**47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition**
**5 - 8 January 2009, Orlando, Florida**

**AIAA 2009-950**

# A $p$-Multigrid Spectral Difference method for viscous compressible flow using 2D quadrilateral meshes

Sachin Premasuthan, [*] Chunlei Liang [†] and Antony Jameson [‡]

*Stanford University, Stanford, CA 94305, USA*

Z.J.Wang[§]

*Iowa State University, IA 50010, USA*

**The work focuses on the development of a 2D quadrilateral element based Spectral Difference solver for viscous flow calculations, and the application of the $p$-multigrid method and implicit time-stepping to accelerate convergence. This paper extends the previous work by Liang et al (2009) on the $p$-multigrid method for 2D inviscid compressible flow,[2] to viscous flows. The high-order spectral difference solver for unstructured quadrilateral meshes is based on the formulation of Sun et al[3] for unstructured hexahedral elements. The $p$-multigrid method operates on a sequence of solution approximations of different polynomial orders ranging from one upto four. An efficient preconditioned Lower-Upper Symmetric Gauss-Seidel (LU-SGS) implicit scheme is also implemented. The spectral difference method is applied to a variety of inviscid and viscous compressible flow problems. The speed-up using the p-Multigrid and Implicit time-stepping techniques is also demonstrated.**

## I. Introduction

Until recently, compressible flow computations on unstructured meshes have generally been dominated by schemes restricted to second order accuracy. However, the need for highly accurate methods in applications such as large eddy simulation, direct numerical simulation, computational aeroacoustics etc., has seen the development of higher order schemes for unstructured meshes such as the Discontinuous Galerkin (DG) Method,[1,7] Spectral Volume (SV) method[9,10] and Spectral Difference (SD) Method.[4,5] The SD method is a newly developed efficient high-order approach based on differential form of the governing equation. It was originally proposed by Liu et al.[4] and developed for wave equations in their paper on triangular grids. Wang et al.[5](2007) extended it to 2D Euler equations on triangular grids and Sun et al.[3](2007) further developed it for three-dimensional Navier-Stokes equations on hexahedral unstructured meshes. The SD method combines elements from finite-volume and finite-difference techniques, and is particularly attractive because it is conservative, and has a simple formulation and implementation.

One of the difficulties encountered in large-scale simulations and high-order solutions is that the rate of convergence slows down dramatically as order is increased. This becomes a serious issue when using explicit time-integration schemes, particularly for viscous flow calculations with clustered boundary layer grids. In order to speed up convergence, an implicit temporal discretization or a multigrid strategy is required. Implicit time-integration schemes are highly desirable for convergence acceleration as they allow significantly larger time-steps as compared to explicit schemes. Many implicit schemes have been developed and applied successfully to unstructured grids. In this paper we have implemented and used a preconditioned Implicit Lower-Upper Symmetric Gauss-Seidel (LU-SGS) solution algorithm for convergence acceleration. The original LU-SGS approach was developed by Yoon and Jameson on structured grids.[16,17] The implicit

[*]Doctoral Candidate, Department of Aeronautics and Astronautics, Stanford University, AIAA Member.
[†]Post-Doctoral Candidate, Department of Aeronautics and Astronautics, Stanford University, AIAA Member.
[‡]Professor, Department of Aeronautics and Astronautics, Stanford University, AIAA Fellow.
[§]Professor, Department of Aerospace Engineering, Iowa State University, AIAA Associate Fellow.

LU-SGS scheme has been used with the Spectral Difference method successfully on both triangular[2] and hexahedral[3] meshes.

The issue with implicit solution methods is that they require a considerable amount of memory to store the Jacobian matrix, which becomes prohibitive for large scale 3D problems and high-order solutions. The $p$-multigrid method is an iterative scheme which solves the discretized equations by recursively iterating on solution approximations of different polynomial orders. This method was initially proposed by Ronquist and Patera,[11] and extended by Maday and Munoz.[12] The convergence acceleration by the $p$-multigrid algorithm on Euler equations was demonstrated by Liang et al[2] for unstructured triangular mesh. In this paper, we extend the previous work by Liang et al to viscous flow problems.

## II. Formulation of 2D Spectral Difference Scheme on quadrilateral meshes

The formulation of the equations for the 2D spectral difference scheme on quadrilateral meshes is similar to the formulation of Sun et al[3] for unstructured hexahedral grids

Consider the unsteady compressible 2D Navier Stokes equations in conservative form

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \tag{1}$$

where $Q$ is the vector of conserved variables; $F$ and $G$ are the total fluxes including both inviscid and viscous flux vectors.

To achieve an efficient implementation, all elements in the physical domain $(x, y)$ are transformed into a standard square element. $0 < \xi < 1$, $0 < \eta < 1$. The transformation can be written as:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \sum_{i=1}^{K} M_i\left(\xi, \eta\right) \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{2}$$

where $K$ is the number of points used to define the physical element, $(x_i, y_i)$ are the cartesian coordinates at those points, and $M_i\left(\xi, \eta\right)$ are the shape functions. The metrics and the Jacobian of the transformation can be computed for the standard element. The governing equations in the physical domain are then transfered into the computational domain, and the transformed equations take the following form:

$$\frac{\partial \tilde{Q}}{\partial t} + \frac{\partial \tilde{F}}{\partial \xi} + \frac{\partial \tilde{G}}{\partial \eta} = 0 \tag{3}$$

where $\tilde{Q} = |J| \cdot Q$ and

$$\begin{pmatrix} \tilde{F} \\ \tilde{G} \end{pmatrix} = |J| \begin{pmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{pmatrix} \begin{pmatrix} F \\ G \end{pmatrix} \tag{4}$$

In the standard element, two sets of points are defined, namely the solution points and the flux points, illustrated in figure 1. In order to construct a degree $(N - 1)$ polynomial in each coordinate direction, solution at $N$ points are required. The solution points in 1D are chosen to be the Gauss points defined by:

$$X_s = \frac{1}{2}\left[1 - cos\left(\frac{2s-1}{2N} \cdot \pi\right)\right], s = 1, 2, \cdots, N. \tag{5}$$

The flux points are selected to be Legendre-Gauss-Quadrature points plus the two end points 0 and 1, as suggested by Huynh.[19] They are found to be more stable than the Gauss-Lobatto flux points[3] and produce more accurate solutions for high-order spectral difference schemes.

Using the solutions at $N$ solution points, a degree $(N - 1)$ polynomial can be built using the following Lagrange basis defined as:

$$h_i\left(X\right) = \prod_{s=1, s\neq i}^{N} \left(\frac{X - X_s}{X_i - X_s}\right) \tag{6}$$

Similarly, using the fluxes at $(N + 1)$ flux points, a degree $N$ polynomial can be built for the flux using a similar Lagrange basis defined as:

$$l_{i+1/2}\left(X\right) = \prod_{s=0, s\neq i}^{N} \left(\frac{X - X_{s+1/2}}{X_{i+1/2} - X_{s+1/2}}\right) \tag{7}$$

The reconstructed solution for the conserved variables in the standard element is just the tensor products of the two one-dimensional polynomials,

$$Q(\xi, \eta) = \sum_{j=1}^{N} \sum_{i=1}^{N} \frac{\tilde{Q}_{i,j}}{|J_{i,j}|} h_i(\xi) \cdot h_j(\eta) \tag{8}$$

Similarly, the reconstructed flux polynomials take the following form:

$$\begin{aligned} \tilde{F}(\xi, \eta) &= \sum_{j=1}^{N} \sum_{i=0}^{N} \tilde{F}_{i+1/2,j} l_{i+1/2}(\xi) \cdot h_j(\eta), \\ \tilde{G}(\xi, \eta) &= \sum_{j=0}^{N} \sum_{i=1}^{N} \tilde{G}_{i,j+1/2} h_i(\xi) \cdot l_{j+1/2}(\eta) \end{aligned} \tag{9}$$

The reconstructed fluxes are only element-wise continuous, but discontinuous across cell interfaces. For the inviscid flux, a Riemann solver is employed to compute a common flux at interfaces to ensure conservation and stability. In our case, we have used the Rusanov solver[15] to compute the interface fluxes.

In summary, the algorithm to compute the inviscid flux derivatives consists of the following steps:

1. Given the conservative variables at the solution points, the conservative variables are computed at the flux points
2. The inviscid fluxes at the interior flux points are computed using the solutions computed at Step 1
3. The inviscid fluxes at the element interfaces are computed using the Rusanov solver.
4. The derivative of the fluxes are computed at the solution points according to (equation)

$$\left(\frac{\partial \tilde{F}}{\partial \xi}\right)_{i,j} = \sum_{r=0}^{N} \tilde{F}_{r+1/2,j} \cdot l'_{r+1/2}(\xi_i), \tag{10}$$

$$\left(\frac{\partial \tilde{G}}{\partial \eta}\right)_{i,j} = \sum_{r=0}^{N} \tilde{G}_{i,r+1/2} \cdot l'_{r+1/2}(\eta_j) \tag{11}$$

The viscous flux is a function of both the conserved variables and their gradients. Therefore, the solution gradients have to be calculated at the flux points. In our solver, the average approach described in reference[3] is used to compute the viscous fluxes.

## III. $p$-Multigrid method

The $p$-multigrid method is a natural extension of geometric multigrid methods to high-order accurate formulations. Classical multigrid method begins with a two-level process. First, iterative relaxation is applied using the higher order polynomial, thus basically reducing high-frequency errors. Then, a "coarse-grid" correction is applied, in which the smooth error is determined at the lower polynomial level. This error is interpolated to the higher polynomial level and used to correct the existing higher order solutions. Applying this method recursively to solve at lower polynomial levels leads to the $p$-multigrid method.

The key idea of the $p$-multigrid method is to solve the nonlinear equations using a lower order polynomial such that "smooth becomes rough" and low frequencies act like high frequencies. The $p$-Multigrid method operates on a sequence of solution approximations of different polynomial orders. Therefore it offers the flexibility of switching between higher and lower polynomial levels without changing the actual geometrical nodal grid points.

In this study a multi-level V-cycle has been used to drive the iterations. To accomplish the communication between different levels, we use restriction and prolongation operators derived from cardinal basis functions of the solution points for the starting $p$-level. Currently a Runge-Kutta explicit scheme is employed as a smoother at all levels

The following steps summarize the standard two-level scheme with a V-cycle at $p$ and $p-1$ levels:

Firstly, we are trying to solve $R_p(Q_p) = r_p$, where $R_p(Q_p)$ corresponds to the residual of current solution $Q_p$, $r_p$ is the rhs and equals 0 for the higher polynomial level p.

1. Apply smoothing iterations at level $p$ to improve the solution $Q_p$.
2. Compute the defect at level $p$.
$$d_p = r_p - R_p\left(Q_p\right) = -R_p\left(Q_p\right) \tag{12}$$
3. Restrict the latest solution and the defect to the lower approximation level $p-1$
$$Q^0_{p-1} = I^{p-1}_p\left(Q_p\right) \tag{13}$$
4. Compute the rhs of equation at level $p-1$.
$$r_{p-1} = R_{p-1}\left(Q^0_{p-1}\right) + I^{p-1}_p d_p \tag{14}$$
5. Apply smoothing iterations at level $p-1$ to improve $Q_{p-1}$
6. Compute the correction at level $p-1$.
$$C_{p-1} = \bar{Q}_{p-1} - Q^0_{p-1} \tag{15}$$
7. Extrapolate the correction from level $p-1$ to update the solution at level $p$.
$$\tilde{Q}_p = Q_p + I^p_{p-1} C_{p-1} \tag{16}$$
8. Improve $Q_p$ by application of iterative smoother

The $p$-multigrid has been used to accelerate convergence for the DG method.[6–8] For SD schemes, the $p$-multigrid has been used for inviscid flows on triangular grids.[2] This paper intends to study the performance and effectiveness of $p$-multigrid with quadrilateral grids using up to four-level multigrid cycles. Earlier studies for Euler flow calculations have indicated that the use of explicit smoothing iterations at all levels adversely affects the performance of the $p$-multigrid. This has been attributed to the severely anisotropic (hyperbolic) nature of the Euler equations and the isotropic (elliptic) nature of $p$-multigrid iterations.[7] Efforts will be made to address this issue with regards to the viscous flow equations. The speed-up obtained using p-multigrid with explicit RK smoothers, is compared to the speed-up obtained using the (single p-level) implicit LU-SGS scheme.

## IV. Implicit Scheme

The implicit LU-SGS scheme has been used with the Spectral Difference Method for inviscid flows on triangular meshes,[2] and also been used for viscous computations on hexahedral meshes.[3] A similar approach is used here for implementation with quadrilateral meshes.

At each cell c, the governing equations can be discretized using the backward Euler difference to give
$$\frac{Q^{n+1}_c - Q^n_c}{\Delta t} - \left[R_c(Q^{n+1}) - R_c(Q^n)\right] = R_c(Q^n) \tag{17}$$
Let $\Delta Q_c = Q^{n+1}_c - Q^n_c$ be the change in the solution over the implicit time step. Linearizing the residual, we obtain
$$R_c(Q^{n+1}) - R_c(Q^n) \approx \frac{\partial R_c}{\partial Q_c}\Delta Q_c + \sum_{nb\neq c}\frac{\partial R_c}{\partial Q_{nb}}\Delta Q_{nb}, \tag{18}$$
where nb indicates all the neighboring cells contributing to the residual of c. Therefore, the fully linearized equations can be written as
$$\left(\frac{I}{\Delta t} - \frac{\partial R_c}{\partial Q_c}\right)\Delta Q_c - \sum_{nb\neq c}\frac{\partial R_c}{\partial Q_{nb}}\Delta Q_{nb} = R_c(Q^n). \tag{19}$$
However, it costs too much memory to store the LHS implicit Jacobian matrices. We employ a LU-SGS scheme to solve equation 19, thus using the most recent solution in the neighboring cells,

$$\left(\frac{I}{\Delta t} - \frac{\partial R_c}{\partial Q_c}\right) \Delta Q_c^{(k+1)} = R_c(Q^n) + \sum_{nb \neq c} \frac{\partial R_c}{\partial Q_{nb}} \Delta Q_{nb}^*. \tag{20}$$

Note that the superscript "$*$" refers to the most recent solution when doing the forward and backward sweeps of the LU-SGS scheme, and the superscript "k+1" denotes the new solution that is obtained after a particular sweep.

The left-hand side matrix given by,

$$D = \left(\frac{I}{\Delta t} - \frac{\partial R_c}{\partial Q_c}\right) \tag{21}$$

is the element cell matrix. For a 3rd order computation in 2D, the size of this matrix is $(4 \times 3 \times 3)^2 = 576$ elements. The computation of the element matrix involves the computation of the cell Jacobian matrix $\frac{\partial R_c}{\partial Q_c}$ which is not a trivial operation. It must be pointed out that for the Navier-Stokes equation the residual in a cell depends not only on its immediate neighbors but also the neighbors' neighbors and hence the computation of the summation term on the right-hand side of equation 20 can be quite computationally intensive. This can be avoided however, by further linearization to eliminate the off-diagonal Jacobian matrices $\frac{\partial R_c}{\partial Q_{nb}}$, thus giving a diagonally dominant system.

$$\left(\frac{I}{\Delta t} - \frac{\partial R_c}{\partial Q_c}\right) \Delta^2 Q_c^{(k+1)} = R_c(Q^*) - \frac{\Delta Q_c^*}{\Delta t}. \tag{22}$$

where $\Delta^2 Q_c^{(k+1)} = \Delta Q_c^{(k+1)} - \Delta Q_c^*$.

Equation 22 is solved with a direct LU decomposition solver and by sweeping symmetrically forward and backward through the computational grid. For steady flow calculations, the term $\frac{\Delta Q_c^*}{\Delta t}$ in the right-hand side can be neglected, leading to faster convergence.

The cell Jacobian $\frac{\partial R_c}{\partial Q_c}$ is computed using a numerical approach as shown below

$$\frac{\partial R_c}{\partial Q_c} = \frac{R_c(Q_{nb}, Q_c + \epsilon) - R_c(Q_{nb}, Q_c)}{\epsilon} \tag{23}$$

The numerical approach for Jacobian computation has been previously used.[2,3] Even though this approach is easy to implement, the computational effort involved is large since each variable at each solution point has to be changed, and the residuals recomputed. Numerical tests have shown that it is not necessary to compute the Jacobian matrix at every iteration. In practice it need be computed every 40-100 iterations. This technique has been called matrix-freezing and it does not degrade convergence to steady state.

## V. Results

In the section *V.A*, the results obtained from the testing and validation of the inviscid flow solver are presented. The supersonic vortex test case and the inviscid flow past circle and airfoil are discussed in this context. In the next section, the testing and validation of the viscous flow solver is discussed. The test cases demonstrated include the planar Couette flow test case, viscous flow past a circle and viscous flow past an airfoil. We then discuss (in section *V.C*), the application of p-multigrid method and implicit scheme to the inviscid and viscous flow computations.

It should be mentioned that all explicit time-marching calculations for steady flows have been done using a Jameson type four-stage Runge Kutta scheme (RK4), which is 2nd order accurate in time. For the unsteady problems, we have used a 4th order accurate, strong-stability-preserving five-stage Runge-Kutta scheme[18] to advance in time. In all p-multigrid computations, the explicit RK4 smoother is used at all p-levels. In all implicit calculations, unless otherwise stated, each iteration consists of 11 forward-backward sweeps, and the element Jacobian matrix is computed only every 40 timesteps.

### V.A. Inviscid flow calculations

#### *V.A.1. Validation using supersonic vortex flow*

This test case is used to verify the implementation of the developed computer code and assess the order of accuracy of the spectral difference method used. The supersonic vortex flow problem is one of the few non-trivial problems of the steady compressible 2D Euler equations for which a smooth analytical solution is known. The inviscid, isentropic, supersonic flow of a compressible fluid between concentric circular arcs presents a flow where the velocity varies inversely with radius. The expression for density as a function of radius r is given by:

$$\rho\left(r\right)=\rho_i\left\{1+\frac{\gamma-1}{2}M_i^2\left[1-\left(\frac{r_i}{r}\right)^2\right]\right\}^{\frac{1}{\gamma-1}} \tag{24}$$

where $M_i$ and $r_i$ are the Mach number and the radius at the inner arc. In the present calculation, the Mach number, density and pressure at the inner radius $r_i$ are specified to be 2.25, 1 and $1/\gamma$ respectively. The inner and outer radii are 1 and 1.384. A curved inviscid wall BC is used as wall boundary condition. The zero-gradient extrapolation boundary is employed for the exit. In the following, the numerical solution to this problem are computed for the 2nd, 3rd and 4th order SD method on successively refined grids. All the computations are initialized using constant density and pressure. The $L^2$-error of the density is evaluated.

The four meshes used in the computation were of sizes $10 \times 4$, $15 \times 6$, $30 \times 12$, and $60 \times 24$. A sample $15 \times 6$ mesh is shown in figure 2 (a). Figure 2 (b) shows the density contours in the flow field obtained by the 3rd order SD method. The details of the order calculation and verification are shown in Table 1. The table clearly indicates that the SD method applied to the steady compressible Euler equations exhibits a full order of convergence on smooth solutions. Figure 3 (a) provides the details of the spatial accuracy of the SD method for different orders for this numerical experiment. Figure 3 (b) shows the $L^2$-error of the SD method at different order SD method plotted against the number of degrees of freedom. One can clearly see that a higher order SD method requires a lesser number of degrees of freedom than a lower order SD method to achieve the same accuracy.

#### *V.A.2. Subsonic flow past a circle*

In this section, the results obtained for 2D steady, subsonic flow around a circle at Mach number $M_\infty$=0.2, are presented. Computations were made for second, third and fourth order approximation of unknowns, and linear, quadratic and cubic geometric mapping for the curved boundary elements. The computations have been performed on a $32\times32$ grid . This test case demonstrates the significance of higher order representation of boundary elements to obtain accurate solutions.

Figures 4 (a) show the Mach number contours using 4th order SD with linear representation of boundary. This solution is very inaccurate, as put in evidence by the non-physical "boundary layer" which develops along the solid wall, and by the associated "wake" in the downstream region of the circle. Moreover the computations do not converge to a steady solution due to the unsteadiness of the unphysical wake. Figure 4 (b) shows the Mach number contours using the 4th order SD method with cubic boundary representation. It clearly indicates that the use of higher order representation for the boundary results in a smoother and more accurate solution.

#### *V.A.3. Inviscid flow past airfoil*

The inviscid subsonic flow at $M_\infty = 0.5$ past a NACA0012 airfoil at zero degree angle of attack is studied. The computational grid with $80 \times 16$ cells was used and is shown in figure 5(a). The airfoil surface is represented using a high-order boundary representation - cubic spline in this case. The outer boundary is about 25 chords away from the airfoil mid-chord. Figure 5(b) shows the pressure contours for a 3rd order SD computation. The computed $C_D$ for the 3rd order simulation is 0.00002.

### V.B. Viscous flow calculations

#### *V.B.1. Validation using planar Couette flow*

The planar Couette flow test case was used to test and validate the viscous solver, and to confirm the order of accuracy of viscous computations. This problem models the viscous flow between two parallel plates located at $y = 0$ and $y = H$. The plate at $y = 0$ is stationary and kept at a temperature $T_0$, and the plate at $y = H$ is moving at speed U in the x-direction and is at fixed temperature $T_1$. Under the assumption of constant viscosity coefficient, this problem has an exact solution, which can be expressed as

$$u\left(y\right)=\frac{U}{H}y, v=0, \tag{25}$$

$$T\left(y\right) = T_0 + \frac{y}{H}\left(T_1 - T_0\right) + \frac{\mu U^2}{2k} \cdot \frac{y}{H}\left(1 - \frac{y}{H}\right), \tag{26}$$

$$p = const, \rho = \frac{p}{R \cdot T} \tag{27}$$

The numerical solution to this problem was computed for the 2nd, 3rd and 4th order SD methods on successively refined grids. All the computations are initialized using constant density and pressure. Isothermal wall BC was used for top and bottom walls, and periodic BC was used for left and right boundary. The $L^2$-error of the density is evaluated. The four meshes used in the computation were of sizes $2 \times 1$, $4 \times 2$, $8 \times 4$, and $16 \times 8$. Figure 6 shows the density contours in the flow field obtained by 3rd order SD method using the $4 \times 2$ grid. The details of the order calculation and verification are shown in Table 2. The table clearly indicates that the SD method applied to the steady compressible Navier Stokes equations exhibits a full order of convergence on smooth solutions.

#### *V.B.2. Viscous flow past a circle*

The viscous solver was also tested for flow past a circle. The results obtained for viscous flow past a circle are discussed below. The grid used for viscous computation had 800 elements and is shown in figure 7 (a). The freestream Mach number is 0.2 and the Reynolds number is 100. Computations were performed using both 3rd and 4th order schemes with corresponding quadratic and cubic boundary representations. The computed Mach number contours for 4th order unsteady calculation are shown in figure 7 (b). The unsteady von Karman vortex street was predicted by both the 3rd and 4th order schemes. It was observed that the flow reached a periodic state in both the cases. For the fourth order solution, the frequency of the periodic variation of lift coefficient $C_L$ gave a Strouhal number of 0.164. The SSPRK5 scheme was used time-stepping scheme for this unsteady problem.

#### *V.B.3. Viscous flow past NACA0012 airfoil*

The results obtained for viscous fluid flow past NACA0012 airfoil at zero degree angle of attack are reported here. The free stream Mach number is 0.5 and the flow Reynolds number is 5000. These flow conditions are identical to one of the test-cases in Bassi and Rebay (1997).[14] The 120 × 24 C-grid (see figure 8) with 78 points on the airfoil surface is used in these calculations. Figure 9(a) shows the Mach contours obtained for a 3rd order SD calculation. The surface $C_p$ distribution compares well with that obtained by Bassi and Rebay for a 4th order DG computation (see figure 9(b)). The coefficient of drag $C_D = 0.531$ obtained with the 3rd order SD compares well with $C_D = 0.535$ obtained with the 3rd order DG calculation. Also, the skin friction coefficient indicates the separation point at 0.84c, which compares well with the value predicted by Bassi and Rebay.

## V.C. Application of $p$-multigrid and Implicit time-stepping to accelerate convergence

#### *V.C.1. Subsonic inviscid flow past bump*

This test case consists of subsonic flow in a channel with a 10% thick circular bump on the bottom. The length of the channel is 3 units and its height 1 unit. The inlet Mach number is 0.5. This test case has been used by $p$-multigrid for DG formulation[7] for Euler equation as well as for SD formulations.[2] The mesh which contains 3201 grid-points, 3072 elements in total and 32 nodes to resolve the bump, is depicted in figure 10. It must be mentioned that for this grid, a 3rd order computation would involve $(3072 \times 9 =)$ 27648 computational nodes. The circular surface of the bump is represented as a quadratic boundary. The pressure contours obtained using the 4th order SD scheme with 4-level p-multigrid scheme is shown in figure 11, and is almost identical to those obtained in references.[2,7] All p-multigrid calculations for the bump case are done using RK4 scheme for time marching at all levels. As mentioned earlier, for the implicit LU-SGS calculation, 11 forward and backward sweeping iterations are done at each time-step, and the jacobian matrix is computed every 40 time-steps.

A comparison of the convergence histories using the explicit RK4 scheme shows that the convergence deteriorates drastically when the SD order is increased from 1 to 2. For 3rd and 4th order SD, the RK scheme failed to converge. These convergence trends for higher order SD using RK time-stepping for the subsonic bump case have been documented earlier.[7,13] The application of the $p$-multigrid method results in superior convergence characteristics for second, third and fourth order SD schemes. However, the single p-level Implicit LU-SGS time-stepping is able to obtain fastest convergence.

Figure 12(a) shows the convergence acceleration in terms of the CPU time for the second order SD scheme. A 2-level V-cycle *p*-multigrid method has been used with single smoothing iteration at the higher level (p1) and 2 smoothing iterations at the lower level (p0). (It should be noted that p0 corresponds to 1st order SD, p1 to 2nd order and so on.) It is evident that speed-up is obtained using the 2-level *p*-multigrid cycle. The speed-up using implicit time-stepping is about 5 times compared to the 2-level *p*-multigrid cycle.

In the absence of p-multigrid both 3rd and 4th order fail to converge. However, the use of *p*-multigrid results in favourable convergence characteristics as seen in figure 12(b). For the 3rd order SD, a 3 level V-cycle is used with 1-1-2-1-0 smoothing iterations at p2-p1-p0-p1-p2 levels. For the 4th order SD case, a 4-level V-cycle with 1-1-1-2-1-1-0 smoothing iterations at p3-p2-p1-p0-p1-p2-p3 levels is used. A comparison of *p*-multigrid and implicit speed-up for 3rd and 4th order is presented in figure 13(a) and (b).

The effect of changing the V-cycle on the convergence of 3rd order SD was studied. As expected the p2-p1-p0 3-level multigrid was found to give best results, as seen in figure 13(a). However, the p2-p0 2-level (3,1 (a)) V-cycle produces comparable convergence acceleration. This can be attributed to the fact that for a single level, p0 has the fastest convergence. The p2-p1 2-level cycle has poor convergence characteristics due to the slow convergence at p1 level. The speed-up using implicit scheme is about 7 times faster than the fastest *p*-multigrid cycle for the residual to drop to 1e-8.

In the case of the 4th order SD method, the 4-level V-cycle (4,3,2,1 p-MG) with 1-1-1-2-1-1-0 iterations at p3-p2-p1-p0-p1-p2-p3 is found to be most effective for convergence acceleration, as seen in figures 13(b). The 3-level V-cycle with 1-1-2-1-0 iterations at p3-p1-p0-p1-p3 (4,2,1 p-MG) gives the next best convergence. The use of p3-p2-p0-p2-p3 (4,3,1 p-MG) and the p3-p2-p1-p2-p3 (not shown in figure) 3-level cycles does not show good convergence characteristics. This could be attributed to the fact that two of the levels p3 and p2 both have very poor convergence. The implicit scheme is about 10 times faster than the fastest *p*-multigrid cycle.

#### *V.C.2. Inviscid flow past airfoil*

The details of this test-case have been described earlier in the section *V.A.*3. Figure 17(a) shows the convergence speed-up using the p-multigrid and implicit schemes for 3rd order SD computation. The 3-level cycle with 1-1-2-1-0 smoothing iterations at p2-p1-p0-p1-p2 levels is found to give best convergence. It is about 1.5 times faster than the 2-level p-multigrid cycle with 1-2-0 smoothing iterations at p2-p1-p2 levels. For a residual drop to 1e-8, a p-multigrid speed up of about 7-8 is obtained as compared to the explicit RK4. In comparison, an implicit speed up obtained is about 300 times compared to explicit RK4.

Figure 17(b) shows the convergence speed-up using the p-multigrid and implicit schemes for 3rd order SD computation. The 4-level p-multigrid cycle with 1-1-1-2-1-1-0 iterations at p3-p2-p1-p0-p1-p2-p3 is found to give a convergence speed up of about 9-10 times compared to the explicit RK4 case. The 2-level and 3-level p-multigrid cycles were found to give poor speed-up compared to the 4-level cycle in this case. The convergence using the implicit scheme is about 3 orders of magnitude faster than that of the explicit RK case.

#### *V.C.3. Planar Couette flow*

The p-multigrid method and implicit scheme were used for convergence acceleration for the planar Couette flow test case. All multigrid calculations were done using the four stage RK scheme for time marching at all levels. It must be mentioned that for these computations, the multilevel *p*-multigrid cycle used has a single smoothing iteration at the higher levels and 2 smoothing iterations at the lowest level (p0). Also, the computations were done on the $8 \times 4$ grid corresponding to 128, 288 and 512 computational nodes for the 2nd, 3rd and 4th order SD respectively.

In figure 14, we see that for a second order computation, the application of p-multigrid accelerates convergence by about 3 times. The implicit speed-up obtained is about 13 times compared to explicit RK4. For the third order computation, the 2-level V-cycle (3,1 p-MG) is found to be more effective for convergence acceleration than the 3-level V-cycle (3,2,1 p-MG) (see figure 15). In this case, the application of p-multigrid is able to reduce the computation time by a factor of about 5, and the implicit scheme reduces the computational time by about 40. In the case of the fourth order computation, the 3-level V-cycle (4,2,1 p-MG) gives the best convergence characteristics (see figure 16). A speed-up of about 8 is obtained for 4th order SD. The implicit speed-up is about 110 compared to the explicit RK.

### *V.C.4. Viscous flow past NACA0012 airfoil*

The details of this test case have been described in section *V.C.*3. Figure 18(a) shows the convergence speed-up using the p-multigrid and implicit schemes for 2nd order SD computation. The 2-level p-multigrid gives a speed-up of about 3.5 for the viscous flow case. The implicit scheme once again gives a much higher speed up of about 40 times compared to the explicit RK. For the 3rd order case, a 3-level p-multigrid V-cycle is used with 1-1-2-1-0 smoothing iterations at the p2-p1-p0-p1-p2 levels. A speed-up of about 8 is obtained until the residual drops to about 1e-7. As the residual drops further the p-multigrid performance deteriorates slightly as shown in figure 18(b). In comparison, the implicit scheme still obtains a very high speed up of more than 2 orders of magnitude.

In comparison to the inviscid flow past airfoil case, the performance of the p-multigrid for the viscous flow past airfoil shows relatively unsatisfactory convergence characteristics. As indicated in earlier studies using p-multigrid for viscous Discontinuous Galerkin[8] and Spectral Volume,[20] this issue may be resolved by using implicit smoothers at the different $p$-multigrid levels.

Also, when using the implicit LU-SGS time-stepping, the speed-up obtained for the inviscid airfoil flow case is much higher than that for the viscous case. This could possibly be due to two reasons. Firstly, the implicit LU-SGS implementation requires much higher computational effort for viscous flow calculations. For inviscid flow, the residual within an element (cell) depends only on its immediate neighbors, whereas for the viscous case, residual computations involve communication with its neighbors' neighbors, thus increasing computational effort and time. Another explanation could be that the LU-SGS performance degrades when using stretched grids.

## VI. Conclusions and Future work

We have developed a 2D spectral difference solver for inviscid and viscous flows. The solver has been tested and validated for a number of test cases. $p$-Multigrid has been implemented with upto four multigrid levels, and the results have been promising. An implicit LU-SGS scheme has been implemented with very effective convergence speed-up for both the inviscid and viscous case. Efforts will be made to incorporate implicit smoothing at different levels of the p-multigrid. The p-multigrid will be applied to other test cases to study its applicability and effectiveness to accelerate convergence. Different multigrid cycles such as W-cycle and Full Multigrid (FMG) will be implemented to find the fastest and most efficient technique.

We plan to extend our p-multigrid spectral difference implementation to 3D. The p-multigrid method can have highly beneficial effects when used for 3D flow problems. This is because the use of implicit methods for 3D flow problems are not very feasible due to the high storage requirements. This memory restriction is aggravated for problems where we seek a high-order solution ($> 3$). Therefore it is expected that for high order SD schemes for 3D flow problems, $p$-multigrid could be an efficient and effective technique to accelerate convergence. Efforts will also be made to extend the p-multigrid method to unsteady flows.

## References

[1]F.Bassi and S. Rebay, High-order accurate discontinuous finite element solution of the 2D euler equations, *Journal of Computational Physics*, Vol. 138, pp. 251-285, 1997

[2]C. Liang, R. Kannan and Z.J. Wang, A p-Multigrid Spectral Difference Method with explicit and implicit smoothers on unstructured triangular grids, *Computers and Fluids*, vol. 38, pp. 254-265, 2009.

[3]Y. Sun, Z. J. Wang, & Y. Liu, High-order multidomain spectral difference method for the Navier-Stokes equations on unstructured hexahedral grids, *Communication in Computational Physics*, Vol. 2, pp. 310-333, 2007.

[4]Y. Liu, M. Vinokur, & Z. J. Wang, Spectral difference method for unstructured grids I:Basic formulation, *J. of Comput. Phys.*, Vol. 216 , pp. 780-801, 2006.

[5]Z.J. Wang, Y. Liu, G. May and A. Jameson, Spectral Difference Method for Unstructured Grids II: Extension to the Euler Equations, *Journal of Scientific Computing*, Vol. 32, pp. 45-71, 2007.

[6]B. T. Helenbrook and H. L. Atkins, Application of p-Multigrid to Discontinuous Galerkin formulations of the Poisson equation, *AIAA Journal*, Vol. 44, pp. 566-575, 2006.

[7]H. Luo, J. D. Baum and R. Löhner, A p-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids, *Journal of Computational Physics*, Vol. 211, pp 767-783, 2006.

[8]K.J. Fidkowski, T.A. Oliver, J. Lu, D.L. Darmofal, p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible NavierStokes equations, *Journal of Computational Physics*, Vol. 207, pp 92-113, 2005.

[9]Y. Liu, M. Vinokur, & Z. J. Wang, Spectral (finite) volume method for conservation laws on unstructured grids V: Extension to three-dimensional systems, *Journal of Computational Physics*, Vol. 212, pp 454-472, 2006.

[10]Z. J. Wang & Y. Liu, Extension of the spectral volume method to high-order boundary representation, *Journal of Computational Physics*, Vol. 211, pp. 154-178, 2006.

[11]E.M. Ronquist, A.T.Patera, Spectral Element Multigrid, I. Formulation and numerical results, *Journal of Scientific Computing*, Vol 2, pp. 389-406, 1987

[12]Y. Maday, R. Munoz, Spectral element multigrid, Part 2: Theoretical justification, Tech. Rep 88-73, ICASE, 1988

[13]Y. Sun, Z. J. Wang, & Y. Liu, Efficient Implicit LU-SGS Algorithm for high-order spectral difference method on unstructured hexahedral grids, *AIAA paper*, AIAA-2007-313, 2007.

[14]F. Bassi & S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations, *Journal of Computational Physics*, Vol. 131, pp.267-279, 1997

[15]V.V. Rusanov, Calculation of interaction of non-steady shock waves with obstacles, *Journal of Computational and Mathematical Physics USSR*, Vol. 1, pp. 267-279, 1961.

[16]A. Jameson & S. Yoon, Lowerupper implicit schemes with multiples grids for the Euler equations, *AIAA Journal*, Vol. 25, pp. 929935, 1987.

[17]S. Yoon & A. Jameson, A Lower-upper symmetric-Gauss-Seidel method for the Euler and Navier-Stokes equations, *AIAA Journal*, Vol. 26, pp. 1025-1026, 1988.

[18]R. J. Spiteri, S. J. Ruuth, A new class of optimal high-order strong-stability-preserving time discretization methods, *SIAM J. Numer. Anal.* , Vol. 40, pp. 469-491, 2002.

[19]H.T. Huynh, A Flux Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin Methods, *AIAA paper*, AIAA-2007-4079, 2007.

[20]R. Kannan, Y. Sun, and Z.J. Wang, A Study of Viscous Flux Formulations for an Implicit p-Multigrid Spectral Volume Navier Stokes Solver, *AIAA paper*, AIAA-2008-783, 2008.

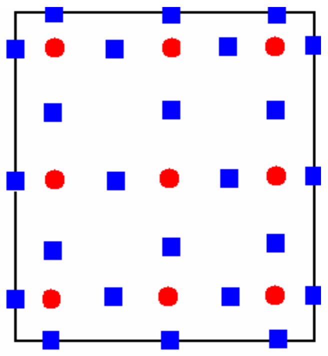
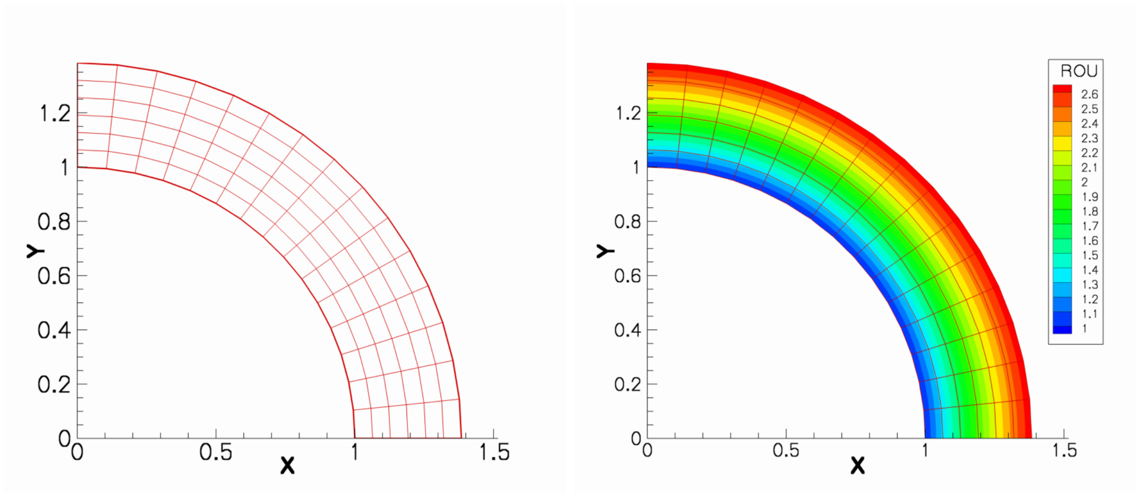
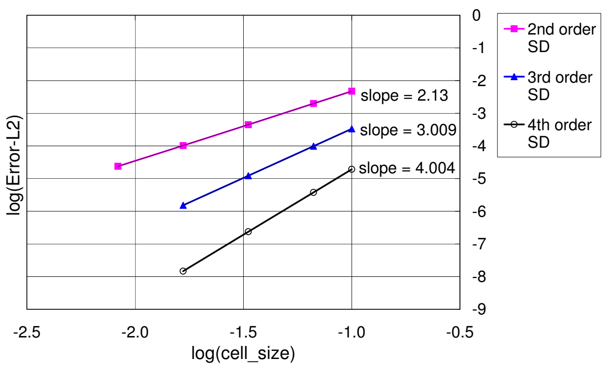


**Figure 1. Position of solution (circles) and flux (squares) points on standard square element for 3rd order SD**

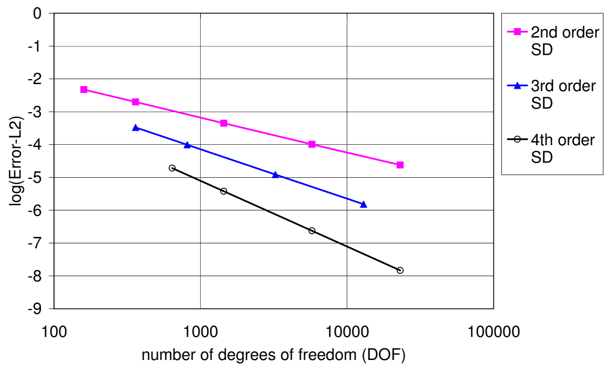


(a) (b)

**Figure 2. (a) $15 \times 6$ mesh for supersonic vortex flow test case; (b) Density contours using 3rd order SD.**



(a)

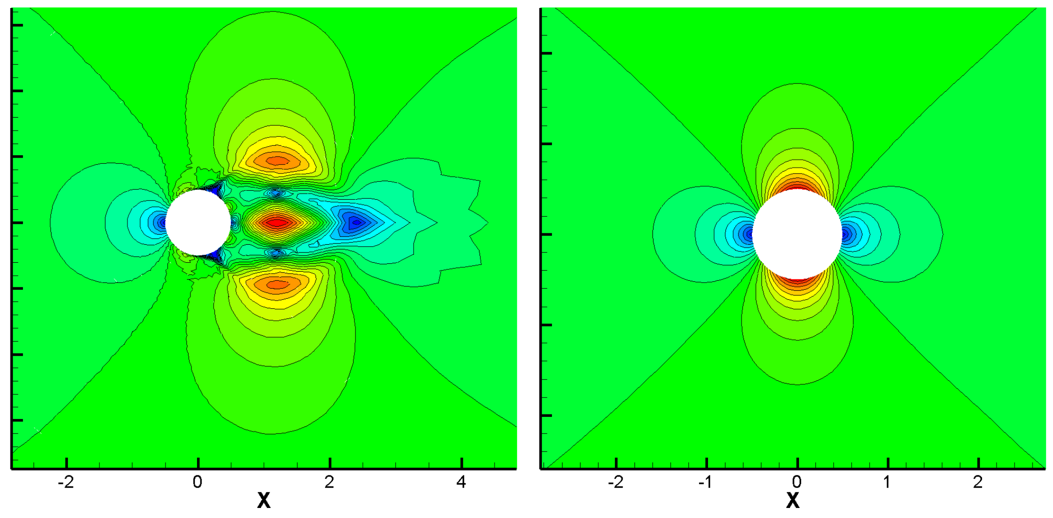


(b)

**Figure 3. Accuracy summary for the supersonic vortex flow case for SD method using 2nd, 3rd and 4th order solution approximations (a) $L^2$ error vs cell-size; (b) $L^2$ error vs degrees of freedom.**

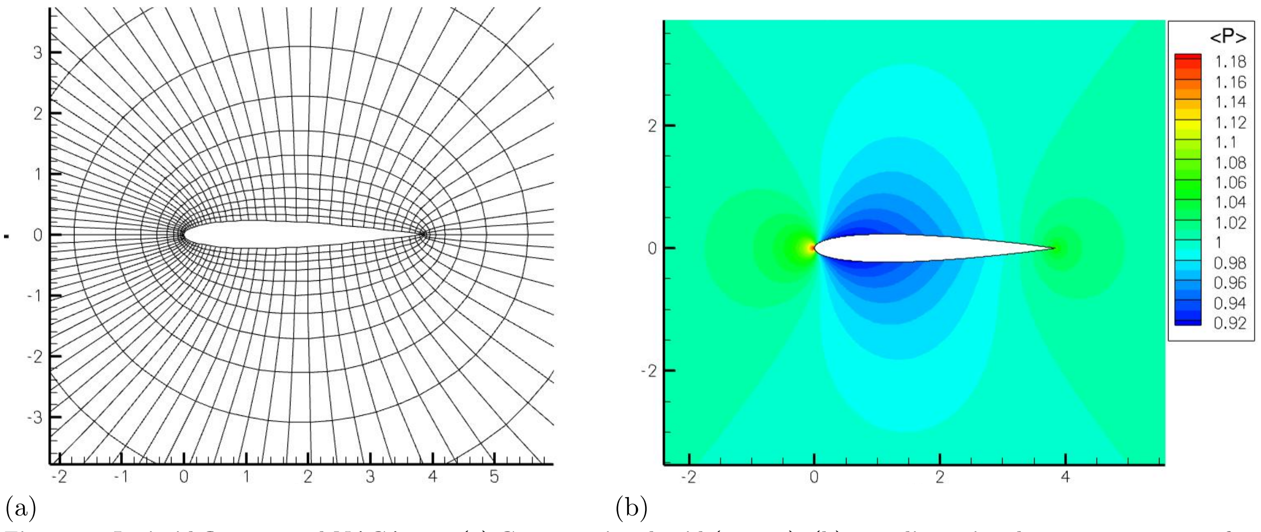
**Table 1. $L^2$ errors and orders of accuracy for supersonic vortex case**

| No. of elements | No. of DOFs | L2-error | Order |
|---|---|---|---|
| 2nd order SD | | | |
| 40 | 160 | 4.7249E-03 | - |
| 90 | 360 | 1.9881E-03 | 2.135 |
| 360 | 1440 | 4.4721E-04 | 2.152 |
| 1440 | 5760 | 1.0196E-04 | 2.133 |
| 3rd order SD | | | |
| 40 | 360 | 3.3393E-04 | - |
| 90 | 810 | 9.8833E-05 | 3.003 |
| 360 | 3240 | 1.2242E-05 | 3.013 |
| 1440 | 12960 | 1.5230E-06 | 3.007 |
| 4th order SD | | | |
| 40 | 640 | 1.9238E-05 | - |
| 90 | 1440 | 3.7883E-06 | 4.008 |
| 360 | 5760 | 2.3651E-07 | 4.002 |
| 1440 | 23040 | 1.4743E-08 | 4.004 |



(a) (b)

**Figure 4. Mach contours for inviscid flow past a circle (a) 4th order SD with linear boundary (b) 4th order SD with cubic boundary**



(a) (b)

**Figure 5. Inviscid flow around NACA0012 (a) Computational grid ($80 \times 16$); (b) non-dimensional pressure contours for 3rd order SD.**

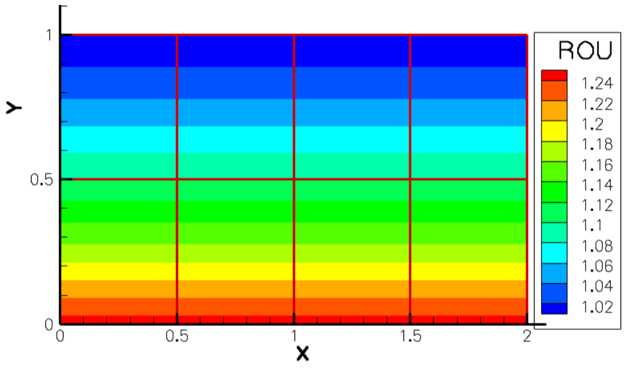


**Figure 6. Density contours for Couette flow problem using 3rd order SD** ($4 \times 2$ **grid)**

**Table 2.** $L^2$ **errors and orders of accuracy for Couette flow case**

| No. of elements | No. of DOFs | L2-error | Order |
|---|---|---|---|
| 2nd order SD | | | |
| 2 | 8 | 1.4180E-02 | - |
| 8 | 32 | 3.3520E-03 | 2.081 |
| 32 | 128 | 9.1210E-04 | 1.878 |
| 128 | 512 | 2.4350E-04 | 1.905 |
| 3rd order SD | | | |
| 2 | 18 | 1.4783E-03 | - |
| 8 | 72 | 1.5199E-04 | 3.282 |
| 32 | 288 | 1.6525E-05 | 3.201 |
| 128 | 1152 | 1.7991E-06 | 3.199 |
| 4th order SD | | | |
| 2 | 32 | 1.9784E-04 | - |
| 8 | 128 | 1.1827E-05 | 4.064 |
| 32 | 512 | 7.9780E-07 | 3.980 |
| 128 | 2048 | 4.7330E-08 | 4.075 |

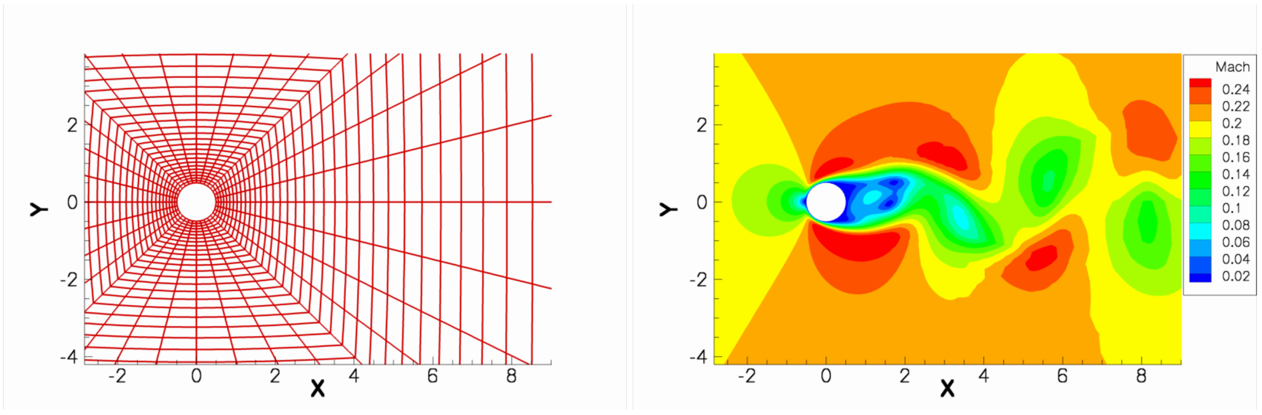


**Figure 7. Viscous flow around circle (a) Computational grid; (b) Mach number contours using 4th order SD.**

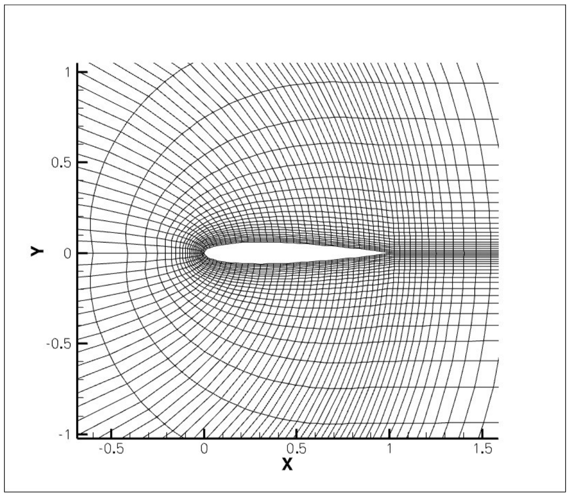
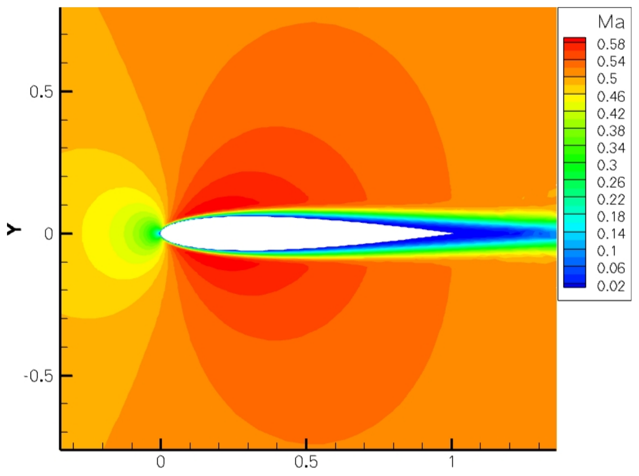
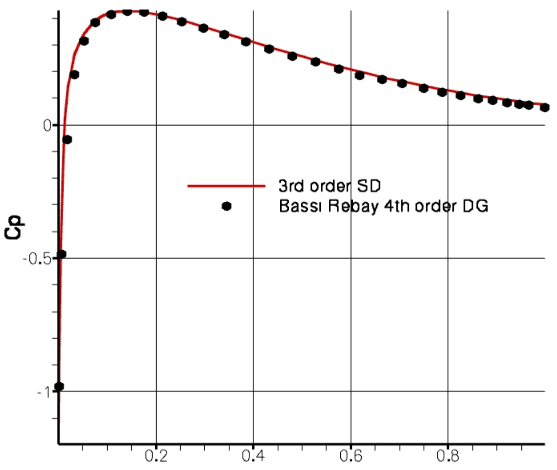


**Figure 8. 120 × 24 C-grid for NACA0012 airfoil**





(a) (b)

**Figure 9. Viscous flow around NACA0012 (a) Mach number contours using 3rd order SD; (b) Surface $C_p$ distribution.**

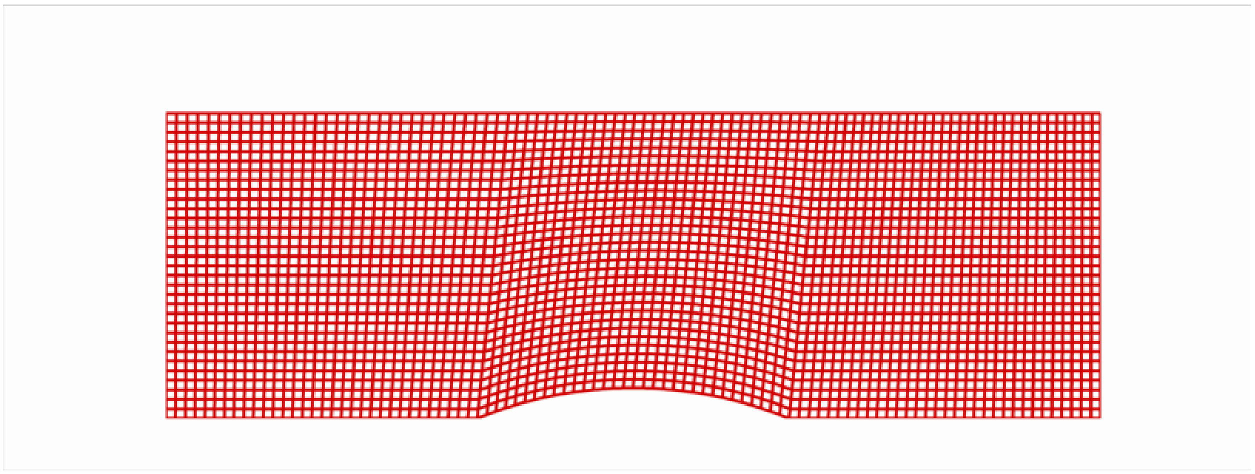


**Figure 10. Grid used for subsonic bump case**

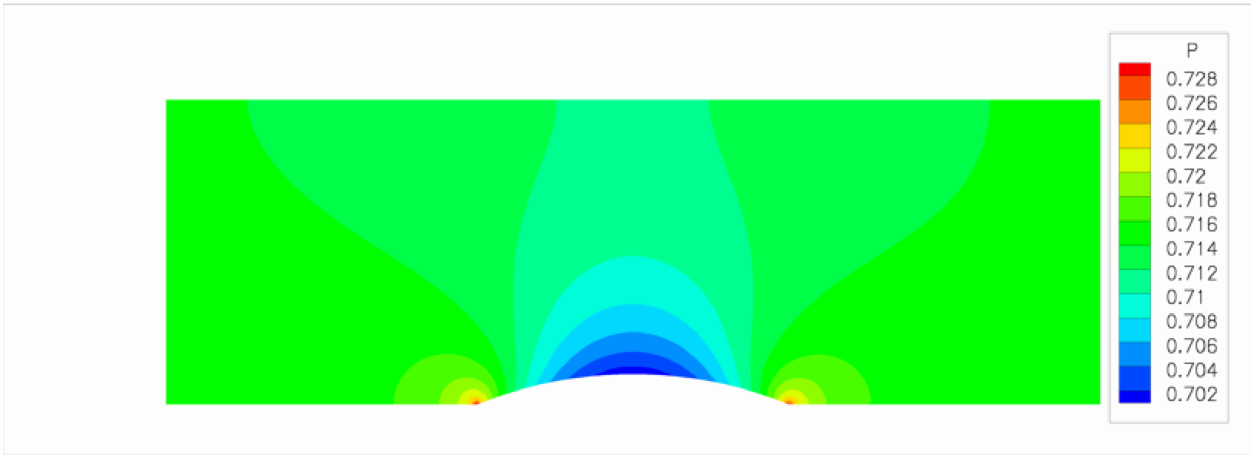
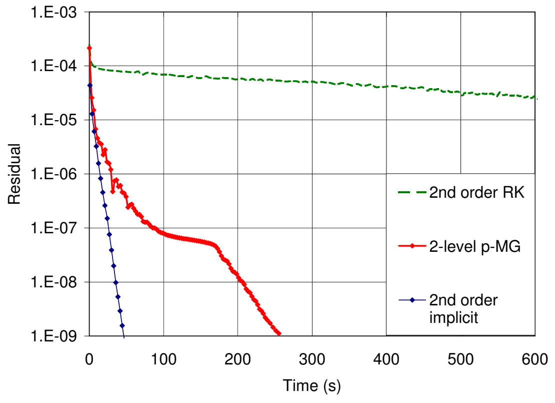
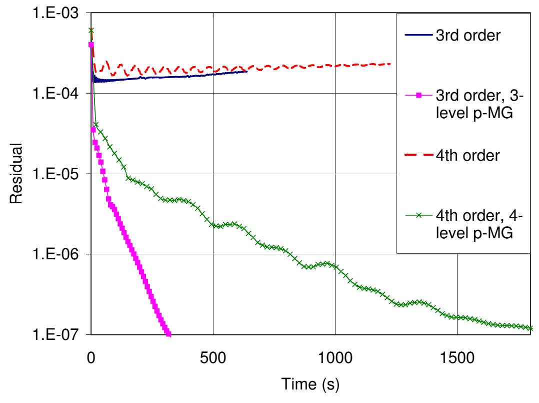


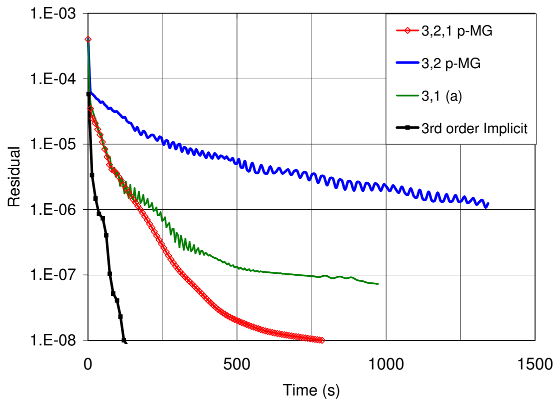
**Figure 11. Pressure Contours using 4th order SD**

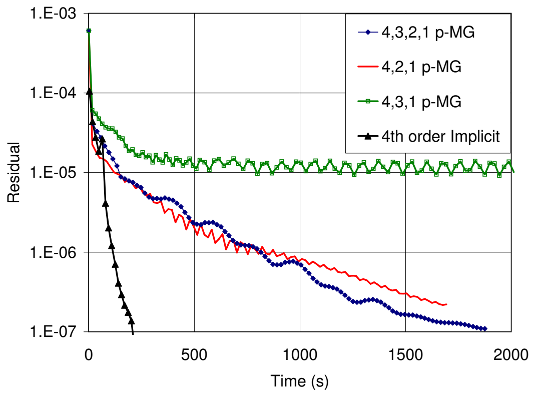




(a) (b)

**Figure 12. Convergence history for subsonic flow over bump testcase (a) Plot of residual vs CPU time for 2nd order computations; (b) Plot of residual vs. CPU time for 3rd and 4th order computations**





(a) (b)

**Figure 13. Convergence history for subsonic flow over bump testcase (a) Plot of residual vs CPU time for 3rd order computations; (b) Plot of residual vs. CPU time for 4th order computations.**

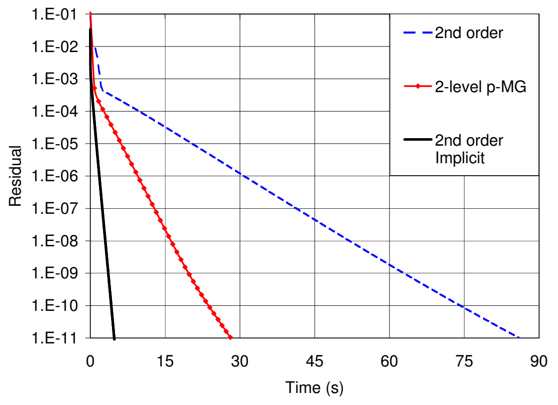


**Figure 14. Convergence history for 2nd order SD Couette flow case**

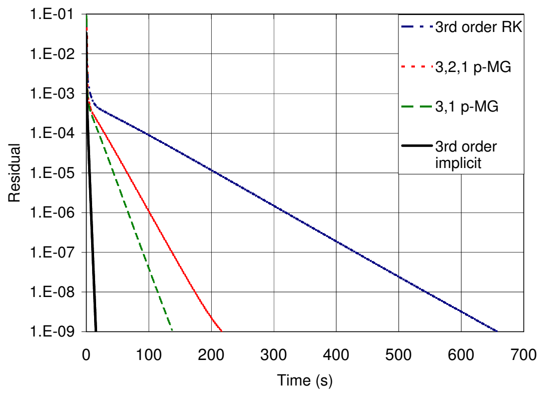


**Figure 15. Convergence history for 3rd order SD Couette flow case**

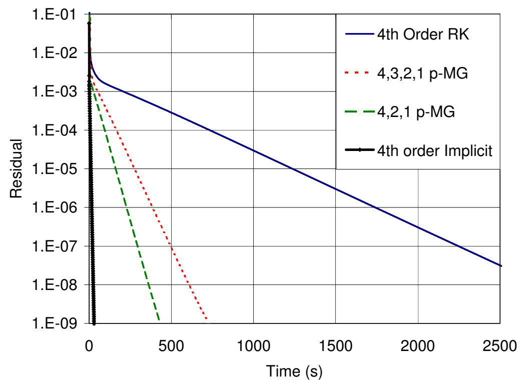


**Figure 16. Convergence history for 4th order SD Couette flow case**

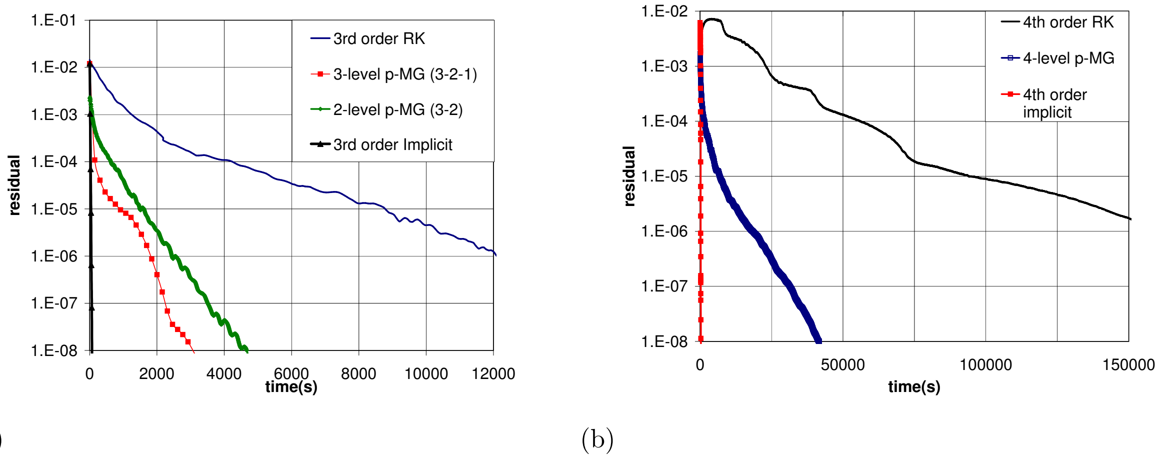


**Figure 17. Convergence history for inviscid flow past NACA0012. (a) Plot of residual vs CPU for 3rd order computations; (b) Plot of residual vs. CPU time for 4th order computations.**

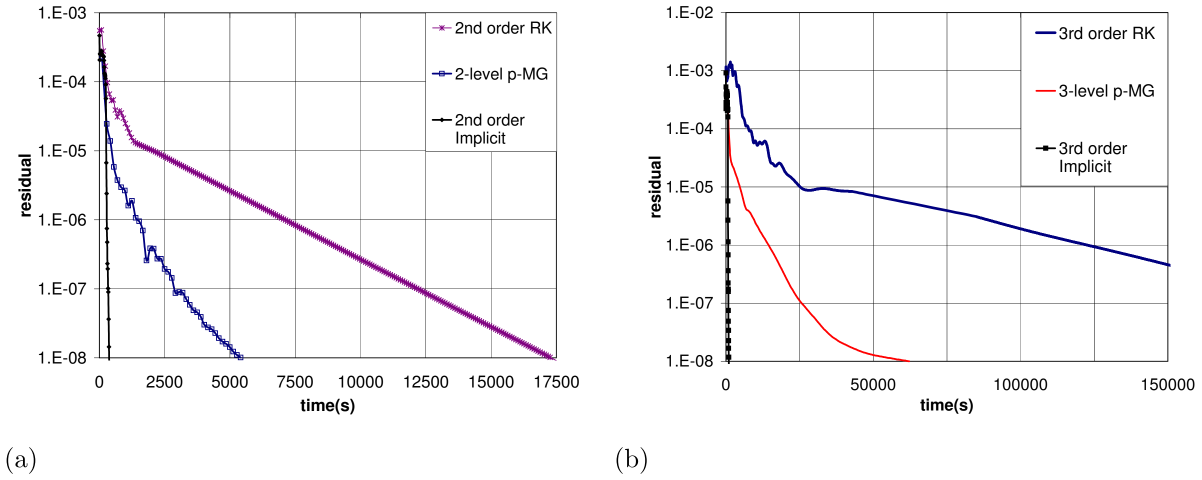


**Figure 18. Convergence history for viscous flow past NACA0012. (a) Plot of residual vs CPU for 2nd order computations; (b) Plot of residual vs. CPU time for 3rd order computations.**