polynomial by reducing the reconstructed solution derivatives, affecting the accuracy of the reconstructed solution. In particular, if the limiter value differs from 1 by more than $\mathcal{O}(h^k)$ in smooth regions, the solution cannot be kth-order accurate. High-order methods are particularly sensitive to problems with limiters, especially with regard to accuracy.

We choose to use Venkatakrishnan's limiter [44]. This limiter is only semi-differentiable (it is not fully differentiable when the control volume with the maximum value changes, for instance), but this is enough for excellent convergence behavior. To address accuracy issues, we apply the limiter selectively: our experience as well as other research [16,18,17] shows that applying the limiter to all derivatives yields a more diffusive solution. Therefore, we employ a differentiable switch σ to drop the non-linear terms in the reconstruction when the limiter ϕ differs significantly from one:

$$U_G^{(k)}(x_G, y_G) = \overline{U}_i + [(1-\sigma)\phi_i + \sigma]\{\text{Linear part}\} + \sigma\{\text{Higher-order part}\} \tag{10}$$

In smooth regions, the full high-order reconstruction is applied by choosing $\sigma = 1$. Near discontinuities we choose $\sigma \approx 0$ to recover the good behavior of the limited linear reconstruction. Because using a switch to set σ to either zero or one stalls the convergence, we define σ as a smooth function of ϕ, such that σ is nearly one for the regions that $\phi \geqslant \phi_0$ and it quickly goes toward zero for other values of ϕ [31]

$$\sigma = \frac{1 - \tanh(S(\phi_0 - \phi))}{2} \tag{11}$$

S in Eq. (11) determines the sharpness of the transition function, and ϕ_0 defines the limiter value that activates the switch. We have found that choosing $\phi_0 = 0.8$ and $S = 20$ provides a reasonable switch function whose good behavior is relatively case independent.

2.5. Boundary conditions

For a scheme to be fully high-order, boundary condition enforcement must also be high-order. We apply boundary conditions in two ways: as constraints on the reconstruction or by applying special flux formulae at boundaries. When applying boundary constraints, we require that the reconstruction polynomial satisfy the boundary conditions at boundary Gauss points; these constraints, which are written as Taylor series, are additional constraints on the least-squares problem. When boundary constraints are used, the analytic Euler flux (Eq. (2)) is then applied at boundary Gauss points. At the far field, we apply characteristic boundary conditions to determine data for computing boundary fluxes, with interior data computed from the high-order accurate reconstruction [33].

3. Convergence to steady state

Assuming the control volumes do not change with time, we can re-write Eq. (1) as a time evolution equation for the control volume averages:

$$\frac{\mathrm{d}\overline{U}_i}{\mathrm{d}t} = -\frac{1}{A_{CV_i}} \oint_{CS_i} F \mathrm{d}A \equiv -R_i(\overline{U}) \tag{12}$$

The right-hand side of Eq. (12) is called the flux integral or residual of control volume i, which is a non-linear function of the solution. We arrive at an implicit time advance formula by applying backward Euler time differencing:

$$\left(\frac{I}{\Delta t} + \frac{\partial R}{\partial \overline{U}}\bigg|^n\right) \delta U_i = -R_i^n(\overline{U}), \quad \delta U_i = \overline{U}_i^{n+1} - \overline{U}_i^n \tag{13}$$

where $\frac{\partial R}{\partial \overline{U}}$ is the Jacobian matrix resulting from residual linearization. Eq. (13) is a large linear system of equations which must be solved at each time step to obtain an update for the vector of unknowns. We use a preconditioned matrix-free GMRES solver for this linear system. Section 3.1 describes some fine points of implementation of matrix-free GMRES for high-order methods, while the application and assembly of a suitable preconditioning matrix are discussed in Sections 3.2 and 3.3, respectively.

If we take an infinite time step, Eq. (13) reduces to Newton iteration

$$\frac{\partial R}{\partial U} \delta U_i = -R_i^n(\overline{U}) \tag{14}$$

Newton's method generally converges quadratically in the vicinity of the solution. Overall solver robustness depends critically on finding an approximate solution near enough to the steady-state solution that Newton iteration will succeed, while efficiency considerations dictate that the start-up procedure by which we compute such an approximate solution be as cheap as possible. Section 3.4 discusses our start-up process, including criteria for switching to the Newton phase, while Section 3.5 describes our Newton solver.

3.1. Linear system solver

We solve the linear systems arising from Eqs. (13) and (14) by using the generalized minimal residual (GMRES) algorithm [43], which was designed for non-symmetric systems. GMRES minimizes the L_2-norm of the residual of the linear system, implying that if the linearization of the non-linear system is accurate then GMRES provides the best update for the solution at each iteration. The linear system arising from a high-order discretization has four to five times as many non-zero entries as a second-order scheme. Because of the size of these matrices and the difficulty in computing their entries analytically (even for second-order), we use a matrix-free implementation of GMRES (which to our knowledge was first applied to CFD problems by Johan et al. [25]) when using a Jacobian for a residual that is more than first-order accurate. In this approach, matrix–vector products are approximated by a directional derivative formula

$$\frac{\partial R}{\partial U}\cdot z \approx \frac{R(U+\varepsilon z)-R(U)}{\varepsilon}, \quad \varepsilon = \frac{\varepsilon_0}{\|z\|_2} \tag{15}$$

ε_0 is a very small number typically equal to the square root of machine precision. However, for fine meshes where the mesh length scale is very small, we need to use a larger ε_0 to accurately account for perturbation of the fourth-order terms due to round-off error considerations. Consequently, we choose $\varepsilon_0 = 10^{-6}$ instead of the more common 10^{-8}.

3.2. Preconditioning

Convergence of iterative techniques, including Krylov subspace methods, is highly dependent on the conditioning of the linear system, *i.e.*, the Jacobian matrix. Using a high-order discretization introduces more off-diagonal entries and increases the bandwidth of the Jacobian matrix considerably. In addition, in the case of Euler equations (compressible flow), with a non-linear flux function and possible discontinuities in the solution, the Jacobian matrix is off-diagonally dominant. All these factors lead to poor convergence of the linear solver, and consequent slowing or stalling of convergence of the non-linear problem. Improving the spectral properties and eigenvalue clustering of the linear system by preconditioning improves the convergence characteristics of GMRES [11], although GMRES is not as sensitive to the eigenvalue spectrum of the linear system as symmetric iterative solvers such as the conjugate gradient (CG) method. We apply right preconditioning to leave the right-hand side of the linear system intact:

$$AM^{-1}\overbrace{(Mx)}^{z} = b, \quad x = M^{-1}z \tag{16}$$

where M is a non-singular matrix which approximates A in the linear system $Ax = b$. Finding the optimal preconditioner matrix is something of an art, since it is dependent both on the problem and on how the preconditioner is applied. In general, three factors are considered in choosing a preconditioner:

1. M should be a reasonably good approximation to the coefficient matrix A.
2. M should be better conditioned, more narrowly banded, and less expensive to build than A.
3. The system $Mx = z$ should be much easier to solve than $Ax = b$.

The bottom line is that the cost of constructing and applying the preconditioner should be relatively small compared to the cost of solving the original linear system.

We solve the linear system $Mx = z$ approximately by factoring M. Complete factorization of the matrix M into two triangular matrices,

$$M = LU$$

where L is a lower triangular matrix, and U is an upper triangular matrix, gives an exact solution of $Mx = z$, but these factored matrices are far less sparse than the original matrix M, with concomitant increases in memory and CPU usage for the factorization. Instead of full factorization, therefore, we apply an incomplete upper–lower (ILU) factorization, with the non-zero pattern for the factors chosen in advance [42]. The amount of fill allowed is specified by a fill level p, written as ILU(p). In ILU(0), the factorized matrix and

the original, non-factored matrix have the same graph or non-zero element locations. Choosing $p > 0$ allows some additional fill-in in the factorized matrix improving the accuracy of factorization and the preconditioning quality. However, increasing the fill-level comes at the expense of memory usage and extra computing cost, imposing a restriction in increasing fill-level in practice.

Several researchers have implemented ILU methods for preconditioning of the GMRES linear solver for compressible fluid flows [45,10,18,12,26]. As discussed in Section 1, their results show that ILU(p) with some additional fill-in (*e.g.*, $p = 1$ or 2) is a reliable and robust preconditioning strategy for a variety of test cases for second-order unstructured mesh schemes, while ILU(0) fails to provide fast convergence in some cases.

Another variant in the ILU family is ILU(p, τ), where in addition to the static non-zero pattern, all fill-in entries smaller than a tolerance τ are set to zero [11]. Since the proper fill-level and tolerance criterion in ILU(p, τ) for efficient preconditioning of the compressible flows are highly dependent on the test case, we restrict ourselves to using ILU(p) and do not consider ILU(p, τ) in this research.

Reordering is another important factor in ILU factorization. Reordering is designed to reduce the bandwidth of a matrix and consequently the number of fill-in entries that appear during factorization. Knowing that increasing the fill level in ILU preconditioning has its own disadvantages, reordering the original preconditioner matrix becomes essential to keep the accuracy of preconditioning for a low fill-level factorization. Our experience shows that quite often a non-reordered preconditioner matrix with low fill-level works poorly, while the same fill-level factorization performs perfectly well when the original matrix is reordered. Like many other researchers, we use the reverse Cuthill–McKee (RCM) [13] reordering technique.

3.3. Preconditioner matrix

For reasons of memory and computation time, 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 depends both on the problem and on how the preconditioner is applied. Because a low-order Jacobian matrix captures the essential physical information about the flow (though not all the details), it is a reasonable choice for preconditioning the linear system arising from the high-order discretization. The first-order Jacobian is narrowly banded after reordering and is much better conditioned than the high-order Jacobian. Therefore, constructing and applying the preconditioner will require significantly less effort than 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. As we shall see, the former is cheaper to compute, while the latter is a more effective preconditioner, presumably because it is a more accurate linearization.

3.3.1. 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. 2. 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:

$$R_i = \sum_{m=1,2,3} (F \cdot \hat{n}\,\mathrm{d}s)_m = F(U_i, U_{N_1}) \cdot \hat{n}_1 l_1 + F(U_i, U_{N_2}) \cdot \hat{n}_2 l_2 + F(U_i, U_{N_3}) \cdot \hat{n}_3 l_3 \tag{17}$$

where $\hat{n}_m$ and l_m are the outward unit normal and the length of face m of 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:

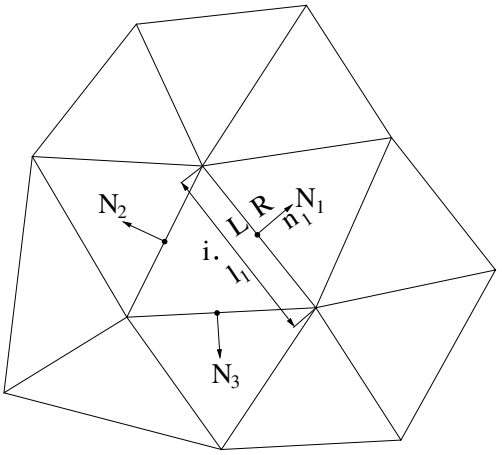


Fig. 2. Schematic of direct neighbors.

$$\frac{\partial R_i}{\partial U_{N_1}} = \frac{\partial F(U_i, U_{N_1})}{\partial U_{N_1}} \hat{n}_1 l_1 \tag{18}$$

$$\frac{\partial R_i}{\partial U_{N_2}} = \frac{\partial F(U_i, U_{N_2})}{\partial U_{N_2}} \hat{n}_2 l_2 \tag{19}$$

$$\frac{\partial R_i}{\partial U_{N_3}} = \frac{\partial F(U_i, U_{N_3})}{\partial U_{N_3}} \hat{n}_3 l_3 \tag{20}$$

$$\frac{\partial R_i}{\partial U_i} = \frac{\partial F(U_i, U_{N_1})}{\partial U_i} \hat{n}_1 l_1 + \frac{\partial F(U_i, U_{N_2})}{\partial U_i} \hat{n}_2 l_2 + \frac{\partial F(U_i, U_{N_3})}{U_i} \hat{n}_3 l_3 \tag{21}$$

Since both the flux F and solution U are 4-component vectors, each entry in the Jacobian matrix $\frac{\partial R}{\partial U}$ 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

$$\frac{\partial\left(\left|\widetilde{A}\right|\Delta U\right)}{\partial U_j} = \overbrace{\frac{\partial\left|\widetilde{A}\right|}{\partial U_j}\Delta U}^{1} + \overbrace{\left|\widetilde{A}\right|\frac{\partial(\Delta U)}{\partial U_j}}^{2} \tag{22}$$

where the index j represents either control volume i or its neighbor.

Differentiation of $\frac{\partial|A|}{\partial U}$ produces third-rank tensors which not only are difficult to derive but also are quite expensive to compute. Barth [5] has found the full derivative of $\frac{\partial(|\widetilde{A}|\Delta U)}{\partial U}$ with some clever modifications to eliminate the tensor computations, reducing the complexity of the Jacobian computation to some degree. Through spectral radius analysis for 1-D flow he showed that for a smooth flow the approximate Jacobian is reasonably accurate up to CFL = 1000 or even above. However, for the shock tube problem the difference between the true Jacobian and the approximate Jacobian grows after CFL = 10, and becomes noticeable after CFL=100, showing that the approximate Jacobian will not be accurate enough for larger CFL numbers. This result is consistent with what we would expect from inspection of Eq. (22). For a smooth flow, ΔU – the difference in the two reconstructed solutions at a Gauss point – is on the order of truncation error, so $\frac{\partial\widetilde{|A|}}{\partial U}\Delta U$ is very small compared to $|\widetilde{A}|\frac{\partial(\Delta U)}{\partial U}$, and the resulting approximate Jacobian will be acceptable. Near a discontinuity, however, this approximation is not accurate anymore, because $\frac{\partial\widetilde{|A|}}{\partial U}$ and ΔU will be O(1).

Even though ignoring changes in $\widetilde{A}$ (treating it as a constant) introduces an error in the Jacobian, it is still accurate enough to serve as a preconditioning matrix or for the start-up linearization, greatly reducing the overall Jacobian computation cost for these purposes.This simplification of the Jacobian of Roe's flux can be written in the following form for cells i and N_1:

$$\frac{\partial F(U_i, U_{N_1})}{\partial U_{N_1}} = \frac{1}{2}\left[\frac{\partial F(U_{N_1})}{\partial U_{N_1}} - \left|\widetilde{A}\right|\right] \tag{23}$$

$$\frac{\partial F(U_i, U_{N_1})}{\partial U_i} = \frac{1}{2}\left[\frac{\partial F(U_i)}{\partial U_i} + \left|\widetilde{A}\right|\right] \tag{24}$$

The other Jacobian terms in Eq. (19)–(21) 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 in each control volume at the Gauss point produces a more effective preconditioning matrix; in this paper, we use linear reconstruction data in computing the Jacobian. Because the reconstruction has already been computed, the additional cost of using this more accurate data is negligible.

Also, 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 of the cost of a second-order residual evaluation; reconstruction and limiting costs are not included here, as the limited reconstruction from the residual evaluation is re-used.

3.3.2. 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. Again, 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.

3.4. Start-up phase

Finding a good initial guess or reasonable approximate solution requires understanding of the physics of the problem; in addition to using analytic approximate solutions where possible, there are various numerical techniques for rapidly computing a good initial solution, including mesh sequencing, multigrid, and mixed explicit/implicit iterations. Our start-up process is based on an implicit defect correction procedure [31], in which the linearization is based on the inexpensive first-order discretization and the flux calculation remains high-order:

$$\left(\frac{I}{\Delta t} + \left.\frac{\partial R}{\partial U}\right|_{1st}^{n}\right)\Delta U^{n+1} = -R_{\text{High}}(U^n) \tag{25}$$

With this approach, high-order Jacobian computation – which is very expensive and not sufficiently accurate to allow large time steps and solution updates at this early stage – is avoided. Furthermore, the resultant linear system is easy to solve using (matrix-explicit) GMRES, because the left-hand side is constructed based on the first-order discretization and can be preconditioned very effectively by an ILU(1) factorization of itself. Since we are using an approximate linearization, solving the linear system exactly is pointless, and so the linear system in these defect correction *pre-iterations* is solved approximately with the tolerance of $5 \times 10^{-2} \cdot \|\text{Res}(U)\|_2$. Pre-iterations are performed until we reach a good initial state before switching to Newton's method; how good is good enough is problem dependent, and will be discussed further in Section 4.

3.5. Newton phase

By this point in convergence, most of the transients in the non-linear CFD problem have been removed from the flow field, and major steady features of the solution have appeared. Consequently, the linearization of the non-linear residual is accurate enough that the use of Newton iterations is effective in achieving quadratic, or at least super-linear, convergence

$$\left.\frac{\partial R}{\partial U}\right|_{\text{High}}^{n} \Delta U^{n+1} = -R_{\text{High}}(U^n) \tag{26}$$

For sufficiently accurate linearization, the high-order Jacobian must be applied during the Newton iterations. As discussed earlier, we use matrix-free GMRES to avoid the necessity of forming the high-order Jacobian explicitly. Note, however, that the matrix-free Jacobian is not exact because of both truncation error in the directional derivatives and – especially for high-order – round-off error. These two factors imply that achieving true quadratic convergence may be impossible in practice. The linear system in Eq. (26) is right preconditioned by the first-order Jacobian. ILU($p = 2$–4) is used for preconditioning; normally for high-order computation (especially for fourth-order and transonic flow), increasing the fill-level is highly beneficial [32]. A fixed number of search directions is employed ($K = 30$); limiting the subspace size is important, since matrix-free GMRES evaluates the non-linear residual once per search direction, which can become very expensive for high-order residual computation. The linear system is again solved approximately but this time with a tighter tolerance ($10^{-2}\|\text{Res}(U)\|_2$), as an accurate update is required. No restart is allowed, and if the tolerance is not reached, the next outer iteration starts with the best update from the last inner GMRES iteration. Approximately solving the linear system in this way is called the *inexact Newton method* [19], and although it typically increases the number of non-linear outer iterations, considerable computation time is saved overall by not solving the linear system exactly [39,12,26,31].

4. Results

This section focuses on the effect of variants in preconditioning on convergence of our Newton-GMRES solver, and on the effect of discretization order on preconditioning quality and convergence rate. We will examine two test cases in detail: a subsonic case (smooth flow) and a transonic case (flow with discontinuity).

4.1. Subsonic flow over NACA 0012, $M = 0.63$, $\alpha = 2°$

The subsonic flow for a NACA 0012 airfoil using an unstructured mesh with 9931 control volumes is computed at $M = 0.63$, $\alpha = 2°$ for all discretization orders. The far field is located at 25 chords and characteristic boundary conditions are implemented implicitly. The initial condition is the free stream flow. The tolerance in solving the linear system is 5×10^{-2} of the L_2 norm of the non-linear residual for the start-up phase and 1×10^{-2} for the Newton phase. These criteria are often not reached within the allowable number of inner iterations. For all discretization orders, a subspace size of 30 has been set and no restart is allowed for either the pre-iterations or the inexact Newton iterations. The preconditioning for the pre-iterations is performed by employing the approximate analytical Jacobian matrix with ILU(1) factorization; for the Newton iterations, the first-order finite-difference Jacobian matrix is used. The convergence criterion for the steady-state solution is 1×10^{-12} for the L_2 norm of the non-linear residual.

A closeup of the mesh used for this case is shown in Fig. 3. On this mesh, the results for all orders of accuracy are indistinguishable from each other to plotting accuracy. The computed lift coefficient for all orders fall in the range 0.3250 ± 0.0003, while the drag coefficient is 3–4 counts. We have shown elsewhere [33], in a thorough analysis of the accuracy of our scheme for this case, that the computed drag coefficient is a strong function and lift coefficient a weak function of far field distance; use of an improved far field boundary conditions would also improve these results, but is beyond our present scope. De Zeeuw and Powell [15] report comparable results, with a lift coefficient of 0.3289 and drag coefficient of 0.0004 for a second-order computation, on a locally refined Cartesian mesh of 10694 control volumes with a far field distance of 128 chords.

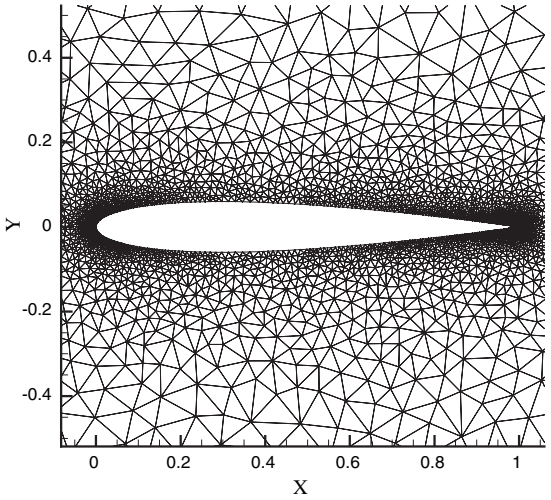


Fig. 3. Mesh for subsonic NACA 0012 case.

Our 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 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 to reach a good initial solution before switching to Newton iterations. As shown in Table 1, the CFL number starts at 2 and is increased gradually 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 using defect correction with a constant CFL of 100, where the first-order Jacobian is used both for constructing the Jacobian and for preconditioning the linear system. The residual is evaluated to second-order accuracy for all cases. 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.

After start-up, the solution process is switched to Newton iteration and an infinite CFL is employed. For the Newton phase three different strategies are applied:

1. Inexact Newton iteration with ILU(4) preconditioning.
2. Inexact Newton iteration with LU preconditioning.
3. Exact Newton iteration with ILU(4) preconditioning; up to 10 restarts are allowed for reaching machine accuracy in the linear solver.

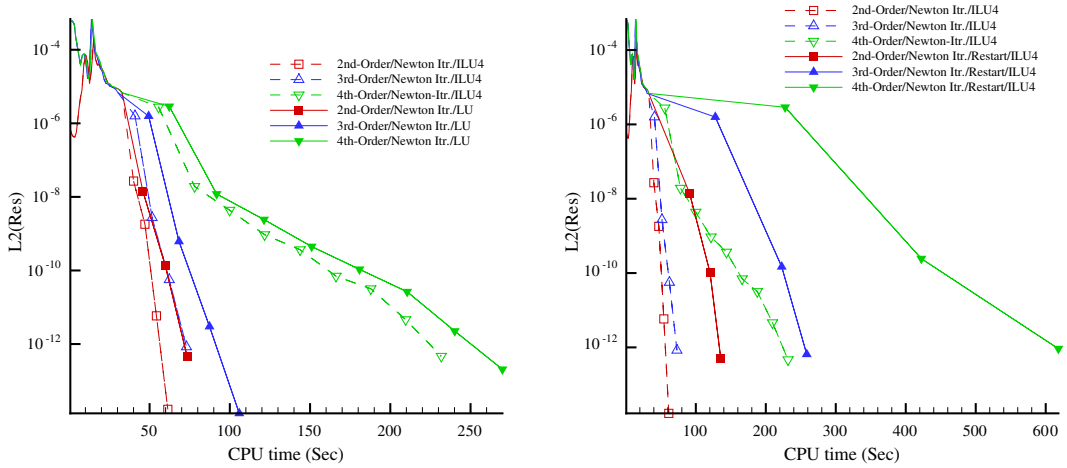
To allow comparison of computing cost between orders of accuracy, we normalize CPU time by a work unit, defined as the cost of one residual evaluation for the corresponding order of accuracy for a specific mesh. Table 2 shows the convergence summary for all orders of spatial discretization accuracy in terms of total number of residual evaluations, total CPU time, total work units, number of Newton iterations, and cost of the Newton phase in terms of work units. Also, the convergence history in terms of CPU time is shown in Fig. 4. For all discretization orders, after 30 pre-iterations, the solution has converged after a few Newton iterations. Full convergence is achieved for all orders of accuracy, but the CPU time for the fourth-order case is

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

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

Order	Resid. Eval.	Time (s)	Work units	Newton iterations	Newton phase work units
ILU(4)/Inexact Newton					
2nd	158	60.4	399.9	4	182.8
3rd	158	72.9	271.3	4	158.7
4th	318	231.8	377.5	9	324.2
LU/Inexact Newton					
2nd	105	73.4	476.5	3	260
3rd	149	105.9	386.4	4	264.3
4th	285	269.9	460.5	8	404.0
ILU(4)/Exact Newton					
2nd	574	135.7	875.4	3	659.6
3rd	766	258.5	953.8	3	831.2
4th	963	618.1	1060.2	3	1003.6



(a) ILU4 versus LU preconditioning

(b) Inexact versus Exact Newton/ILU4 preconditioning

Fig. 4. Convergence history, NACA 0012 (9931 CVs), $M = 0.63$, $\alpha = 2°$.

much larger than the other two orders of accuracy. Part of this effect is expected, since the fourth-order discretization requires more operations due to the rising cost of the reconstruction with increasing discretization order. However, a considerable part of the computing cost is due to a noticeable increase in the number of outer iterations. The linear system arising from the fourth-order discretization is more poorly conditioned than for lower orders, even with preconditioning, making it more difficult to compute an accurate update to the non-linear problem. On the other hand, both the second- and third-order cases quickly converge, demonstrating the effectiveness of the preconditioning for these cases.

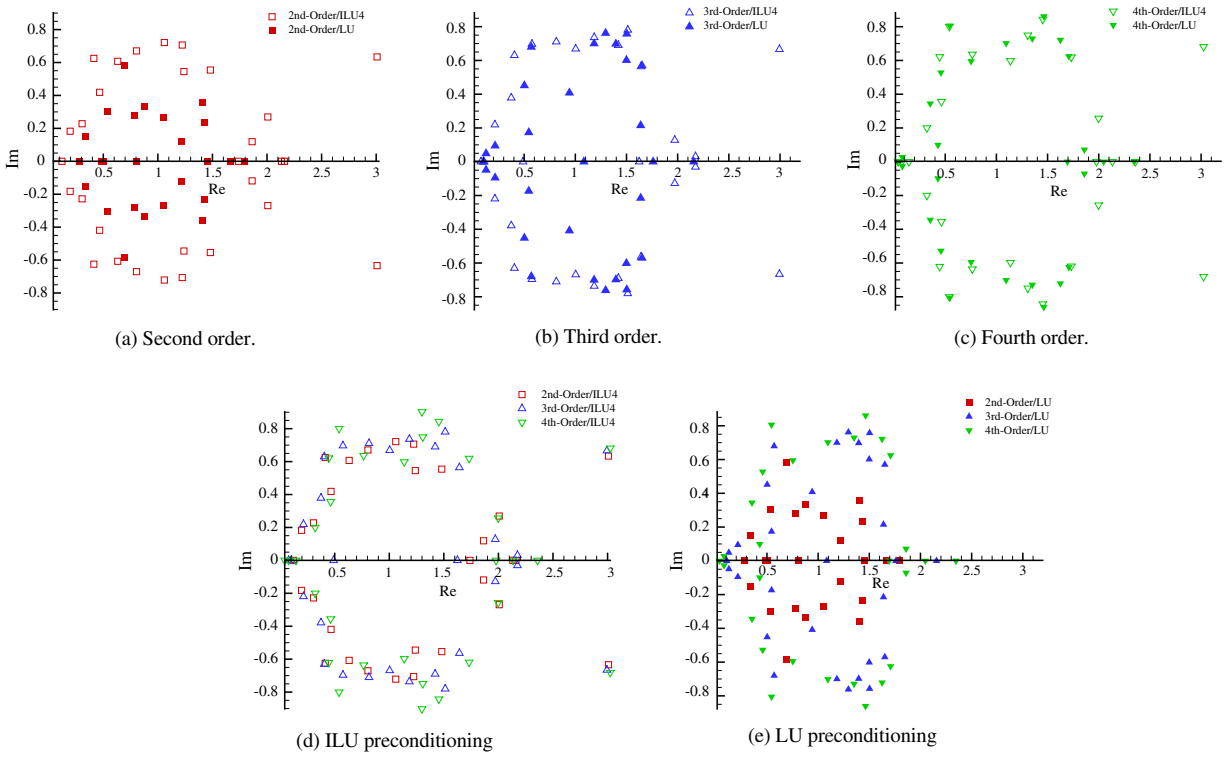
LU factorization provides the best possible application of the preconditioner matrix. As such, it is not surprising that LU improves convergence as measured by non-linear iteration count. However, the difference from ILU(4) is slight, suggesting that ILU(4) is sufficient to exploit all the information in the preconditioner matrix considering the fixed subspace size and tolerance in the linear solver. In terms of both CPU time and memory requirements, ILU(4) is definitely superior to LU.

For the exact Newton case where the linear system in each Newton outer iteration is solved to machine accuracy using multiple restarts, super-linear convergence is achieved for all discretization orders. However, due to multiple restarts the total number of non-linear residual evaluations increases dramatically, resulting in increased solution time.

Perfect preconditioning would cluster all eigenvalues at one. For the proposed preconditioning strategy, it is impossible to exactly cluster all eigenvalues at one for two reasons. First, the preconditioning matrix – the first-order Jacobian – is only an approximation of the true high-order Jacobian. Second, incomplete factorization is used as the preconditioning technique instead of full factorization. Therefore, the best that can be expected is that all eigenvalues of the preconditioned operator will be clustered near unity. As a result, the distance of the eigenvalues from unity can be used as one of the preconditioning quality indicators. Also, it is desired to have eigenvalues located far from the origin to avoid ill-conditioning and singularity issues. To evaluate and compare the quality of the preconditioning for different discretization orders, the approximate eigenvalue spectrum of the preconditioned linear system is computed as a byproduct of the Arnoldi process that builds the Krylov subspace [4]. While the largest eigenvalues are likely to be reasonably accurate, the smallest eigenvalues are much less reliable. Notwithstanding, all reported eigenvalues are plotted in Fig. 5 at the last iteration (*i.e.*, the converged solution) for both the LU and ILU(4) factorizations.

The eigenvalues associated with high-order discretizations are scattered with larger distances from one compared to the second-order eigenvalues. This clearly indicates a reduction in the quality of preconditioning with increasing discretization order, which is consistent with the convergence results. Also, the smallest eigenvalues approach the origin as the discretization order increases, shifting the matrix toward singularity. The fourth-order discretization in particular has one eigenvalue very close to the origin. Based on the plotted eigenvalue patterns, the LU factorization provides better preconditioning compared with the ILU(4) factorization, which is also consistent with the total number of residual evaluations shown in Table 2.

Condition numbers for the linear systems solved during Newton iterations are also estimated during the Arnoldi process and are shown in Fig. 6. The condition number is shown as a function of drop in residual. The $\|\mathrm{Res}_0\|_2$, or reference residual, is the L_2-norm of the initial non-linear residual computed based on the far field flow condition. Therefore the ratio of the non-linear residual at the end of each Newton iteration to $\|\mathrm{Res}_0\|_2$, reflects the relative convergence after each Newton iteration. The first iteration condition number shows the conditioning of the linear system formed based on the solution linearization at the end of the



(a) Second order. (b) Third order. (c) Fourth order.

(d) ILU preconditioning (e) LU preconditioning

Fig. 5. Eigenvalue pattern for the preconditioned system (converged solution), NACA 0012 (9931 CVs), $M = 0.63$, $\alpha = 2°$.

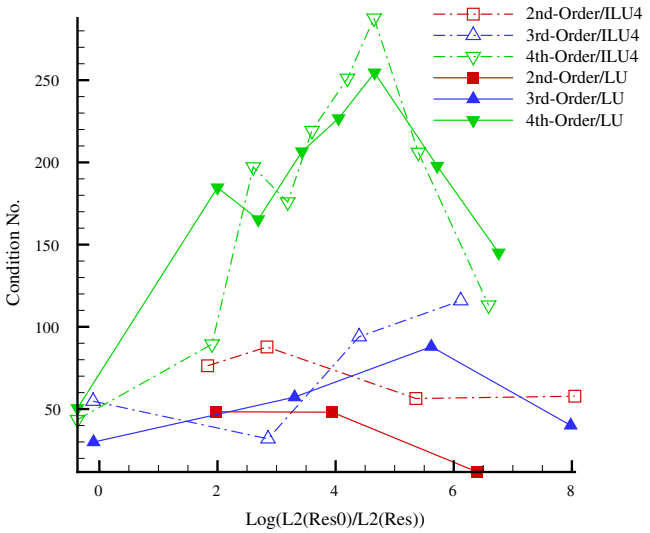


Fig. 6. Condition number of the preconditioned system (Newton iterations), NACA 0012 (9931 CVs), $M = 0.63$, $\alpha = 2°$.

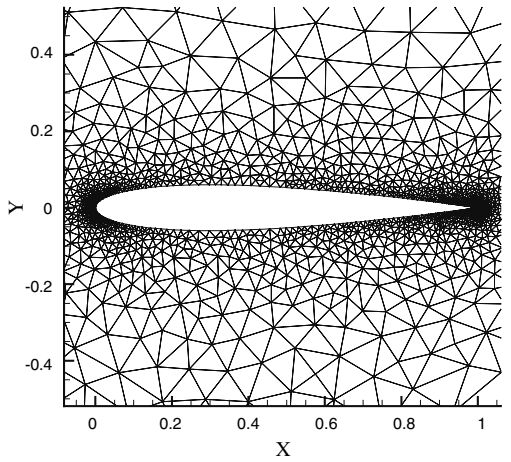
start-up phase. The rest of the reported condition numbers are associated with linear systems formed based on the solution at the end of successive Newton iterations. While the second-order discretization initially has a larger condition number that the third-order discretization, the condition number decreases gradually with convergence; note also that the reason the second-order results start at a two order of magnitude residual drop is that the pre-iterations are more successful in reducing the non-linear residual for this case than the others. In high-order cases, initially, the condition number is smaller compared to the second-order case but starts to grow as Newton iteration proceeds.

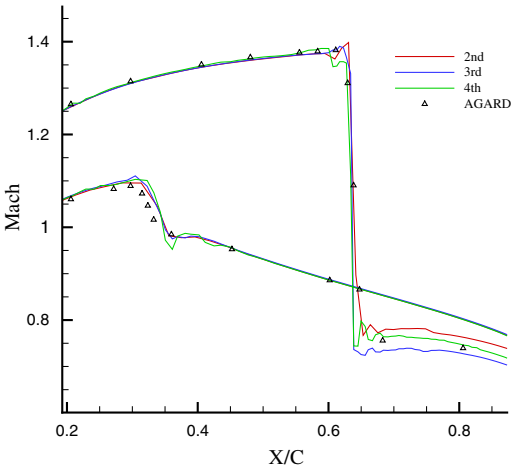
4.2. Transonic flow over NACA 0012, $M = 0.85$, $\alpha = 1°$

For transonic flow, in general, it is more difficult to get fast convergence than for subsonic flow. This is because of the mixed subsonic/supersonic nature of the flow and the existence of discontinuities in the solution. The methodology for handling discontinuities can increase the complexity of the problem, especially for implicit schemes, where the update in each iteration can be large and 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 solution convergence. All these factors make efficient solution of transonic flows more difficult.

The transonic flow around a NACA 0012 airfoil at $M = 0.85$, $\alpha = 1.0°$ is studied; this is the classic AGARD test case 2 [3]. An unstructured mesh with 5354 control volumes is employed for this test case; the far field boundary is 25 chords away. The flow is computed for all orders of accuracy using the Venkatakrishnan limiter with proper high-order modification, as described in Section 2.4. The limiter values are allowed to change through all iterations and no freezing is considered. The tolerance of solving the linear system, like the previous test case, is 5×10^{-2} of the L_2 norm of the non-linear residual for the start-up phase, and 1×10^{-2} for the Newton phase. For all test cases a subspace size of 30 has been set and no restart is allowed. For fourth-order, the subspace in the Newton phase has been reduced to 20 to reduce the cost of solving the linear system, but after reaching a non-linear residual of 1×10^{-9} a subspace of 30 is employed again to improve accuracy in the linear solver. Preconditioning is performed using the approximate analytical Jacobian matrix with ILU(1) factorization for the start-up pre-iterations and the finite-difference Jacobian matrix with ILU(4) factorization for the Newton iterations. The initial condition is free stream flow. The global convergence tolerance is an L_2 norm of the non-linear density residual of 1×10^{-12}.

Fig. 7 shows a close-up of this mesh and of the Mach number distributions on the surface, including both the upper- and lower-surface shocks. In all cases, agreement with the AGARD data [3] is very good, and over-





	2nd	3rd	4th	AGARD (192×39)
C_L	0.3376	0.3394	0.3451	0.3474
C_D	0.0221	0.0223	0.0225	0.0221

Fig. 7. Mesh and Mach number distribution for the transonic test case.

shoots are minimal. Our lift and drag results compare favorably to the AGARD results obtained on a structured mesh with 40% more control volumes, despite a deliberate lack of adaptive refinement at the shocks on our part.

For transonic flow, the shock locations in the flow field and their strengths need to be captured relatively accurately before Newton iterations can decrease the residual of the non-linear problem effectively. For the second and third-order start-up phases, defect correction pre-iterations continue until the residual of the non-linear problem drops by a factor of $10^{1.4}$ compared with the initial residual. As shown in Table 3, for the second and third-order cases the starting CFL is 2.0, increasing gradually to CFL = 500 after 50 pre-iterations. The CFL number remains constant for the rest of the pre-iterations. For the fourth-order start-up, the CFL number is not increased above 200 as larger time steps do not help convergence when the linearization is not accurate. For the fourth-order Newton iterations, limiter oscillations and poor conditioning of the linear system (even with preconditioning) contribute to an inaccurate solution update and poor overall convergence. Using a finite CFL = 10,000 instead improves performance for the "Newton"-GMRES phase for the fourth-order scheme.

The convergence summary is tabulated in Table 4 and the convergence history is displayed in Fig. 8; exact Newton results are not given here, because that approach was shown not to be competitive, on a time basis, for the subsonic case. A constant subspace size without restart leads to poor linear system solution for the fourth-order linear system for this case. Consequently more outer (Newton) iterations are needed to reduce the non-linear residual where the fourth-order discretization is employed. For this transonic case, the third-order solution is about 1.3 times and the fourth-order solution is about 3.2 times more expensive than the second-order solution, where ILU(4) is used as a preconditioning technique. Just as with the subsonic case,

Table 3
Variation of CFL number with iteration for the transonic airfoil case

	CFL at iteration ...						Total # of pre-iterations
	1–10	11–30	31–50	51+	101+	Newton	
2nd/3rd	2	20	100	500	–	10^6	100
4th	2	10	10	100	200	10^4	187

Table 4
Convergence summary for NACA 0012 airfoil, $M = 0.85$, $\alpha = 1°$

Order	Resid. Eval.	Time (s)	Work units	Newton iterations	Newton phase work units
ILU(4)/Inexact Newton					
2nd	292	103.4	431	6	139
3rd	292	137.7	355	6	144
4th	499	330.8	609	11	273
LU/Inexact Newton					
2nd	272	126.8	526	7	230
3rd	292	201.0	500	6	289
4th	509	371.0	665	11	339

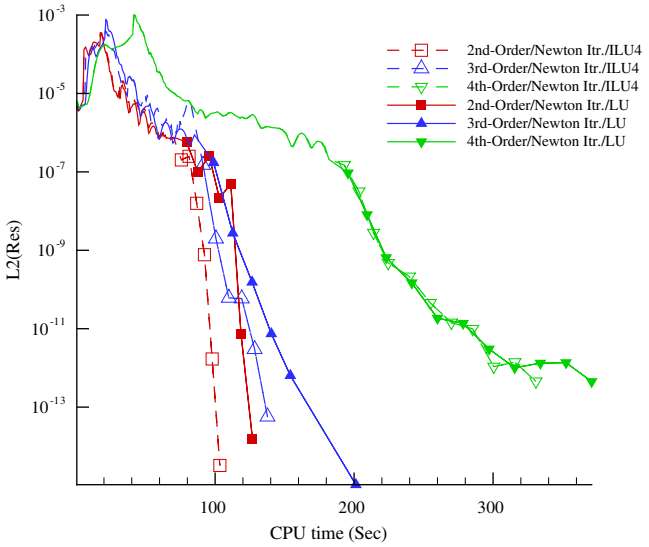
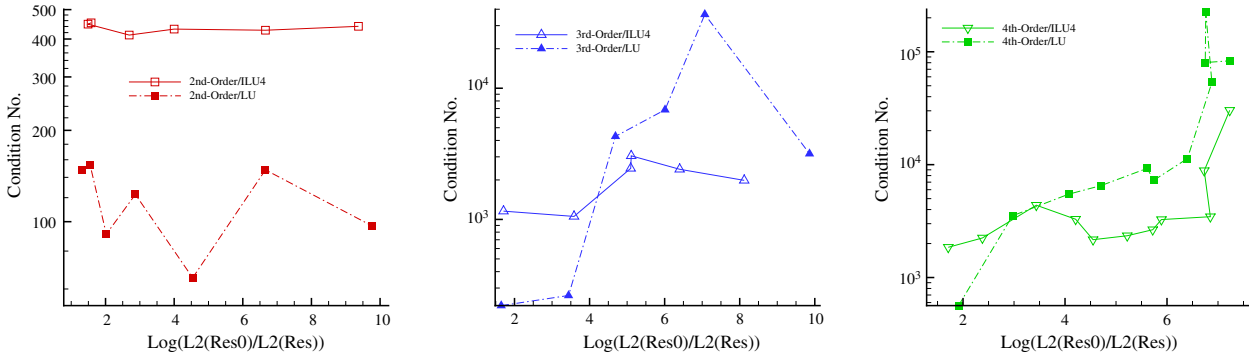


Fig. 8. Convergence history, NACA 0012 (5354 CVs), $M = 0.85$, $\alpha = 1°$.

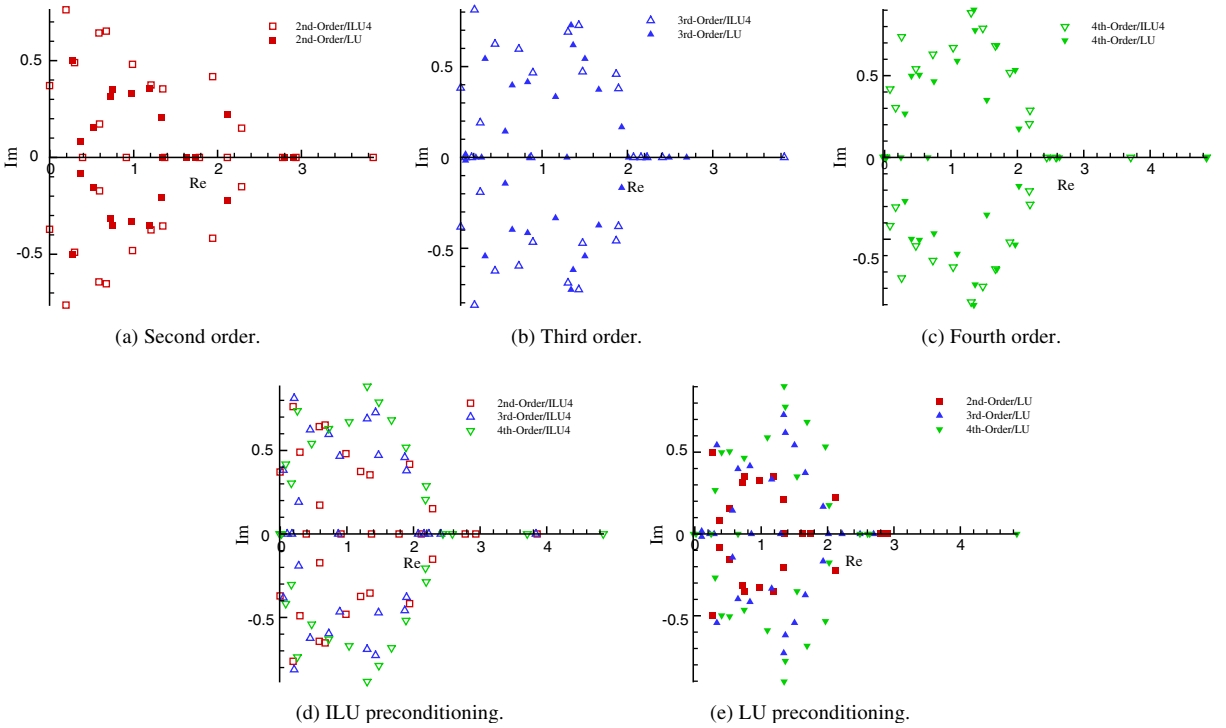
employing full factorization of the first-order Jacobian preconditioner matrix does not help the overall convergence performance.

The estimated condition numbers of the preconditioned linear systems for Newton iterations at all discretization orders are shown in Fig. 9; the condition number becomes larger as we increase the order of accuracy



(a) Second order. (b) Third order. (c) Fourth order.

Fig. 9. Condition number of the preconditioned system (Newton iterations), NACA 0012 (5354 CVs), $M = 0.85$, $\alpha = 1°$.



(a) Second order. (b) Third order. (c) Fourth order.

(d) ILU preconditioning. (e) LU preconditioning.

Fig. 10. Eigenvalue pattern for the preconditioned system (converged solution), NACA 0012 (5354 CVs), $M = 0.85$, $\alpha = 1°$.

showing the rising complexity of the associated linear system. Compared to the subsonic case, the conditioning of the transonic case is worse because of the existence of discontinuities in the flow field. For third- and fourth-order accuracy, the preconditioned system has larger condition number when LU is used than when ILU(4) is used. This shows that for flows with discontinuities the exact factorization, LU, will not help the conditioning of the preconditioned linear system (i.e. the quality of preconditioning) when the preconditioning matrix is formed based on the first-order Jacobian. For the third and fourth discretization orders the condition number of the linear system rises suddenly just before reaching the converged solution. Because this phenomenon is particularly strong when LU factorization is used, we speculate that it may be caused by trying to extract too much information about the high-order solution from a low-order Jacobian that simply cannot support the higher level of detail.

The eigenvalue patterns for all orders of accuracy are shown in Fig. 10 for the preconditioned operator at the last Newton iteration. The eigenvalues of the preconditioned system are again clustered near one, though not particularly tightly. The LU preconditioning compared to ILU(4) clusters eigenvalues better for the second-order case but loses its effectiveness as the accuracy of the discretization increases. Also eigenvalue scatter increases for high-order discretizations, illustrating the reduction in quality of preconditioning. Like the subsonic case, the smallest eigenvalues are closer to the origin for high-order discretizations.

5. Conclusion

An ILU preconditioned Newton-GMRES algorithm has been presented for the high-order solution of inviscid compressible flows. The robustness and fast convergence of the approach have been demonstrated. We have investigated the effect of the discretization order on preconditioning and convergence of the Newton-GMRES solver for subsonic and transonic test cases by examining the conditioning and eigenvalue patterns of the preconditioned system.

Our subsonic results clearly show that the convergence rate of the flow solver depends both on the accuracy of the linearization of the non-linear problem and on the conditioning of the linear system. The dependence on

linearization accuracy implies that the use of highly accurate Jacobians is counterproductive when far from the converged solution because of non-linear effects, but mandatory to attain optimal convergence rate near steady state. For high-order discretizations, the quality of the preconditioned system is not as good as it is for the second-order discretization and more outer or Newton iterations are needed for full convergence, especially for fourth-order discretization. The quality of the preconditioning – as measured by condition number and eigenvalue spectra – can be improved by using LU factorization instead of ILU for smooth flows. Furthermore, the number of Newton iterations can be reduced dramatically if the exact Newton approach is used. However, the overall solver efficiency is still best with ILU(4) and an inexact Newton approach.

For a transonic case, we found similar trends. Again, both accuracy of the linearization and conditioning of the linear system are important for convergence. In this case, the start-up procedure must provide a solution with shock strengths and locations reasonably accurately known before Newton iteration is effective. Degradation of convergence rate with increasing order of accuracy, as measured by both iteration count and CPU time, is more severe for the transonic problem. Also, we found that the quality of the preconditioning by exact (LU) factorization can be degraded compared to ILU(4) for a high-order transonic flow. More effective preconditioning of transonic cases will require careful study within the context of the discontinuity handling methodology.

The number of Newton iterations for ILU(4) and LU for all cases were almost the same, showing the approximate equivalence of the performance of these two preconditioning techniques given the pre-set tolerance and the first-order preconditioner matrix. Choices made about preconditioning are critical for efficient convergence for high-order schemes. Incomplete factorization as a preconditioning method and a low-order Jacobian as a preconditioner matrix provide a satisfactory convergence rate for all discretization orders. However, the fourth-order discretization convergence is considerably slower than the other two accuracy orders using the current matrix-free approach.

Acknowledgments

This work was supported by the Canadian Natural Sciences and Engineering Research Council under Grant OPG-0194467. We would like to thank the reviewers of this paper for many helpful comments which improved the presentation significantly.

References

[1] Rémi Abgrall, Design of an essentially non-oscillatory reconstruction procedure on finite-element type meshes, ICASE Report No. 91-84, NASA Langley Research Center, 1991, NASA CR 189574.
[2] Rémi Abgrall, On essentially non-oscillatory schemes on unstructured meshes: analysis and implementation, Journal of Computational Physics 114 (1) (1994) 45–58.
[3] AGARD Fluid Dynamics Panel, Test Cases for Inviscid Flow Field Methods, AGARD Advisory Report AR-211, AGARD, May 1985.
[4] S. Balay, K. Buschelman, D. Gropp, W.D. Kaushik, M. Knepley, B.F. McInnes, L.C. Smith, H. Zhang, PETSc home page. <http://www.mcs.anl.gov/petsc>, 2004.
[5] Timothy J. Barth, Analysis of implicit local linearization techniques for upwind the TVD algorithms. Twenty-fifth Aerospace Sciences Meeting, January 1987, AIAA Paper 87-0595.
[6] Timothy J. Barth, Recent developments in high order *k*-exact reconstruction on unstructured meshes, AIAA paper 93-0668, January 1993.
[7] Timothy J. Barth, Aspects of unstructured grids and finite-volume solvers for the Euler and Navier–Stokes equations, in: Lecture Series 1994-05, Rhode-Saint-Genèse, Belgium, March 1994. von Karman Institute for Fluid Dynamics.
[8] 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.
[9] Timothy J. Barth, Dennis C. Jespersen, The design and application of upwind schemes on unstructured meshes, AIAA paper 89-0366, January 1989.
[10] 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.
[11] M. Benzi, Preconditioning techniques for large linear systems: a survey, Journal of Computational Physics 182 (2002) 18–477.
[12] Max Blanco, David W. Zingg, Fast Newton–Krylov method for unstructured grids, American Institute of Aeronautics and Astronautics Journal 36 (4) (1998) 607–612.

[13] E.H. Cuthill, J.M. McKee, Reducing the bandwidth of sparse symmetric matrices, in: Proceedings of the 24th National Conference of the Association for Computing Machinery, 1969, pp. 157–172.
[14] 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.
[15] 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.
[16] 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.
[17] Michel Delanaye, Polynomial Reconstruction Finite Volume Schemes for the Compressible Euler and Navier–Stokes Equations on Unstructured Adaptative Grids, Ph.D. Thesis, Universite de Liege, Faculte des Sciences Appliquees, 1998.
[18] Michel Delanaye, Phillippe Geuzaine, J.A. Essers, Compressible flows on unstructured adaptive grids, in: Proceedings of the Thirteenth AIAA Computational Fluid Dynamics Conference, 1997.
[19] Ron S. Dembo, Stanley C. Eisenstat, Trond Steihaug, Inexact Newton methods, SIAM Journal on Numerical Analysis 19 (1982) 400–408.
[20] Olivier Friedrich, Weighted essentially non-oscillatory schemes for the interpolation of mean values on unstructured grids, Journal of Computational Physics 144 (1) (1998) 194–212.
[21] 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.
[22] Gene H. Golub, Charles F. Van Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, 1983.
[23] Ami Harten, High-resolution schemes for hyperbolic conservation-laws, Journal of Computational Physics 49 (3) (1983) 57–393.
[24] C.Q. Hu, C.W. Shu, Weighted essentially non-oscillatory schemes on triangular meshes, Journal of Computational Physics 150 (1) (1999) 97–127.
[25] Z. Johan, T.J.R. Hughes, F. Shakib, A globally convergent matrix-free algorithm for implicit time-marching schemes arising in finite-element analysis in fluids, Computer Methods in Applied Mechanics and Engineering 87 (2–3) (1991) 281–304.
[26] L.M. Manzano, J.V. Lassaline, P. Wong, D.W. Zingg. A Newton–Krylov Algorithm for the Euler Equations Using Unstructured Grids, AIAA Pap. 2003-0274, 41th AIAA Aerospace Sciences Meeting and Exhibit, 2003.
[27] Krzysztof Michalak, Carl Ollivier-Gooch. Differentiability of slope limiters on unstructured grids, in: Proceedings of the Fourteenth Annual Conference of the Computational Fluid Dynamics Society of Canada, SociTtT canadienne de CFD/CFD Society of Canada, 2006.
[28] Krzysztof Michalak and Carl Ollivier-Gooch. Limiters for unstructured higher-order accurate solutions of the euler equations, in: Forty-sixth Aerospace Sciences Meeting, 2008.
[29] Amir Nejat, A Higher-Order Accurate Unstructured Finite Volume Newton-Krylov Algorithm for Inviscid Compressible Flows, Ph.D. Thesis, University of British Columbia, Department of Mechanical Engineering, 2007.
[30] Amir Nejat and 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.
[31] 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.
[32] Amir Nejat, Carl Ollivier-Gooch, On preconditioning of Newton-GMRES algorithm for a higher-order accurate unstructured solver, in: Proceedings of the Fourteenth Annual Conference of the Computational Fluid Dynamics Society of Canada, cfdsc, 2006.
[33] Amir Nejat, Carl Ollivier-Gooch, A high-order accurate unstructured finite volume Newton–Krylov algorithm for inviscid compressible flows, Journal of Computational Physics (2007), doi:10.1016/j.jcp.2007.11.011.
[34] 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.
[35] Carl Ollivier-Gooch, Amir Nejat, Christopher Michalak, On obtaining high-order finite-volume solutions to the Euler equations on unstructured meshes, in: Proceedings of the Eighteenth AIAA Computational Fluid Dynamics Conference, 2007.
[36] 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.
[37] 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.
[38] A. Pueyo, D.W. Zingg, Progress in Newton–Krylov methods for aerodynamic calculations, in: Thirty-fifth Aerospace Sciences Meeting, 1997, AIAA Paper 97-0877.
[39] A. Pueyo, D.W. Zingg, Efficient Newton–Krylov solver for aerodynamic computations, American Institute of Aeronautics and Astronautics Journal 36 (11) (1998) 1991–1997.
[40] P.L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (1981) 357–372.
[41] 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.
[42] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., Society for Industrial and Applied Mathematics, 2003.
[43] 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.

[44] V. Venkatakrishnan, Convergence to steady-state solutions of the Euler equations on unstructured grids with limiters, Journal of Computational Physics 118 (1995) 120–130.
[45] V. Venkatakrishnan, D. Mavriplis, Implicit solvers for unstructured meshes, Journal of Computational Physics 105 (1993) 83–91.
[46] 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.