

# Matrix-Explicit GMRES for a Higher-Order Accurate Inviscid Compressible Flow Solver

Krzysztof Michalak\* and Carl F. Ollivier-Gooch†

*Department of Mechanical Engineering, University of British Columbia, Vancouver, BC, V6T 1Z4, Canada*

**Implicit methods for finite-volume schemes on unstructured grids typically rely on a matrix-free implementation of GMRES and an explicit first-order Jacobian for preconditioning. We show that it is possible to form the higher-order Jacobian explicitly at a reasonable computational cost. This is demonstrated for cases using both limited and unlimited reconstruction. The benefits resulting from improvements in preconditioning and the elimination of residual evaluations in the inner iterations of the matrix-free GMRES method are substantial. Computational results are presented for 2nd, 3rd, and 4th order accurate schemes. Overall computational cost for the matrix-explicit method is lower than the matrix-free method for all cases. The fourth-order matrix-explicit scheme is a factor of 2.5 to 3 faster than the matrix-free scheme while using only about 50% more memory.**

## I. Introduction

Higher-order discretizations on structured grids have been shown<sup>1,2</sup> to reduce computational effort for a given level of solution accuracy. Although higher-order finite-volume methods on unstructured grids are relatively well known,<sup>3-5</sup> means to efficiently converge the solution have not been adequately studied. For this reason, the advantages of higher-order methods have not been convincingly demonstrated.

An increasingly popular approach for solution convergence for second-order unstructured flow solvers is use of Newton-like methods with the GMRES<sup>6</sup> linear solver. Previous work has focused on using the matrix-free<sup>7</sup> approach. With this method the full flux Jacobian does not need to be explicitly constructed. Instead, the matrix-vector multiplies required by GMRES are computed by using Frechet derivatives. However, a preconditioning matrix is still required for good GMRES convergence; the typical choice here is to compute a Jacobian with first-order fill (though perhaps using second-order data to compute the entries in the matrix) and use an incomplete LU factorization of this matrix to precondition the linear system. Our group's previous work in developing an efficient high-order accurate flow solver extended this approach to high-order residual calculations, including careful study of preconditioning methods.<sup>8,9</sup>

While those studies were successful in dramatically reducing CPU time requirements for high-order schemes, especially for third-order accuracy, there are also clear indications that the fourth-order scheme in particular is poorly preconditioned by the reduced-order Jacobian (in this case, first-order fill, with entries computed using fourth-order data). The research described in this paper explores the possibility of forming and exploiting the full order Jacobian. On the one hand, we expect that preconditioning will be much better if we use the full order Jacobian. Also, having the Jacobian on hand will eliminate all the flux evaluations done within GMRES inner iterations in matrix-free schemes; this savings in CPU time should help offset the cost of computing the Jacobian. The major drawback to forming the full order Jacobian is the memory requirements. Whether overall CPU time would decrease with the full order Jacobian was an unknown at the beginning of the work. This paper will describe our research on this topic to date. We will demonstrate how to efficiently construct the full flux Jacobian and compare the overall performance of the matrix-explicit and matrix-free schemes for subsonic and transonic compressible flow problems.

---

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

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

## II. Algorithm Description

### II.A. High-Order Accurate Solution Reconstruction

As in previous work,<sup>10</sup> the high-order accurate reconstruction is obtained by formulating and solving a least-squares problem.

To obtain an accurate estimate of the solution at flux quadrature points, we represent the solution within each control volume by the Taylor series expansion

$$\begin{aligned} u_i^R(x - x_i, y - y_i) &= u|_i + \frac{\partial u}{\partial x}\Big|_i (x - x_i) + \frac{\partial u}{\partial y}\Big|_i (y - y_i) + \\ &\frac{\partial^2 u}{\partial x^2}\Big|_i \frac{(x - x_i)^2}{2} + \frac{\partial^2 u}{\partial x \partial y}\Big|_i (x - x_i)(y - y_i) + \\ &\frac{\partial^2 u}{\partial y^2}\Big|_i \frac{(y - y_i)^2}{2} + \dots \end{aligned}$$

where  $u_i$  is the value of the reconstructed solution and  $\frac{\partial^{k+l} u_i}{\partial x^k \partial y^l}$  are its derivatives at the reference point  $(x_i, y_i)$  of control volume  $i$ . The coefficients of the polynomial are computed so that the mean value of the solution in the control volume is conserved and the reconstruction approximates nearby control volume averages.

Conservation of the mean within a control volume is satisfied when

$$\begin{aligned} \bar{u}_i = \frac{1}{A_i} \int_{V_i} u_i^R dA &\equiv u|_i + \frac{\partial u}{\partial x}\Big|_i \bar{x}_i + \frac{\partial u}{\partial y}\Big|_i \bar{y}_i + \\ &\frac{\partial^2 u}{\partial x^2}\Big|_i \frac{\bar{x}_i^2}{2} + \frac{\partial^2 u}{\partial x \partial y}\Big|_i \bar{x}_i \bar{y}_i + \\ &\frac{\partial^2 u}{\partial y^2}\Big|_i \frac{\bar{y}_i^2}{2} + \dots \end{aligned} \quad (1)$$

where

$$\overline{x^n y^m}_i \equiv \frac{1}{A_i} \int_{V_i} (x - x_i)^n (y - y_i)^m dA$$

As we shall see, this mean constraint can be eliminated analytically from the least-squares system. The terms used to construct the reconstruction least-squares problem have the form

$$\begin{aligned} \frac{1}{A_j} \int_{V_j} u_i^R(\vec{x} - \vec{x}_i) dA &= u|_i + \frac{\partial u}{\partial x}\Big|_i \widehat{x}_{ij} + \frac{\partial u}{\partial y}\Big|_i \widehat{y}_{ij} \\ &+ \frac{\partial^2 u}{\partial x^2}\Big|_i \frac{\widehat{x}_{ij}^2}{2} + \frac{\partial^2 u}{\partial x \partial y}\Big|_i \widehat{x}_{ij} \widehat{y}_{ij} + \frac{\partial^2 u}{\partial y^2}\Big|_i \frac{\widehat{y}_{ij}^2}{2} + \dots \end{aligned} \quad (2)$$

The geometric terms in this equation are of the general form

$$\begin{aligned} \widehat{x^n y^m}_{ij} &\equiv \frac{1}{A_j} \int_{V_j} ((x - x_j) + (x_j - x_i))^n \cdot ((y - y_j) + (y_j - y_i))^m dA \\ &= \sum_{l=0}^m \sum_{k=0}^n \frac{m!}{l!(m-l)!} \frac{n!}{k!(n-k)!} (x_j - x_i)^k \cdot (y_j - y_i)^l \cdot \overline{x^{n-k} y^{m-l}}_j \end{aligned}$$

The resulting least-squares problem to be solved for each solution variable in each control volume takes the form

$$\begin{bmatrix} 1 & \bar{x}_i & \bar{y}_i & \frac{\bar{x}_i^2}{2} & \bar{x}_i \bar{y}_i & \frac{\bar{y}_i^2}{2} & \dots \\ w_{i1} & w_{i1} \widehat{x}_{i1} & w_{i1} \widehat{y}_{i1} & w_{i1} \frac{\widehat{x}_{i1}^2}{2} & w_{i1} \widehat{x}_{i1} \widehat{y}_{i1} & w_{i1} \frac{\widehat{y}_{i1}^2}{2} & \dots \\ w_{i2} & w_{i2} \widehat{x}_{i2} & w_{i2} \widehat{y}_{i2} & w_{i2} \frac{\widehat{x}_{i2}^2}{2} & w_{i2} \widehat{x}_{i2} \widehat{y}_{i2} & w_{i2} \frac{\widehat{y}_{i2}^2}{2} & \dots \\ w_{i3} & w_{i3} \widehat{x}_{i3} & w_{i3} \widehat{y}_{i3} & w_{i3} \frac{\widehat{x}_{i3}^2}{2} & w_{i3} \widehat{x}_{i3} \widehat{y}_{i3} & w_{i3} \frac{\widehat{y}_{i3}^2}{2} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\ w_{iN} & w_{iN} \widehat{x}_{iN} & w_{iN} \widehat{y}_{iN} & w_{iN} \frac{\widehat{x}_{iN}^2}{2} & w_{iN} \widehat{x}_{iN} \widehat{y}_{iN} & w_{iN} \frac{\widehat{y}_{iN}^2}{2} & \dots \end{bmatrix} \begin{pmatrix} u \\ \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{1}{2} \frac{\partial^2 u}{\partial x^2} \\ \frac{\partial^2 u}{\partial x \partial y} \\ \frac{1}{2} \frac{\partial^2 u}{\partial y^2} \\ \vdots \end{pmatrix}_i = \begin{pmatrix} \bar{u}_i \\ w_{i1} \bar{u}_1 \\ w_{i2} \bar{u}_2 \\ w_{i3} \bar{u}_3 \\ \vdots \\ w_{iN} \bar{u}_N \end{pmatrix} \quad (3)$$

where the weights used are the inverse of distance between control-volume reference locations squared:

$$w_{ij} = \frac{1}{|\vec{x}_j - \vec{x}_i|^2} \quad (4)$$

The first equation in this linear system is the mean constraint, which can be removed by Gauss elimination, leaving an unconstrained least-squares problem.

In previous work,<sup>8</sup> the least-squares problem was solved at each flux evaluation using QR factorization.<sup>11</sup> However, since the matrix contains only geometric terms, it is identical for each solution variable in a given control volume and does not change between iterations. Therefore, substantial savings in computational time can be achieved by precomputing and storing the pseudo-inverse of the reconstruction matrix for each control volume. To obtain the pseudo-inverse in a numerically stable manner, the present work uses the singular value decomposition (SVD)<sup>11</sup> method. Given an SVD of a reconstruction matrix  $A$ , the pseudo-inverse can easily be obtained

$$\begin{aligned} A &= U\Sigma V^T \\ A^+ &= V\Sigma^+U^T \end{aligned} \quad (5)$$

Once this precomputation has been performed, the reconstruction coefficients in each control volume can be obtained by performing a matrix-vector multiplication; the number of columns in this matrix equals the number of control volumes in the reconstruction stencil, while the number of rows equals the number of required reconstruction coefficients. From these coefficients the reconstructed values of the solution can easily be computed at the flux quadrature points.

## II.B. Flux Calculation and Integration

The finite-volume formulation of the inviscid Euler equations for a control volume  $\Omega$  with boundary  $\partial\Omega$  takes the form

$$\frac{\partial}{\partial t} \int_{\Omega} \mathbf{u} dV + \oint_{\partial\Omega} (\mathbf{f} \cdot \mathbf{n}) dS = 0$$

The conserved variables  $\mathbf{u}$  are the density, two components of momentum and total energy. The vector  $\mathbf{f}$  represents the inviscid flux. A median-dual control volume is associated with each vertex in the triangular mesh. The contour integral is discretized using Gauss quadrature over the dual edges. One Gauss point per edge is used for the second-order scheme. Two Gauss points are used for the third and fourth order schemes. The procedure to compute the numerical flux  $\mathbf{f}$  is:

1. Translate the control-volume average solution from conserved variables (density, momentum, energy) to primitive variables (density, velocity, and pressure).
2. Compute the reconstruction coefficients using the precomputed pseudo-inverses of the reconstruction matrices.
3. At each Gauss point, compute the left and right solutions using the reconstruction coefficients from both adjacent control volumes.
4. The reconstructed solution at each Gauss point from both adjacent control volumes is used as input to an approximate Riemann solver. In the present work, the AUSM+-up<sup>12</sup> scheme is used.

## II.C. Implicit Time Advance

The spatial discretization leads to a system of ordinary differential equations:

$$\frac{dU}{dt} + R(U) = 0$$

where  $U$  are the control volume averages of the solution.  $R$  is the discrete flux integral which we will call the residual. An implicit scheme is obtained by the following discretization :

$$\begin{aligned} \left( \frac{I}{\Delta t} + \frac{\partial R}{\partial U} \right) \delta U &= -R \\ U^{n+1} &= U^n + \delta U \end{aligned}$$

Since we are strictly concerned with the steady-state solution,  $\Delta t$  could be taken as infinity and the equation simplified to Newton's method. In most practical cases this is not possible due to the highly non-linear nature of the residual. However, the time-step can be gradually increased to infinity as the solution process proceeds. A local time-step is used where  $\Delta t$  is taken as proportional to the control-volume size. The linear system at each time-step is solved approximately using a preconditioned GMRES solver.

## II.D. Matrix-Free GMRES

The GMRES algorithm only requires the result of matrix-vector products. These can be approximated without forming the matrix explicitly by:

$$\frac{\partial R}{\partial U}(u) a = \frac{R(u + h \cdot a) - R(u)}{h}$$

where  $h$  is a small value. This scheme requires one flux evaluation per inner GMRES iteration plus one flux evaluation per outer iteration.

## II.E. Forming the Higher-Order Jacobian Matrix

### II.E.1. Without Limiters

The analytic Jacobian can be represented explicitly as

$$\frac{\partial FluxInt}{\partial CVars} = \frac{\partial FluxInt}{\partial Flux} \frac{\partial Flux}{\partial RecSol} \frac{\partial RecSol}{\partial RecCoef} \frac{\partial RecCoef}{\partial PVars} \frac{\partial PVars}{\partial CVars}$$

where  $FluxInt$  is the flux integral,  $Flux$  is the numerical flux,  $RecSol$  is the reconstructed solution at Gauss points,  $RecCoef$  are the reconstruction coefficients,  $PVars$  are the control volume averages of the primitive variables, and  $CVars$  are the control volume averages of the conserved variables. To compute the Jacobian, the following procedure is used at each time-step:

1.  $\frac{\partial PVars}{\partial CVars}$  is computed for each control volume and stored. This is the standard Jacobian for the change of variables from  $(\rho \ u \ v \ P)^T$  to  $(\rho \ \rho u \ \rho v \ E)^T$ .
2. For each Gauss point, do the following for each of the two adjacent control volumes:
  - (a)  $\frac{\partial RecSol}{\partial PVars} = \frac{\partial RecSol}{\partial RecCoef} \frac{\partial RecCoef}{\partial PVars}$  is computed. The  $\frac{\partial RecCoef}{\partial PVars}$  term is simply the pseudo-inverse of the reconstruction matrix precomputed in Equation 5, while the  $\frac{\partial RecSol}{\partial RecCoef}$  term is a geometric term that depends on the location of the Gauss quadrature point. Although the entire  $\frac{\partial RecSol}{\partial PVars}$  term is purely geometric and could be precomputed and stored, this would dramatically increase the memory requirements since, unlike  $\frac{\partial RecCoef}{\partial PVars}$ , this term is unique for each Gauss point of every control volume. Since the computational efficiency gains would be small, this increase in memory is not worthwhile.
  - (b)  $\frac{\partial Flux}{\partial RecSol}$ , the Jacobian of the AUSM+up flux, is computed. Due to its complexity, the code required to compute this term was automatically generated using the ADIC<sup>13</sup> automatic differentiation tool.
  - (c) The product  $\frac{\partial Flux}{\partial PVars} = \frac{\partial Flux}{\partial RecSol} \frac{\partial RecSol}{\partial PVars}$  is computed efficiently by taking advantage of the sparsity of the reconstruction terms that is due to the lack of coupling between solution variables.  $\frac{\partial Flux}{\partial PVars}$  couples all solution variables in the reconstruction stencil.
  - (d) The product  $\frac{\partial Flux}{\partial CVars} = \frac{\partial Flux}{\partial PVars} \frac{\partial PVars}{\partial CVars}$  is computed. Since  $\frac{\partial Flux}{\partial PVars}$  couples all solution variables in the reconstruction stencil, this step is computationally intensive.
  - (e)  $\frac{\partial FluxInt}{\partial CVars} = \frac{\partial FluxInt}{\partial Flux} \frac{\partial Flux}{\partial CVars}$  is the contribution to the flux integral due to one side of this Gauss point. This component is computed by using the appropriate Gauss integration weight. The result is added to the total flux Jacobian.

The sparse analytic Jacobian is found once for every Newton iteration and used to produce the matrix-vector products needed by the GMRES solver. These products require fewer operations to compute than the flux evaluation needed by the matrix-free method. They are also easy to optimize to fully benefit from caching and vector operations that may be available on the processor.

### II.E.2. With Limiters

For transonic and supersonic flows, a limiter must be used to eliminate overshoots in the solution. We will describe a means of including the effect of Venkatakrishnan’s limiter<sup>14</sup> in the Jacobian matrix for second-order reconstructions and our adaptation<sup>15</sup> of the limiter to higher-order accurate reconstruction. The limiting procedure consists in computing the maximum limiting factor  $\phi \in [0, 1]$  which eliminates overshoots. The limiter is then applied as:

$$RecCoef = \begin{bmatrix} 1 & (\phi - 1)\bar{x}_i & (\phi - 1)\bar{y}_i & (\phi - 1)\bar{x}_i^2 & (\phi - 1)\bar{x}_i\bar{y}_i & (\phi - 1)\bar{y}_i^2 & \cdots \\ & \phi & & & & & \cdots \\ & & \phi & & & & \cdots \\ & & & \phi & & & \cdots \\ & & & & \phi & & \cdots \\ & & & & & \phi & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{pmatrix} u \\ \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{1}{2} \frac{\partial^2 u}{\partial x^2} \\ \frac{\partial^2 u}{\partial x \partial y} \\ \frac{1}{2} \frac{\partial^2 u}{\partial y^2} \\ \vdots \end{pmatrix}_{unlmt} \quad (6)$$

where *unlmt* represents the coefficients found by using unlimited reconstruction.

The Jacobian is therefore computed exactly as for the case without limiters except for the the  $\frac{\partial RecCoef}{\partial PVars}$  term which is computed as:

$$\frac{\partial RecCoef}{\partial PVars} = \frac{\partial RecCoef}{\partial \phi} \frac{\partial \phi}{\partial PVars} + \frac{\partial RecCoef}{\partial RecCoef_{unlmt}} \frac{\partial RecCoef_{unlmt}}{\partial PVars}$$

where  $RecCoef_{unlmt}$  are the coefficients of the unlimited reconstruction. To balance computational time and memory use, the  $\frac{\partial \phi}{\partial PVars}$  term is computed and stored once for each flow variable in each control-volume in the same way as  $\frac{\partial PVars}{\partial CVars}$  was for the unlimited case. The computation of the remaining terms and the assembly of  $\frac{\partial RecCoef}{\partial PVars}$  is performed “on the fly” while iterating over Gauss points:

- $\frac{\partial RecCoef}{\partial \phi}$  term is easily found based on the linear relationship in Equation 6.
- $\frac{\partial RecCoef}{\partial RecCoef_{unlmt}}$  term is also evident from Equation 6.
- $\frac{\partial RecCoef_{unlmt}}{\partial PVars}$  is the pseudo-inverse of the reconstruction matrix precomputed in Equation 5.

The limiter value  $\phi$  computed using Venkatakrishnan’s scheme<sup>14</sup> or the scheme presented in our other work,<sup>15</sup> requires the control-volume average value, the neighboring maximum and minimum control-volume average values, and the unlimited reconstructed value of the solution at the Gauss points. During the limiting procedure, the minimum value of  $\phi$  from all the Gauss points is selected. For the purposes of the Jacobian, the choice of which Gauss point produced the smallest value of  $\phi$  is “frozen”. Similarly, the control-volume which contains the minimum or maximum control-volume average is also preselected. The  $\frac{\partial \phi}{\partial PVars}$  term can therefore be decomposed as:

$$\frac{\partial \phi}{\partial PVars} = \frac{\partial \phi}{\partial CVave} \frac{\partial CVave}{\partial PVars} + \frac{\partial \phi}{\partial NeighMin} \frac{\partial NeighMin}{\partial PVars} + \frac{\partial \phi}{\partial NeighMax} \frac{\partial NeighMax}{\partial PVars} + \frac{\partial \phi}{\partial RecVal} \frac{\partial RecVal}{\partial PVars}$$

where *CVave*, *NeighMin*, and *NeighMax* are simply select components of *PVar* and *RecVal* is the unlimited reconstructed value of the solution at the Gauss point producing the smallest value of  $\phi$ . The components  $\frac{\partial \phi}{\partial CVave}$ ,  $\frac{\partial \phi}{\partial NeighMin}$ ,  $\frac{\partial \phi}{\partial NeighMax}$ , and  $\frac{\partial \phi}{\partial RecVal}$  are obtained by computing the derivative of the limiter function. The component  $\frac{\partial RecVal}{\partial PVars}$  is one row of  $\frac{\partial RecSol}{\partial RecCoef_{unlmt}} \frac{\partial RecCoef_{unlmt}}{\partial PVars}$  and is found as discussed in the previous section.

## II.F. Preconditioning

Since the rate of convergence of the GMRES method is strongly dependent on the condition number of the matrix, preconditioning is used to alter the spectrum and hence accelerate the convergence rate of iterative technique. Left preconditioning is applied by:

$$M^{-1}Ax = M^{-1}b$$

where  $M^{-1}$  is an approximate inverse of the preconditioning matrix  $M \approx A$ . A common approach<sup>7,16</sup> is to use the flux Jacobian of the first-order scheme for  $M$  and to use ILU decomposition to form the approximate inverse. This approach has the advantages of being easier to compute and requiring less memory than using the full-order accurate Jacobian. To form the preconditioning matrix  $M$ , a procedure similar to that presented in Section II.E is used except that the terms  $\frac{\partial Flux}{\partial RecSol}$   $\frac{\partial RecSol}{\partial RecCoeff}$  are eliminated. This method is used for the matrix-free results presented in the present work.

Since the high-order Jacobian already needs to be computed in the matrix-explicit method, its ILU decomposition can easily be used as a preconditioner. The increase in memory use can be partially mitigated by using a lower level of fill; as we will show, our results demonstrate that even with low levels of fill, the matrix-explicit method is much better conditioned than the matrix-free method.

## II.G. Startup

Since Newton-like quadratic convergence cannot be expected during initial time-stepping, computing the full-order Jacobian or using a matrix-free method is unnecessarily expensive at that stage. Instead, the first-order Jacobian is used on the left-hand-side and the full-order residual is used on the right-hand-side.

For robustness, our scheme adjusts the magnitude of the update vector  $\delta U$  by using a cubic line-search. If the result indicates that the magnitude did not need to be reduced the time-step at the next iteration is doubled. Otherwise, the next time-step is reduced based on the adjusted relative size of the update vector. This scheme requires additional residual evaluations at each time-step due to the line-search. However, since the time-step is aggressively increased when appropriate, the overall number of residual evaluations is not adversely affected. Furthermore, the convergence process is very robust and does not require manual adjustments to the startup procedure to obtain good performance on a variety of test cases.

In the present work, we use time-stepping with the first-order Jacobian until a relative residual drop of  $5 \times 10^{-3}$  is achieved. We then switch to using an infinite time-step with either the matrix-free GMRES or full-order matrix-explicit GMRES. If the linear system was solved to machine precision, the resulting scheme would be expected to provide quadratic convergence. In practice, we achieve very rapid convergence by solving the linear system to a relative residual of  $10^{-2}$ .

# III. Results

## III.A. Subsonic

The test case consists of subsonic flow over a NACA 0012 airfoil at Mach 0.5 and an angle of attack of 1 degree. The computational mesh consists of 4656 control-volumes and is shown in Figure 1. The pressure from the steady-state solution using the second- and fourth-order schemes is shown in Figure 2; the fourth-order solution clearly has a much smaller entropy layer along the upper surface of the airfoil, as well as much smaller jumps in the solution at control volume boundaries.

### III.A.1. Computational Time

We begin by comparing the relative CPU time needed to compute a flux integral, a first-order Jacobian, and a higher-order Jacobian. The results are given in Table 1. The large increase in computational time needed for flux and Jacobian evaluations for the 3rd order scheme relative to the 2nd order scheme can in large part be attributed to the doubling of the number of Gauss points. Since the 4th order scheme uses the same number of Gauss points as the 3rd order scheme, a smaller increase in flux and Jacobian evaluation times is observed. For flux evaluations, this increase is purely due to the cost of the reconstruction procedure. For the Jacobian, the increased cost of matrix-matrix products needed for its assembly is important.

Next, we compare the relative effectiveness of the ILU decomposition of the first-order and higher-order Jacobians. The average number of inner GMRES iterations needed per Newton iteration to obtain a relative

residual drop of  $10^{-2}$  is shown in Table 2. The results indicate that the first-order Jacobian is a reasonably good preconditioner for the second order scheme if ILU with enough levels of fill is used. However, this preconditioner does a poor job for the 3rd and 4th order schemes. Using the full-order Jacobian for the preconditioner, the convergence properties of the higher-order schemes is comparable to that of the second-order scheme and a lower level of fill can be used.

Finally, the total run time to obtain a steady-state solution is presented in Table 3. The residual versus computational time plot is shown in Figure 3. For this plot, the matrix-free method is preconditioned using the ILU(4) decomposition of the first-order Jacobian while the matrix-explicit scheme is preconditioned using ILU(0) of the full-order Jacobian. The same startup procedure is used in both cases. The results show that the matrix-explicit method is faster for all orders of accuracy. The benefit of the matrix-explicit method increases with the order of accuracy. This can be attributed to two main factors. Firstly, as was shown in Table 2, the 1st-order preconditioner is less effective for the higher-order schemes. Secondly, the higher-order schemes spend a relatively smaller portion of time in the startup stage which is identical for the two schemes.

### III.A.2. Memory Requirement

The major components contributing to the memory requirement for both matrix-explicit and matrix-free methods are:

- *The pseudo-inverse of the reconstruction matrix.* To avoid solving the least-squares problem at each flux evaluation, these matrices need to be precomputed and stored for each control-volume.
- *The Jacobian.* For the matrix-free scheme, this will always be the first-order Jacobian. For the matrix-explicit scheme, the higher-order Jacobian will inevitably have more fill and require more storage.
- *ILU decomposition of the Jacobian.* The memory required for this depends not only on the fill of the Jacobian but also by the additional fill due to the decomposition which increases with  $n$  for  $ILU(n)$ .
- *Krylov subspace.* The maximum number of inner GMRES iterations required to solve a Newton iterations determines the memory requirement of the Krylov solver.

The memory required per control-volume for the subsonic case is presented in Table 4. The additional memory required by the matrix-explicit scheme is due to the increased fill of the Jacobian and resulting preconditioning matrix. However, this is partially offset by the lower fill-ratio of the ILU decomposition and the reduced memory use of the Krylov solver. On average, the matrix-explicit method required 59% more memory than the matrix-free method.

### III.B. Transonic

The test case consists of transonic flow over a NACA 0012 airfoil at Mach 0.8 and an angle of attack of 1.25 degrees. The computational mesh used was the same as for the subsonic case. For the second-order results, Venkatakrishnan's Limiter<sup>14</sup> was used while the third- and fourth-order results used our variation<sup>15</sup> of this limiter. The pressure along the airfoil produced by the second- and fourth-order schemes is shown in Figure 4. The pressure from the steady-state solution using the second- and fourth-order schemes is shown in Figure 2; the most visible difference in the solutions is the sharpness of the lower-surface shock in the fourth-order solution. The startup procedure used for the second- and fourth-order schemes was the same as for the subsonic case. For the third-order case convergence during the startup phase stalled and it was necessary to use time-stepping together with the matrix-free or matrix-explicit schemes rather than transitioning to a Newton approach.

The relative CPU time needed to compute a flux integral, a first-order Jacobian, and a higher-order Jacobian are given in Table 5. The flux integral is considerably more costly than in the subsonic case due to the computation of the limiter value  $\phi$ . Although the Jacobian is also more costly to compute, *relative to a flux evaluation* it is cheaper than in the subsonic case. Table 6 shows a similar trend to the subsonic case in the number of inner iterations required per outer iterations. The relatively good conditioning of the third-order scheme is partially due to the use of a finite timestep rather than the Newton method.

The total runtime presented in Table 7 and the convergence plot shown in Figure 5 exhibit similar trends to the subsonic case for the second- and fourth-order schemes. The performance of the third order scheme

suffers due to the need for continued use of timestepping after startup. However, the performance advantages of the matrix-explicit scheme relative to the matrix-free scheme remain present.

The memory requirement for the transonic case are very similar to that of the subsonic case. For brevity, these values have not been tabulated.

## IV. Conclusion

We have shown how to efficiently assemble the higher-order accurate Jacobian of inviscid flow on unstructured grids for use with a Newton-Krylov solver. The increased computational cost of forming this Jacobian is more than compensated for by improved preconditioning and the elimination of the residual evaluations associated with the matrix-free method. The gains in computational efficiency were greatest for the fourth order scheme and equally prevalent in subsonic and transonic solutions.

## Acknowledgments

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

## References

- <sup>1</sup>Zingg, D., De Rango, S., Nemeec, M., and Pulliam, T., "Comparison of Several Spatial Discretizations for the Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 160, 2000, pp. 683–704.
- <sup>2</sup>De Rango, S. and Zingg, D. W., "Higher-order spatial discretization for turbulent aerodynamic computations," *American Institute of Aeronautics and Astronautics Journal*, Vol. 39, No. 7, July 2001, pp. 1296–1304.
- <sup>3</sup>Barth, T. J. and Frederickson, P. O., "Higher Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction," AIAA paper 90-0013, Jan. 1990.
- <sup>4</sup>Barth, T. J., "Aspects of Unstructured Grids and Finite-Volume Solvers for the Euler and Navier-Stokes Equations," *Lecture Series 1994-05*, von Karman Institute for Fluid Dynamics, Rhode-Saint-Genèse, Belgium, March 1994.
- <sup>5</sup>Delanaye, M. and Essers, J. A., "Quadratic-reconstruction finite volume scheme for compressible flows on unstructured adaptive grids," *American Institute of Aeronautics and Astronautics Journal*, Vol. 35, No. 4, April 1997, pp. 631–639.
- <sup>6</sup>Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal of Scientific and Statistical Computing*, Vol. 7, No. 3, July 1986, pp. 856–869.
- <sup>7</sup>Barth, T. J. and Linton, S. W., "An Unstructured Mesh Newton Solver for Compressible Fluid Flow and Its Parallel Implementation," AIAA paper 95-0221, Jan. 1995.
- <sup>8</sup>Nejat, A. and Ollivier-Gooch, C., "A High-Order Accurate Unstructured GMRES Algorithm for Inviscid Compressible Flows," *Proceedings of the Seventeenth AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, June 2005.
- <sup>9</sup>Nejat, A. and Ollivier-Gooch, C., "On Preconditioning of Newton-GMRES algorithm for a Higher-Order Accurate Unstructured Solver," *Proceedings of the Fourteenth Annual Conference of the Computational Fluid Dynamics Society of Canada*, cfdsc, 2006.
- <sup>10</sup>Ollivier-Gooch, C. F. and Van Altena, M., "A High-order Accurate Unstructured Mesh Finite-Volume Scheme for the Advection-Diffusion Equation," *Journal of Computational Physics*, Vol. 181, No. 2, 2002, pp. 729–752.
- <sup>11</sup>Golub, G. H. and Loan, C. F. V., *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 1983.
- <sup>12</sup>Liou, M.-S., "A further development of the AUSM+ scheme towards robust and accurate solutions for all speeds," *Proceedings of the Sixteenth AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, June 2003, AIAA paper 2003-4116.
- <sup>13</sup>Bischof, C., Roh, L., and Mauer, A., "ADIC — An Extensible Automatic Differentiation Tool for ANSI-C," Preprint ANL/MCS-P626-1196, Argonne National Laboratory, 1996.
- <sup>14</sup>Venkatakrishnan, V., "On the Accuracy of Limiters and Convergence to Steady-State Solutions," AIAA paper 93-0880, Jan. 1993.
- <sup>15</sup>Michalak, K. and Ollivier-Gooch, C., "Limiters for Unstructured Higher-Order Accurate Solutions of the Euler Equations," *Forty-sixth Aerospace Sciences Meeting*, 2008.
- <sup>16</sup>Pueyo, A. and Zingg, D. W., "Improvements to a Newton-Krylov Solver for Aerodynamic Flows," AIAA 98-0619, Jan. 1998.

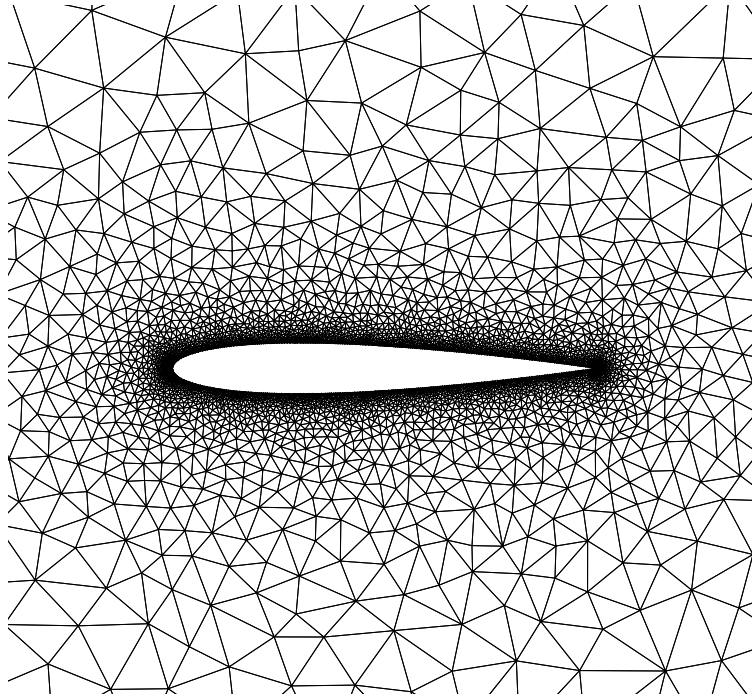


Figure 1. Computational mesh consisting of 4656 control volumes

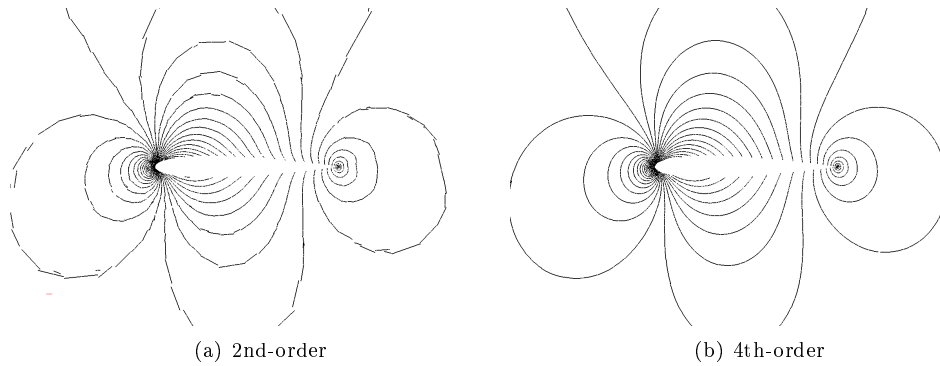


Figure 2. Pressure isolines for subsonic case.

Order	Flux	1st Order Jacobian		Full Order Jacobian	
	seconds	seconds	relative to flux	seconds	relative to flux
1	0.014	0.236	17.06		
2	0.023	0.236	10.20	0.320	13.82
3	0.051	0.236	4.59	1.231	23.91
4	0.079	0.236	3.00	1.305	16.58

Table 1. Relative computational time of flux and Jacobian evaluation without limiter for subsonic case.

Order	1st Order Preconditioner		Full Order Preconditioner	
	ILU(1)	ILU(4)	ILU(0)	ILU(1)
2	66.0	30.0	41.5	22.2
3	88.0	53.8	33.8	18.0
4	92.8	75.5	28.2	14.0

Table 2. Number of inner GMRES iterations per Newton iteration for subsonic case.

Order	Matrix-Free		Matrix-Explicit	
	ILU(1)	ILU(4)	ILU(0)	ILU(1)
2	14.4	8.6	6.3	5.7
3	31.6	21.1	12.0	11.7
4	47.2	40.8	13.4	14.0

Table 3. Run time in seconds for the subsonic case.

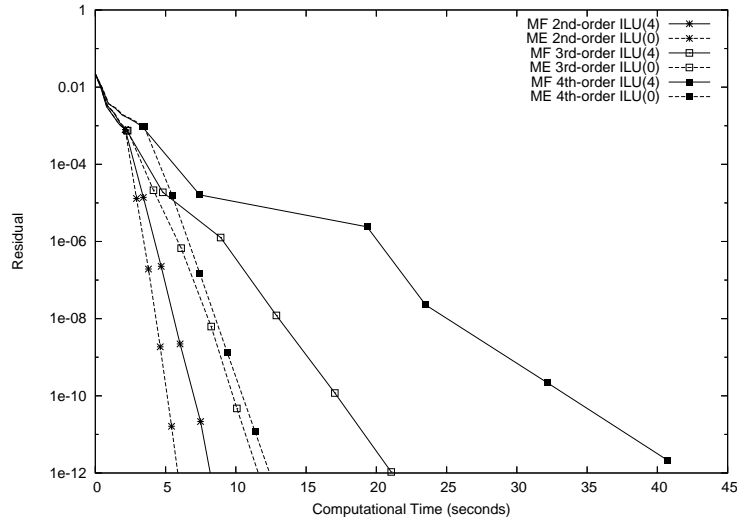


Figure 3. Residual versus computational time for the subsonic case.

Order	Scheme	Precon- ditioner	Recon- struction	Jacobian	ILU Fill-Ratio	ILU Decomposition	Krylov Subspace	Total (floats)	Total (bytes)
2	MF	ILU(1)	11.7	108.7	1.31	142.5	296	558.9	4471
2	MF	ILU(4)	11.7	108.7	2.68	291.7	128	540.1	4321
2	ME	ILU(0)	11.7	297.2	1.00	297.2	176	782.1	6257
2	ME	ILU(1)	11.7	297.2	1.58	470.2	92	871.1	6969
3	MF	ILU(1)	88.0	108.7	1.31	142.5	408	747.2	5978
3	MF	ILU(4)	88.0	108.7	2.68	291.7	244	732.4	5859
3	ME	ILU(0)	88.0	580.2	1.00	580.2	156	1404.3	11234
3	ME	ILU(1)	88.0	580.2	1.70	986.0	84	1738.1	13905
4	MF	ILU(1)	179.2	108.7	1.31	142.5	532	962.4	7699
4	MF	ILU(4)	179.2	108.7	2.68	291.7	500	1079.6	8637
4	ME	ILU(0)	179.2	616.7	1.00	616.7	120	1532.6	12260
4	ME	ILU(1)	179.2	616.7	1.74	1073.4	60	1929.2	15434

Table 4. Memory use in number of floating point numbers per control-volume for subsonic case.

Order	Flux	1st Order Jacobian		Full Order Jacobian	
	seconds	seconds	relative to flux	seconds	relative to flux
1	0.014	0.236	17.06		
2	0.045	0.236	5.229	0.367	8.14
3	0.088	0.236	2.687	1.569	17.87
4	0.127	0.236	1.863	2.007	15.84

Table 5. Relative computational time of flux and Jacobian evaluation with limiter for transonic case.

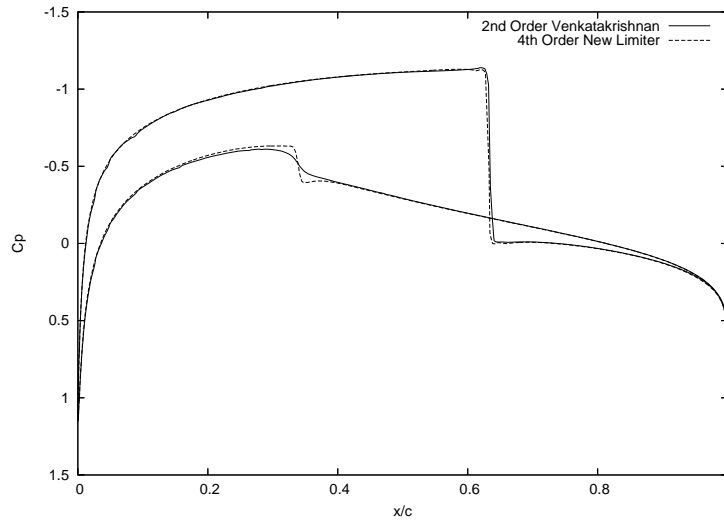


Figure 4. Surface pressure profiles for transonic flow over a NACA0012 airfoil

Order	1st Order Preconditioner		Full Order Preconditioner	
	ILU(1)	ILU(4)	ILU(0)	ILU(1)
2	36.2	18.5	33.0	19.0
3	58.6	41.5	17.7	9.7
4	105.3	92.2	30.9	15.7

Table 6. Number of inner GMRES iterations per Newton iteration for transonic case.

Order	Matrix-Free		Matrix-Explicit	
	ILU(1)	ILU(4)	ILU(0)	ILU(1)
2	23.9	20.2	16.8	16.4
3	305.3	252.1	138.4	139.2
4	133.1	118.1	49.3	49.6

Table 7. Run time in seconds for the transonic case.

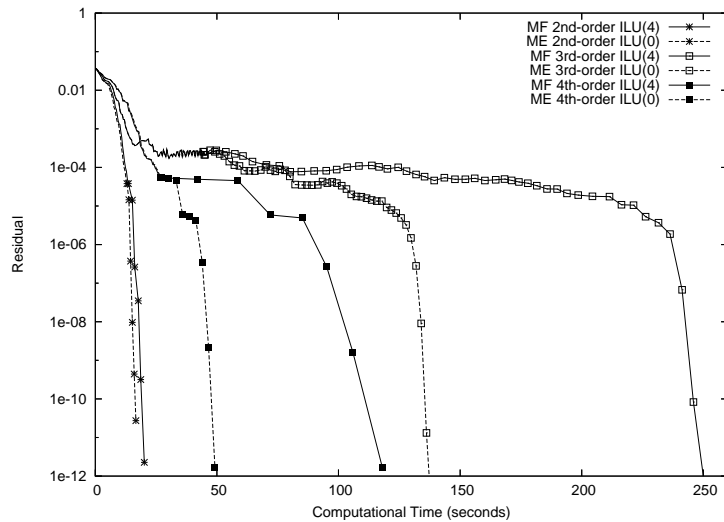


Figure 5. Residual versus computational time for the transonic case.