COMMUNICATIONS IN COMPUTATIONAL PHYSICS
Vol. **5**, No. 2-4, pp. 760-778

Commun. Comput. Phys.
February 2009

Efficient Implicit Non-linear LU-SGS Approach for Compressible Flow Computation Using High-Order Spectral Difference Method

Yuzhi Sun[1], Z. J. Wang[1,*] and Yen Liu[2]

[1] *Department of Aerospace Engineering, Iowa State University, Ames, IA 50011, USA.*
[2] *NASA Ames Research Center, Moffett Field, CA 94035, USA.*

Received 21 September 2007; Accepted (in revised version) 25 February 2008

Available online 1 August 2008

Abstract. An implicit non-linear lower-upper symmetric Gauss-Seidel (LU-SGS) solution algorithm has been developed for a high-order spectral difference Navier-Stokes solver on unstructured hexahedral grids. The non-linear LU-SGS solver is preconditioned by a block element matrix, and the system of equations is then solved with the LU decomposition approach. The large sparse Jacobian matrix is computed numerically, resulting in extremely simple operations for arbitrarily complex residual operators. Several inviscid and viscous test cases were performed to evaluate the performance. The implicit solver has shown speedup of 1 to 2 orders of magnitude over the multi-stage Runge-Kutta time integration scheme.

AMS subject classifications: 65M70, 76M20, 76M22

Key words: High order, unstructured grids, spectral difference, Navier-Stokes, implicit.

1 Introduction

The past three decades have seen tremendous development in computational fluid dynamics (CFD) as a discipline. CFD is now used routinely to complement the wind tunnel in engineering design. Nearly all production flow solvers are based on second-order numerical methods. They are capable of delivering design-quality Reynolds Averaged Navier-Stokes results with several million cells (degrees of freedom or DOFs) on commercial Beowulf clusters within a few hours.

As impressive as these second order codes are, there are still many flow problems considered as out of reach, e.g., vortex dominated flows including helicopter blade vortex interaction and flows over high-lift configurations. Unsteady propagating vortices are the

*Corresponding author. *Email addresses:* sunyuzhi@iastate.edu (Y. Sun), zjw@iastate.edu (Z. J. Wang), Yen.Liu@nasa.gov (Y. Liu)

main features of these flow problems, and second-order methods are usually too dissipative to resolve those unsteady vortices for a significant distance. The advantage of high-order methods (order of accuracy >2) over first- and second-order ones is well-known in the CFD community. Generally speaking, with the same number of DOFs or solution unknowns, high-order methods are capable of producing much more accurate results. For problems requiring very high accuracy, e.g., wave propagation problems in computational aeroacoustics, high-order methods have been the main choice. Many high-order methods were developed for structured grids, e.g., ENO/WENO methods [28], compact methods [16, 38], optimized methods [34], to name just a few. In the last two decades, there have been intensive research efforts on high-order methods for unstructured grids since many real world applications have complex geometries. An incomplete list of notable examples includes the spectral element method [22], multi-domain spectral method [14,15], k-exact finite volume method [2,10,17,24], WENO methods [11], discontinuous Galerkin (DG) method [3,7,8], high-order residual distribution methods [1], spectral volume (SV) [20,31,39,41,42] and spectral difference (SD) methods [12,18,19,32,33,43]. Among those methods, some are based on the weighted residual form of the governing equations, for instance, the DG method. Some are based on the integral form of the governing equations, e.g., the k-exact finite volume method and SV methods. Others, such as the staggered grid multi-domain spectral method and the SD method, are based on the differential form. In fact, the staggered-grid multi-domain spectral method and the SD method are identical on quadrilateral or hexahedral grids. More comprehensive reviews of high-order methods are given in [9, 40].

When one chooses a particular method for three-dimensional applications, the cost and the complexity in implementing the method is often an important factor. It is obvious that methods based on the differential form are the easiest to implement since they do not involve surface or volume integrals. This is particularly true when high-order curved boundaries need to be dealt with. We recently developed a high order SD method [32, 33] for the three dimensional Navier-Stokes equations on unstructured hexahedral grids. High-order of accuracy and spectral convergence are achieved for several benchmark problems. It was also shown that the wall boundaries must be approximated with high-order surfaces. An explicit Runge-Kutta time integration scheme [27] was used in the implementation. Although the explicit scheme is easy to implement and has high-order accuracy in time, it suffered from slow convergence, especially for viscous grids which are clustered in the viscous boundary layer. It is well-known that high-order methods are restricted to a smaller CFL number than low order ones. In addition, they also possess much less numerical dissipation. Therefore it takes excessive CPU to reach a state-steady solution with explicit high-order schemes. The computation cost of high-order explicit methods for many steady-state problems is so high that they become less efficient than low-order implicit methods in terms of the total CPU time given the same level of solution error. It is therefore imperative to develop efficient implicit solution approaches for high-order methods to fully realize the potentials, which is the objective of the present study.

Implicit time-integration schemes are highly desired for improved efficiency since

they can advance the solution with significantly larger time steps compared with the explicit methods. Many implicit schemes have been developed and applied successfully to unstructured grids to accelerate convergence to steady state [4, 6, 30, 36, 37] in the last one and a half decades. In this paper, an efficient implicit lower-upper symmetric Gauss-Seidel (LU-SGS) solution algorithm is developed to solve both inviscid and viscous compressible flows for the high order spectral difference method on unstructured hexahedral grids. The original LU-SGS algorithm was developed in [44], and later more efficient preconditioned versions were developed in [6, 13]. The main difficulty in the development is the computation of the element Jacobian matrices, which involve both the solution and the solution gradients. A numerical approach [5,23] to compute the Jacobian is employed to avoid the difficulty.

The paper is organized as follows. In the next section, the formulation of the 3D spectral difference method, including both explicit and implicit schemes, is described for a hexahedral element. In Section 3, several representative test cases are selected to demonstrate the efficiency of the implicit approach and to study the effects of several parameters on the convergence rate. Conclusions and possible future works are summarized in Section 4.

2 Formulation of multidomain spectral difference method

2.1 Governing equation

Consider the unsteady compressible 3D Navier-Stokes equations in conservative form written as

$$\frac{\partial Q}{\partial t}+\frac{\partial F}{\partial x}+\frac{\partial G}{\partial y}+\frac{\partial H}{\partial z}=0, \tag{2.1}$$

where Q is the vector of conserved variables, and F, G, H are the total fluxes including both the inviscid and viscous flux vectors, i.e.,

$$F=F^i-F^v, \quad G=G^i-G^v, \quad H=H^i-H^v.$$

We employ the non-overlapping unstructured hexahedral cells or elements to fill the computational domain. The use of hexahedral cells for viscous boundary layers is preferred over tetrahedral cells because of the efficiency and accuracy. In order to handle curved boundaries, both linear and quadratic isoparametric elements are employed, with linear elements used in the interior domain and quadratic elements near high-order curved boundaries. In order to achieve an efficient implementation, all elements are transformed from the physical domain (x,y,z) into a standard cubic element $(\xi,\eta,\varsigma)\in[0,1]^3$. The transformation can be written as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}=\sum_{i=1}^{K} M_i(\xi,\eta,\varsigma)\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}, \tag{2.2}$$

where K is the number of points used to define the physical element, (x_i,y_i,z_i) are the Cartesian coordinates of those points, and $M_i(\xi,\eta,\varsigma)$ are the shape functions. For the transformation given in (2.2), the Jacobian matrix J takes the following form

$$J=\frac{\partial(x,y,z)}{\partial(\xi,\eta,\varsigma)}=\begin{bmatrix} x_\xi & x_\eta & x_\varsigma \\ y_\xi & y_\eta & y_\varsigma \\ z_\xi & z_\eta & z_\varsigma \end{bmatrix}.$$

For a non-singular transformation, its inverse transformation must also exist, and the Jocobian matrices are related to each other according to

$$\frac{\partial(\xi,\eta,\varsigma)}{\partial(x,y,z)}=\begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \varsigma_x & \varsigma_y & \varsigma_z \end{bmatrix}=J^{-1}.$$

The governing equations in the physical domain are then transformed into the computational domain (standard element), and the transformed equations take the following form

$$\frac{\partial\widetilde{Q}}{\partial t}+\frac{\partial\widetilde{F}}{\partial\xi}+\frac{\partial\widetilde{G}}{\partial\eta}+\frac{\partial\widetilde{H}}{\partial\varsigma}=0, \tag{2.3}$$

where

$$\widetilde{Q}=|J|\cdot Q,\quad \begin{bmatrix} \widetilde{F} \\ \widetilde{G} \\ \widetilde{H} \end{bmatrix}=|J|\begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \varsigma_x & \varsigma_y & \varsigma_z \end{bmatrix}\cdot\begin{bmatrix} F \\ G \\ H \end{bmatrix}.$$

Let $\vec{S}_\xi=|J|(\xi_x,\xi_y,\xi_z)$, $\vec{S}_\eta=|J|(\eta_x,\eta_y,\eta_z)$ and $\vec{S}_\varsigma=|J|(\varsigma_x,\varsigma_y,\varsigma_z)$. Then we have

$$\widetilde{F}=\vec{f}\bullet\vec{S}_\xi,\quad \widetilde{G}=\vec{f}\bullet\vec{S}_\eta,\quad \widetilde{H}=\vec{f}\bullet\vec{S}_\varsigma$$

with $\vec{f}=(F,G,H)$.

2.2 Space discretization

In the standard element, two sets of points are defined, namely the solution points and the flux points. The solution unknowns or degrees-of-freedom (DOFs) are the conserved variables at the solution points, while fluxes are computed at the flux points. In order to construct a degree $(N-1)$ polynomial in each coordinate direction, solutions at N points are required. The solution points in 1D are chosen to be the Chebyshev-Gauss points defined by

$$X_s=\frac{1}{2}\left[1-\cos\left((s-1/2)\pi/N\right)\right],\qquad s=1,2,\cdots,N. \tag{2.4}$$

The flux points are selected to be the Chebyshev-Gauss-Lobatto points given by

$$X_{s+1/2}=\frac{1}{2}\left[1-\cos\left(s\pi/N\right)\right],\qquad s=0,1,\cdots,N. \tag{2.5}$$

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

$$h_i(X)=\prod_{s=1,s\neq i}^{N}\left(\frac{X-X_s}{X_i-X_s}\right). \tag{2.6}$$

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

$$l_{i+1/2}(X)=\prod_{s=0,s\neq i}^{N}\left(\frac{X-X_{s+1/2}}{X_{i+1/2}-X_{s+1/2}}\right). \tag{2.7}$$

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

$$Q(\xi,\eta,\varsigma)=\sum_{k=1}^{N}\sum_{j=1}^{N}\sum_{i=1}^{N}\frac{\widetilde{Q}_{i,j,k}}{|J_{i,j,k}|}h_i(\xi)\cdot h_j(\eta)\cdot h_k(\varsigma). \tag{2.8}$$

Similarly, the reconstructed flux polynomials take the following forms:

$$\widetilde{F}(\xi,\eta,\varsigma)=\sum_{k=1}^{N}\sum_{j=1}^{N}\sum_{i=0}^{N}\widetilde{F}_{i+1/2,j,k}l_{i+1/2}(\xi)\cdot h_j(\eta)\cdot h_k(\varsigma), \tag{2.9a}$$

$$\widetilde{G}(\xi,\eta,\varsigma)=\sum_{k=1}^{N}\sum_{j=0}^{N}\sum_{i=1}^{N}\widetilde{G}_{i,j+1/2,k}h_i(\xi)\cdot l_{j+1/2}(\eta)\cdot h_k(\varsigma), \tag{2.9b}$$

$$\widetilde{H}(\xi,\eta,\varsigma)=\sum_{k=0}^{N}\sum_{j=1}^{N}\sum_{i=1}^{N}\widetilde{H}_{i,j,k+1/2}h_i(\xi)\cdot h_j(\eta)\cdot l_{k+1/2}(\varsigma). \tag{2.9c}$$

The reconstructed fluxes are only element-wise continuous, but discontinuous across cell interfaces. For the inviscid flux, a Riemann solver, such as the Rusanov [26] or Roe [25] flux, is employed to compute a common flux at interfaces to ensure conservation and stability. In summary, the algorithm to compute the inviscid flux derivatives consists of the following steps:

• Given the conserved variables at the solution points $\{\widetilde{Q}_{i,j,k}\}$, compute the conserved variables at the flux points $\{Q_{i+1/2,j,k},Q_{i,j+1/2,k},Q_{i,j,k+1/2}\}$ using (2.8) (Note that $h_m(X_n)=\delta_{mn}$);

• Compute the inviscid fluxes at the interior flux points using the solutions computed at Step 1, i.e., $\{\widetilde{F}^i_{i+1/2,j,k},i=1,\cdots,N-1\}$, $\{\widetilde{G}^i_{i,j+1/2,k},j=1,\cdots,N-1\}$ $\{\widetilde{H}^i_{i,j,k+1/2},k=1,\cdots,N-1\}$;

• Compute the inviscid flux at element interfaces using a Riemann solver, in terms of the left and right conserved variables of the interface. Given the normal direction of

the interface $\vec{n}$, and the averaged normal velocity component $\overline{V}_n$ and sound speed $\overline{c}$, the Rusanov flux on the interface is computed with

$$\widetilde{F}^i=\frac{1}{2}\left[\widetilde{F}^i_L+\widetilde{F}^i_R-(|\overline{V}_n|+\overline{c})\cdot(Q_R-Q_L)\cdot\vec{n}\bullet\vec{S}_\xi\right],$$
$$\widetilde{G}^i=\frac{1}{2}\left[\widetilde{G}^i_L+\widetilde{G}^i_R-(|\overline{V}_n|+\overline{c})\cdot(Q_R-Q_L)\cdot\vec{n}\bullet\vec{S}_\eta\right],$$
$$\widetilde{H}^i=\frac{1}{2}\left[\widetilde{H}^i_L+\widetilde{H}^i_R-(|\overline{V}_n|+\overline{c})\cdot(Q_R-Q_L)\cdot\vec{n}\bullet\vec{S}_\varsigma\right].$$

• Compute the derivatives of the fluxes at all the solution points according to

$$\left(\frac{\partial\widetilde{F}}{\partial\xi}\right)_{i,j,k}=\sum_{r=0}^{N}\widetilde{F}_{r+1/2,j,k}\cdot l'_{r+1/2}(\xi_i), \tag{2.10a}$$
$$\left(\frac{\partial\widetilde{G}}{\partial\eta}\right)_{i,j,k}=\sum_{r=0}^{N}\widetilde{G}_{i,r+1/2,k}\cdot l'_{r+1/2}(\eta_j), \tag{2.10b}$$
$$\left(\frac{\partial\widetilde{H}}{\partial\varsigma}\right)_{i,j,k}=\sum_{r=0}^{N}\widetilde{H}_{i,j,r+1/2}\cdot l'_{r+1/2}(\varsigma_k). \tag{2.10c}$$

The viscous flux is a function of both the conserved variables and their gradients, e.g.,

$$\widetilde{F}^v_{i+1/2,j,k}=\widetilde{F}^v(Q_{i+1/2,j,k},\nabla Q_{i+1/2,j,k}).$$

Therefore the key is how to compute the solution gradients at the flux points. The gradient of the conserved variables in the physical domain can be easily computed using

$$\nabla Q=\frac{1}{|J|}\left[\frac{\partial(Q\vec{S}_\xi)}{\partial\xi}+\frac{\partial(Q\vec{S}_\eta)}{\partial\eta}+\frac{\partial(Q\vec{S}_\varsigma)}{\partial\varsigma}\right]. \tag{2.11}$$

The derivative along each coordinate direction is computed using

$$\left[\frac{\partial(Q\vec{S}_\xi)}{\partial\xi}\right]_{j,k}=\sum_{r=0}^{N}\left(Q\vec{S}_\xi\right)_{r+1/2,j,k}\cdot l'_{r+1/2}(\xi), \tag{2.12a}$$
$$\left[\frac{\partial(Q\vec{S}_\eta)}{\partial\eta}\right]_{i,k}=\sum_{r=0}^{N}\left(Q\vec{S}_\eta\right)_{i,r+1/2,k}\cdot l'_{r+1/2}(\eta), \tag{2.12b}$$
$$\left[\frac{\partial(Q\vec{S}_\varsigma)}{\partial\varsigma}\right]_{i,j}=\sum_{r=0}^{N}\left(Q\vec{S}_\varsigma\right)_{i,j,r+1/2}\cdot l'_{r+1/2}(\varsigma). \tag{2.12c}$$

The following steps are taken to compute the viscous fluxes:

• Same as Step 1 for the inviscid flux computations;

• When computing the derivatives using (2.12), the solution Q at the cell interface is not uniquely defined. The solution at the interface is simply the average of the left and right solutions,

$$\hat{Q}=(Q_L+Q_R)/2.$$

• Compute the gradients of the solution at the solution points using the solutions at the flux points with (2.11) and (2.12). Then the gradients are interpolated from the solution points to the flux points using the same Lagrangian interpolation approach given in (2.8).

• Compute the viscous flux at the flux points using the solutions and their gradients at the flux points. Again at cell interfaces, the gradients have two values, one from the left and one from the right. The gradients used in the viscous fluxes at the cell interface are simply the averaged ones, i.e.,

$$\widetilde{F}^v=\widetilde{F}^v((Q_L+Q_R)/2,(\nabla Q_L+\nabla Q_R)/2).$$

2.3 Time marching

Explicit scheme: Denote the residuals at all the solution points at cell c as $R_c(\widetilde{Q}^n)$. Obviously, the semi-discrete equation can be written as

$$\frac{\partial \widetilde{Q}_c}{\partial t}=R_c(\widetilde{Q}^n)=-\left(\frac{\partial \widetilde{F}}{\partial \xi}+\frac{\partial \widetilde{G}}{\partial \eta}+\frac{\partial \widetilde{H}}{\partial \varsigma}\right), \tag{2.13}$$

where $\widetilde{Q}_c$ represents the solutions at the solution points of cell c. A multi-stage TVD Runge-Kutta scheme is used as the explicit scheme [27].

Implicit scheme: At each cell c, using the backward Euler difference, (2.3) can be written as

$$\frac{\widetilde{Q}_c^{n+1}-\widetilde{Q}_c^n}{\Delta t}-\left[R_c(\widetilde{Q}^{n+1})-R_c(\widetilde{Q}^n)\right]=R_c(\widetilde{Q}^n). \tag{2.14}$$

Let $\Delta\widetilde{Q}_c=\widetilde{Q}_c^{n+1}-\widetilde{Q}_c^n$ and linearizing the residual. We obtain

$$R_c(\widetilde{Q}^{n+1})-R_c(\widetilde{Q}^n)\approx\frac{\partial R_c}{\partial \widetilde{Q}_c}\Delta\widetilde{Q}_c+\sum_{nb\neq c}\frac{\partial R_c}{\partial \widetilde{Q}_{nb}}\Delta\widetilde{Q}_{nb}, \tag{2.15}$$

where subscript nb indicates all the neighboring cells contributing to the residual of cell c. Therefore, the fully linearized equations for (2.14) can be written as

$$\left(\frac{I}{\Delta t}-\frac{\partial R_c}{\partial \widetilde{Q}_c}\right)\Delta\widetilde{Q}_c-\sum_{nb\neq c}\frac{\partial R_c}{\partial \widetilde{Q}_{nb}}\Delta\widetilde{Q}_{nb}=R_c(\widetilde{Q}^n). \tag{2.16}$$

However, it is expensive in memory to store the full LHS implicit Jacobian matrices. Therefore, we employ a preconditioned LU-SGS scheme to solve (2.16). The contributions from the neighboring cells are included in the right hand side, i.e.,

$$\left(\frac{I}{\Delta t}-\frac{\partial R_c}{\partial \widetilde{Q}_c}\right)\Delta\widetilde{Q}_c^{(k+1)}=R_c(\widetilde{Q}^n)+\sum_{nb\neq c}\frac{\partial R_c}{\partial \widetilde{Q}_{nb}}\Delta\widetilde{Q}_{nb}^*, \tag{2.17}$$

where superscript $(k+1)$ is an iteration index, and superscript $*$ indicates the most recently updated solutions. The matrix

$$D=\left(\frac{I}{\Delta t}-\frac{\partial R_c}{\partial \widetilde{Q}_c}\right) \tag{2.18}$$

is the element (or cell) matrix, which serves as the preconditioning matrix. Eq. (2.17) is then solved with a direct LU decomposition solver. Since we do not want to store the matrices $\partial R_c/\partial\widetilde{Q}_{nb}$, (2.17) is further manipulated as follows. Note that

$$\begin{aligned}
&R_c(\widetilde{Q}^n)+\sum_{nb\neq c}\frac{\partial R_c}{\partial \widetilde{Q}_{nb}}\Delta\widetilde{Q}_{nb}^*=R_c(\widetilde{Q}_c^n,\{\widetilde{Q}_{nb}^n\})+\sum_{nb\neq c}\frac{\partial R_c}{\partial \widetilde{Q}_{nb}}\Delta\widetilde{Q}_{nb}^*\\
\approx&R_c(\widetilde{Q}_c^n,\{\widetilde{Q}_{nb}^*\})\approx R_c(\widetilde{Q}_c^*,\{\widetilde{Q}_{nb}^*\})-\frac{\partial R_c}{\partial \widetilde{Q}_c}\Delta\widetilde{Q}_c^*\\
=&R_c(\widetilde{Q}^*)-\frac{\partial R_c}{\partial \widetilde{Q}_c}\Delta\widetilde{Q}_c^*.
\end{aligned} \tag{2.19}$$

In (2.19), note that both *approximations* can be obtained using the first-order Taylor series expansion. Combining (2.17) and (2.19) together, we obtain

$$\left(\frac{I}{\Delta t}-\frac{\partial R_c}{\partial \widetilde{Q}_c}\right)\left(\widetilde{Q}_c^{(k+1)}-\widetilde{Q}_c^{(k)}\right)=R_c(\widetilde{Q}^*)-\frac{\Delta\widetilde{Q}_c^*}{\Delta t}. \tag{2.20}$$

Eq. (2.20) is then solved with the symmetric forward and backward sweeps. Note that once (2.20) is solved to machine zero, the unsteady residual is zero at each time step. For steady state problems, the last term in (2.20) can often be dropped resulting in faster convergence rate.

2.4 Computation of the Jacobian matrix

Because of the way in which the viscous fluxes are computed, the present SD method also uses cells, which are neighbors' neighbors. If the analytical approach is used to compute the element Jacobian matrix $\partial R_c/\partial\widetilde{Q}_c$, the formulation would be complex. Instead, the following numerical approach is used based on the definition

$$\frac{\partial R_c}{\partial \widetilde{Q}_c}\approx\frac{R_c(\{\widetilde{Q}_{nb}\},\widetilde{Q}_c+\varepsilon)-R_c(\{\widetilde{Q}_{nb}\},\widetilde{Q}_c)}{\varepsilon}, \tag{2.21}$$

where ε is a small parameter, e.g., $\varepsilon \approx \|\widetilde{Q}_c\| \times 10^{-8}$. This numerical approach is not new, and was previously used in [5, 23]. Although this approach is very easy to implement for arbitrarily complex residual operators, it is quite expensive because each variable has to be changed, and the flux be computed. In practice, we have found it unnecessary to compute the matrix at each iteration. Therefore, we often re-compute the matrix every 40-100 iterations. Numerical tests showed that this matrix-freezing approach did not significantly degrade the convergence rate to the steady state.

The main memory usage of the block pre-conditioned LU-SGS approach hinges on the size of the element matrix $\partial R_c/\partial \widetilde{Q}_c$, which is dimensioned $(5\times \text{order}\times \text{order}\times \text{order})^2 = 25\times \text{order}^6$. Therefore as the order of accuracy is increased, the memory use is increased much faster. Just to give a quick reference, for a mesh with 100,000 cells, the memory required to store the element matrices is 160 MW (mega words) for a 2^{nd}-order scheme with 800K DOFs, 1.82 GW (giga words) for a 3^{rd}-order scheme with 2.7 million DOFs, and 10.2 GW for a 4^{th}-order scheme with 6.4 M DOFs. For practical 3D computations, it is probably unrealistic to use implicit schemes higher than 4^{th}-order accurate.

3 Numerical experiments

3.1 Inviscid flow over a sphere

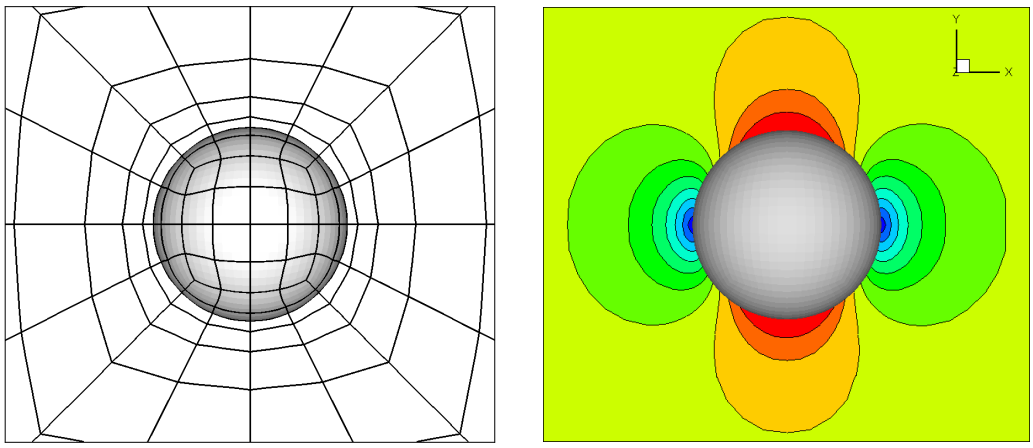
An inviscid flow over a sphere with a free stream Mach number of 0.2535 is selected as the first test to demonstrate the efficiency of the implicit scheme, and also to study the effects of *CFL* number and inner iteration control parameters on convergence characteristics. Fig. 1a shows the computational grid used in the simulation, which includes 768 hexahedral cells. Fig. 1b depicts the Mach number distribution computed with the 4^{th}-order SD scheme.

Both the three-stage Runge-Kutta explicit and the LU-SGS implicit schemes with 2^{nd}-, 3^{rd}-, and 4^{th}-order spatial accuracy were employed in the simulation. The implicit scheme dramatically accelerates the convergence rate to the steady state for this external flow. This is illustrated in the Fig. 2, which displays the convergence histories in terms of CPU time. The convergence rate with the implicit scheme is more than an order of magnitude faster than the explicit scheme.

Next, we study the effects of several parameters on the convergence rate of the simulation. Obviously the *CFL* number is an important convergence parameter. In the present study, the *CFL* number is computed based on the following power law form and bounded by the minimum and maximum *CFL* number:

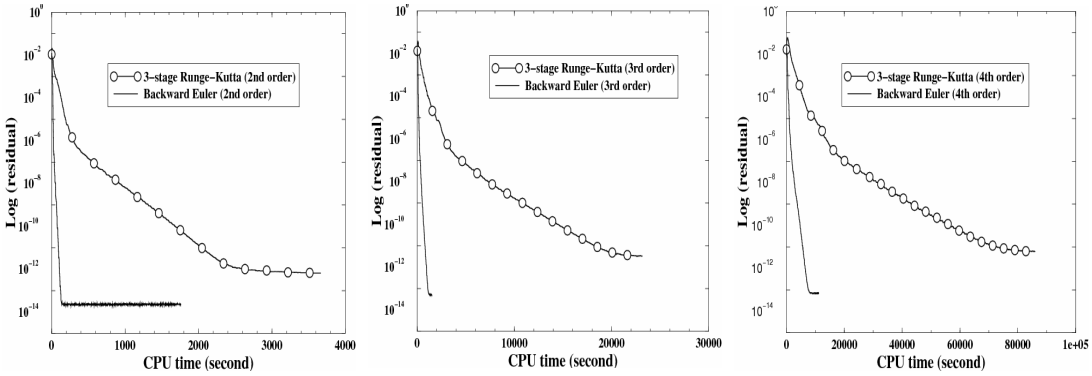
$$CFL = MIN(CFL_{\min}\cdot \alpha^n, CFL_{\max})$$

where $\alpha \geq 1$ is the amplification factor, and n is the iteration number. In the first test, $CFL_{\min}$ and α are fixed at certain values, while $CLF_{\max}$ is a variable. The effects of $CLF_{\max}$ on the convergence rates are showed in Fig. 3a with $CLF_{\min} = 1.0$ for the 3^{rd}-order SD



(a) (b)

Figure 1: Sphere grid with quadratic boundary (768 elements) and computed Mach number contours.



(a) 2nd Order (b) 3rd Order (c) 4th Order

Figure 2: Convergence histories of flow over a sphere with different orders of spatial accuracy.

scheme. The amplification factor is set to be 1.25. It is easily observed that the convergence rate strongly depends on the *CFL* number. The larger *CFL* number results in higher convergence rate. However, we also want to emphasize that a too large *CFL* number can sometimes cause the simulation to diverge.

The second parameter on the convergence rate is the number of the inner iterations, i.e., the number of forward and backward Gauss-Seidel sweeps in the LU-SGS approach. One sweep is defined to include both the forward and backward step here, and is denoted by *inner_sweep*. The unsteady residual in each time iteration step can be driven to machine zero if *inner_sweep* is big enough. In the present simulations, *inner_sweep*=3 is the smallest number to guarantee stability and convergence to the steady state. In this test, we let *inner_sweep* vary and fix the other parameters. From Fig. 3b, it seems that the number of inner iterations does not strongly influence the convergence rate for the inviscid flow

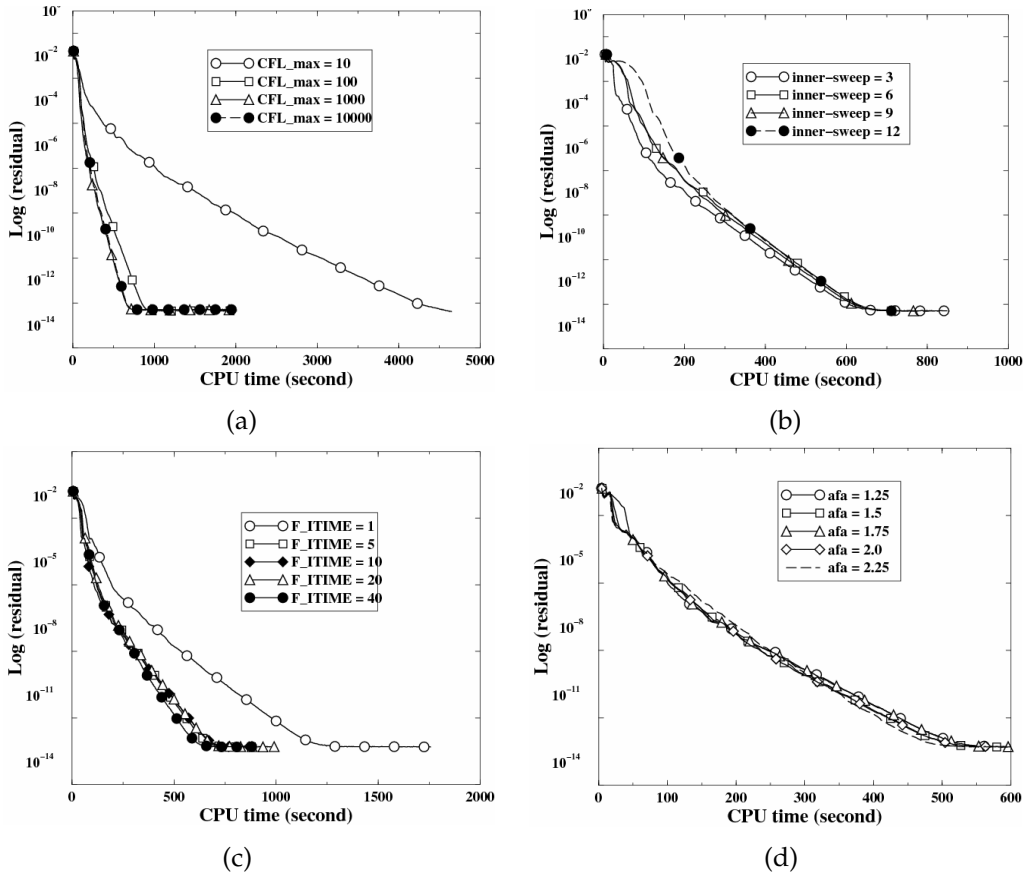


Figure 3: Effects of various parameters on the convergence characteristics (a) CFL number, (b) number of inner iterations, (c) matrix updating frequency, (d) amplification factor.

over a sphere.

In the LU-SGS approach, the computation of the element Jacobian matrix is quite time consuming since its size is quite large. One idea to improve the efficiency is to freeze this matrix for several time steps. The frequency in which this matrix is updated is denoted by *F_ITIME*. For example, *F_ITIME*$=5$ means the element Jacobian matrix is computed every 5 time steps. Fig. 3c shows the convergence histories with different *F_ITIME* with a 3$^{\text{rd}}$-order SD scheme. In this test, the *inner_sweep* is set to be 5, and CFL ranges from 1 to 10^6. It can be observed that the bigger *F_ITIME* results in higher efficiency.

Finally the effect of the amplification factor on the convergence rate is studied. In this test, varies from 1.25 to 3 with a 0.25 interval. Other parameters are set as follows: $CFL_{\min}=1$, $CFL_{\max}=10^6$, *inner_sweep*$=5$ and *F_ITIME*$=40$. In Fig. 3d, the convergence rates are plotted together for $1.25\leq\alpha\leq 2.25$. It appears the convergence rate does not strongly depend on the amplification factor when $1.25\leq\alpha\leq 2.25$. However, the simulation diverges when $\alpha=2.5$ or $\alpha=3.0$.

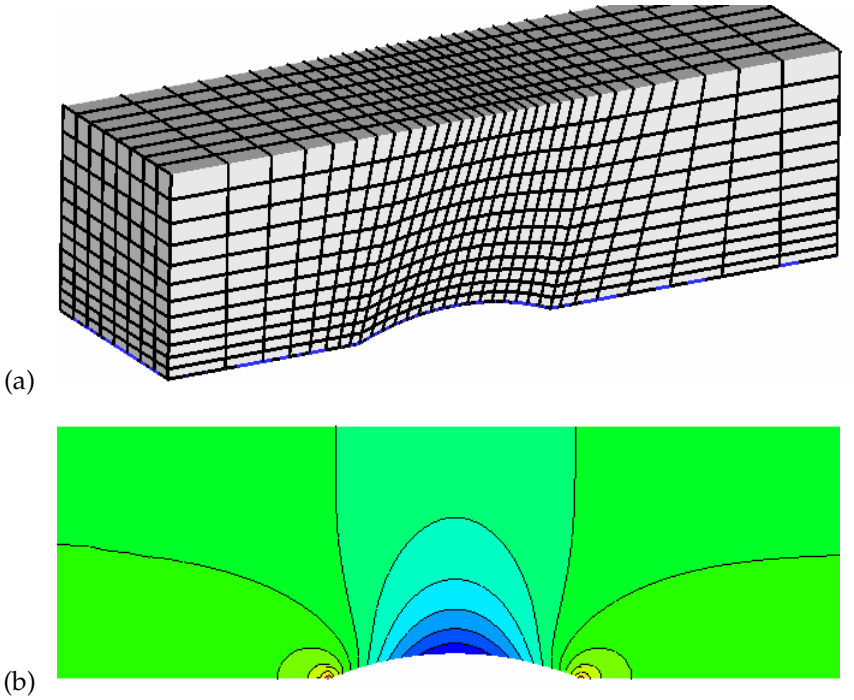


Figure 4: Grid for inviscid flow over a bump and computed pressure contours at a cutting plane.

3.2 Inviscid flow over a 3D bump

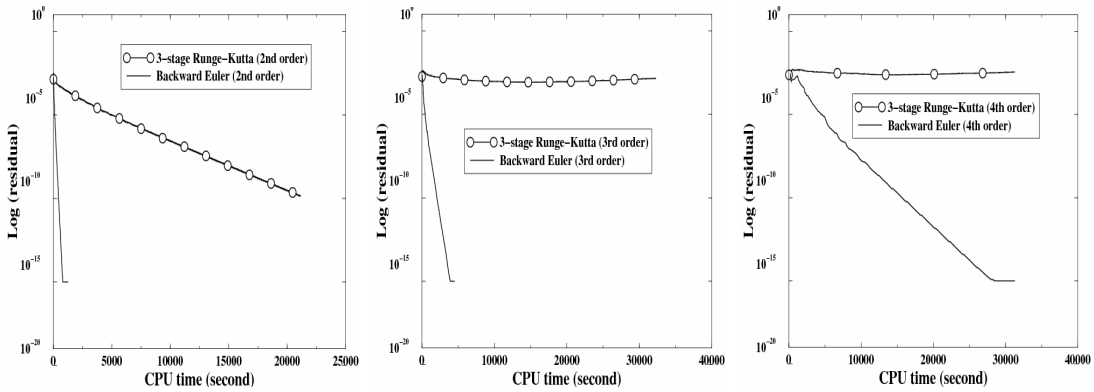
An inviscid flow over a 3D bump was selected to represent internal flow problems. Fig. 4a shows the computation grid with 3,072 hexahedral cells. Fig. 4b shows the steady state pressure contours on the middle cutting plane computed with the 4^{th}-order SD scheme.

Both the three-stage Runge-Kutta explicit and LU-SGS implicit schemes were employed for this problem with 2^{nd}-, 3^{rd}-, and 4^{th}-order spatial accuracies. Fig. 5 shows the convergence histories. From Fig. 5a, we can observe that the convergence to the steady state is accelerated by more than 20 times using the 2^{nd}-order SD scheme. For 3^{rd}- and 4^{th}-order SD schemes, the three-stage Runge-Kutta schemes failed to converge, as shown in Figs. 5b and 5c.

The effects of other convergence parameters are quite similar to the previous case, and the results are thus omitted.

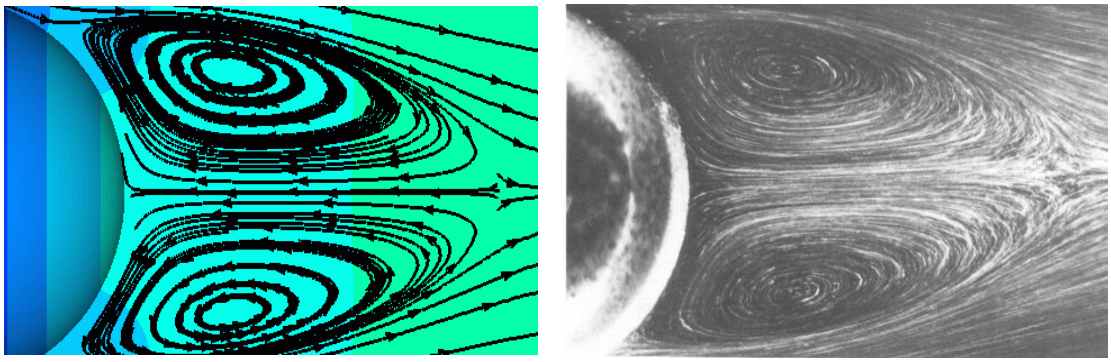
3.3 Steady viscous flow around a sphere

A steady viscous flow around a sphere is used here to demonstrate the performance with the implicit LU-SGS method. The mesh used is the same as in the inviscid flow case. The Reynolds number based on the diameter was chosen to be 118 since an experimental streamline picture [35] is available for comparison. The computations were performed



(a) 2nd Order (b) 3rd Order (c) 4th Order

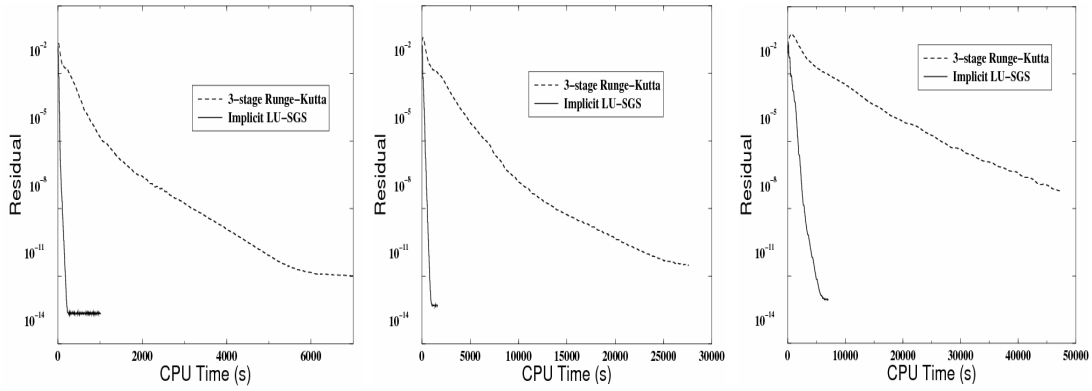
Figure 5: Residual histories of inviscid flow over a 3D-bump.



(a) Computation (b) Experiment

Figure 6: Comparison of streamlines of the flow field between computation and experiment.

using the 2nd- to 4th-order schemes. The computational streamlines and the size of the separation region agree well with experimental streamlines, as shown in Fig. 6 using the 4th-order SD scheme. The explicit and implicit schemes are compared in Fig. 7, which shows the convergence histories in terms of CPU times. For all the tested schemes, the speedup with the LU-SGS algorithm is more than an order of magnitude and up to 2 orders, fully demonstrating the effectiveness of the implicit algorithm. The implicit SD schemes of various orders of accuracy are also compared in Fig. 8. Note that the convergence in terms of iterations is nearly order independent. It is also obvious that the CPU times for high-order schemes increase non-linearly with respect to the order of accuracy. This is partly because there are far more DOFs in the higher order simulations than the lower order ones. For example, the simulation with a 3rd-order SD scheme has 3.4 times the number of DOFs than that with a 2nd-order SD scheme.



(a) 2nd Order (b) 3rd Order (c) 4th Order

Figure 7: Residual histories in term of CPU times for viscous flow over a sphere.

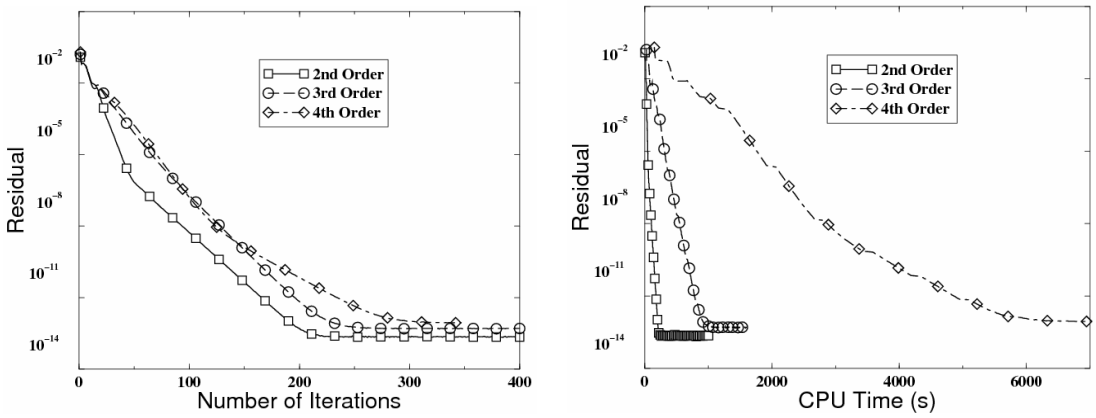


Figure 8: Convergence histories in terms of number of iterations and CPU times for various SD schemes.

3.4 Laminar flow over NACA0012 airfoil

In this test, we consider a subsonic viscous flow problem over the NACA0012 airfoil at an angle of attack $\alpha=0^0$, free stream Mach number $M=0.5$, and Reynolds number Re$=5000$. This is a widely used validation case for viscous flow solvers, and for example used in [21, 24]. The computational grid is displayed in Fig. 9 with $72\times24\times1$ cells (72 cells along the airfoil surface and 24 cells in the radial direction and 1 cell in the span-wise direction). The grid extends about 20 chords away form the airfoil. The computations were performed using 2nd- to 5th-order SD schemes. The wall is assumed to be non-slip and adiabatic, and is represented by a quadratic patch. The Reynolds number is near the upper limit for a steady laminar flow. A distinguishing feature of this test case is the separation region of the flow occurring near the trailing edge, which causes the formation of a small recirculation bubble that extends in the near-wake region of the airfoil.

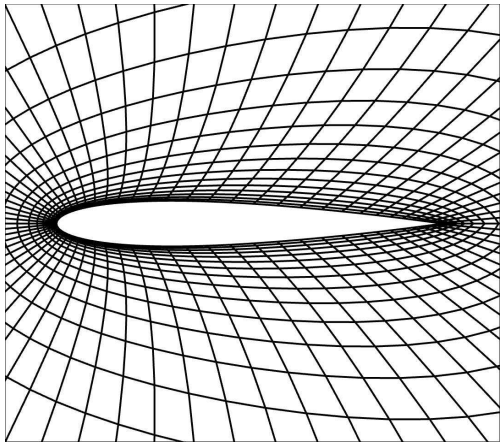
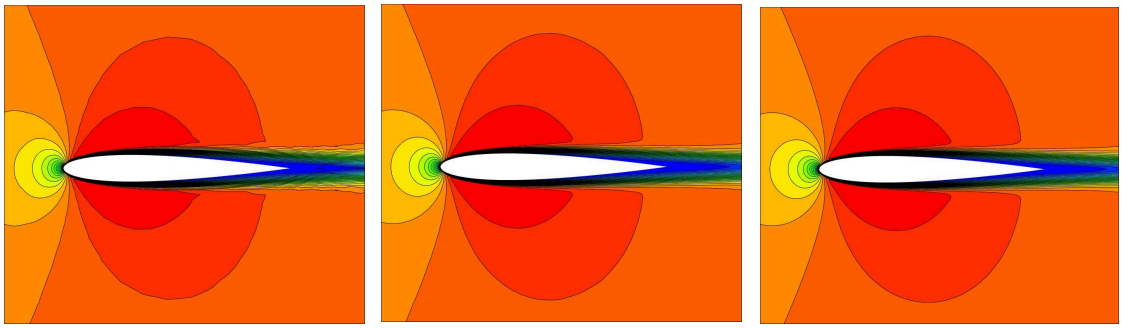


Figure 9: Computational grid ($72\times24\times1$) around a NACA0012 airfoil.



(a) 2^{nd} Order (b) 3^{rd} Order (c) 4^{th} Order

Figure 10: Computed Mach number contours using 2^{nd}- to 4^{th}-order SD schemes.

Fig. 10 shows the Mach contours computed with SD schemes of 2^{nd}-, 3^{rd}- and 4^{th}-order of accuracies. It is obvious that the solution is becoming smoother and smoother with the increasing order of the polynomial reconstruction, indicating the solution is more accurate. In fact, the solutions between the 3^{rd}- and 4^{th}-order schemes are nearly indistinguishable, demonstrating order-independent flow convergence. The convergence histories of the explicit and implicit 3^{rd}-order SD schemes are plotted in Fig. 11. The speedup factor in this case is estimated to be more than 2 orders. The residual histories of different orders of accuracies are again weakly dependent on the order of accuracy, and are thus not shown. The computed skin friction coefficients with 2^{nd}- to 4^{th}-order SD schemes are displayed in Fig. 12. The plot shows a clear convergence of C_f from the 2^{nd}- to 4^{th}-order prediction. In fact, the difference between the 3^{rd}- and 4^{th}-order results is less than 0.5%. An enlarged view near the separation point where $C_f = 0$ is shown in Fig. 12b. From this figure, we can also obtain the location of the separation

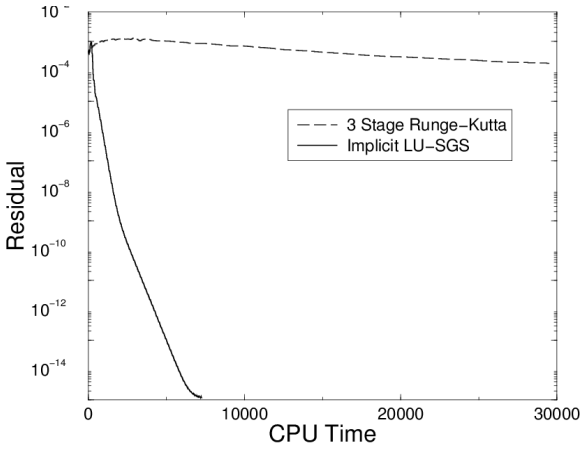


Figure 11: Convergence histories for 3rd-order SD scheme.

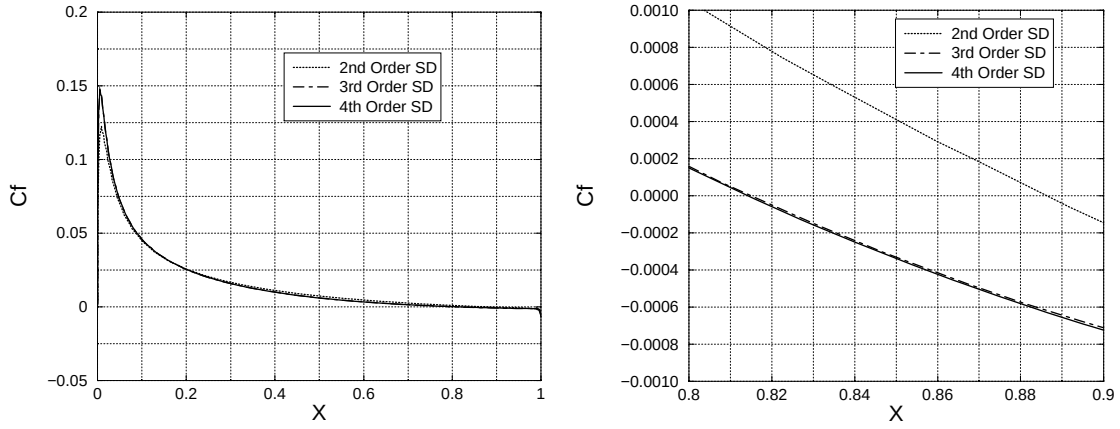


Figure 12: Computed skin friction coefficient using SD schemes of different orders of accuracy.

point. The 2nd-, 3rd- and 4th-order schemes predict the separation point at 88.6%, 81.5% and 81.4% respectively. Next we compare the present simulation with that performed in [21, 24]. In addition to the separation point, the drag coefficients due to pressure and friction can also be computed. The comparisons between the present simulation and those of [21, 24] are summarized in Table 1. The 5th-order prediction is also shown to verify order-independent convergence. Note that the first three decimal places of drag coefficients and separation point location nearly reached convergence with the 3rd-order SD scheme. In the other two computations, such convergence was not demonstrated. However, there is a perfect match among the three in the prediction of the separation point. The maximum difference between the predicted drag coefficients is between 0.4% to 2.9%.

Table 1: Comparison of pressure and viscous drag coefficients and location of separation point between the present simulation and other simulations in the literature.

Method	***NDOFs***	CD_p	CD_f	**Separation Point**
2^{nd}-Order SD	144×48 (6,912)	0.02174	0.03272	88.6%
3^{rd}-Order SD	216×72 (15,552)	0.02219	0.03250	81.5%
4^{th}-Order SD	288×96 (27,648)	0.02225	0.03251	81.4%
5^{th}-Order SD	360×120 (43,200)	0.02225	0.03251	81.4%
Triangle Scheme $\kappa_4=1/64$ [21]	$320\times64\times2$ (40,960)	0.0225	0.0344	83.4%
Triangle Scheme $\kappa_4=1/128$ [21]	$320\times64\times2$ (40,960)	0.0228	0.0336	82.4%
Triangle Scheme $\kappa_4=1/256$ [21]	$320\times64\times2$ (40,960)	0.0229	0.0332	81.4%
Cell Centered FV [24]	256×64 (16,384)	0.02256	0.03301	80.9%
Cell Centered FV [24]	512×128 (65,536)	0.02235	0.03299	81.4%

4 Conclusions

In this paper, an efficient implicit lower-upper symmetric Gauss-Seidel (LU-SGS) solution algorithm has been developed for compressible flow simulations using a high-order multi-domain spectral difference method on unstructured hexahedral grids. The LU-SGS solver is preconditioned by the block element matrix. A numerical approach is developed to compute the element Jacobian matrix, resulting in straightforward operations for arbitrarily complex residual operators. The implicit scheme has shown 1 to 2 orders of magnitude of speed-up relative to the multi-stage Runge-Kutta explicit time integration scheme for several demonstration problems. The issue of shock-capturing is very important, and will be discussed in a future publication. Although the method is shown to be effective, the memory requirement for higher than 4^{th} order schemes is prohibitive as the element matrix size is equal to $(25\times\text{order}^6)$. The development of lower-memory solvers for higher order schemes is a future challenge.

Acknowledgments

The study was partially funded by Rockwell Scientific/DARPA contract W911NF-04-C-0102, AFOSR grant FA9550-06-1-0146, and 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 DARPA, AFOSR, DOE, or the U.S. Government.

References

[1] R. Abgrall and P.L. Roe, High order fluctuation schemes on triangular meshes, J. Sci. Comput., 19 (2003), 3-36.

[2] T.J. Barth and P.O. Frederickson, High-order solution of the euler equations on unstructured grids using quadratic reconstruction, AIAA Paper, (1990), 90-0013.

[3] F. Bassi and S. Rebay, High-order accurate discontinuous finite element solution of the 2D Euler equations, J. Comput. Phys., 138 (1997), 251-285.
[4] M. Blanco and D.W. Zingg, A fast solver for the Euler equation on unstructured grids using a Newton-GMRES metod, AIAA Paper, (1997), 97-0331.
[5] P.N. Brown and Y. Saad, Hybrid Krylov methods for nonlinear systems of equations, SIAM J. Sci. Comp., 11(3) (1990), 450-481.
[6] R.F. Chen and Z.J. Wang, Fast, block lower-upper symmetric Gauss-Seidel scheme for arbitrary grids, AIAA J., 38(12) (2000), 2238-2245.
[7] B. Cockburn and C.-W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: General framework, Math. Comput. 52 (1989), 411-435.
[8] B. Cockburn and C.-W. Shu, The Runge-Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems, J. Comput. Phys., 141 (1998), 199-224.
[9] J.A. Ekaterinaris, High-order accurate, low numerical diffusion methods for aerodynamics, Progress Aerospace Sci., 41 (2005), 192-300.
[10] N.T. Frink, Assessment of an unstructured-grid method for predicting 3-D turbulent viscous flows, AIAA Paper, (1996), 96-0292.
[11] C. Hu and C.-W. Shu, Weighted essentially non-oscillatory schemes on triangular meshes, J. Comput. Phys., 150 (1999), 97-127.
[12] P.G. Huang, Z.J. Wang and Y. Liu, An implicit space-time spectral difference method for discontinuity capturing using adaptive polynomials, AIAA, (2005), 2005-5255.
[13] A. Jameson and D.A. Caughey, How many steps are required to solve the Euler equations of steady compressible flow: In search of a fast solution algorithm, 15th AIAA Computational Fluid Dynamics Conference, June 2001, Anaheim, CA.
[14] D.A. Kopriva, A staggered-grid multidomain spectral method for the compressible Navier-Stokes equations, J. Comput. Phys., 143 (1998), 125-158.
[15] D.A. Kopriva and J.H. Kolias, A conservative staggered-grid Chebyshev multidomain method for compressible flows, J. Comput. Phys., 125 (1996), 244-261.
[16] S.K. Lele, Compact finite difference schemes with spectral-like resolution, J. Comput. Phys., 103 (1992), 16-42.
[17] Y. Liu, C.-W. Shu, E. Tadmor and M. Zhang, Non-oscillatory hierarchical reconstruction for central and finite volume schemes, Commun. Comput. Phys., 2 (2007), 933-963.
[18] Y. Liu, M. Vinokur and Z.J. Wang, Discontinuous spectral difference method for conservation laws on unstructured grids, in: Proceeding of the 3rd International Conference in CFD, Toronto, Canada July 2004.
[19] Y. Liu, M. Vinokur and Z.J. Wang, Spectral difference method for unstructured grids I: Basic formulation, J. Comput. Phys., 216 (2006), 780-801.
[20] Y. Liu, M. Vinokur and Z.J. Wang, Spectral (finite) volume method for conservation laws on unstructured grids V: Extension to three-dimensional systems, J. Comput. Phys., 212 (2006), 454-472.
[21] D. J. Mavriplis, A. Jameson and L. Martinelli, Multigrid solution of the Navier-Stokes equations on triangular meshes, AIAA paper, (1989), 89-0120.
[22] A.T. Patera, A Spectral element method for fluid dynamics: Laminar flow in a channel expansion, J. Comput. Phys., 54 (1984), 468-488.
[23] N. Qin and B.E. Richards, Sparse quasi-Newton methods for Navier-Stokes solution, Notes Numer. Fluid Mech., 29 (1990), 474-480.
[24] R. Radespiel and R.C. Swanson, An investigation of cell-centered and cell vertex multigrid

schemes for Navier-Stokes equations, AIAA Paper, (1989), 89-0543.
[25] P.L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys., 43 (1981), 357-372.
[26] V.V. Rusanov, Calculation of interaction of non-steady shock waves with obstacles, J. Comput. Math. Phys. USSR, 1 (1961), 267-279.
[27] C.-W. Shu, Total-Variation-Diminishing time discretizations, SIAM J. Sci. Stat. Comput., 9 (1988), 1073-1084.
[28] C.-W. Shu, Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws, in: A. Quarteroni (Ed.), Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, Lecture Notes in Mathematics, Springer-Verlag, Berlin/New York, 1998, Vol. 1697, p. 325.
[29] B. Sjogreen and N. A. Petersson, A Cartesian embedded boundary method for hyperbolic conservation laws, Commun. Comput. Phys., 2 (2007), 1199-1219.
[30] M. Soetrisno, S.T. Imlay and D.W. Roberts, A zonal implicit procedure for hybrid structured-unstructured grids, AIAA Paper, (1994), 94-0645, .
[31] Y. Sun, Z.J. Wang and Y. Liu, Spectral (finite) volume method for conservation laws on unstructured grids VI: Extension to viscous flow, J. Comput. Phys., 215 (2006), 41-58.
[32] Y. Sun, Z.J. Wang and Y. Liu, High-order multi-domain spectral difference method for the Navier-Stokes equations on unstructured hexahedral grids, Commun. Comput. Phys., 2(2) (2007), 310-333.
[33] Y. Sun, Z.J. Wang, Y. Liu and C.L. Chen, Efficient implicit LU-SGS algorithm for high-order spectral difference method on unstructured hexahedral grids, AIAA Paper, (2007), 2007-0313.
[34] C.K.W. Tam and J.C. Webb, Dispersion-relation-preserving finite difference schemes for computational acoustics, J. Comput. Phys., 107(2) (1993), 262-281.
[35] S. Taneda, Experimental investigations of the wake behind a sphere at low Reynolds numbers, J. Phys. Soc. Japan, 11 (1956), 1104-1108.
[36] V. Venkatakrishnan and D.J. Mavriplis, Implicit solvers for unstructured meshes, J. Comput. Phys., 105(1) (1993), 83-91.
[37] V. Venkatakrishnan and D.J. Mavriplis, Implicit method for the computation of unsteady flows on unstructured grids, J. Comput. Phys., 127(2) (1996), 380-397.
[38] M. Visbal and D. Gaitonde, Shock capturing using compact- differencing- based methods, AIAA Paper, (2005), 2005-1265.
[39] Z.J. Wang, Spectral (finite) volume method for conservation laws on unstructured grids: Basic formulation, J. Comput. Phys., 178 (2002), 210-251.
[40] Z.J. Wang, High-order methods for the Euler and Navier-Stokes equations on unstructured grids, J. Progress Aerospace Sci., 43 (2007), 1-41.
[41] Z.J. Wang, and Y. Liu, Spectral (finite) volume method for conservation laws on unstructured grids III: Extension to one-dimensional systems, J. Sci. Comput., 20(1) (2004), 137-157.
[42] Z.J. Wang, L. Zhang and Y. Liu, Spectral (finite) volume method for conservation laws on unstructured grids IV: Extension to two-dimensional systems, J. Comput. Phys., 194 (2004), 716-741.
[43] Z.J. Wang, Y. Liu, G. May and A. Jameson, Spectral difference method for unstructured grids II: Extension to the Euler equations, J. Sci. Comput., 32(1) (2007), 45-71.
[44] S. Yoon and A. Jameson, Lower-upper symmetric-Gauss-Seidel method for the Euler and Navier-Stokes equations, AIAA J., 26 (1988), 1025-1026.