

Effect of Discretization Order on Preconditioning and Convergence of a Higher-Order Unstructured Newton-Krylov Solver for Inviscid Compressible Flows

Amir Nejat* and Carl Ollivier-Gooch†

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

The effect of discretization order on preconditioning and convergence of a higher-order Newton-Krylov unstructured flow solver is studied in detail for the 2nd, 3rd, and 4th-orders of accuracy. 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, and their performances are compared for all discretization orders. The conditioning and eigenvalue spectrum of the preconditioned system are examined to investigate the quality of preconditioning.

I. Introduction

Employing high-order discretization schemes is a major area of interest for solver developers to enhance solution quality and improve the overall accuracy of the solution for unstructured meshes. For structured meshes, application of high-order algorithms has progressed considerably and it has been shown that, for practical levels of accuracy, using a high-order accurate method can be more efficient both in terms of accuracy and solution time. With high-order accurate methods, the cost of flux computation, integration, and other associated numerical calculations increase per control volume. However, as we can use a coarser mesh, computation time and memory are saved overall while accuracy is preserved by high-order discretization.^{23, 24, 36} 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.

Although high-order accurate methods for unstructured meshes are reasonably well established,^{2, 10, 18} application of these methods for physically complicated flows is still a challenge due to very slow convergence. This eliminates the efficiency benefits of higher-order unstructured discretization and limits its application for practical purposes. Consequently, fast convergence and robustness become the key issue for the practical usage of higher-order unstructured solvers; convergence acceleration techniques such as implicit time advance play an important role in robustness of higher-order methods.

Newton-Krylov solvers^{5, 20, 33} are used extensively in CFD simulations because of their property of semi-quadratic convergence when starting from a good initial solution. The Newton-GMRES algorithm, among

*PhD Candidate, nejat@mech.ubc.ca, Student Member AIAA

†Associate Professor, cfog@mech.ubc.ca, Member AIAA

other Krylov techniques, is a very practical and attractive technique for dealing with the complicated Jacobian matrices arising from higher-order discretization since it only needs matrix vector products and these products can be computed by a matrix free approach.^{5, 11} This approach saves memory usage considerably and removes the problem of explicitly forming the higher-order Jacobian matrix.

For CFD problems, the Jacobian matrix is typically ill-conditioned, and effective preconditioning becomes crucial since the performance of any implicit methods such as Newton-Krylov is highly dependent on the quality of preconditioned system. Preconditioning consists of two parts i.e., preconditioning method and preconditioner matrix. In most cases the preconditioner matrix is an approximation of the Jacobian matrix and needs to be available explicitly. A good preconditioner matrix should be easy to compute and a relatively accurate representation of the flux integral's derivatives. Considering the complexity of the unstructured higher-order discretization method, computing the higher-order Jacobian matrix is an expensive task in terms of memory usage and computation cost. Even if memory were not an issue, computing a higher-order Jacobian is quite expensive in terms of CPU time. Therefore, the accuracy of Jacobian calculation involves a trade off.

For effective preconditioning, in addition to applying a good preconditioner matrix, we need to employ a good preconditioning technique.²⁷ Several authors have studied the effect of various preconditioning strategies on convergence of the matrix-free Newton-GMRES both for structured and unstructured meshes.^{5, 7, 13, 22, 33} Their research shows incomplete lower-upper (ILU) factorization of the approximate Jacobian is a very efficient preconditioning strategy for CFD problems.

The objective of this research is to investigate the effect of discretization order on preconditioning and convergence of a higher-order unstructured Newton-GMRES solver. A defect correction procedure is used for the start-up phase before performing Newton iterations. In Newton phase, infinite time step is taken (except for the 4th-order discretization) to achieve the desirable super-linear convergence rate. Both LU and ILU are employed in the Newton phase to study the effect of factorization accuracy on convergence rate. In addition, our experience shows a first-order Jacobian can be used for effective preconditioning of the linear system resulting from a higher-order discretization provided that a high-fill (typically 4) incomplete factorization is employed.^{16, 17} The restarted version of the GMRES algorithm is used to demonstrate the possibility of reducing the number of Newton (outer) iterations (especially for the 4th-order discretization) at the increasing expense of the overall solution time. The conditioning and eigenvalue spectrum of the preconditioned operator are examined for subsonic and transonic cases.

II. Flow solver

II.A. Governing equations

The finite volume formulation of the unsteady 2D Euler equations for an arbitrary control volume can be written in the following form of a volume and a surface integral (1), where U is the solution vector in conservative variables, and F is the flux vector, Eq. (2).

$$\frac{d}{dt} \int_{CV} U dV + \oint_{CS} F dA = 0 \quad (1)$$

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

II.B. High-Order Reconstruction Procedure

The goal here is to reconstruct a solution polynomial, $U_R^{(K)}(x, y)$, based on the control volume data such that the truncation error of the solution in each of our cell-centered control volumes will be $\mathcal{O}(\Delta x^k)$ where Δx is the local mesh length scale. To achieve this, we perform a k -exact least-squares reconstruction, as described first by Barth and Frederickson.³ By design, this reconstruction procedure ensures conservation of the mean of each variable within each control volume, Eq. (5).

$$\begin{aligned}
 U_R^{(K)}(x, y) = & U(x_c, y_c) + \left. \frac{\partial U}{\partial x} \right|_C \Delta x + \left. \frac{\partial U}{\partial y} \right|_C \Delta y + \\
 & \left. \frac{\partial^2 U}{\partial x^2} \right|_C \frac{\Delta x^2}{2} + \left. \frac{\partial^2 U}{\partial x \partial y} \right|_C \Delta x \Delta y + \left. \frac{\partial^2 U}{\partial y^2} \right|_C \frac{\Delta y^2}{2} + \\
 & \left. \frac{\partial^3 U}{\partial x^3} \right|_C \frac{\Delta x^3}{6} + \left. \frac{\partial^3 U}{\partial x^2 \partial y} \right|_C \frac{\Delta x^2 \Delta y}{2} + \left. \frac{\partial^3 U}{\partial x \partial y^2} \right|_C \frac{\Delta x \Delta y^2}{2} + \left. \frac{\partial^3 U}{\partial y^3} \right|_C \frac{\Delta y^3}{6} + \dots \quad (3)
 \end{aligned}$$

where,

$$\Delta x = x - x_c, \quad \Delta y = y - y_c \quad (4)$$

$$\int_{CV} U_R^{(K)}(x, y) = \bar{U}_{CV} \quad (5)$$

In addition to the mean constraint, we impose wall boundary conditions ($u_n = 0$) as constraints on the reconstruction.¹⁸ As a result of these boundary constraints, the reconstruction in boundary control volumes automatically satisfies the physical boundary conditions exactly at boundary Gauss points while satisfying the boundary conditions to within truncation error elsewhere on the boundary.

Notice that the reconstructed polynomial, $\langle U_R^{(K)}(x, y) \rangle_i$, is piecewise and changes from one control volume to another. Therefore integrating the reconstructed function of a control volume over its neighbors would not satisfy the average value of the neighboring control volumes exactly. For the K th-order solution reconstruction, we are reconstructing a $(K-1)$ th order polynomial which satisfies the mean property for the control volume i and minimizes the error in computing control volume averages of the neighboring control volumes. The neighboring control volumes used in reconstruction constitutes the reconstruction stencil, Fig (1). The reconstruction stencil should include a proper number of neighboring control volumes for computing the mentioned derivatives. The number of unknowns for the linear, quadratic, and cubic reconstructions are 2, 5 and 9 leading to 2nd, 3rd and 4th-order accuracy. Generally speaking, for a K th-order accurate reconstruction, at least $K(K+1)/2$ neighboring control volumes must be included in the reconstruction stencil. However, for practical reasons one may use a few more control volumes in the reconstruction stencil. This leads to a larger least-square system but provides additional information for reconstruction resulting usually in more robust solution reconstruction in the presence of non-smooth and/or vigorous oscillatory data. In this research, the reconstruction stencil of 2nd, 3rd, and 4th-order methods include 4, 9 and 16 control volumes respectively.

II.C. Implicit algorithm

Assuming the discretized physical domain does not change in time, U can be brought out from the integral in Eq. (1), as the average solution vector of the control volume:

$$\frac{dU}{dt} = -\frac{1}{A_{CV}} \oint_{CS} F dA \quad (6)$$

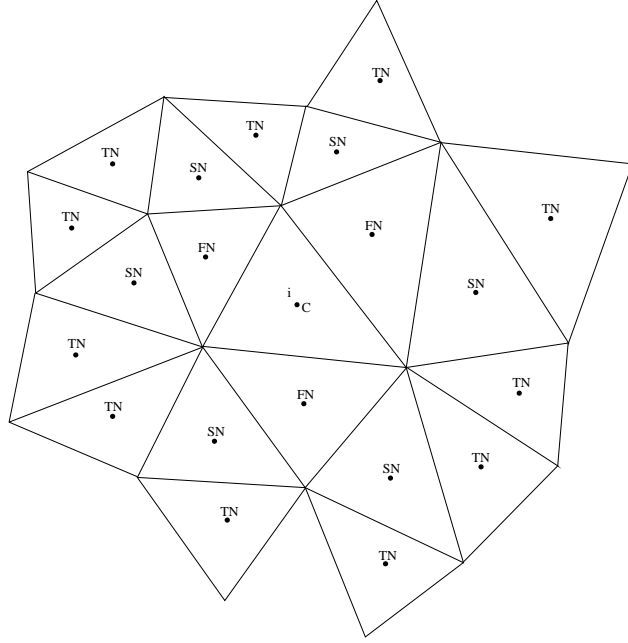


Figure 1. A typical cell center control volume and its reconstruction stencil, including three layers of neighbors

The left hand side of (6) is the first time derivative of the average solution vector in each control volume and the right hand side represents the spatial discretization for the same control volume. The right hand side of (6) is called the flux integral or residual of the control volume, which is a nonlinear function of the solution vector, and can be re-written in the form of (7) for control volume i :

$$\frac{dU_i}{dt} = -R(U_i) \quad (7)$$

An implicit time advance formula is derived by backward time differencing of the left hand side and residual linearization in time for the right hand side:

$$\left(\frac{I}{\Delta t} + \frac{\partial R}{\partial U} \Big|_i^n \right) \delta U_i = -R(U_i^n), \quad U_i^{n+1} = U_i^n + \delta U_i \quad (8)$$

where $\frac{\partial R}{\partial U}$ is the Jacobian matrix resulting from residual linearization. Equation (8) is a large linear system of equations which should be solved at each time step to obtain an update for the vector of unknowns. To reach the steady-state solution, advancing the solution in time continues until the residual of the non-linear problem converges to machine zero.

If we take an infinite time step, Eq. (8) becomes Newton iteration (9). Newton's method generally converges quadratically in the vicinity of the solution. If the solution starts from an initial condition which is not close to the steady-state solution a start-up process may be needed for the sake of stability and robustness, section (III.A).

$$\left(\frac{\partial R}{\partial U} \right) \delta U_i = -R(U_i^n), \quad U_i^{n+1} = U_i^n + \delta U_i \quad (9)$$

II.D. Residual calculation

The high-order accurate least-squares reconstruction scheme of Section II.B enables us to compute all the flow variables in the interior and at the boundaries up to 4th order accuracy. Flow quantities at the flux integration

points (Gauss points) are computed to the desired accuracy and fluxes at control volume boundaries are computed by Roe’s flux differencing,²⁵ using reconstructed flow variables. Having computed the fluxes, we use Gauss quadrature to integrate the fluxes to the same order of accuracy as the reconstruction. High-order accurate boundary treatment is also imposed.¹⁸ For the far field boundary, characteristic boundary conditions are used, and for the wall boundary a slip boundary condition is used. The wall boundary condition is implemented by employing a zero normal velocity constraint, $u_n = 0$, in the velocity component reconstruction; the flux at the wall is then computed using the analytic flux of Eq. (2).

II.E. Monotonicity

Enforcing monotonicity is one of the main issues both for the 2nd-order and higher-order upwind schemes. Limiters are often needed to suppress oscillations around discontinuities and to avoid reconstructing non-physical solution (negative density) at gauss points located close to such locations. However limiters cause two major problems. First they hamper the convergence as their values oscillate across the shock, especially in the case of a non-differentiable limiter formulation such as the Barth-Jespersen limiter.⁴ Even using a differentiable limiter does not guarantee good convergence behavior. Secondly, they change the reconstruction polynomial by reducing the reconstructed solution derivatives, affecting the accuracy of the reconstructed solution. These issues for higher-order methods are even more complicated. 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. In this research, Venkatakrishnan limiter^{31, 32} (semi-differentiable), which addresses most of the aforementioned issues, has been employed with some modifications.

Our experience as well as other research^{9, 11} shows that applying the limiter to all derivatives yields a more diffusive solution. Therefore the limiter, ϕ , is applied only to the linear part of the reconstruction, and the non-linear part of the reconstruction is dropped.^{10, 11} This also helps the limiting process around the shock as higher derivatives tend to show oscillatory behavior in the vicinity of discontinuities. Relating ϕ and σ by a semi-step function, as explained below, greatly enhance the convergence. The limited reconstruction polynomial is expressed as:

$$U_G^{(K)}(x_G, y_G) = \bar{U}_i + [(1 - \sigma)\phi_i + \sigma] \{\text{Linear Part}\} + \sigma \{\text{Higher-Order Part}\} \quad (10)$$

where σ is a limiter for higher-order terms. In smooth regions, the full higher-order reconstruction is applied by choosing $\sigma = 1$. Near discontinuities we switch from the higher-order to the limited linear polynomial to prevent possible oscillatory behavior of second and third derivatives. This is done by a discontinuity detector employing the value of limiter; if the limiter fires aggressively we assign zero to σ .

Using a switch to set σ to either zero or one stalls the convergence. To overcome this problem, σ is defined as a differentiable function of ϕ , such that σ is nearly one for the regions that $\phi \geq \phi_0$ and it quickly goes toward zero for other values of ϕ . In other words, we use a semi-step function to perform as a differentiable switch.

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

S in (11) determines the sharpness of the step function, and it adjusts how fast is the switching transition between zero and one; ϕ_0 defines the limiter value that activates the switch. It appears that $\phi_0 = 0.8$ and $S = 20$ provide a reasonable switch function whose good behavior is relatively case independent in this research.

II.F. Linear system solver

We use the Generalized Minimal Residual (GMRES) algorithm, which was designed for non-symmetric systems such as those resulting from unstructured discretizations, as a linear solver for the solution of Eq. (9) at each iteration. GMRES minimizes the residual of the linearized 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,⁵ where matrix vector products are approximated by a directional derivative formula, Eq. (12). ε_0 is a very small number typically equal to the square root of machine accuracy. However, for fine meshes where the mesh length scale is very small, we need to use larger ε_0 to accurately account for perturbation of the 4th-order terms due to round off error considerations. In this research $\varepsilon_0 = 10^{-6}$ is used.

$$\frac{\partial R}{\partial U} \cdot v \approx \frac{R(U + \varepsilon v) - R(U)}{\varepsilon}, \quad \varepsilon = \frac{\varepsilon_0}{\|v\|_2} \quad (12)$$

II.G. 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 higher-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 consequently slowing or stalling the solution process of the non-linear problem. To remove this obstacle and enhance the performance of the linear solver, the linear system needs to be modified through a process called preconditioning, such that the preconditioned system has better spectral properties and clustered eigenvalues. Although in the case of the GMRES method, eigenvalues may not describe the convergence of a nonsymmetric matrix iterations as much as they do for symmetric iterative solvers such as the Conjugate Gradient (CG) method, a clustered eigenvalue spectrum (not too close to zero) normally improves the convergence characteristics of the linear solver.⁶

Consider a preconditioner M as a nonsingular matrix which approximates the Jacobian matrix A in the linear system of $Ax = b$. The linear system then can be modified by introducing the preconditioner operator in the right side of the matrix A , and leave the right hand side intact:

$$AM^{-1} \overbrace{(Mx)}^z = b, \quad x = M^{-1}z \quad (13)$$

Of course the best apparent choice for M is A , as $AM^{-1} = I$, but if we could solve the system with the A matrix easily we never needed preconditioning in the first place. Finding the optimal preconditioner matrix is neither unique nor purely scientific, since it is highly problem dependent, and also depends on how the preconditioner is applied. Also there are circumstances where the matrix M need not be computed explicitly and a preconditioner operator replaces M , like polynomial preconditioners.²⁹ But in general three factors are considered in choosing a preconditioner:

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

The bottom line is the cost of the construction and applying the preconditioner should be relatively cheap with respect to the cost of solving the original linear system.

A preconditioning technique is a sparse linear solver itself, and it can be a direct or iterative method. Iterative stationary methods such as successive over-relaxation (SOR) rely on the fixed iteration Eq. (14), where B is the fixed iteration matrix and C is a fixed vector:

$$x_{i+1} = Bx_i + C \quad (14)$$

Stationary methods are easy to implement, often have parallelization advantages, are low cost (memory and CPU-time), and are effective in damping high frequency errors. However, they often have a restrictive stability condition, reducing the benefits of Newton’s method. This is especially true if the preconditioner matrix is off-diagonal which is the case for compressible flow. At the same time, for relatively large systems slow damping of the low frequency errors becomes a noticeable issue for these techniques in the absence of a multigrid augmentation strategy.

Another preconditioning technique is the Incomplete Lower-Upper Factorization (ILU) method. Factoring matrix M into two triangular matrices, $M = LU$, where L is a lower triangular matrix, and U is an upper triangular matrix normally results in factored matrices that are far less sparse than the original matrix M . As a consequence, a considerable amount of memory needs to be assigned for factorization, and the factorization process is quite expensive. One solution to this problem is incomplete factorization in which additional nondiagonal fill-in entries are allowed only for a predefined set of locations in the LU factorizations. In other words, we choose a non-zero pattern in advance for the elements of the factored matrices.²⁷ This is called ILU-P where P is the fill-level in the factorized matrix. P equal to zero means no fill is permitted during ILU decomposition. In ILU-0 the factorized matrix and the original (non-factored) matrix have the same graph or non-zero element locations. Choosing P larger than zero 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.

Both the preconditioner matrix and preconditioning technique may be developed based on the specific problem approach, but first one should have a detailed knowledge about the numerical aspects of the problem which is not always possible; second, generally speaking, such a preconditioning procedure is sensitive to the type of the problem, meaning that by changing the test case the benefits of the approach may be lost. Therefore, application of general preconditioning techniques such as ILU is more common for CFD solvers. Another modification in the ILU family is ILU(P, τ), where in addition to the static non-zero pattern, a tolerance τ is added as a dropping criterion for entries of the factored matrices.⁶ In other words, all small fill-in entries within the level P are changed to zero. 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, and they are not determined uniquely, ILU(P, τ) is not considered in this research.

Several researchers have implemented ILU methods for preconditioning of the GMRES linear solver for compressible fluid flows.^{5, 7, 11, 13, 34} Their results show that ILU with some additional fill-in (i.e. ILU-1&2) is a reliable and robust preconditioning strategy for a variety of test cases, while ILU-0 fails to provide fast convergence in some cases.

Reordering is another important factor in ILU factorization. Reordering is designed to reduce the bandwidth of a matrix. Smaller bandwidth leads to a sparser matrix, and consequently during factorization (Gaussian elimination process) of the matrix fewer fill-in entries will appear. 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. 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. The most common available reordering technique is Reverse Cuthill-McKee (RCM)²⁷ which has been successfully used for ILU-P preconditioning.

II.H. Preconditioner Matrix

The Jacobian matrix can be built by either analytical approach or numerical (finite difference) differentiation; both approaches have been employed quite successfully.³⁰ Analytical differentiation can be performed either manually, symbolically³⁵ or automatically.²⁸ The analytical approach computes the exact Jacobian (ignoring round off error in numerical evaluation) but it is relatively difficult to apply this approach for complicated flux functions and/or higher-order discretizations by manual or even symbolic differentiation. Automatic

differentiation (AD), a very powerful tool for computing complex Jacobians, augments the computer program to compute the derivatives by applying the chain rule repeatedly to the elementary arithmetic operations performed by the computer program. The derivatives are computed relatively cheaply by this method and are accurate to machine accuracy (i.e. round off error) without any truncation error. However, AD requires very careful programming and a considerable amount of memory,⁸ and has not been considered in this research. Numerical differentiation is fairly straight forward even for higher-order and complicated flux functions. It also works reasonably well where the flux function is not differentiable, but like any other numerical approach, the numerical differentiation technique suffers from truncation error as well as round off error.

In this research a first-order Jacobian matrix is used as a preconditioner matrix. In the case of the 2D Euler equations discretized over a cell-centered unstructured mesh, each control volume has 3 direct (first) neighbors, and the first-order flux integral of the control volume i only depends on these three neighbors and the control volume itself (Fig(2)):

$$R_i = \sum_{m=1,2,3} (F\hat{n}ds)_m = F(U_i, U_{N_1})\hat{n}_1l_1 + F(U_i, U_{N_2})\hat{n}_2l_2 + F(U_i, U_{N_3})\hat{n}_3l_3 \quad (15)$$

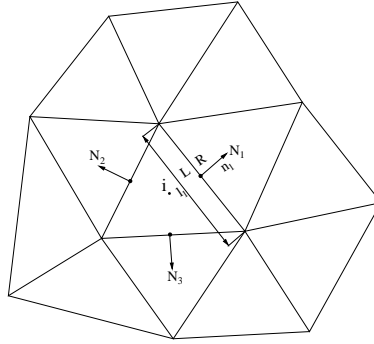


Figure 2. Schematic of Direct Neighbors

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. To compute the flux Jacobian of the control volume i , we need to take the derivatives of the flux integral or residual function, Eq. (15), with respect to all control volume averages involved in the residual function evaluation as is shown in the following formulas:

$$J(i, N_1) = \frac{\partial R_i}{\partial U_{N_1}} = \frac{\partial F(U_i, U_{N_1})}{\partial U_{N_1}}\hat{n}_1l_1 \quad (16)$$

$$J(i, N_2) = \frac{\partial R_i}{\partial U_{N_2}} = \frac{\partial F(U_i, U_{N_2})}{\partial U_{N_2}}\hat{n}_2l_2 \quad (17)$$

$$J(i, N_3) = \frac{\partial R_i}{\partial U_{N_3}} = \frac{\partial F(U_i, U_{N_3})}{\partial U_{N_3}}\hat{n}_3l_3 \quad (18)$$

$$J(i, i) = \frac{\partial R_i}{\partial U_i} = \frac{\partial F(U_i, U_{N_1})}{\partial U_i}\hat{n}_1l_1 + \frac{\partial F(U_i, U_{N_2})}{\partial U_i}\hat{n}_2l_2 + \frac{\partial F(U_i, U_{N_3})}{\partial U_i}\hat{n}_3l_3 \quad (19)$$

Since both the flux F and solution U are 4-component vectors, each entry in the Jacobian matrix J is a 4×4 matrix and the total size of the block matrix is $n \times n$ where n is the total number of control volumes. However, most of these blocks are just zeros as in general there are no more than three neighboring control

volumes in the first-order cell-center formulation and the Jacobian matrix is very sparse, with at most four non-zero blocks per row.

The approximate derivatives of fluxes in Eq. (16) through Eq. (19) can be evaluated analytically^{1, 15, 16} where the differencing term in Roe's flux formula²⁶ is assumed to be constant for simplification purposes. This leads to inaccurate linearization for non-smooth flows and poor preconditioning especially in Newton iterations. As an alternative, finite difference technique can be employed to accurately compute the mentioned flux derivatives.¹⁹

III. Solution Strategy

The final goal of all CFD solver developers is reducing the cost of computation as well as improving the accuracy of the solution. The latter could be achieved by applying higher-order discretization methods. However, in a practical sense that is possible if and only if the cost of the computation is comparable to the cost of the common second-order methods. That is why implicit techniques (Newton family of methods) play a crucial role in achieving that goal since the best explicit techniques are far less efficient with respect to computation cost making higher-order computation quite uncompetitive. In addition to the computation cost, i.e. CPU-time, memory usage is also an issue for solver developers. But in most cases, CPU-time is the main concern since if storing a set of data (i.e. Jacobian matrix) is expensive, then computing that data would be expensive too except in the case that the expensive computation is performed once and the resultant work is used in multiple iterations.

Knowing the over all approach, which is the implicit formulation, and the main objective which is the steady-state solution, we would like to lay out an efficient strategy to reach the steady-state solution as fast as possible.

The overall computing cost for a steady-state CFD problem by an implicit approach, can be analyzed by dividing the solution process into three major parts:

1. Forming the linear system including linearization and the non-linear residual computation.
2. Solving the linear system including preconditioning.
3. Number of implicit iterations needed for steady state convergence.

It is obvious that we would like to minimize the cost of the first and second part of the solution process and reach the steady-state solution with minimum number of iterations. The third part has a cumulative effect on the overall solution cost since in each (non-linear) iteration the first two parts are covered once already.

Most CFD problems, including compressible flows, are highly non-linear, and that makes them not too easy to solve via linearization. In addition to non-linearity in the mathematical sense, the behavior of a compressible flow (the physics of the flow) can dramatically change in some flow conditions for a fixed geometry. Therefore if the solution process is started from an arbitrary initial condition which is not reasonably close to the steady-state solution, large updates to the solution based on linearization are not likely to be helpful. Also, Newton's method is well known to converge to the solution quadratically if started from the vicinity of the solution; otherwise it will diverge or stall quickly. Since finding a good approximate solution in general (that is, except the cases for which some close solution is available through analytical or experimental data) is impossible without actually starting to solve the problem, the solution process should consist of two phases:

1. Start-up phase
2. Newton phase

In the start-up phase, multiple linearizations with small or moderate solution updates are required to advance the solution to the point that the linearization is accurate enough for finding the steady state solution, i.e. a good approximate solution as initial starting point for Newton phase. Having a good initial guess, the solution process is switched to the Newton iteration where super-linear or quadratic convergence is possible by taking a very large (or infinite) time step.

III.A. Start-up Phase

Finding a good initial guess or reasonable approximate solution is about knowing the physics of the problem and finding a proper way to get to that good initial solution faster by employing various techniques such as mesh sequencing, multigrid, mixed explicit/implicit iterations, exact solution of a simplified problem, potential flow solution, and so on. Hence, like preconditioning, start-up is problem dependent and is more an art than an exact science, implying that the proper start-up process is not unique either. In this research a start-up process for compressible flows is suggested which is based on the defect correction procedure.¹⁴

In the start-up process, considering the fact that multiple iterations are required for advancing the solution toward a good initial guess, an implicit iterative process is used where the linearization is performed based on the inexpensive first-order discretization and the flux calculation remains higher-order:

$$\left(\frac{I}{\Delta t} + \frac{\partial R}{\partial U} \Big|_{1st}^n \right) \Delta U^{n+1} = -R_{High}(U^n), \quad U^{n+1} = U^n + \omega \Delta U^{n+1} \quad (20)$$

A relaxation factor, ω , is applied for the solution update; $\omega = 1$ results in standard defect correction, while using different values for ω (typically $\omega = 0.9-1.3$) can over-relax or under-relax the update.¹⁷ Over-relaxation often is helpful in subsonic flows where it accelerates the solution while under-relaxation can prevent divergence resulting from an inaccurate solution update, for instance in transonic cases.

The Jacobian matrix on the left hand side is the first-order Jacobian and its computation cost is approximately the same as the cost of the second-order residual calculation. The cost of one Jacobian evaluation is 0.6-0.7 of the cost of a second-order residual evaluation (limiting cost excluded) for the approximate analytical Jacobian and it is 1.3-1.5 of the computation cost of the same residual evaluation if the finite-difference Jacobian is employed (for moderate mesh sizes).

To reduce the cost of forming the linear system, the Jacobian matrix may be computed and stored for several iterations.¹² With this approach the Jacobian is updated every j iteration(s) (typically $j = 1-4$).¹⁷

As was noted before using the slope of the residual function for estimating the behavior of the residual function over a large span of the independent variables is not accurate in the early stage of the solution process. Therefore it makes sense to start from a relatively small Δt or CFL, and to gradually increase CFL to some modest number as we are getting closer to the solution of the residual function, i.e. $F(U) = 0$. This justifies applying the first-order linearization in the start-up process instead of the correct order Jacobian calculation; the linearization does not allow large steps towards the steady-state solution, so why should we spend too much work to compute it, especially if implementation of the Jacobian in the linear solver requires several matrix-vector multiplications? At the same time, solving a linear system based on the first-order Jacobian is much easier than doing so for higher-order Jacobian (even for the second-order one), because of the structure of the higher-order Jacobian matrix. Therefore we are better off to form and solve a less expensive system, since we must inevitably perform several iterations at the start-up phase.

The next step is solving the linear system. Since we are using an approximate linearization, it does not make sense to solve the approximate linear system exactly, therefore the linear system in each defect correction iteration, referred to as pre-iteration from now on, is solved approximately. The goal of each pre-iteration is to reduce the non-linear higher-order residual by cheap linearization. Consequently it is logical to solve the linear system up to some fraction of the non-linear residual, i.e. tolerance = $C \times Res(U)$, $C \ll 1$. Employing this approach saves us from spending too much effort on finding a precise solution to a linear system that we know will not give us the correct solution to the non-linear problem. The right-preconditioned

GMRES(m) algorithm is used with a limited subspace size ($K=15-20$) and relatively a loose tolerance ($0.1-0.05 Res(U)$) is set for solving the linear system. No restart is allowed, and if the tolerance is not reached within the maximum search directions (subspace size) GMRES is terminated anyway and the computed solution update up to that point is taken to move to another non-linear outer iteration. The preconditioner matrix is the same as the linear system matrix which is the perfect choice for preconditioning because the first-order Jacobian has been built already for forming the linear system, and it is the most accurate preconditioner matrix for that case. To reduce the preconditioning cost, a low fill-level incomplete factorization, ILU(1), is applied, which is accurate and efficient enough for this first order linear system.

Now the question is how close we should be to the solution before switching to Newton iterations. Normally the decrease of some norm of the non-linear residual at the current iteration compared to the initial residual, $\frac{\|Res(U)\|_n^0}{\|Res(U)\|_n}$, is chosen for that purpose. That is, if the solution reaches a specific relative norm of the non-linear residual, then the linearization is accurate enough to take a very large time step at each iteration. It should be noted that the value for this criterion varies from problem to problem. However, it is possible to find reasonable values for different categories of the compressible flows.

III.B. Newton Phase

By this stage most of the transient behaviors of the non-linear CFD problem have been removed from the flow field solution, and major steady features of the fluid flow have appeared in the solution domain. Consequently the linearization of the non-linear residual is fairly accurate for seeking the steady-state solution, and taking infinite or very large time step accelerates the solution toward quadratic or super-linear convergence, i.e. switching to the Newton iteration formula:

$$\left. \frac{\partial R}{\partial U} \right|_{High}^n \Delta U^{n+1} = -R_{High}(U^n), \quad U^{n+1} = U^n + \Delta U^{n+1} \quad (21)$$

To have an accurate linearization, the higher-order Jacobian must be applied during the Newton iterations, Eq. (21). This has been implemented through matrix-free GMRES, using finite-difference directional matrix-vector multiplication technique. The linear system is right preconditioned by the first-order Jacobian, which indeed is essential due to bad conditioning of the left hand side. ILU(P=2-4) is used for preconditioning and normally for higher-order computation (especially the fourth-order transonic flow), increasing the fill-level is greatly beneficial.¹⁷ A fixed number of search directions is employed ($K=30$) and it is important to keep them limited since the matrix-free GMRES evaluates the non-linear residual once per search direction, which is very expensive for higher-order residual computation. The linear system is again solved approximately but this time with a tighter tolerance ($10^{-2}Res(U)$) 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, and although it can increase the non-linear outer iteration number, it reduces the overall CPU-time as considerable computation time is saved by not computing the exact solution of the linear system.^{7,13,16,21} Notice that the left hand side is not quite exact because of the truncation error in the matrix-vector multiplications and possibly is polluted by round off error, especially for higher-order terms. Therefore it is preferable to give up the quadratic convergence instead of increasing the solution CPU time.

IV. Results

To investigate the effect of discretization order on preconditioning and convergence of the proposed Newton-GMRES solver a subsonic case (smooth flow) and a transonic case (flow with discontinuity) have been studied which include most features of the solver performance characteristics.

IV.A. Subsonic flow over NACA 0012, $M = 0.63$, $\alpha = 2^\circ$

The subsonic flow for a NACA 0012 airfoil using an unstructured mesh with 9931 control volumes is computed at $M = 0.63$, $\alpha = 2^\circ$ 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 for the start-up phase is 5×10^{-2} and for the Newton phase (inexact Newton iteration) is 1×10^{-2} of the l_2 norm of the non-linear residual which often is not reached within the allowable inner iterations. For all discretization orders a subspace of 30 has been set and no restart is allowed for the inexact Newton iterations. The preconditioning for the start-up pre-iterations is performed by employing the approximate analytical Jacobian matrix¹⁵ with ILU-1 factorization and for the Newton iteration 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.

No attempt for optimizing the start-up process is made. The solution starts with 30 pre-iterations in the start-up process to reach a good initial solution before switching to Newton iterations. The CFL starts at 2.0 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 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 (Eq. (20)) is evaluated to the second-order of accuracy. The cost of each pre-iteration includes one first-order Jacobian evaluation and its incomplete factorization, one flux evaluation, and one system solve using GMRES, which is not matrix-free since the Jacobian matrix is available explicitly.

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.

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:

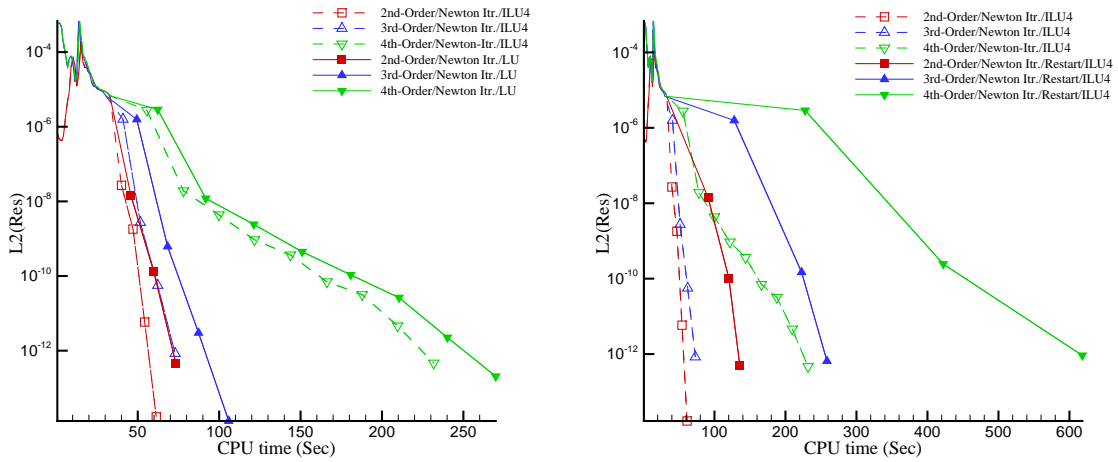
1. Inexact Newton iteration with ILU-4 preconditioning.
2. Inexact Newton iteration with LU preconditioning.
3. Exact Newton iteration with ILU-4 preconditioning; up to 10 restarts are allowed for reaching machine accuracy in the linear solver.

To be able to compare the computing cost a work unit has been used, which is simply equivalent to the cost of one residual evaluation for the corresponding order of accuracy for a specific mesh. Table 1 shows the convergence summary for the 2nd, 3rd and 4th-order discretizations in terms of total number of residual evaluations, total CPU time, total work units, number of Newton iterations, and the cost of Newton phase in terms of work units. Also, the convergence history in terms of CPU time is shown in Fig (3). For all discretization orders, after 30 pre-iterations, the solution has converged after a few Newton iterations, Fig (3). As it is evident full convergence is achieved, but the CPU-Time for the 4th-order case is much larger than the other two orders of accuracy. This was partially expected since the 4th-order discretization requires more operations, and the cost of the reconstruction rises quadratically by increasing the discretization order. However for inexact cases, a considerable part of the computing cost is due to noticeable increase in outer iteration numbers. Furthermore, the linear system arising from the 4th-order discretization is ill-conditioned and difficult to solve which results in relatively ineffective linearization of the non-linear problem (considering the fixed subspace size and Inexact Newton approach) demanding more outer iterations. On the contrary, both the 2nd and 3rd-order cases quickly converge displaying the effectiveness of the linearization.

Using the LU factorization, the most accurate factorization for preconditioning, slightly enhances the convergence in terms of the Newton iteration numbers as the preconditioning quality is increased. However, since the LU factorization cost for preconditioning is more expensive than the ILU4 factorization, the overall

Table 1. Convergence summary for NACA 0012 airfoil, $M = 0.63$, $\alpha = 2^0$

Test Case	No. of Res.	Time (Sec)	Work Unit	No. of Newton Itr.	Newton Phase Work Unit
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
ILU4/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

Figure 3. Convergence history, NACA 0012 (9931 CVs), $M = 0.63$, $\alpha = 2^0$

solution time increases. In the LU case, it is clear because of the huge memory requirements,^{17,27} its practical application is limited to small size meshes; but it is worth mentioning that ILU4 has almost the same performance in terms of iteration count as LU has in the Newton phase given the first-order preconditioner matrix and the inexact approach.

For the exact Newton case where the linear system in each Newton outer iteration is solved to the machine accuracy using multiple restarts, the semi-quadratic convergence is achieved for all discretization orders including the 4th-order case. However, due to multiple restarts the total number of the non-linear residual evaluation increases dramatically resulting in inefficient solution time. It should be mentioned that the exact Newton approach in the matrix-free version is not quite exact in practice because of the truncation error of the matrix vector products.

The 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. Firstly, the first-order linearization matrix (Jacobian matrix) is employed which is an approximation for higher-order linearization and not equal to the original linearization. Secondly, an 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 scattered around unity while some of them could be very close to one. 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 (zero) 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 (estimated via GMRES algorithm) is plotted at the last iteration (i.e. converged solution) for both the LU and ILU-4 factorizations, Fig (4).

The eigenvalues associated with higher-order discretizations are scattered with larger distances from one compared to the 2nd-order eigenvalues. This clearly indicates a reduction in the quality of preconditioning with increasing discretization order, which of course was not hard to predict in the first place. At the same time, the higher the discretization order the closer to the origin eigenvalues are located shifting the matrix toward singularity. This is especially the case for the 4th-order discretization where one of the eigenvalues is very close to origin. Based on the plotted eigenvalue patterns, as it was expected, the quality of preconditioning by LU factorization is improved with respect to ILU4 factorization. This fact can also be inspected by comparing the total number of residual evaluations, Table 1.

The estimated condition numbers of the arising linear system for Newton iterations are shown in Fig (5). The condition number is shown as a function of drop in residual. The *Res0* or reference residual is the initial non-linear residual of the corresponding mesh computed based on the far field flow condition. Therefore the ratio of the non-linear residual at the end of each Newton iteration to *Res0*, 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 start-up phase. The rest of the reported condition numbers are associated to the linear system formed based on the solution at the end of successive Newton iterations. While the 2nd-order discretization graph initially has larger condition number, the condition number decreases gradually with convergence. In higher-order cases, initially, the condition number is smaller compare to the 2nd-order case, but it starts to grow with Newton iterations. In conclusion, the over all performance of the flow solver depends not only on the conditioning of the linear system but also how well the non-linear problem is represented by the linearization at each iteration. For higher-order discretizations the quality of the linearization is not as good as it is for the 2nd-order discretization and more outer or Newton iterations are needed for full convergence. The overall condition number of the preconditioned operator through most part of the Newton iterations is smaller for the LU compared to ILU4 confirming the quality (not efficiency) improvement of the preconditioning using LU factorization.

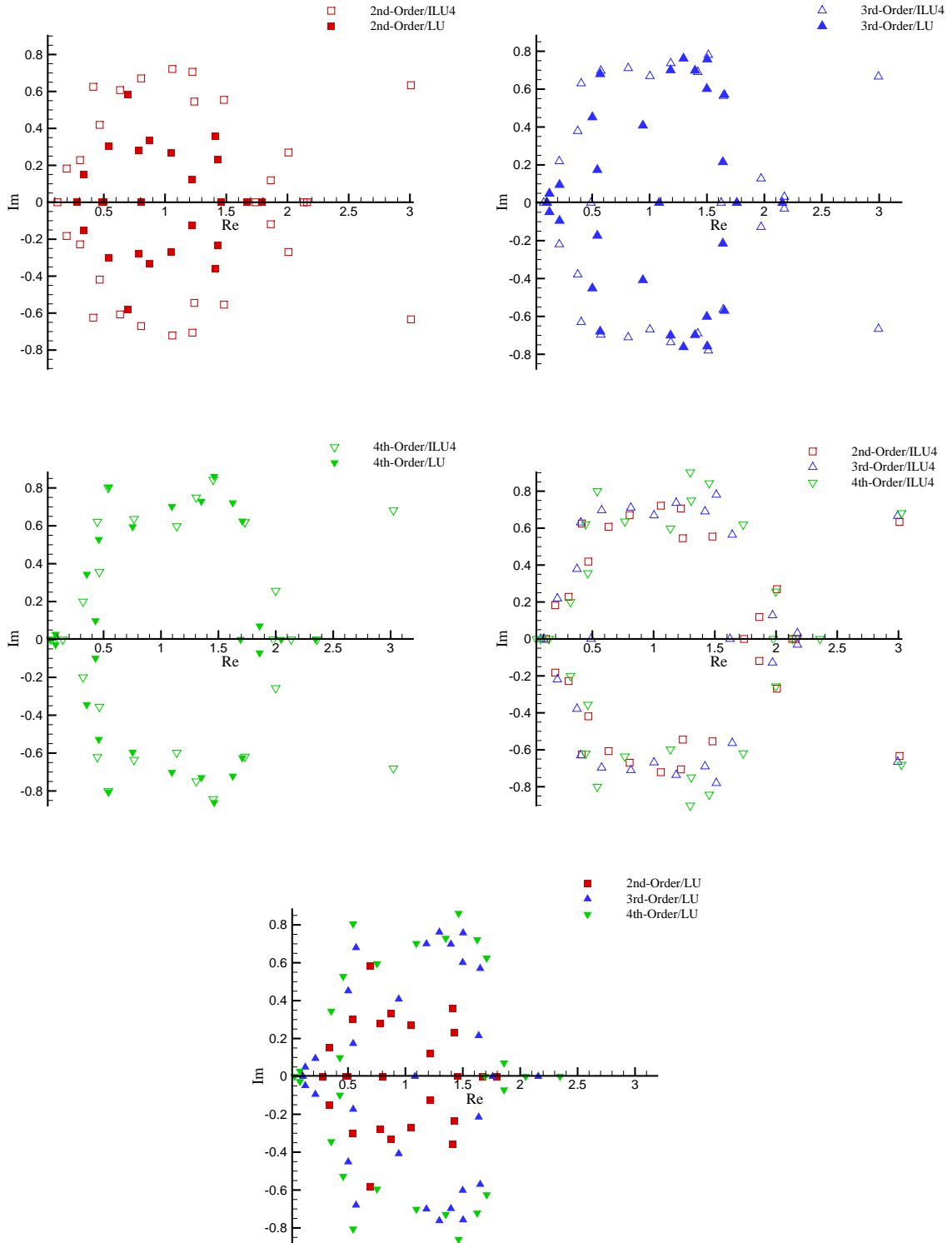


Figure 4. Eigenvalue pattern for the preconditioned system (converged solution), NACA 0012 (9931 CVs), $M = 0.63$, $\alpha = 2^0$

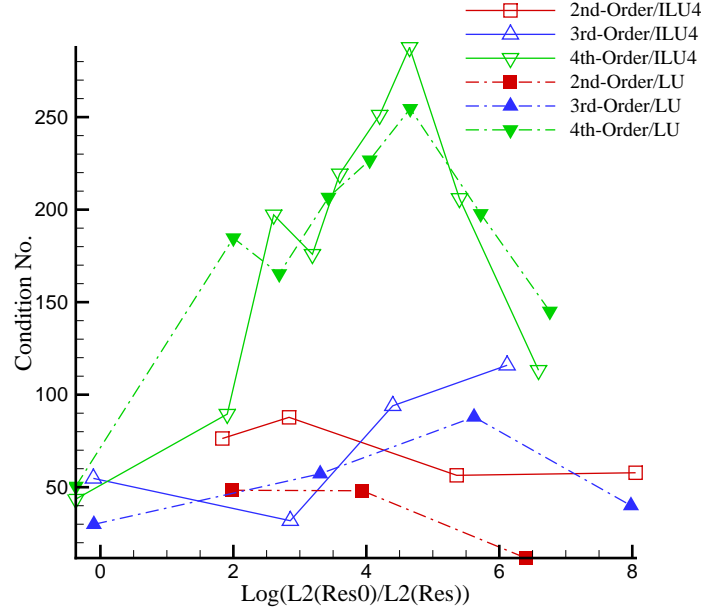


Figure 5. Condition No. of the preconditioned system (Newton iterations), NACA 0012 (9931 CVs), $M = 0.63$, $\alpha = 2^0$

IV.B. Transonic flow over NACA 0012, $M = 0.85$, $\alpha = 1^0$

For 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 (shock) in the solution. The methodology for handling discontinuities can increase the complexity of the problem. This is true especially for implicit schemes, where there could be a large change in solution update in each iteration and limiter values could have large oscillations. In the case of matrix-free approach which matrix-vector multiplication is computed through flux perturbation, any oscillatory behavior in limiter could severely degrade the solution convergence. All these facts amount to rising the complexity and difficulty in solving transonic flows.

The transonic flow around NACA 0012 at $M = 0.85$, $\alpha = 1.0^0$ is studied. An unstructured mesh with 5354 control volumes is employed for this test case. Flow is solved for all orders of accuracy using the Venkatakrishnan limiter with proper higher-order modification, section II.E. The applied 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} of the l_2 norm of the non-linear residual. For all test cases a subspace of 30 has been set and no restart is allowed. For the 4th-order, the subspace in the Newton phase has been reduced to 20 to partially reduce the cost of the 4th-order linear system solving, but after reaching the tolerance of 1×10^{-9} (non-linear residual) a subspace of 30 is employed again. 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 initial condition is the free stream flow at the far field (25 chords). The convergence is reached (solution process is stopped) when the L_2 norm of the non-linear density residual falls below 1×10^{-12} .

For transonic flow before switching to Newton iteration, the shock locations in the flow field and their strengths need to be captured relatively accurately otherwise Newton iterations would not decrease the residual of the non-linear problem effectively. This normally is achieved by reducing the residual of the non-linear residual by some orders, typically 1.3-2, with respect to the initial residual in the course of the

start-up process.

Several implicit pre-iterations are performed in the form of defect correction, before switching to Newton iteration. For the 2nd and 3rd-order start-up phases, pre-iterations in the form of defect correction continue until the residual of the non-linear problem drops 1.4 order below the residual of the initial condition. For the 2nd and 3rd 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 4th-order start-up, the CFL number is not increased above the value of 200 as increasing CFL does not help convergence when the linearization is not accurate. For the 4th-order Newton iterations, using an infinite time step causes ineffective linearization (considering the fixed subspace size) and limiter oscillation leading to inaccurate solution update. Since this slows the convergence, CFL=10,000 has been set for Newton-GMRES phase of the 4th-order.

Table 2. Convergence summary for NACA 0012 airfoil, $M = 0.85$, $\alpha = 1^\circ$

Test Case	No. of Res.	Time (Sec)	Work Unit	No. of Newton Itr.	Newton Phase Work Unit
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

The convergence summary is tabulated in the Table 2 and the convergence history is displayed in Fig (6). Using a constant subspace size without restart becomes less and less effective in solving the 4th-order linear system as the complexity of the system increases resulting in reducing the linearization effectiveness. Consequently more outer (Newton) iterations are needed to reduce the non-linear residual where the 4th-order discretization is employed. For this transonic case, the 3rd-order solution is about 1.3 times and the 4th-order solution is about 3.2 times more expensive than the 2nd-order solution, where ILU-4 is used as a preconditioning technique. Similar to the subsonic case, employing full factorization of the first-order Jacobian (as the preconditioner matrix) does not help the overall convergence performance.

The estimated condition numbers of the arising linear system for Newton iterations are shown in Fig (7); the condition number becomes larger as we increase the order of accuracy showing the rising complexity of the associated linear system. Compared to the subsonic case, in general, the conditioning of the transonic case is worse because of the existence of discontinuities in the flow field. For all orders of accuracy (2nd-order excluded) the preconditioned system has larger condition number where LU 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 3rd and 4th discretization orders the conditioning of the linear system rises suddenly for last Newton iterations right before reaching the converged solution.

The Eigenvalue patterns for all orders of accuracy are shown in Fig (8) for the preconditioned operator (at the last Newton iteration). The eigenvalues of the preconditioned system are scattered around one with some distance. The LU preconditioning compared to ILU-4 does a better eigenvalue clustering for the 2nd-order case but it loses its effectiveness as order of discretization increases. Also eigenvalues scatter with the larger radius (around one) for higher-order discretizations displaying the reduction in quality of preconditioning. Like the subsonic case, the higher the discretization order, the closer to the origin eigenvalues are located shifting the matrix toward singularity. This fact is particularly observed for the 4th-order discretization

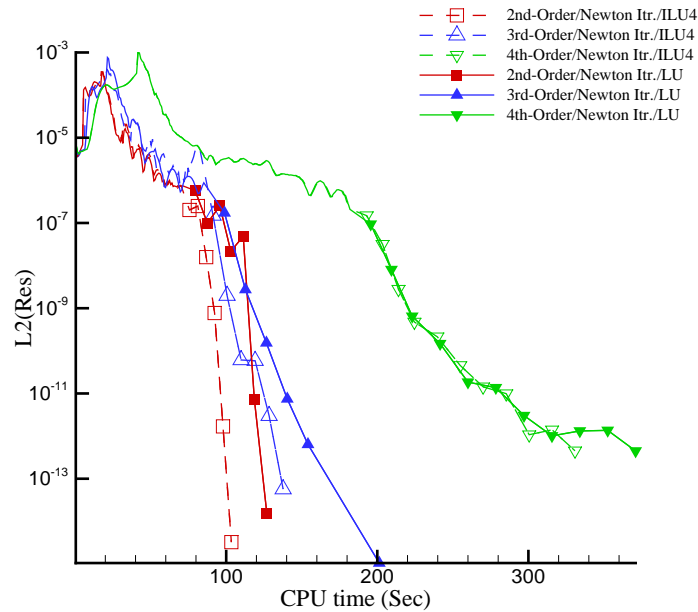


Figure 6. Convergence history, NACA 0012 (5354 CVs), $M = 0.85$, $\alpha = 1^0$

where eigenvalues are very close to zero.

V. Conclusion

An ILU preconditioned Newton-GMRES algorithm has been presented for the higher-order solution of inviscid compressible flows. The robustness and fast convergence of the approach have been demonstrated. The effect of the discretization order on preconditioning and convergence of the Newton-GMRES solver for subsonic and transonic test cases have been studied by inspecting the conditioning and eigenvalue patterns of the preconditioned system. The quality of the preconditioning, but not the overall solver efficiency, can be improved using LU factorization for smooth flows. However, the quality of the preconditioning by exact (LU) factorization can be degraded compared to ILU-4 for a higher-order transonic flow. 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. The incomplete factorization as a preconditioning method and lower-order Jacobian as a preconditioner matrix provide satisfactory convergence rate for all discretization orders. However, the 4th-order discretization convergence is considerably slower than the other two accuracy orders using the current matrix-free approach.

Acknowledgement

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

References

¹T. J. Barth. Analysis of Implicit Local Linearization Techniques for Upwind and TVD Algorithms. AIAA Paper 87-0595, 25th Aerospace Sciences Meeting and Exhibit, 1987.

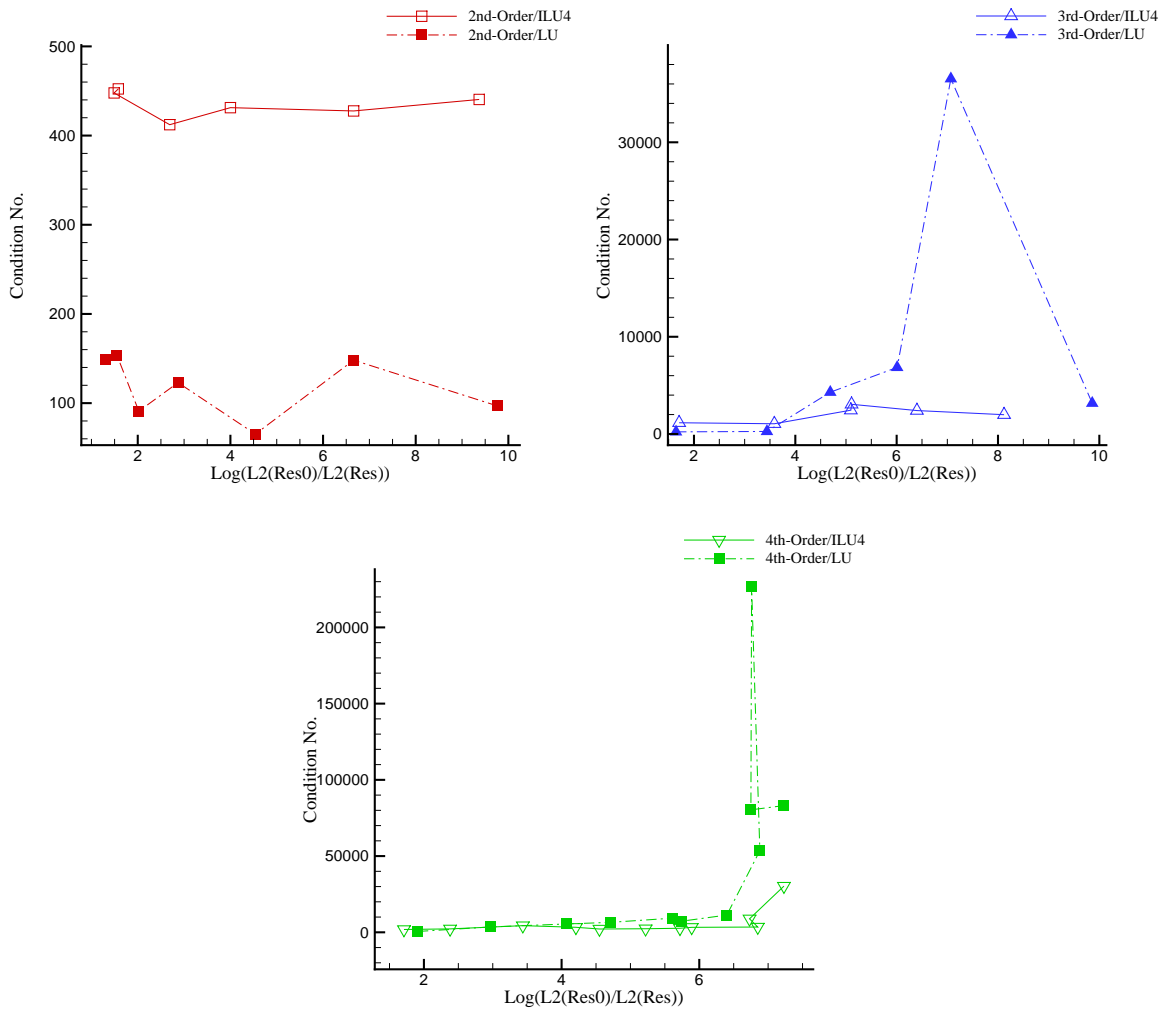


Figure 7. Condition No. of the preconditioned system (Newton iterations), NACA 0012 (5354 CVs), $M = 0.85$, $\alpha = 1^0$

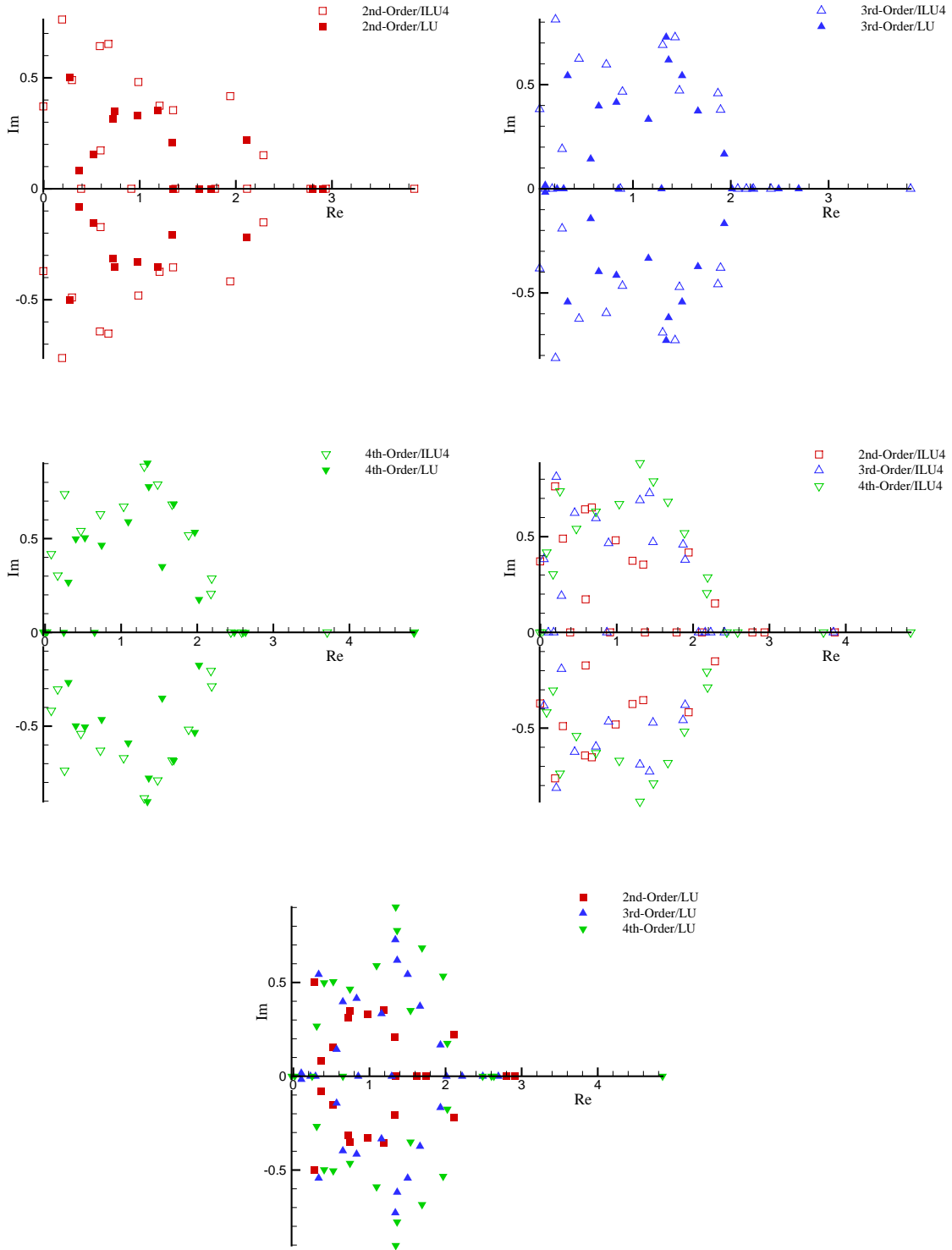


Figure 8. Eigenvalue pattern for the preconditioned system (converged solution), NACA 0012 (5354 CVs), $M = 0.85$, $\alpha = 1^{\circ}$

- ²T. J. Barth. Recent Development in High Order K-Exact Reconstruction on Unstructured Meshes. AIAA Paper 93-0668, 31th Aerospace Sciences Meeting and Exhibit, 1993.
- ³T. J. Barth and P. O. Frederickson. Higher-Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction. AIAA Paper 90-0013, 28th Aerospace Sciences Meeting and Exhibit, 1990.
- ⁴T. J. Barth and D. C. Jespersen. The Design and Application of Upwind Schemes on Unstructured Meshes. AIAA Paper 89-0366, 27th Aerospace Sciences Meeting and Exhibit, 1989.
- ⁵T. J. Barth and S. Linton. An Unstructured Mesh Newton Solver for Compressible Fluid Flow and its Parallel Implementation. AIAA Paper 95-0221, 33rd Aerospace Sciences Meeting and Exhibit, 1995.
- ⁶M. Benzi. Preconditioning Techniques for Large Linear Systems: A Survey. *Journal of Computational Physics*, 182:418–477, 2002.
- ⁷M. Blanco and D. Zingg. A Fast Solver for the Euler Equations on Unstructured Grids Using a Newton-GMRES Method. Number AIAA Paper 97-0331, 1997.
- ⁸G. Corliss, Ch. Faure, A. Griewank, L. Hascoet, and U. Naumann. *Automatic Differentiation of Algorithms From Simulation to Optimization*. Springer, 2002.
- ⁹M. Delanaye. *Polynomial Reconstruction Finite Volume Schemes for the Compressible Euler and Navier-Stokes Equations on Unstructured Adaptive Grids*. PhD thesis, Universite de Liege, Faculte des Sciences Appliquees, 1998.
- ¹⁰M. Delanaye and J. A. Essers. Quadratic-Reconstruction Finite-Volume Scheme for Compressible Flows on Unstructured Adaptive Grids. *AIAA Journal*, 35(4):631–639, 1997.
- ¹¹M. Delanaye, Ph. Geuzaine, and J. A. Essers. Compressible Flows on Unstructured Adaptive Grids. AIAA Paper 97-2120, 13th AIAA Computational Fluid Dynamics Conference, 1997.
- ¹²D. B. Kim and P. D. Orkwis. Jacobian Update Strategies for Quadratic and Near-Quadratic Convergence of Newton and Newton-Like Implicit Schemes. AIAA Paper 93-0878, 1993.
- ¹³L. M. Manzano, J. V. Lassaline, P. Wong, and D. W. Zingg. A Newton-Krylov Algorithm for the Euler Equations Using Unstructured Grids. AIAA Paper 2003-0274, 41th AIAA Aerospace Sciences Meeting and Exhibit, 2003.
- ¹⁴D. 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.
- ¹⁵A. Nejat and C. Ollivier-Gooch. A High-Order Accurate Unstructured GMRES Algorithm for Inviscid Compressible Flows. AIAA Paper 2005-5341, 17th AIAA Computational Fluid Dynamics Conference, 2005.
- ¹⁶A. Nejat and C. Ollivier-Gooch. A High-Order Accurate Unstructured Newton-Krylov Solver for Inviscid Compressible Flows. Number AIAA-2006-3711 in 36th AIAA Fluid Dynamics Conference, 2006.
- ¹⁷A. Nejat and C. Ollivier-Gooch. On Preconditioning of Newton-GMRES algorithm for a Higher-Order Accurate Unstructured Solver. 14th Annual Conference of the CFD Society of Canada, 2006.
- ¹⁸C. Ollivier-Gooch and M. Van Altena. A Higher-Order Accurate Unstructured Mesh Finite-Volume Scheme for the Advection-Diffusion Equation. *Journal of Computational Physics*, 181:729–752, 2002.
- ¹⁹O. Onur and S. Eyi. Effects of the Jacobian Evaluation on Newton’s Solution of the Euler Equations. *International Journal for Numerical Methods In Fluids*, 49:211–231, 2005.
- ²⁰P. D. Orkwis. Comparison of Newton’s and Quasi-Newton’s Method Solvers for Navier-Stokes Equations. *AIAA Journal*, 31:832–836, 1993.
- ²¹A. Pueyo and D. W. Zingg. Progress in NewtonKrylov Methods for Aerodynamic Calculations. AIAA Paper 97-0877, 35th Aerospace Sciences Meeting and Exhibit, 1997.
- ²²A. Pueyo and D. W. Zingg. Improvement to a Newton-Krylov Solver for Aerodynamic Flows. AIAA Paper 98-0619, 36th Aerospace Sciences Meeting and Exhibit, 1998.
- ²³S. De Rango and D. W. Zingg. Aerodynamic Computations Using a Higher-Order Algorithm. AIAA Paper 99-0167, 37th AIAA Aerospace Sciences Meeting and Exhibit, 1999.
- ²⁴S. De Rango and D. W. Zingg. Further Investigation of Higher-Order Algorithm for Aerodynamic Computations. AIAA Paper 2000-0823, 38th Aerospace Sciences Meeting and Exhibit, 2000.
- ²⁵P. Roe. Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. *Journal of Computational Physics*, 43:357–372, 1981.
- ²⁶P. Roe. Characteristic-Based Schemes for the Euler Equations. *Annual Review of Fluid Mechanics*, 18:337–365, 1986.
- ²⁷Y. Saad. *Iterative Methods for Sparse Linear Systems*. Siam, second edition, 2003.
- ²⁸M. Tadjouddine, S. A. Forth, and N. Qin. Elimination AD Applied to Jacobian Assembly for an Implicit Compressible CFD Solver. *International Journal for Numerical Methods in Fluids*, 47:1315–1321, 2005.
- ²⁹H. A. Van Der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press, 2003.
- ³⁰K. J. Vanden and P. D. Orkwis. Comparison of Numerical and Analytical Jacobians. *AIAA Journal*, 34(6):1125–1129, 1996.
- ³¹V. Venkatakrishnan. On the Accuracy of Limiters and Convergence to Steady State Solutions. AIAA Paper 93-0880, 31st Aerospace Sciences Meeting and Exhibit, 1993.
- ³²V. Venkatakrishnan. Convergence to Steady state Solutions of the Euler Equations on Unstructured Grids With Limiters. *Journal of Computational Physics*, 118:120–130, 1995.

³³V. Venkatakrishnan and D. Mavriplis. Implicit Solvers for Unstructured Meshes. *Journal of Computational Physics*, 105(83-91), 1993.

³⁴V. Venkatakrishnan and D. J. Mavriplis. Implicit Solvers for Unstructured Meshes. AIAA Paper 91-1537, 1991.

³⁵L. Wigton. Application of MACSYMA and Sparse Matrix Technology to Multielement Airfoil Calculations. AIAA Paper 87-1142, 1987.

³⁶D. W. Zingg, S. De Rango, M. Nemec, and T. H. Pulliam. Comparison of Several Spatial Discretizations for the Navier-Stokes Equations. *Journal of Computational Physics*, 160:683–704, 2000.