AIAA JOURNAL
Vol. 44, No. 10, October 2006

Convergence Acceleration of an Upwind Least Squares Finite Difference based Meshless Solver

D. Sridar* and N. Balakrishnan†
Indian Institute of Science, Bangalore 560 012, India

DOI: 10.2514/1.15362

One of the important trends in the present day CFD research is the development of new class of flow solvers called meshless solvers. All that this class of solvers require is a distribution of points in the computational domain. The development of upwind least squares finite difference method can be considered as a landmark event in the advances being made in the area of meshless solvers. In this paper, we propose an implicit time integration methodology for upwind least squares finite difference procedure. The idea of matrix-free implicit procedure in the framework of finite volume solver has been exploited in the present work to obtain a cheap and robust implicit time integration procedure. The present formulation shows excellent convergence acceleration over the explicit upwind least squares finite difference procedure.

I. Introduction

NUMERICAL estimation of spatial derivatives holds the key to the solution of fluid dynamic equations in most of the real life flows. Different methodologies such as the finite element or finite volume procedures essentially differ in the way these derivatives are estimated. Most of these classical methods strongly depend on the discretization of the computational domain into finer cells. The class of methods referred to as "meshless" or "gridfree" solvers do not depend on such a discretization of the computational domain [1]. These methods merely require distribution of points in the computational domain. For obvious reasons, getting a point distribution is a far simpler task compared with "grid generation." For example, what is popularly referred to as a Cartesian mesh generator [2,3] can in fact provide the required point distribution. It is well known that the performance of finite volume solvers, in conjunction with Cartesian grids, is seriously hampered by the appearance of small cut cells, particularly for viscous flows [4]. It is in this context the research in meshless solvers becomes very important. In recent times, the development of the upwind least squares finite difference scheme (LSFD–U) [5–8] can be considered a landmark event in the advances being made in the area of "meshless solvers," particularly in the context of CFD. A detailed discussion of various possibilities toward the development of upwind meshless solvers, proof of order of accuracy of generalized finite difference schemes, and conservation studies are presented along with results for a number of validating test cases in Sridar and Balakrishnan [8]. In the earlier works, the authors have employed simple local time stepping procedure to accelerate convergence to steady state [7,8]. It is well known that the acceleration offered by such a procedure is still limited by the stability constraint of an explicit time integration procedure. One of the means to circumvent this problem is to employ an implicit time integration procedure. In this work we establish the implicit LSFD–U procedure. It is worthwhile to remark that implicit acceleration of codes based on meshless solvers is not very common, with the only exception being the work of Morinishi [9] and Anandhanarayanan et al. [10]. While the earlier work [9] does not clearly bring out the speedup obtained using the implicit procedure, a speedup of about 4 times the explicit procedure is reported in [10]. On the contrary, in the present work, which extensively exploits the advancements made in the area of implicit finite volume solvers [11,12], convergence acceleration up to 30 times the explicit procedure has been observed. This work also gains significance in the light of more recent developments in Cartesian mesh calculations, wherein meshless solver is used in close vicinity of the body surface for solution update [13,14].

Received 2 January 2005; revision received 27 October 2005; accepted for publication 31 March 2006.

*Research Student, CAd Lab, Department of Aerospace Engineering.
†Assistant Professor, CAd Lab, Department of Aerospace Engineering; nbalak@aero.iisc.ernet.in (corresponding author).

II. Upwind–Least Squares Finite Difference Method

The meshless framework offered by the LSFD–U procedure is more recent. Therefore, the details are not commonly known to the researchers. Based on this consideration the explicit LSFD–U procedure is briefly introduced in this section.

Consider the 2-D Euler equation of gas dynamics, given by

$$\frac{\partial U}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0 \tag{1}$$

A typical point distribution required by the present methodology to solve the Eq. (1) is shown in Fig. 1. To determine the derivatives f_x and g_y, using upwind least squares finite difference (LSFD–U) method, we introduce a fictitious interface J associated with the neighbor $j \in S_i$ of node i, where S_i is the connectivity set of node i. Let $\Delta \boldsymbol{r_J} = (\Delta x_J, \Delta y_J)$ represent the vector $\boldsymbol{iJ}$ and $\hat{n}_J = (n_{x_J}, n_{y_J})$ represent the unit vector along this direction. The directional flux F along $\boldsymbol{iJ}$ is given by

$$F = f n_{x_J} + g n_{y_J} \tag{2}$$

The interfacial flux F_J can be determined using any upwind scheme. Also, we use a linear reconstruction procedure to determine the left and right states at the fictitious interface to determine the interfacial flux F_J. Then the flux difference associated with the neighbor j is given by

$$\Delta F_J = F_J - F_i \tag{3}$$

Now, the task at hand is to recover the 2-D gradients of the fluxes from these directional flux differences, which are essentially unidirectional. An estimate of these flux differences obtained by expanding the interfacial flux F_J using Taylor series about i is

$$\Delta F_J^e = \frac{1}{\boldsymbol{\Delta} r_J}\left[\frac{\partial f}{\partial x}\Delta x_J^2 + \left(\frac{\partial f}{\partial y} + \frac{\partial g}{\partial x}\right)\Delta x_J \Delta y_J + \frac{\partial g}{\partial y}\Delta y_J^2\right] \tag{4}$$

The above equation, along with (3), represents an overdetermined system for f_x, g_y, and $f_y + g_x$, and we solve this using the method of least squares. Therefore, the error associated with node j is

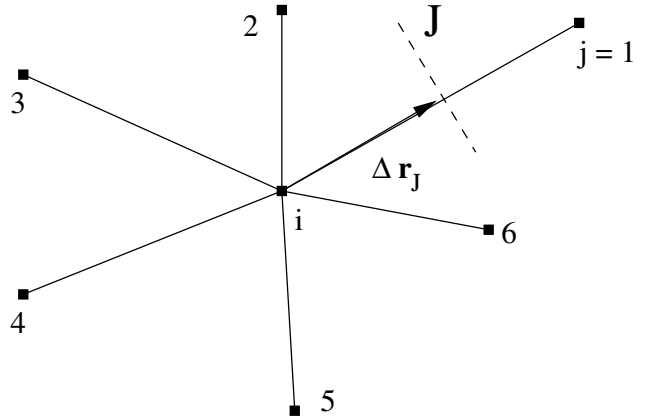


Fig. 1 2-D implementation of LSFD–U.

$$E_J = (\Delta F_J - \Delta F_J^e)|\boldsymbol{\Delta r}_J| \tag{5}$$

Minimizing the L_2 norm of error E_J with respect to the flux derivatives, we get

$$\bar{\bar{G}}\boldsymbol{\partial F} = \boldsymbol{\delta}(\Delta F_J) \tag{6}$$

where $\bar{\bar{G}}$ is a real symmetric geometric matrix, $\boldsymbol{\partial F}$ is a vector of flux derivatives and $\delta(\boldsymbol{\Delta F}_J)$ is a RHS vector. They are given by

$$\bar{\bar{G}} = \begin{bmatrix} \sum \Delta x_J^4 & \sum \Delta x_J^3 \Delta y_J & \sum \Delta x_J^2 \Delta y_J^2 \\ \sum \Delta x_J^3 \Delta y_J & \sum \Delta x_J^2 \Delta y_J^2 & \sum \Delta x_J \Delta y_J^3 \\ \sum \Delta x_J^2 \Delta y_J^2 & \sum \Delta x_J \Delta y_J^3 & \sum \Delta y_J^4 \end{bmatrix} \tag{7}$$

$$\boldsymbol{\partial F} = \begin{bmatrix} f_x \\ f_y + g_x \\ g_y \end{bmatrix} \tag{8}$$

$$\boldsymbol{\delta}(\Delta F_J) = \begin{bmatrix} \sum \Delta F_J |\boldsymbol{\Delta r}_J| \Delta x_J^2 \\ \sum \Delta F_J |\boldsymbol{\Delta r}_J| \Delta x_J \Delta y_J \\ \sum \Delta F_J |\boldsymbol{\Delta r}_J| \Delta y_J^2 \end{bmatrix} \tag{9}$$

In the above equations, the summations are over $j \in S_i$. If $\partial_l F$ and $\delta_l(\Delta F_J)$ represent the components of the vectors $\boldsymbol{\partial F}$ and $\boldsymbol{\delta}(\Delta F_J)$, respectively, and if the elements of $\bar{\bar{G}}^{-1}$ are represented by G_{kl}^{-1}, we have

$$\partial_k F = \sum_{l=1}^{3} G_{kl}^{-1} \delta_l(\Delta F_J) \quad \text{for } k = 1, 2, 3 \tag{10}$$

This leads to the following explicit state update formula:

$$\frac{\Delta_t U_i^{\exp}}{\Delta t_i} = -(\partial_1 F + \partial_3 F) \tag{11}$$

where $\Delta_t(\cdot)_i = (\cdot)_i^{n+1} - (\cdot)_i^n$. For more details pertaining to the order of accuracy of LSFD–U and some of its interesting variants, the readers are referred to [8].

III. Implicit LSFD–U

In the earlier works, the authors have used a simple strategy like local time stepping to accelerate convergence to steady state [8]. It is well known that the acceleration offered by such a procedure is still limited by the stability constraint of an explicit time stepping scheme. One of the means to circumvent this problem is to employ an implicit time stepping procedure. In this section we establish the implicit LSFD–U procedure.

The implicit state update for LSFD–U can be written as

$$\frac{\Delta_t U_i}{\Delta t} = -(\partial_1 F^{n+1} + \partial_3 F^{n+1})_i \tag{12}$$

In the above equation, the spatial derivatives appearing on the RHS are given by

$$\boldsymbol{\partial} \boldsymbol{F}^{n+1} = \bar{\bar{G}}^{-1} \boldsymbol{\delta}(\Delta F_J^{n+1}) \tag{13}$$

Similar to the definition of $\boldsymbol{\delta}(\cdot)$ in the previous section, we define the following vector to be used in the implicit procedure:

$$\boldsymbol{\delta}^j(\cdot) = \begin{bmatrix} (\cdot)|\boldsymbol{\Delta r}_J| \Delta x_J^2 \\ (\cdot)|\boldsymbol{\Delta r}_J| \Delta x_J \Delta y_J \\ (\cdot)|\boldsymbol{\Delta r}_J| \Delta y_J^2 \end{bmatrix}$$

such that $\boldsymbol{\delta}(\cdot) = \sum_{j \in S_i} \boldsymbol{\delta}^j(\cdot)$.

Substituting for spatial derivatives from Eq. (10) in Eq. (12), we get

$$\frac{\Delta_t U_i}{\Delta t} = -\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l(\Delta F_J^{n+1}) \tag{14}$$

Using the flux vector splitting formulation and the homogeneity property of the Euler fluxes, after a temporal linearization, we have

$$F_J^{n+1} = F_J^n + A^+(U_i^n)\Delta_t U_i + A^-(U_j^n)\Delta_t U_j \tag{15}$$

Substituting Eq. (15) in Eq. (14), we get

$$\left[\frac{I}{\Delta t} - \sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l(A^-(U_i^n))\right] \Delta_t U_i + \sum_{j \in S_i} \sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l^j(A^-(U_j^n)) \Delta_t U_j = R_i^n \tag{16}$$

where

$$R_i^n = -\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l(\Delta F_J^n)$$

Using Jameson and Yoon's split flux Jacobians [15] given by

$$A^{\pm} = \frac{1}{2}[A \pm \rho_A I]$$

where ρ_A is spectral radius of directional flux Jacobians, we have

$$\left\{\frac{I}{\Delta t} - \frac{1}{2}\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l[A(U_i^n) - \rho_{A_i} I]\right\} \Delta_t U_i + \frac{1}{2} \sum_{j \in S_i} \sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l^j[A(U_j^n) - \rho_{A_j} I] \Delta_t U_j = R_i^n \tag{17}$$

Rewriting the above equation in matrix–vector form, we have

$$\mathbb{M}\, \Delta_t W = \mathbb{R}^n \tag{18}$$

where $\mathbb{M}$ is a sparse banded matrix, $\Delta_t W$ is the unknown vector given by $\Delta_t W = [\Delta_t U_1, \Delta_t U_2, \ldots, \Delta_t U_{\mathrm{NP}}]^T$, $\mathbb{R}^n = [R_1^n, R_2^n, \ldots, R_{\mathrm{NP}}^n]^T$ is the residue vector, and NP is the total number of points in the computational domain. The elements of matrix $\mathbb{M}$ are given by

$$M_{ii} = \left\{\frac{I}{\Delta t} - \frac{1}{2}\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l[A(U_i^n) - \rho_{A_i} I]\right\}$$

$$M_{ij} = \begin{cases} \frac{1}{2}\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l^j[A(U_j^n) - \rho_{A_j} I], & j \in S_i \\ 0, & \text{otherwise} \end{cases}$$

It is well known that the solution of the matrix–vector system given by Eq. (18), involving direct inversion of a large sparse banded matrix $\mathbb{M}$, is computationally intensive; therefore, it can undermine the benefits one can obtain through use of an implicit procedure. It is a general practice to solve this system iteratively using a suitable relaxation procedure. In this work, we have evaluated the performance of three such relaxation procedures as against an explicit computation. To solve the matrix–vector system given by Eq. (18) iteratively, the matrix $\mathbb{M}$ is decomposed into three matrices: an upper triangular matrix $\mathbb{C}$, a diagonal matrix $\mathbb{D}$, and a lower triangular matrix $\mathbb{E}$; thus, $\mathbb{M} = \mathbb{C} + \mathbb{D} + \mathbb{E}$. The elements of these matrices are

$$C_{ij} = \begin{cases} M_{ij}, & i < j \\ 0, & i \geq j \end{cases}, \qquad D_{ij} = \begin{cases} M_{ij}, & i = j \\ 0, & i \neq j \end{cases},$$
$$E_{ij} = \begin{cases} M_{ij}, & i > j \\ 0, & i \leq j \end{cases}$$

With the above decomposition, Eq. (18) reads

$$(\mathbb{C} + \mathbb{D} + \mathbb{E})\Delta_t W = \mathbb{R}^n \tag{19}$$

A. Lower-Upper Symmetric Gauss–Seidel

First of the three relaxation procedures used in this work is lower-upper symmetric Gauss–Seidel (LU–SGS). This is based on the approximate LU decomposition proposed by Jameson and Turkel [16]; which was later modified as a matrix-free implicit procedure [11]. The procedure is given by

$$(\mathbb{E} + \mathbb{D})\mathbb{D}^{-1}(\mathbb{D} + \mathbb{C})\Delta_t W = \mathbb{R}^n \tag{20}$$

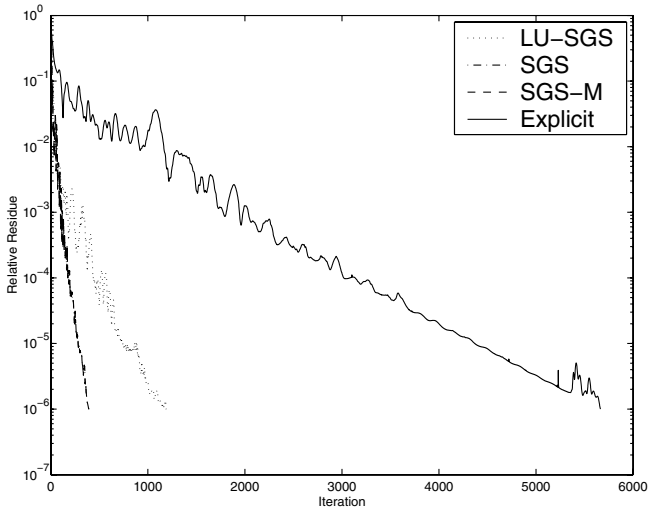
The Eq. (20) is inverted in two steps as

$$(\mathbb{E} + \mathbb{D})\Delta_t W^* = \mathbb{R}^n,$$
$$\mathbb{D}^{-1}(\mathbb{D} + \mathbb{C})\Delta_t W = \Delta_t W^* \tag{21}$$

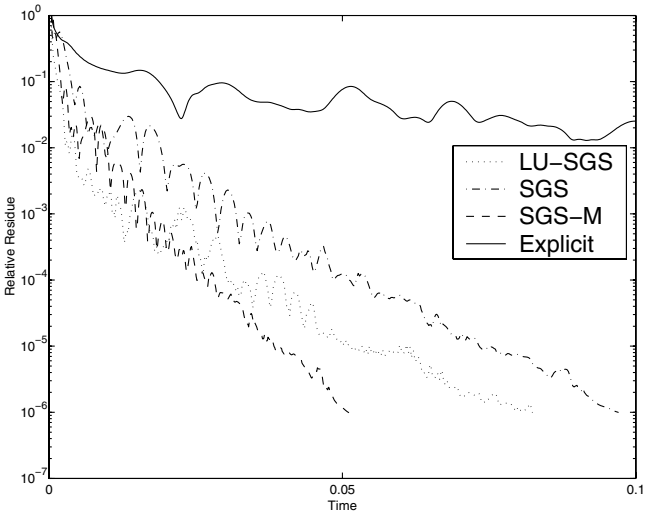
This procedure results in the following forward and reverse sweeps:

1) Forward sweep

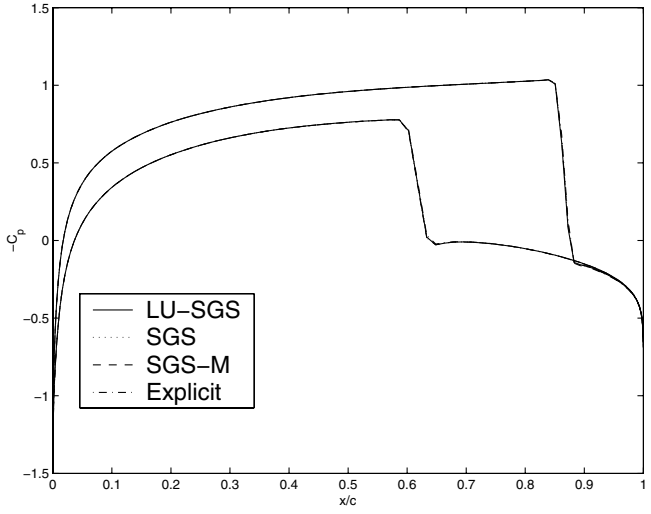
$$\Delta_t U_i^* = D_{ii}^{-1}\left\{R_i^n - \frac{1}{2}\sum_{j<i} \times \left[\sum_{l=1}^{3}(G_{1l}^{-1} + G_{3l}^{-1})\delta_l^j(\Delta_t F_j^* - \rho_{A_j}\Delta_t U_j^*)\right]\right\}$$



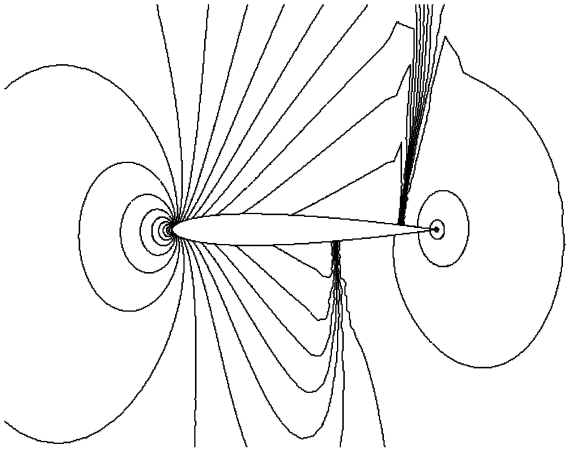
a)



b)



c)



d)

Fig. 2 Comparison of different implicit procedures against explicit computation for transonic flow past NACA 0012.

Table 1 Summary of the details of computational domain and interfacial formulas

Test case	Domain		Numerical flux formula
	No. of nodes	No of nodes on the wall	
1. Transonic flow past NACA 0012 ($M_\infty = 0.85, \alpha = 1.0$ deg)	8913[a]	198	AUSM [19]
2. Subsonic flow past NACA 0012 ($M_\infty = 0.63, \alpha = 2.0$ deg)	8913[a]	198	Roe [20]
3. Supersonic flow past ramp in channel ($M_{\text{inlet}} = 2.0$)	3293	120	KFVS [21,22]
4. Supersonic flow past semicylinder ($M_\infty = 3.0$)	6921[b]	120	KFVS [21,22]
5. Transonic flow past NACA 0012 ($M_\infty = 0.85, \alpha = 1.0$ deg)	4385[a]	118	AUSM [19]

[a]The far-field = 10 chords.
[b]The far-field = 3 radii.

Table 2 C_L and C_D values for transonic flow past NACA 0012 airfoil

	LU–SGS	SGS	SGS–M	Explicit	AGARD
C_L	0.3863	0.3894	0.3894	0.3881	0.330–0.389
C_D	0.0565	0.0566	0.0566	0.0556	0.0464–0.0590

Table 3 C_L and C_D values for subsonic flow past NACA 0012 airfoil

	LU–SGS	SGS	SGS–M	Explicit	GAMM
C_L	0.3333	0.3335	0.3329	0.3339	0.329–0.336
C_D	−0.0005	−0.0009	−0.0007	−0.0010	0.003–0.07

2) Reverse sweep

$$\Delta_t U_i = \Delta_t U_i^* - D_{ii}^{-1} \times \left\{ \frac{1}{2} \sum_{j>i} \left[\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l^j (\Delta_t F_j - \rho_{A_j} \Delta_t U_j) \right] \right\} \quad (22)$$

The solution of above equation still involves inversion of the D_{ii} matrix. The conserved variables at time level $n+1$ are calculated from

$$U_i^{n+1} = U_i^n + \Delta_t U_i$$

B. Symmetric Gauss–Seidel

In this procedure Eq. (19) is solved iteratively. Every iteration comprises of a forward and reverse sweep as described below:

$$\begin{aligned} (\mathbb{E} + \mathbb{D}) \Delta_t W^* &= \mathbb{R}^n - \mathbb{C} \Delta_t W^{(k-1)}, \\ (\mathbb{D} + \mathbb{C}) \Delta_t W^{(k)} &= \mathbb{R}^n - \mathbb{E} \Delta_t W^* \end{aligned} \quad (23)$$

With $\Delta_t W^{(0)} = 0$, the above formula translates to

1) Forward sweep

$$\Delta_t U_i^* = D_{ii}^{-1} \left\{ R_i^n - \frac{1}{2} \sum_{j<i} \times \left[\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l^j (\Delta_t F_j^* - \rho_{A_j} \Delta_t U_j^*) \right] - \frac{1}{2} \sum_{j>i} \left[\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l^j (\Delta_t F_j^{(k-1)} - \rho_{A_j} \Delta_t U_j^{(k-1)}) \right] \right\}$$

2) Reverse sweep

$$\Delta_t U_i^{(k)} = D_{ii}^{-1} \left\{ R_i^n - \frac{1}{2} \sum_{j<i} \times \left[\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l^j (\Delta_t F_j^* - \rho_{A_j} \Delta_t U_j^*) \right] - \frac{1}{2} \sum_{j>i} \left[\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l^j (\Delta_t F_j^{(k)} - \rho_{A_j} \Delta_t U_j^{(k)}) \right] \right\} \quad (24)$$

Similar to the LU–SGS procedure described in the previous section, the SGS procedure also involves inversion of matrix D_{ii}. The conserved variables at time level $n+1$ are obtained through

$$U_i^{n+1} = U_i^n + \Delta_t U_i^{(k_{\max})}$$

C. Modified Symmetric Gauss–Seidel

Here we propose a simple modification of the SGS procedure, which makes it matrix free. This is accomplished by allowing $\Delta_t F_i$ term lag by one inner iteration. The forward and reverse sweeps associated with the modified SGS procedure are given by

1) Forward sweep

$$\Delta_t U_i^* = d_i^{-1} \left\{ R_i^n - \frac{1}{2} \sum_{j<i} \times \left[\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l^j (\Delta_t F_j^* - \Delta_t F_i^{(k-1)} - \rho_{A_j} \Delta_t U_j^*) \right] - \frac{1}{2} \sum_{j>i} \left[\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l^j (\Delta_t F_j^{(k-1)} - \Delta_t F_i^{(k-1)} - \rho_{A_j} \Delta_t U_j^{(k-1)}) \right] \right\}$$

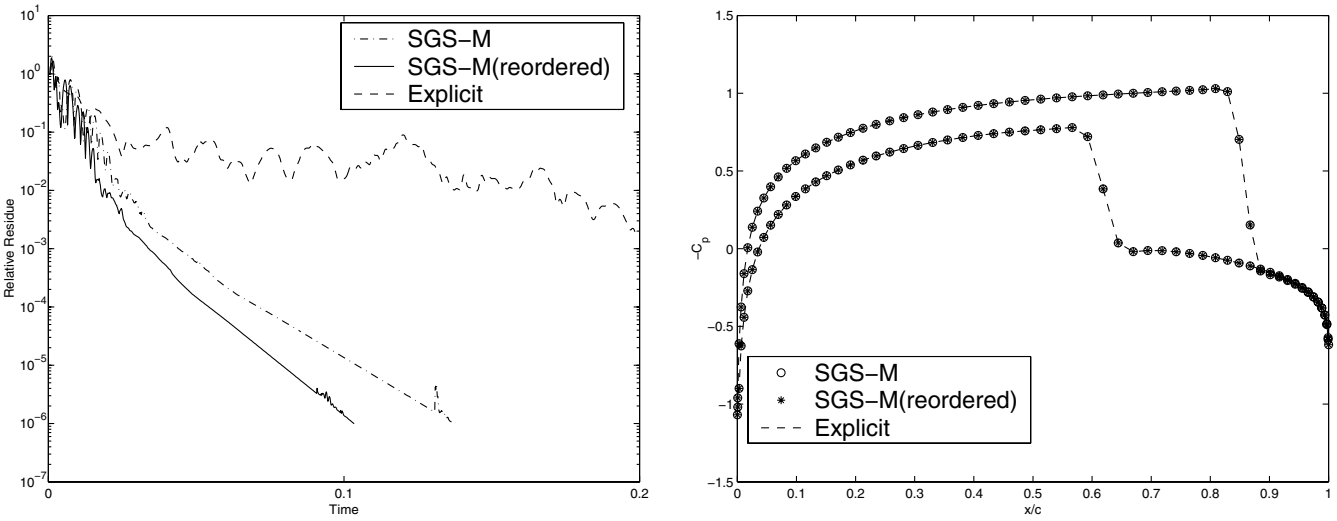
2) Reverse sweep

$$\Delta_t U_i^{(k)} = d_i^{-1} \left\{ R_i^n - \frac{1}{2} \sum_{j<i} \times \left[\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l^j (\Delta_t F_j^* - \Delta_t F_i^* - \rho_{A_j} \Delta_t U_j^*) \right] - \frac{1}{2} \sum_{j>i} \times \left[\sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l^j (\Delta_t F_j^{(k)} - \Delta_t F_i^* - \rho_{A_j} \Delta_t U_j^{(k)}) \right] \right\} \quad (25)$$

where

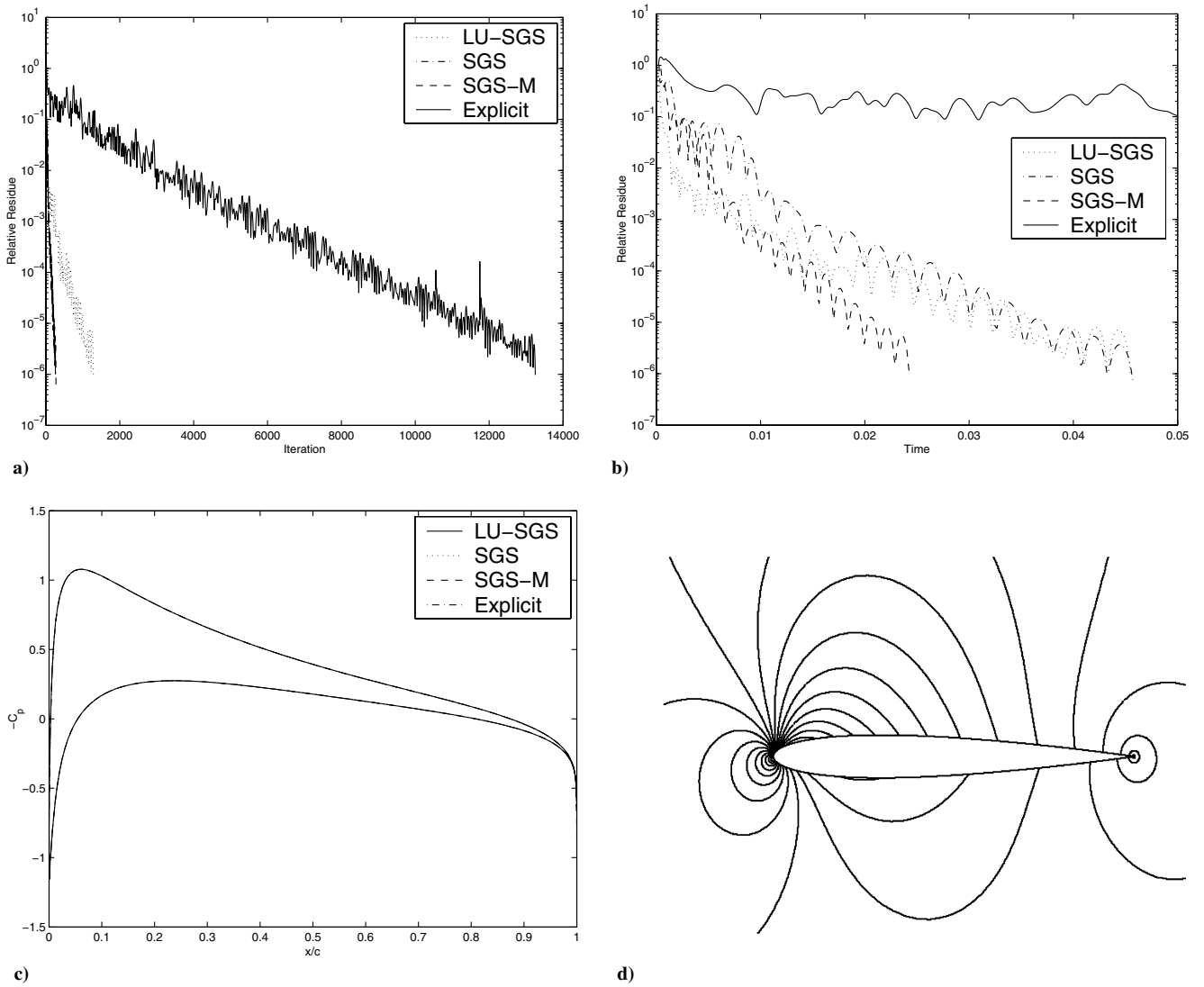
$$d_i = \left[\frac{1}{\Delta t} - \frac{1}{2} \sum_{l=1}^{3} (G_{1l}^{-1} + G_{3l}^{-1}) \delta_l(\rho_{A_i}) \right] \quad (26)$$

Again, the conserved variables at time level $n+1$ is obtained through



a) b)

Fig. 3 Comparative performance of SGS–M with node reordering.



a) b)

c) d)

Fig. 4 Comparison of different implicit procedures against explicit computation for subsonic flow past NACA 0012.

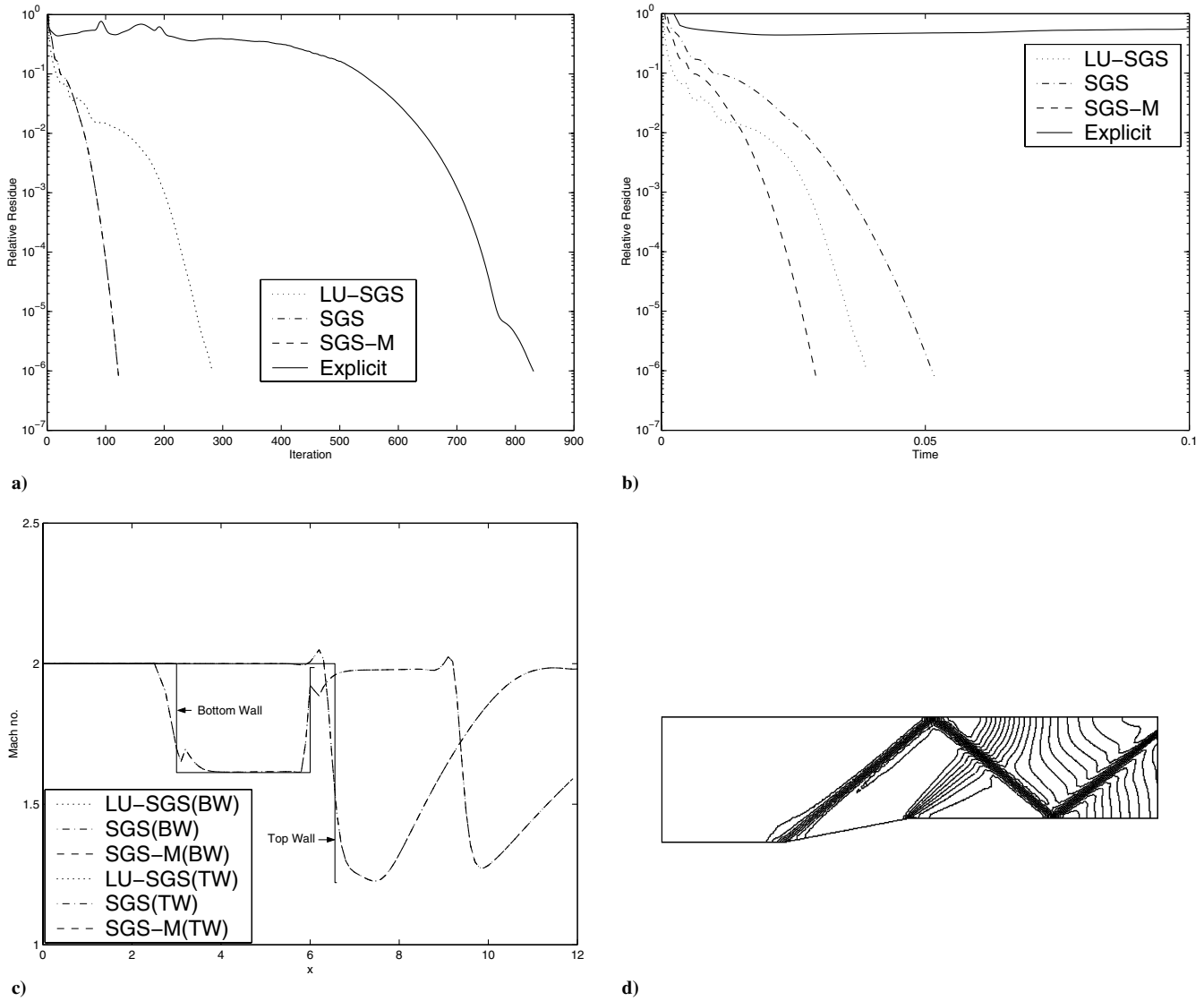


Fig. 5 Comparison of different implicit procedures against explicit computation of supersonic flow past ramp in channel; TW: top wall, BW: bottom wall.

$$U_i^{n+1} = U_i^n + \Delta_t U_i^{(k_{\max})}$$

In all the above procedures one has to still invert a real symmetric geometric matrix $\bar{\bar{G}}$ given by Eq. (7). This matrix can be inverted at the preprocessing stage itself, and thereby saving computational time.

IV. Results and Discussion

The 2-D computations performed to test the implicit LSFD–U are presented in this section. These 2-D cases are chosen to test the performance of the implicit scheme over a wide range of Mach numbers. In all the computations performed using implicit LSFD–U, the boundary terms are treated explicitly. On the solid wall, a strong wall boundary condition due to Balakrishnan and Fernandez [17] is used. On the outer boundary, a far-field boundary condition due to Thomas and Salas [18] is used. For the supersonic flow, a nonreflective boundary condition on the inflow and outflow boundaries is used. The convergence is assumed after a 6 decades fall in relative residue based on density. The explicit computations are performed using four step Runga–Kutta scheme with local time stepping. The connectivity set S_i for the present computations is obtained employing the same procedure described in Sridar and Balakrishnan [8]. This procedure ensures well conditioning of the geometric matrix $\bar{\bar{G}}$. One of the advantages of LSFD–U procedure is the flexibility to choose different interfacial flux formula–based on their relative performance at different Mach numbers. Table 1 summarizes the details of computational domain and interfacial formula for various test cases to be presented in this paper. The uniformity of computing hardware among all the computations is maintained using a Pentium 4 computer with a clock speed of 1.8 GHz. The time taken per iteration is computed automatically using appropriate routines in the computing language. In all the relative residue convergence plots, the time is normalized with respect to the explicit convergence time.

The performance of the implicit procedure depends upon parameters, such as convergence of inner iterations and the choice of CFL number. Therefore, a number of numerical experiments were done with the objective of optimizing the performance of the implicit procedure [23]. Based on these experiments, the following strategy has been made use of in generating the results presented in this paper:

1) The convergence of the inner iteration is declared if a relative L_2 norm defined by $\|\Delta_t U^{(k)} - \Delta_t U^{(k-1)}\|_2 / \|\Delta_t U^{(1)}\|_2$ is less than ε ($\varepsilon = 10^{-2}$), subjected to a maximum of $k_{\max}$ ($k_{\max} = 6$) inner iterations.

2) The CFL number is made proportional to the iteration number with a proportionality constant of κ ($\kappa = 10^6$).

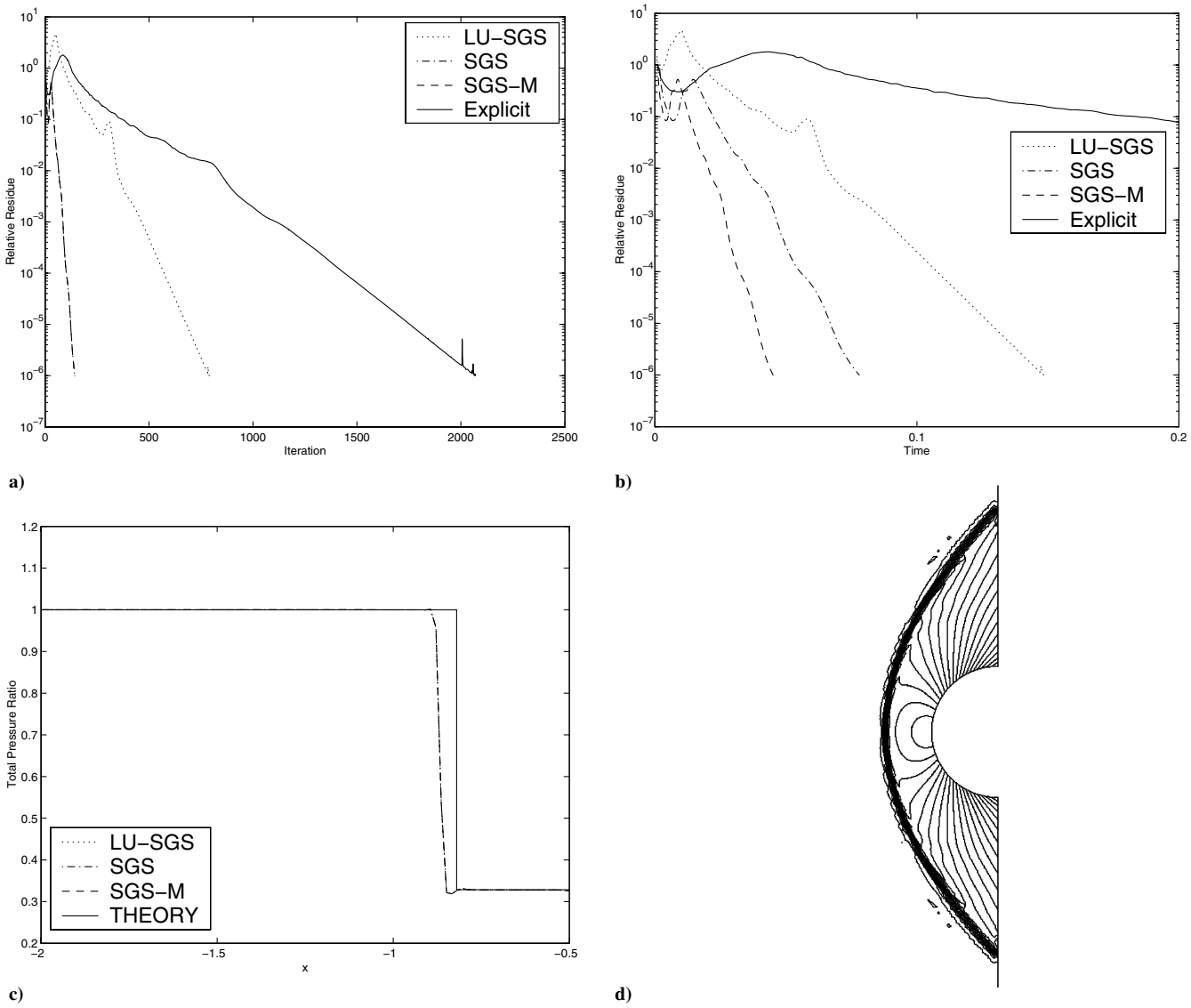


Fig. 6 Comparison of different implicit procedures against explicit computation for high supersonic flow past half-cylinder.

While this strategy can be expected to give an optimal performance, the optimal values of ε, $k_{\max}$ and κ can be problem dependent.

A. Transonic Flow Past NACA0012

This test case, transonic ($M_\infty = 0.85$) flow past a NACA0012 airfoil at 1.0 deg angle of attack, is a 2-D lifting AGARD [24] test case for which standard values of C_L and C_D are available for comparison. The steady state convergence of all the three implicit procedures are compared against the explicit computation in Figs. 2a and 2b. The modified symmetric Gauss–Seidel (SGS–M) procedure converges faster than LU–SGS and SGS procedures. The C_p plot, in Fig. 2c, is strong proof of implicit scheme exactly reproducing the explicit result at the steady state. The Mach contours obtained using SGS–M are presented in Fig. 2d. The contours are identical to other implicit procedures and explicit LSFD–U, which is evident from the Fig. 2c. The C_L and C_D values are compared against the AGARD values in Table 2.

It is well known that the performance of SGS and LU–SGS iterations are sensitive to the ordering of the nodes. This aspect is reemphasized in this section for meshless solvers. For this purpose we choose domain 5 (see Table 1) with 4385 nodes, and reorder the nodes using Cuthill and McKee [25] algorithm. From Figs. 3a and 3b, one can certainly see a performance improvement with node reordering, which for the present case is about 20%. Again, Fig. 3b clearly demonstrates SGS–M exactly reproducing the explicit result at steady state.

B. Subsonic Flow Past NACA0012

This test case, subsonic ($M_\infty = 0.63$) flow past a NACA0012 airfoil at 2 deg angle of attack, is chosen to test the convergence of implicit procedures at low speed. The computational domain and other flow parameters, except Mach number and angle-of-attack, for this computation is the same as the previous test case. Figure 4c shows the surface C_p distribution. One can see that the solution of both implicit and explicit computations is exactly the same. From Figs. 4a and 4b one can conclude that, again for this subsonic flow test case, SGS–M gives the fastest convergence of all three implicit procedures. In Table 3, the C_L and C_D values are compared against the standard GAMM [26] values. The lower values of C_D suggest that the present computations using the Roe flux formula are less dissipative. The Mach contours for SGS–M are shown in Fig. 4d.

C. Supersonic Flow Past Ramp in Channel

This test case involves simulating a supersonic flow ($M_{\text{inlet}} = 2.0$) past a 10.75 deg ramp in channel. The flow domain consists of 3293 nodes, which is a fairly coarse point distribution. The test case involves accurate capturing of shock reflection and shock-expansion interaction. Figure 5c gives the wall Mach number distribution on

both the top and bottom wall for all three implicit procedures. It is evident from the figure that both the top and bottom shock reflections and shock-expansion interaction are captured fairly accurately. Also plotted in the Fig. 5c is the theoretical values—plotted up to the shock reflection on the top wall and up to expansion on the bottom wall—which shows that the location of shock reflection on the top wall was captured accurately. The pressure contours for SGS–M is shown in Fig. 5d, which demonstrates the accurate resolution of flow features even on a coarse grid. The convergence of relative residue with respect to explicit CPU time is given in Fig. 5b. In this case also the SGS–M is the fastest converging among the three procedures. Figure 5b shows a speedup of as much as 30 times the explicit scheme.

D. High Supersonic Flow Past Half-Cylinder

The last test case involves simulation of a standing bow-shock due to a high supersonic ($M_\infty = 3.0$) flow past a half-cylinder. The flow domain consists of 6921 nodes. Figure 6c shows the variation of total pressure ratio (ratio of total pressure to free stream total pressure) along the symmetry line. The results are compared with the theoretical variation of the same quantity. In calculating the theoretical jump along the symmetry line, the shock stand-off distance is calculated using Ambrosio and Wortman [27] correlation. Though the start of total pressure ratio jump for all three procedures is slightly away from the empirical position, the end of the jump coincides with that of the empirical position. It should be remembered here that the grid used in the present computation is not a solution adapted one, and better results can be expected with an adapted grid. The pressure contours for SGS–M procedure is presented in Fig. 6d. This, along with Fig. 6c, clearly demonstrates the robustness of the implicit LSFD–U method. The relative residue convergence is shown in Fig. 6; it again establishes the superiority of the matrix-free SGS–M procedure.

V. Conclusion

In this work, an implicit time integration procedure for convergence acceleration of the meshless solver LSFD–U is developed. Various relaxation schemes, including a modified SGS procedure that offers a matrix-free framework, for the implicit time integration is discussed. Some comments are made on the convergence of the inner iterations for both SGS and SGS–M procedures. Based on the results obtained for a particular test case, it is observed that use of a relative L_2 norm, for convergence of inner iterations, with a ceiling on the maximum of number of inner iterations (6 in this case) yields best results, when the CFL is allowed to increase proportional to the iterations. From the numerical experiments, it is found that the proposed matrix-free SGS–M procedure offers the best convergence acceleration to steady state. For the test cases considered, this implicit procedure offered a speedup of at least 20, in comparison with explicit convergence. This compares well with the speedup achieved using matrix-free implicit finite volume solvers.

Acknowledgment

The authors thank N. Munikrishna, Research Scholar, Department of Aerospace Engineering, I.I.Sc., Bangalore, India, for fruitful discussions they had and help in debugging the implicit code.

References

[1] MacCormack, R. W., "A Perspective on a Quarter Century of CFD Research," AIAA Paper 93-3291-CP, 1993.

[2] Clarke, D. K., Salas, M. D., and Hassan, H. A., "Euler Calculations for Multi-Element Airfoils Using Cartesian Grids," *AIAA Journal*, Vol. 24, No. 3, 1986, pp. 353–358.

[3] Coirier, W. J., and Powell, K. G., "An Accuracy Assessment of Cartesian-Mesh Approaches for the Euler Equations," *Journal of Computational Physics*, Vol. 117, 1995, pp. 121–131.

[4] Coirier, W. J., "An Adaptively-Refined Cartesian Cell Based Scheme for the Euler and Navier–Stokes Equations," Ph.D. thesis, Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, 1994.

[5] Ninawe, A., Munikrishna, N., and Balakrishnan, N., "Viscous Flow Computations Using a Meshless Solver, LSFD–U," *Computational Fluid Dynamics*, Springer–Verlag, Berlin, 2004.

[6] Balakrishnan, N., "New Least Squares Based Finite Difference Method," Indian Institute of Science, Rept. 1999 FM 9, Department of Aerospace Engineering, Bangalore, India, 1999.

[7] Balakrishnan, N., and Sridar, D., "On a Generalised Finite Difference Framework and its Order of Accuracy," *Computational Fluid Dynamics*, Springer, New York, 2000, pp. 797–798.

[8] Sridar, D., and Balakrishnan, N., "An Upwind Finite Difference Scheme for Mesh-Less Solvers," *Journal of Computational Physics*, Vol. 189, 2003, pp. 1–29.

[9] Morinishi, K., "An Implicit Gridless Type Solver for the Navier–Stokes Equations," *Computational Fluid Dynamics Journal*, Vol. 9, No. 1, 2000.

[10] Anandhanarayanan, K., Nagarathinam, M., and Deshpande, S. M., "An LUSGS Convergence Accelerator for LSKUM-Based Euler Solver," in *Proceedings of the 4th Annual CFD Symposium*, CFD Division, Aeronautical Society of India, Bangalore, India, 2004.

[11] Luo, H., Baum, J. D., and Löhner, R., "A Fast Matrix-Free Implicit Method for Compressible Flows on Unstructured Grids," *Journal of Computational Physics*, Vol. 146, 1998, pp. 664–690.

[12] Shende, N., and Balakrishnan, N., "A New Migratory Memory Algorithm (MMA) for Unstructured Implicit Finite Volume Codes," *AIAA Journal*, Vol. 42, No. 9, 2004, pp. 1863–1870.

[13] Kirshman, D. J., and Liu, F., "A Gridless Boundary Condition Method for the Solution of the Euler Equations on Embedded Cartesian Meshes with Multigrid," *Journal of Computational Physics*, Vol. 201, 2004, pp. 119–147.

[14] Luo, H., Baum, J. D., and Löhner, R., "A Hybrid Cartesian Grid and Gridless Method for Compressible Flows," AIAA Paper 2005-0492, 2005.

[15] Jameson, A., and Yoon, S., "Lower-Upper Implicit Schemes with Multi Grid for the Euler Equations," *AIAA Journal*, Vol. 26, No. 9, 1988, pp. 1025–1026.

[16] Jameson, A., and Turkel, E., "Implicit Schemes and LU Decompositions," *Mathematics of Computation*, Vol. 37, No. 156, 1981, pp. 385–397.

[17] Balakrishnan, N., and Fernandez, G., "Wall Boundary Conditions for Inviscid Compressible Flows on Unstructured Meshes," *International Journal of Numerical Methods in Fluids*, Vol. 28, 1998, pp. 1481–1501.

[18] Thomas, J. L., and Salas, M. D., "Far-Field Boundary Condition for Transonic Lifting Solutions to the Euler Equations," AIAA Paper 85-0020, 1985.

[19] Liou, M.-S., and Steffen, C. J., Jr., "A New Flux Splitting Scheme," *Journal of Computational Physics*, Vol. 107, 1993, pp. 23–39.

[20] Roe, P. L., "Approximate Riemann Solver, Parameter Vectors and Difference Schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.

[21] Mandal, J. C., and Deshpande, S. M., "Kinetic Flux Vector Splitting for Euler Equations," *Computers and Fluids*, Vol. 23, No. 2, 1994, pp. 447–478.

[22] Pullin, D. I., "Direct Simulation Methods for Compressible Inviscid Ideal-Gas Flow," *Journal of Computational Physics*, Vol. 34, 1980, pp. 231–244.

[23] Sridar, D., and Balakrishnan, N., "Convergence Acceleration of LSFD–U Meshless Solver," Indian Institute of Science, Rept. 2004 FM 13, Department of Aerospace Engineering, Bangalore, India, July 2004.

[24] Anon, "Test Cases for Inviscid Flow Field Methods," Fluid Dynamics Panel Working Group, AGARD Rept. AR 211, July 1985.

[25] Cuthill, E., and McKee, J., "Reducing the Band Width of Sparse Symmetric Matrices," in *Proceeding of ACM National Conference*, 1969, pp. 157–172.

[26] *GAMM: Workshop on Numerical Solution of Compressible Euler Flows*, Tech. Rept., 1986.

[27] Billig, F. S., "Shock-Wave Shape Around Spherical and Cylindrical-Nosed Bodies," *Journal of Spacecraft and Rockets*, Vol. 4, No. 6, 1967, pp. 822–823.

S. Mahalingam
Associate Editor