A $p$-multigrid spectral difference method with explicit and implicit smoothers on unstructured triangular grids

C. Liang*, R. Kannan, Z.J. Wang

Department of Aerospace Engineering, Iowa State University, 2271 Howe Hall, Ames, IA 50011, United States

ARTICLE INFO

Article history:
Received 19 July 2007
Received in revised form 19 February 2008
Accepted 22 February 2008
Available online 5 March 2008

ABSTRACT

The convergence of high-order methods, such as recently developed spectral difference (SD) method, can be accelerated using both implicit temporal advancement and a $p$-multigrid ($p$-polynomial degree) approach. A $p$-multigrid method is investigated in this paper for solving SD formulations of the scalar wave and Euler equations on unstructured grids. A fast preconditioned lower–upper symmetric Gauss–Seidel (LU-SGS) relaxation method is implemented as an iterative smoother. Meanwhile, a Runge–Kutta explicit method is employed for comparison. The multigrid method considered here is nonlinear and utilizes full approximation storage (FAS) [Ta’asan S. Multigrid one-shot methods and design strategy, Von Karman Institute Lecture Note, 1997 [28]] scheme. For some $p$-multigrid calculations, blending implicit and explicit smoothers for different $p$-levels is also studied. The $p$-multigrid method is firstly validated by solving both linear and nonlinear 2D wave equations. Then the same idea is extended to 2D nonlinear Euler equations. Generally speaking, we are able to achieve speedups of up to two orders using the $p$-multigrid method with the implicit smoother. The implicit methods are normally formulated by the linearization of a given set of equations. Batten et al. [3] modified the exact Riemann solver fluxes based on average-state flux Jacobians. They obtained better convergence rate and more accurate solutions compared to the schemes containing less detailed physics. The development of implicit methods for higher order methods like discontinuous Galerkin (DG) methods [17,18,23,32] has been very noteworthy. Even though these implicit methods offer extremely high speedup, they need huge memory to store the associated matrices. This is greatly felt for higher order methods. One alternative way to mitigate the above problem is to use the traditional multi stage Runge–Kutta method as the higher level smoother and the implicit scheme for the lower levels [17].

The $p$-multigrid method is a smoothing operator which hastens the convergence by using the coarse levels constructed by reducing the level of the interpolation polynomial $p$. This method was initially proposed by Ronquist and Patera [25], and extended by Maday and Munoz [19]. The acceleration of the convergence by the $p$-multigrid algorithm on Euler equations was demonstrated by Bassi and Rebay [2], Fidkowski et al. [8], Nastase and Mavriplis [21] and Luo et al. [17] with the DG method. Helenbrook et al. [10] also performed Fourier analysis of the $p$-multigrid for both diffusive and convective systems. They suggested isotropic multigrid iteration may not efficient for anisotropic convective system.

In this paper, we adopt an efficient implicit LU-SGS approach [4,13] as the iterative smoother for the $p$-multigrid algorithm. The original LU-SGS approach was developed by Yoon and

* Corresponding author. Present address: Department of Aeronautics and Astronautics, Stanford University, 496 Duand Building, Lamita Mall, Stanford, CA 94305, United States. Tel.: +1 650 723 6021. E-mail address: chliang@stanford.edu (C. Liang).
Jameson on structured grids [12,35]. The preconditioned LU-SGS approach further improved the efficiency. For comparison purpose, we also use explicit TVD Runge–Kutta scheme for the p-multigrid smoothers. We consider both the scalar wave equation as well as the Euler equations. Efforts are made to demonstrate clearly both the effect of implicit scheme and the effect of p-multigrid scheme on convergence speedup. The fastest approach employing the p-multigrid method with an implicit smoother is recommended in this paper.

As explained in the earlier paragraph, the high-order implicit method require much higher storage than the explicit method. Therefore, in this paper, we also intend to adopt explicit smoothers for the high p-levels and simultaneously employ implicit schemes embedded with much smaller size of Jacobian matrix on low p-levels.

The paper is organized as follows. Section 2 presents the governing equations using the corresponding SD method. Temporal relaxation schemes are discussed in Section 3. In Section 4 we focus on the nonlinear FAS p-multigrid method. Section 5 includes the validations over the wave equations for accuracy and speedup followed by the results and discussion of the Euler equations. Some conclusions are drawn in Section 6.

2. Governing equations and the SD formulation

The SD method combines the salient features of structured and unstructured grid methods to achieve high computational efficiency and geometric flexibility. It utilizes the concept of discontinuous and high-order local representations to achieve conservation and high accuracy. Universal reconstructions are obtained by distributing unknown and flux points in a geometrically similar manner for all unstructured cells. Fig. 1 shows the placement of unknown and flux points for a triangular cell. In this paper, we consider 1st-order (p0), 2nd-order (p1) and 3rd-order (p2) schemes. The unknowns are updated using the differential form of the conservation law equations by approximating the flux derivatives at these unknown points. In order to obtain the flux derivatives, we use a polynomial reconstruction of the fluxes from their values at available flux points to the unknown points. As a result, the method is defined as a difference method. The SD method is similar to the multi-domain spectral method developed by Kopriva [14,15] and can be viewed as an extension of the multi-domain spectral method to a simplicial unstructured grid.

We could write some linear or nonlinear equations in 2D conservative form

$$\frac{\partial Q}{\partial t} + \nabla \cdot \mathbf{f} + \frac{\partial g}{\partial y} = 0 \quad \text{and} \quad R_i(Q) = -\frac{\partial f}{\partial x} - \frac{\partial g}{\partial y}, \quad (2.1)$$

If the unsteady 2D Euler equations are considered, Q is a vector of conservative variables, f and g are the inviscid fluxes are given by

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \quad f = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E + p) \end{bmatrix}, \quad g = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E + p) \end{bmatrix}, \quad (2.2)$$

where \( \rho \) is the density, \( u \) and \( v \) are the velocity components in x and y directions, \( p \) is the pressure, and \( E \) is the total energy. The pressure is related to the total energy by \( E = \frac{1}{2} \rho (u^2 + v^2) \) with constant ratio of specific heats \( \gamma = 1.4 \) for air.

The current cell residual term \( R_i(Q) \) can be evaluated once the neighbouring three cells are known. We can denote the unknown and flux points for cell i as \( r_{i,j} \) and \( r_{i,p} \), respectively. The solutions of Q at flux points can be conveniently constructed using a Lagrange-type polynomial basis function \( L_{I,j}(r) \) as

$$Q(r_{i,k}) = \sum_{j=1}^{N_p} L_{I,j}(r_{i,k}) Q_{i,j}, \quad (2.3)$$

where \( N_p \) is the number of unknown points required to support a degree \( p \) polynomial construction as already illustrated in Fig. 1. As a result, \( Q(r_{i,k}) \) is continuous inside individual cell element, while across the element interfaces, it is discontinuous and the inviscid fluxes \( f(r_{i,k}) \) and \( g(r_{i,k}) \) are not uniquely determined. We employ one-dimensional Riemann solvers, namely, the Rusanov [26] or Roe [24] flux to obtain a unique normal component of the flux \( F(r_{i,k}) = f(r_{i,k}) + g(r_{i,k}) \) at the element boundary interface for an edge point. The local cell values are used for the tangential components as shown in Fig. 2, i.e. \( (F(Q_{i,j}) \cdot l, F_{i}) \) and \( (F(Q_{i,p}) \cdot l, F_{p}) \) for the left cell and right cell, respectively, where \( l \) is unit tangential as shown in Fig. 2. However, for the corner flux points, multiple values are allowed for different cells using the procedure as shown in [31]. In other words, two faces are associated with a particular corner point of a cell and we can use either the Rusanov or Roe flux to compute unique normal components of the two fluxes, i.e. \( F_{i1} \) and \( F_{i2} \). A flux vector \( F(r_{i,j}) \) at this corner point for this particular cell can be constructed using \( F_{i1} \) and \( F_{i2} \). Once all the flux vectors are determined, they are used to form a degree \( p + 1 \) polynomial, i.e. one order higher than the polynomial used in (2.3). The flux at any location can be expressed as follows:

$$F_i(r) = \sum_{k=1}^{N_{p+1}} M_{k,i}(r) F_{k,i}, \quad (2.4)$$

where \( M_{k,i}(r) \) are the set of shape functions defined uniquely by the flux point locations. We are now ready to compute the divergence of the flux at any locations inside the particular cell and for the unknown point locations, they can easily computed according to
\[ \nabla \cdot F_i(t_{ij}) = \sum_{k=1}^{N_{ij}} \nabla M_{ki}(t_{ij}) \cdot F_{ki}. \] (2.5)

The residual term \( R_i(Q) \) used in (2.1) is simply the negative value of the divergence. Once the SD spatial discretization is completed, attention now is turned to temporal relaxation schemes.

3. Time relaxation schemes

In order to solve the flow to a steady state from a nearly arbitrary initial guess, a relaxation scheme is needed. The unsteady equation is considered in the current paper and the time integration schemes (either explicit or implicit schemes) can be used as a smoother.

\[ \frac{\partial Q}{\partial t} = R_i(Q) = -\nabla \cdot F_i(t_{ij}). \] (3.1)

An overview of the schemes used for the current study is presented in the following. Efforts are made to list their advantages and disadvantages. This enables us to selectively combine these different approaches and reap the maximum benefit in terms of CPU time and memory storage.

3.1. Explicit scheme

The three-stage explicit Runge–Kutta can be written as follows:

\[ Q_i^{(1)} = Q_i^n - \Delta t R_i(Q_i^n), \]
\[ Q_i^{(2)} = \frac{3}{4} Q_i^n + \frac{1}{4} [Q_i^{(1)} - \Delta t R_i(Q_i^{(1)})], \]
\[ Q_i^{n+1} = \frac{1}{3} Q_i^n + \frac{2}{3} [Q_i^{(2)} - \Delta t R_i(Q_i^{(2)})]. \] (3.2)

The main advantage of the Runge–Kutta scheme is that it requires little memory for storage. In addition, this method is inherently simple and easy to program. These are the main reasons for it being one of the most preferred methods of time integration.

The main bottleneck associated with the Runge–Kutta scheme is the limit imposed on the time step. Euler (and Navier–Stokes) equations for realistic geometries entail a rather strict limit on the time step. Even though the above can be circumvented by using a very higher order (several RK steps) scheme, it is seldom used as it required lots of storage and thus adversely affects its simplistic nature. Therefore, one of the primary efforts in this paper is devoted to a new LU-SGS scheme and it is explained in the following.

3.2. Implicit scheme

At each current cell \( c \), using the backward Euler difference, (2.1) can be written as

\[ \frac{Q_i^{n+1} - Q_i^n}{\Delta t} - R_i(Q_i^{n+1}) = R_i(Q_i^n). \] (3.3)

Let \( \Delta Q_c = Q_i^{n+1} - Q_i^n \) and linearizing the residual, we obtain

\[ R_c(Q_i^{n+1}) - R_i(Q_i^n) \approx \frac{\partial R_c}{\partial Q_c} \Delta Q_c + \sum_{nb=c} \frac{\partial R_c}{\partial Q_{nb}} \Delta Q_{nb}, \] (3.4)

where \( nb \) indicates all the neighbouring cells contributing to the residual of cell \( c \). Therefore, the fully linearized equations for (3.3) can be written as

\[ \left( \frac{I}{\Delta t} - \frac{\partial R_c}{\partial Q_c} \right) \Delta Q_c - \sum_{nb=c} \frac{\partial R_c}{\partial Q_{nb}} \Delta Q_{nb} = R_c(Q_i^n). \] (3.5)

However, it costs too much memory to store the LHS implicit Jacobian matrices. An elegant way to circumvent this difficulty is to take advantage of the unstructured nature of our grid topology. We thus employ a LU-SGS scheme to solve (3.5), i.e. the most recent solution for the neighbouring cells,

\[ \left( \frac{I}{\Delta t} - \frac{\partial R_c}{\partial Q_c} \right) \Delta Q_{c(k-1)} = R_c(Q_i^n) + \sum_{nb=c} \frac{\partial R_c}{\partial Q_{nb}} \Delta Q_{nb}. \] (3.6)

It is noted here that the superscript "*n" refers to the most recent solution when doing forward or backward sweeps for LU-SGS scheme. \( Q_{nb} \) refer to the most recent solutions of conserved variables for neighbour cells. We use superscript "*k + 1" to note that a new solution is obtained.

The matrix \( D = \left( \frac{I}{\Delta t} - \frac{\partial R_c}{\partial Q_c} \right) \) is the element (or cell) matrix, which is not quite small for 2nd- to 3rd-order SD schemes. For instance, the matrix \( D \) has 36 elements for the 3rd-order SD wave equation and 576 elements for the 3rd Euler equations for two-dimensional problems. For 3D problems, the size of \( D \) will be further increased. In Eq. (3.6), the off-diagonal matrices \( \frac{\partial R_c}{\partial Q_{nb}} \) has a much bigger size. Using the relations of \( Q_{nb} = \Delta Q_{nb} + Q_{nb}^c \) and \( Q_i^c = \Delta Q_i^c + Q_i^c \), we can propose the following linearization approach to avoid storing the off-diagonal matrices,

\[ R_c(Q_i^n) + \sum_{nb=c} \frac{\partial R_c}{\partial Q_{nb}} \Delta Q_{nb} = R_c(Q_i^n, Q_{nb}^c) + \sum_{nb=c} \frac{\partial R_c}{\partial Q_{nb}} \Delta Q_{nb}^c \approx R_c(Q_i^n, Q_{nb}^c) \approx R_c(Q_i^n, Q_{nb}^c) - \frac{\partial R_c}{\partial Q_c} \Delta Q_c^c = R_c(Q_i^n, Q_{nb}^c) - \frac{\partial R_c}{\partial Q_c} \Delta Q_c^c. \] (3.7)

Let \( \Delta Q_{c(k-1)} = \Delta Q_{c(k-1)} - \Delta Q_{c(k-1)}^* \) where the superscript "*2" refers to an additional differencing over iteration step. We can combine (3.6) and (3.8) together to obtain a diagonally dominant system

\[ \left( \frac{I}{\Delta t} - \frac{\partial R_c}{\partial Q_c} \right) \Delta Q_{c(k-1)} = R_c(Q_i^n) - \Delta Q_{c(k-1)}^* \] (3.9)

Eq. (3.9) is then solved with a direct LU decomposition solver and sweeping symmetrically forward and backward through the com-
putational grid. Note that once (3.9) is solved to machine zero, the unsteady residual is zero at each time step. For steady flow problem, we found that the term \( \frac{\Delta Q^p_{\mu}}{\Delta t} \) in the right-hand side of (3.9) can be neglected and leading to quicker convergence. Note that \( \Delta Q^p_{\mu} \) is the difference between the current solution \( Q^p_{\mu} \) and the solution at the previous time level \( Q^p \). In reality, the entire system is swept several times in order to proceed to the next time level. As a result, \( \Delta Q^p_{\mu} \) is influenced by the solution occurred several sweeps ago. This introduces an under-relaxation effect. Hence neglecting the \( \frac{\Delta Q^p_{\mu}}{\Delta t} \) term accelerates the convergence. We define the solver obtained using (3.9) as implicit normal approach. If \( \frac{\Delta Q^p_{\mu}}{\Delta t} \) term is dropped, the iterative solver is defined as implicit simplified approach. In Section 5, we will demonstrate higher efficiency achieved using the implicit simplified approach for a linear wave equation. The implicit simplified approach is then applied to the Euler equation and unless otherwise stated, the following discussion for the implicit schemes refers to the simplified approach.

4. p-Multigrid method

The main idea of a multigrid method is related to the observation that error-smoothing operators are generally very efficient in damping high-frequency errors, but much less capable of removing low-frequency errors. In other words, the Gauss–Seidel or Jacobi iterations produce low-frequency errors when applied on the aforementioned nonlinear equations. The error field has its high frequencies nearly removed in a few iterations using a higher order polynomial; but low frequencies are removed very slowly. Thus the p-multigrid method is to solve the nonlinear equations with a lower order polynomial grid on which “smooth becomes rough” and low frequencies act like high frequencies. Such a p-multigrid method has been used for high-order discontinuous Galerking method [9,17,19,34] and one-dimensional spectral volume method [29]. 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 [11].

To communicate between different levels, restriction \( (p_{-1}, p_{-2}) \) and prolongation \( (p_{+1}, p_{+2}) \) operators are required in addition to the aforementioned relaxation scheme as a smoother. Restriction consists of moving solutions and their residuals at the unknown points from a space of higher polynomial order to a lower polynomial order. Prolongation refers to a reverse procedure in which lower order polynomial solution correction is redistributed as corrections to the solutions of the unknown points at a higher polynomial order.

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 low-frequency 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 the lower polynomial level problems leads to multigrid method.

The p-multigrid procedure is summarized below:

Defining three polynomial levels from the highest to the lowest as \( p, p-1 \) and \( p-2 \), we want to solve:

\[
R_p(Q^p) = r_p
\]  
(4.1)

and the RHS \( r_p \) is zero for the highest level polynomial;

\[
R_{p-1}(Q^p_{p-1}) = r_{p-1},
\]  
(4.2)

\[
R_{p-2}(Q^p_{p-2}) = r_{p-2}.
\]  
(4.3)

The following illustration employs the implicit LU-SGS schemes as the smoothers for all three levels. The following steps are used to update the solutions on the highest \( p \) level in one three-level V cycle.

- Improve \( Q^p \) by iterating a few backward and forward sweeps of the smoother similar as Eq. (3.7)

\[
\left[ \frac{1}{\Delta t} \frac{\partial R^p}{\partial Q^p} \right]^n \Delta^2 Q^p = R_p(Q^p) - r_p.
\]  
(4.4)

Once again, we only compute the left-hand side Jacobian matrix at the end of each time step. The superscript “n” denotes evaluation operated at previous time step.

- Restrict the latest solution \( Q^p \) to the coarser level for an approximate solution \( Q_{p-1}^p \)

\[
Q_{p-1}^p = R_p^{-1}(Q^p).
\]  
(4.5)

where the superscript “0” denotes the “starting” phase for the restricted solution at level \( p-1 \).

- Compute the defect on the finest level

\[
d_p = r_p - R_p(Q^p) = -r_p(Q^p).
\]  
(4.6)

- Compute the right-hand side of Eq. (4.2) as

\[
r_{p-1} = R_{p-1}(Q_{p-1}^p) + p_{-1}^{-1}d_p.
\]  
(4.7)

This equation also means that once the solution of the \( p \)-level grid problem was obtained, this \( p-1 \) level grid correction does not introduce any changes via interpolation.

- Improve \( Q^p_{p-1} \) by applying the smoother for a few sweeps

\[
\left[ \frac{1}{\Delta t} \frac{\partial R^p}{\partial Q^p} \right]^n \Delta^2 Q^p = R_p^{-1}(Q^p_{p-1}) - r_{p-1}.
\]  
(4.8)

- Restrict the latest solution \( Q^p_{p-1} \) obtained by \( Q^p_{p-1} = Q^p_{p-1} + \Delta^2 Q^p_{p-1} \) to the coarser level for an approximate solution \( Q_{p-2}^p \), where the superscript “s” denotes the “starting” phase for the restricted solution at level \( p-2 \)

\[
Q_{p-2}^p = R_{p-2}(Q^p_{p-2}) + p_{-2}^{-1}d_{p-1}.
\]  
(4.9)

- Define the defect on the intermediate level as

\[
d_{p-1} = r_{p-1} - R_{p-1}(Q^p_{p-1}).
\]  
(4.10)

- Compute the right-hand side of Eq. (4.3) as

\[
r_{p-2} = R_{p-2}(Q^p_{p-2}) + p_{-2}^{-1}d_{p-1}.
\]  
(4.11)

- Improve \( Q^p_{p-2} \) by iterating a few backward and forward sweeps of the smoother

\[
\left[ \frac{1}{\Delta t} \frac{\partial R^p}{\partial Q^p} \right]^n \Delta^2 Q^p_{p-2} = R_{p-2}(Q^p_{p-2}) - r_{p-2}.
\]  
(4.12)

- Correct the current solution on the intermediate level by

\[
Q_{p-1} = Q_{p-1} + p_{-1}^{-1}(Q^p_{p-2} - Q^p_{p-2}).
\]  
(4.13)

- Improve \( Q^p_{p-1} \) by application of a few sweeps of the smoother same as Eq. (4.8) to get \( Q^p_{p-1} \)

- Correct the current solution on the finest level by

\[
Q^p = Q^p + p_{-1}^{-1}(Q^p_{p-1} - Q^p_{p-1}).
\]  
(4.14)

Note that the defect is calculated using the \( p-1 \) level “starting” phase solution obtained in (4.5).

- Improve \( Q^p \) by application of a few times the iterative smoother (the same as Eq. (4.4)) to get \( Q^p \)
Note that only the implicit LU-SGS smoother is described in the above procedure for simplicity. In practice, we can replace any of the implicit smoothers ((4.4), (4.8) or (4.12)) with a three-stage or five-stage TVD Runge–Kutta scheme. If extension to 3D solver is considered, we intend to use the p2 level with an explicit scheme since its storage is small and implicit LU-SGS for the p1 and p0 levels. We define this as a mixed-smoother three-level scheme. In this paper, we also consider using explicit schemes as smoothers for all three levels. It is defined as an R–K smoother three-level scheme.

If we iterate the steps marked with hollow circle bullets (∗) for one more time, a three-level W cycle is formed. If these steps are removed, it will be a classical two-level scheme as aforementioned. Therefore, the actual implementation is realized using recursive routines.

Note that to numerically compute the D matrix at different p levels, we adopted a rediscretization approach based on the distribution of unknown and flux points at different levels as shown in Fig. 1. The procedure produces very stable results. However, Helenbrook and Atkins [9] used an algebraic approach to get a subset matrix from a higher order polynomial level, i.e. \( A_{p+1} = \frac{p}{(p+1)} A_p^{-1} \) for the DG method. They found that it works better than the rediscretization approach for the Poisson equation. For the Poisson equation, there is a static condensation of the diffusive flux variables. Diffusive DG schemes have adjustable parameters which can change with polynomial order p. As a result, the algebraic approach is more consistent for their case. In the present 3D method, we need to take into account the effect of neighbouring cells (at local polynomial level) during the calculations of fluxes. The rediscretization approach utilizes the most recently reconstructed solutions at the flux points. In contrast, the algebraic approach is very local and only takes information from the current cell. Our numerical tests suggested that the rediscretization approach gives more accurate results. Therefore, all the p-multigrid results presented in this paper are obtained using the rediscretization approach unless otherwise stated.

5. Numerical results and discussion

5.1. Validation using 2D scalar conservation laws

5.1.1. Linear straight wave

We consider the linear straight wave over a square domain. It is also considered in [5] using the explicit scheme. The equation can be described as

\[
\frac{\partial q}{\partial t} + \frac{\partial q}{\partial x} + \frac{\partial q}{\partial y} = 0, \quad (x, y) \in [0, 1]^2 \quad \text{and} \quad t > 0. \tag{5.1}
\]

The computational unstructured grid with 3200 elements is used as shown in Fig. 3. The mesh is firstly generated using quadrilateral elements. The left and bottom boundaries are prescribed with analytical solution \( q(x, y) = \sin(\pi(x + y) - 2\pi) \). The outer values of flux points on the right and top boundaries are extrapolated from the inner cell solutions on the corresponding flux points. We found that all our different schemes used here are not sensitive to the initial condition. The computational grid is relatively coarse and the equation is very simple. We maintained a large maximum CFL number around 10 (\( c_\text{c} = 9.87 \)) for the calculations with implicit scheme. The maximum CFL number is only 0.22 for five-stage explicit R–K scheme. The CFL number is defined by \( c_\text{c} = \frac{\Delta t \sum_{i=1}^{N} V_i A_i}{\Omega} \), where \( \Omega \) is the cell volume and \( A_i \) is the face area. In each time step, five forward and five backward sweeps are sufficient and used for this case. When running the numerical tests, we normally start from a very small CFL number. We then gradually increase the CFL number until it reaches a maximum CFL number where a little more over it will result in solution divergence. The speedup obtained using this particular CFL number is finally reported in the paper. Furthermore, the elements are ordered in an unstructured fashion. For instance element number 5 could be far apart from number 6. However, when we do forward and backward sweeps, we could simply sweep forward through 1 to N and backward from N to 1 to maintain a symmetrical property (where N is the total number of elements).

To illustrate the effect of implicit scheme on convergence speed, we plot the Residual averaged over all computational elements (i.e. Residual refers to \( R(q) = -\frac{\partial q}{\partial x} - \frac{\partial q}{\partial y} \)) against CPU time in seconds. Fig. 4 demonstrates that the simplified implicit scheme without \( \frac{\partial q}{\partial x} \) term in Eq. (3.9) converging speed is about seven times as fast as the one of the explicit scheme, while the normal implicit scheme converging speed is only twice as fast as the explicit scheme. We need less than 20 iteration steps to reduce the Residual to somewhere close to machine zero. The p-multigrid method is also applied to the linear wave case. The acceleration by the p-multigrid
method is small, possibly due to the simple form of the governing equation and the nonlinear nature of the FAS.

It is emphasized here that, for subsequent discussions of the implicit scheme, the implicit scheme will refer to the simplified implicit scheme unless otherwise stated.

5.1.2. Nonlinear wave

Application of the \( p \)-multigrid method to nonlinear wave equations can improve the above situation. This case was formulated with the sole purpose of testing our implicit full multigrid ideology. Unlike the linear wave test cases, the initial time step is limited even for the implicit simulations. The full multigrid approach starts by solving the problem using lower order polynomials, and using these solutions as initial approximations for the next higher level. This process continues until it reaches the finest level where the solution of the problem is required. It results in a smooth, good initial guess. Hence a much higher time step can be used for the initial V or W cycles. The final CFL number is around 40 for this particular case.

We consider a nonlinear 2D scalar initial boundary value problem

\[
\frac{\partial Q}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0 \quad \text{and} \quad f = g = \frac{1}{3}Q^3 \tag{5.2}
\]

Eq. (5.2) has an analytical solution \( Q(x,y) = \sin(\pi(x - y)) \) and \((x,y) \in [-1,1]^2\). The boundary conditions are the same as the ones for linear wave case in Section 5.1.1. As shown in Fig. 5, the speedup factor obtained by the three-level full \( p \)-multigrid scheme is about 5 when compared to the single level implicit scheme. As expected, the computational CPU time of the single level explicit scheme is more than one and a half orders magnitude higher than the three-level implicit \( p \)-multigrid.

It is easy to extend the above \( p \)-multigrid method with V cycles to include W cycles if \( p_0 \) and \( p_1 \) levels are used once again before proceeding to the \( p_2 \) level. The problem requires lesser W cycles than V cycles to attain convergence. This is shown in Fig. 6a. However, W cycles consumed slightly more time than the V cycles. This is shown in Fig. 6b.

Both three-stage and five-stage TVD Runge–Kutta schemes are tested for this problem. Five-stage one is proven more effective when it is applied to all three \( p \)-levels. The maximum CFL number is kept at 0.10 for the explicit scheme with 3rd-order SD method. Therefore, in the subsequent part of the paper, we only discuss the results obtained using the five-stage Runge–Kutta scheme for explicit smoothers.

The problem was also tackled by combining the explicit and implicit smoothers. Since the implicit smoother is more effective than the explicit smoother, we use 20–30 iterations of explicit smoother at the \( p_2 \) level to attenuate the prolongation errors generated by 6 LU–SGS iterations at \( p_0 \) level and a single LU–SGS iteration at \( p_1 \) level. The speedup factor is still more than 20 compared to a single level explicit scheme for the normalized residual reducing to \( 10^{-5} \) (Fig. 7). The second approach is to use explicit smoothers for all the three levels. The speed is about four times that of a single level to reach residual level of \( 10^{-5} \). The last approach is maintain-

![Fig. 5. Comparison of convergence history with respect to time using explicit R–K and implicit LU-SGS schemes for the nonlinear wave case \((\frac{\partial^2 Q}{\partial x^2} + \frac{\partial^2 Q}{\partial y^2} = 0)\).](image)

![Fig. 6. Non-linear wave case using a LU-SGS implicit smoother. (a): Convergence history as a function of multigrid cycles. (b): Convergence history as a function of time.](image)
ing explicit smoothers for \( p2 \) and \( p1 \) levels, but using implicit smoother for the \( p0 \) level. As expected, the speed of residual convergence to \( 10^{-5} \) lies in between the above two approaches and is about eight times as fast as that of single level. Note that the convergence history curve using implicit LU-SGS three-level Full multigrid is also plotted. This has a convergence rate which is faster than any of the above schemes.

Fourth-order accuracy has been verified for the spectral difference method using wave equation examples in [16]. We can also easily extend the Full \( p \)-multigrid method to \( p0–p1–p2–p3 \) levels. However, to demonstrate the idea with both explicit and implicit smoothers, we restrict our discussions to three-level \( p \)-multigrid method unless otherwise stated. Moreover, we have found that the number of cycles each multi-grid level is related to the stability of our numerical tests and it depends on individual testing problems. For the flow problems shown in this paper, we have tested several combinations of numbers. Finally, for each individual case, we report only the information for the fastest and most stable convergence.

5.2. Results for the Euler equations

The Jacobian \( D \) matrix for the Euler equations is bigger than the scalar equations, since we have four conservative variables at each unknown point. For example, \( D \) is a \( 24 \times 24 \) matrix for a 3rd-order SD formulation of the 2D Euler equation. The \( D \) matrix is frozen every four multigrid cycles. The \( D \) matrix is frozen for around 20 multigrid cycles when the steady residual drops to below \( 5 \times 10^{-5} \). The Roe flux is used for the flux vector computation at the element boundary interfaces. A quadratic curved boundary condition is adopted [30] for non-straight wall boundary surfaces.

5.2.1. 2D supersonic vortex flow

This test case is used to assess the order of accuracy of the \( p \)-multigrid method for the SD scheme. An inviscid, isentropic, supersonic flow between concentric circular arcs presents a situation where the velocity and Mach number vary inversely with radius and the entropy is constant. This is a shock-free compressible flow and hence the measured order of accuracy is not contaminated by limiter action near shocks. The behaviour will correspond to what one may expect of smooth regions of inviscid subsonic compressible flow. This particular type of flow was used in [1,17] to verify the order of accuracy of their unstructured grid methods. The expression for density \( \rho \) as a function of radius \( r \) is given by

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

where \( M_i \) and \( r_i \) are the Mach number and radius at the inner arc. In the present calculations, the inner and outer radii are 1 and 1.384, respectively. A typical computational grid is shown in Fig. 8a. The Mach number, density and pressure are specified as constants 2.25, 1, and 1/\( c \), respectively, for the inner arc boundary. The outer arc and bottom inlet boundaries are fixed with analytical solutions. The zero-gradient extrapolation boundary is employed for the exit. Fig. 8b shows the solution of pressure obtained using the single level implicit simplified method.

Four different grids are used for the calculations. There is no difference in terms of \( L_1 \) and \( L_2 \) errors for the implicit LU-SGS and the explicit R–K methods. The results are summarized in Tables 1 and 2. Nearly 3rd-order is demonstrated using \( p2 \) polynomials of unknown points for the SD method and 2nd-order is achieved using \( p1 \) polynomials. For explicit scheme, the CFL number is around 0.3 and the implicit uses a CFL number of 28.7.

Using this case with analytical solution, we have found that the speedup obtained by \( p \)-multigrid and implicit methods are pretty
much grid independent. The high-order accuracy of the SD method is not polluted at all by the \( p \)-multigrid and implicit schemes.

5.2.2. 2D subsonic flow over a bump

We chose a testing case of the subsonic flow over a bump at Mach = 0.5. This case has been used by \( p \)-multigrid method for DG formulations of Euler equations in [17,21]. A 10% thick circular bump is mounted on the center of the channel bottom. The length of the channel is 3, its height 1, and its width 0.5. The computational grid with 3140 elements is shown in Fig. 9a. Once again, the circular surface of the bump needs a higher-order boundary treatment and a quadratic boundary as described in [30] is adopted.

Fig. 9b shows the pressure contour obtained by the three-level \( p \)-multigrid method. It is nearly identical to the pressure contour levels shown in [17]. The maximum CFL number used for all the implicit computations is around 8. It is much bigger than the one for explicit R–K scheme (CFL = 0.3) for the 2nd-order SD method with the same number of degree of freedom. Fig. 10 shows the residual convergence history of the implicit schemes. The speedup obtained by the two-level \( p \)-multigrid method is around 1.6 compared to the single level implicit scheme. The three-level \( p \)-multigrid method accelerates the convergence further and the speed is 3.5 times as fast as the single level implicit method. The overall speedup i.e. the speedup attained by the three-level \( p \)-multigrid is about 25 compared to the single level explicit scheme. Note that all the multigrid methods mentioned so far are based on V cycles for this particular case.

As far as the implicit smoothers are considered for all three \( p \)-levels, we also examined the difference between V cycles and W cycles for the three-level \( p \)-multigrid method. As expected, the current case requires lesser W cycles than V cycles to converge to machine zero as shown in Fig. 11a. However, the total time consumed

<p>| Table 1 | ( L_1 ) and ( L_2 ) errors and orders of accuracy of the supersonic vortex flow case using the 3rd-order SD method |</p>
<table>
<thead>
<tr>
<th>Cell number</th>
<th>( L_1 ) error</th>
<th>( L_1 ) order</th>
<th>( L_2 ) error</th>
<th>( L_2 ) order</th>
</tr>
</thead>
<tbody>
<tr>
<td>131</td>
<td>3.42e–4</td>
<td>–</td>
<td>5.09e–4</td>
<td>–</td>
</tr>
<tr>
<td>528</td>
<td>4.97e–5</td>
<td>2.77</td>
<td>7.83e–5</td>
<td>2.68</td>
</tr>
<tr>
<td>1892</td>
<td>8.93e–6</td>
<td>2.79</td>
<td>1.67e–5</td>
<td>2.52</td>
</tr>
<tr>
<td>7590</td>
<td>1.32e–6</td>
<td>2.77</td>
<td>2.74e–6</td>
<td>2.61</td>
</tr>
</tbody>
</table>

<p>| Table 2 | ( L_1 ) and ( L_2 ) errors and orders of accuracy of the supersonic vortex flow case using the 2nd-order SD method |</p>
<table>
<thead>
<tr>
<th>Cell number</th>
<th>( L_1 ) error</th>
<th>( L_1 ) order</th>
<th>( L_2 ) error</th>
<th>( L_2 ) order</th>
</tr>
</thead>
<tbody>
<tr>
<td>131</td>
<td>3.90e–3</td>
<td>–</td>
<td>4.98e–3</td>
<td>–</td>
</tr>
<tr>
<td>528</td>
<td>1.00e–3</td>
<td>1.95</td>
<td>1.34e–3</td>
<td>1.88</td>
</tr>
<tr>
<td>1892</td>
<td>3.25e–4</td>
<td>1.83</td>
<td>4.52e–4</td>
<td>1.76</td>
</tr>
<tr>
<td>7590</td>
<td>7.68e–5</td>
<td>2.08</td>
<td>1.03e–4</td>
<td>2.13</td>
</tr>
</tbody>
</table>

Fig. 10. Comparison of convergence history with respect to time for the bump testing case using explicit R–K and implicit LU-SGS schemes.

Fig. 9. Subsonic flow over a bump confined in a channel. (a): Computational grid. (b): Computed pressure contours.
is lesser when the V cycles were employed. This is shown in Fig. 13b. Note that, the number of iterative smoothers (total 24 for a cycle) performed for V cycles is 1–1–20–1–1 for 3rd, 2nd, 1st, 2nd and 3rd-order levels, respectively. More coarse level iterative smoothers (total 37 for a cycle) are used in the W cycles, i.e. 1–1–16–1–16–1–1 iterations for \( p_2 \)–\( p_1 \)–\( p_0 \)–\( p_1 \)–\( p_1 \)–\( p_2 \) levels, respectively.

It is known that Full multigrid scheme (starting from the coarsest grid level) is a good choice when the initial guess is very bad. Fig. 12 shows the convergence history with respect to time for a FMG implemented using V cycles. The scheme starts the computation from the \( p_0 \) level and the normalized residual drops nearly two orders. It takes about 50 iterations to realize the above situation. The solution is then extrapolated to the \( p_1 \) level using the standard prolongation operator. This prolongation results in a jump, which can be seen in Fig. 12. A two-level scheme is used to drop the residual by an order. It takes about 25 iterations to realize the above. The solution is then extrapolated to the \( p_2 \) level. As expected, a second jump occurs due to the extrapolation. It should be noted that the CPU time consumed by the above mentioned pre-processing computations on \( p_0 \) and \( p_1 \) levels is pretty short (70 s for this particular case). The three-level solver is now used to further drive the errors out of the domain. It can be seen that the time taken for the pre-processor takes only about a tenth of the total time. It is noted that all the subsequent figures for the Full multigrid schemes (either implicit or explicit) show convergence histories directly from the \( p_2 \) level because the CPU time spent on the pre-processing computations is short.

![Fig. 11. 2D subsonic flow over a bump case using implicit LU-SGS smoother. (a): Convergence history as a function of multigrid cycles. (b): Convergence history as a function of time.](image)

![Fig. 12. Convergence history with respect to time, showing the time taken for the single, two and three levels of the multigrid for the bump case.](image)

![Fig. 13. Comparison of convergence history with respect to time using combinations of explicit R-K and implicit LU-SGS schemes for the bump testing case.](image)
All the above multigrid calculations for the bump case are using implicit smoothers. If the explicit smoothers are used for all three p-levels as shown in Fig. 13 a speedup factor of 8 is attained over the explicit single level case. As expected, the three-level explicit scheme is slower than the mixed scheme (1 R–K + 2 LU-SGS). Note that, we use 1–1–6–1–1–1–iteration explicit smoothers at the p2–p1–p0–p1–p2 levels to form a standard V cycle. The curve with the second fastest convergence rate in Fig. 13 is obtained using a calculation employing explicit smoothers only for the p2 level and implicit smoothers for both p1 and p0 levels. The three-level FMG LU-SGS is also plotted and is much faster than any of the above discussed methods.

5.2.3. Subsonic flow over a NACA0012 airfoil

The final testing case of the Euler equations is the subsonic flow over a NACA0012 airfoil at Mach = 0.4 and angle of attack of 0°. The computational grid is shown in Fig. 14. The outer boundary is 20 chords away from the airfoil centre. Fig. 15 shows pressure contours obtained using two-level p-multigrid scheme, i.e. using the p2 and p1 polynomials. The maximum CFL number used for the implicit computations is around 6.5. The explicit scheme limits the maximum CFL number to 0.06 for 3rd-order SD method. From Fig. 16, we can see that the highest speedup factor obtained using the three-level p-multigrid is around 100 compared with the single p2 level explicit scheme. The three-level p-multigrid method is about four times as fast as the single p2 level implicit method. The two-level scheme using the implicit smoothers for p2 and p1 levels shows a fast convergence too. The speedup factor of the two-level scheme is around 70 compared to the single p2 level explicit scheme.

The effect of the explicit smoothers is also studied and is shown in Fig. 17. Full multigrid approach is used for the three level calculations. The first calculation (three-level R–K) uses explicit smoothers for all p-levels. 1–1–6–1–1–1-iteration explicit smoothers are
employed for \( p_2 - p_1 - p_0 - p_1 - p_2 \) levels to form a three-level V cycle. The second calculation is defined as mixed three-level approach. It employs explicit smoothers only for the \( p_2 \) level. For stability reason, we use around 30 iterations of explicit smoother at the \( p_2 \) level to smooth out the prolongation noises generated by the implicit smoothers of six iterations on the \( p_0 \) level and a single iteration on the \( p_1 \) level. The mixed approach is about 1.4 times as fast as the approach of R–K on three levels. The three-level \( p \)-multigrid method using solely implicit smoother is much faster than any of the above discussed schemes.

For the above computations performed for the airfoil case, the iteration numbers for the V cycles shall be pointed out as follows.

- For three-level \( p \)-multigrid using solely implicit smoothers, two iterations are sufficient for the \( p_2 \) and \( p_1 \) levels, and 20 iterations are performed on the \( p_0 \) level during every V cycle. In short, 2–2–20–2–2 iterations of smoothers are employed on \( p_2 - p_1 - p_0 - p_1 - p_2 \) levels. For the two-level \( p \)-multigrid method, a smaller V-cycle is constructed using one relaxation iteration at the \( p_2 \) level, 2 relaxation iterations at the \( p_1 \) level, and two more smoothing iterations at the \( p_0 \) level to remove the prolongation noise. As shown in Fig. 18a, the three-level scheme requires less cycles than the two-level scheme. It is also evident from Fig. 18b that three-level scheme converges faster than the two-level scheme. In particular, the convergence trend of the three-level scheme is much less oscillatory.

6. Conclusions

We have developed a \( p \)-multigrid spectral difference solver for 2D Euler equations with a preconditioned implicit LU-SGS smoother. The SD method is also relaxed with a Runge–Kutta explicit scheme for comparison. We found that the simplified implicit scheme is very stable and produces a speedup factor around one order for the linear scalar equation. The speedup factor reaches even two order magnitude compared to its explicit counterparts for the nonlinear scalar wave equation as well as the Euler equations. The computational speed is further accelerated by a nonlinear \( p \)-multigrid approach in the context of Full Approximation Scheme. The combination of the implicit LU-SGS relaxation scheme with three-level \( p \)-multigrid method achieved very good stability and speedup for both 2D wave and 2D Euler equations. The \( p \)-multigrid method with implicit smoothers on all three \( p \)-levels achieves a speedup of around 4 over the single level implicit scheme. Explicit Runge–Kutta smoothers are also studied for the \( p \)-multigrid method. They are not as effective as the implicit LU-SGS counterparts. However, they can be used at the highest \( p \)-level for 3D problems to circumvent the problems associated with large storage of implicit schemes. Our calculations in two dimensions show that the approaches using explicit smoother at the high \( p \)-levels and implicit smoother at the low \( p \)-levels are stable and achieve considerable speedup. Nevertheless, a
speedup factor around 8 can be achieved using explicit smoothers for all three $p$-levels. In addition, the simulations using the V cycles are faster than the simulations using the W cycles for all cases considered with the nonlinear equations.

Acknowledgement

The study was supported by AFOSR Grant FA9550-06-1-0146 and partially supported by DOE Grant DE-FG02-05ER25677. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR or DOE.

References