NUMERICAL MATHEMATICS: Theory, Methods and Applications
Numer. Math. Theor. Meth. Appl., Vol. **1**, No. 1, pp. 92-112 (2008)

A Multigrid Block LU-SGS Algorithm for Euler Equations on Unstructured Grids

Ruo Li[1,*], Xin Wang[2] and Weibo Zhao[1]

[1] *LMAM & School of Mathematical Sciences, Peking University, Beijing, China.*
[2] *LMAM, CCSE & School of Mathematical Sciences, Peking University, Beijing, China.*

Received 21 January, 2007; Accepted (in revised version) 28 November, 2007

Abstract. We propose an efficient and robust algorithm to solve the steady Euler equations on unstructured grids. The new algorithm is a Newton-iteration method in which each iteration step is a linear multigrid method using block lower-upper symmetric Gauss-Seidel (LU-SGS) iteration as its smoother. To regularize the Jacobian matrix of Newton-iteration, we adopted a local residual dependent regularization as the replacement of the standard time-stepping relaxation technique based on the local CFL number. The proposed method can be extended to high order approximations and three spatial dimensions in a nature way. The solver was tested on a sequence of benchmark problems on both quasi-uniform and local adaptive meshes. The numerical results illustrated the efficiency and robustness of our algorithm.

AMS subject classifications: 65N22, 65N50, 65N55

Key words: Multigrid, block LU-SGS, Euler equations, aerodynamics, airfoil.

1. Introduction

In the last decades, one of the most active research areas in computational aerodynamics has been concerned with the numerical simulation of the complex flow field of aircrafts with practical configuration. Nowadays its rapid development and daily improvement play an important part in accelerating the revolution of aerofoil designing strategies and methods. Because of the hyperbolic nature of Euler equations in the subsonic, transonic and supersonic regimes, many numerical schemes can be chosen to solve the unsteady Euler equations. The finite volume method [11] is one of the most widely used schemes. The nonlinear algebraic system obtained from the finite volume discretization of Euler equations was often solved by certain Newton-iteration. It is a main challenge to develop efficient and robust iterative algorithms for solving the nonlinear algebraic system, especially on unstructured grids. In spite of the difficulties of this problem, remarkable progress

*Corresponding author. *Email addresses:* rli@math.pku.edu.cn (R. Li), wangxin.tom@gmail.com (X. Wang), wbzhao@pku.edu.cn (W. Zhao)

has been made. The solver developed by Jameson, with a series of customized numerical techniques including multigrid acceleration, local time stepping, implicit residual smoothing and enthalpy damping, on structured grids demonstrated great efficiency with a lot of impressive numerical examples [6–8, 10, 11]. Such highly nonlinear system can now be solved with residual convergent to machine accuracy on current desktop computers within minutes. Among these acceleration techniques, the local preconditioning can be quite effective [13, 14], too, with a judiciously chosen precondition matrix. For the system discretized on unstructured grids, the implicit LU-SGS iterative algorithm has been extensively adopted since it was first introduced by Jameson and Turkel [12]. The LU-SGS method was used as a relaxation method for solving the unfactorized implicit scheme by Yoon and Jameson [22–24]. It was further developed and applied to 3D viscous flow fields by Riger and Jameson [17]. Since then, many authors have applied the LU-SGS method to viscous flows on both structured and unstructured grids [2, 4, 18, 25]. Noticing the special formation of the equations, it is more appropriate to solve the linearized Jacobian matrix block by block. Therefore as a further development of the LU-SGS iteration, Wang [5] proposed a block LU-SGS method together with some numerical examples, converged at a satisfactory speed as expected.

In this paper, we developed a multigrid solver using the block LU-SGS iteration as its smoother. On the unstructured grids, we first discretized the steady Euler equations to obtain the nonlinear algebraic system. Then the nonlinear system was linearized with the standard Newton-iteration. It is popular to regularize the linearized system by adding a local artificial time relaxation. The weight of this time relaxation term was calculated dynamically using a local CFL number. This CFL number is different from the CFL number used in solving a time-dependent conservation law. For a time-dependent conservation law, the essential role of the CFL number is to keep the stability of the numerical schemes. Therefore, it is an $\mathcal{O}(1)$ number to make the time stepping length to be the ratio of the typical mesh size and the maximal wave propagation speed. As one of the basic differences in solving the steady Euler equations, the intermediate state of the solution is out of the main concerns. Only if the iterative algorithm can converge, the CFL number can be chosen as large as possible to achieve a faster convergence rate. Generally, the CFL number should be about $\mathcal{O}(1)$ at the beginning of the iteration as a bootstrap of the total algorithm, and then it can be dynamically increased for better efficiency. A balance between the magnitude of the CFL number and the convergence of the iteration is generally required by maximizing the total convergence rate. Based on such a understanding of the local CFL number choosing strategy, we will not use the standard regularization in which a local artificial time relaxation term is added into the Jacobian matrix of the Newton-iteration. Instead, we used a residual related regularization,i.e. $\alpha \|RHS\|_{l^1}$, where RHS is the residual of the linearized system on each grid cell. The magnitude of the cell residual can locally quantify how close to the steady state the flow field is. Therefore it is quite natural to require the local CFL number to be dependent on the local residual. Due to the same scaling between a norm of the local residual and local grid cell size, the regularization term can be simply set as a constant times of a norm of the local residual. With this local residual dependent regularization, our algorithm can automatically choose a moderate local CFL number so as

to achieve a subtle balance between the stability of the iterative procedure and a satisfactory rate of convergence. It can be seen in the numerical examples that our algorithm with such regularization behaved very robust. For a sequence of very different configurations in terms of far field free-stream conditions and domain geometry, the solver converged without any modification to the parameter α.

In our algorithm, the regularized linear system for every Newton-iteration was solved by a linear multigrid method in which the block LU-SGS proposed by Wang was utilized as the smoother. For the implementation of the multigrid solver, we first developed a mesh aggregation program to get a sequence of coarse mesh grids from the finest one by following the idea of the sketchy mesh aggregation algorithm in Section 9.4 in [3]. Numerical results showed that our codes worked quite robustly, and can generate quite high quality coarse meshes, both from the quasi-uniform and the local refined background meshes. The coarse meshes generated were used to construct the projection operator from finer mesh grids to coarser mesh grids, which is then applied on both the sparse matrix and the right hand side of the regularized linear system. The block LU-SGS iteration, as the smoother of the multigrid solver, can contribute a piecewise constant correction on every cell patch on the coarse meshes. We adopted a V-cycle type multigrid iteration in our implementation since the best performance based on extensive numerical experiments. It should be noted that due to the highly nonlinear nature of the original problem, the multigrid iteration was only applied one or two steps for the linearized system; otherwise the convergence of the nonlinear iteration may be slowed down or even impeded occasionally.

The remaining of the paper is organized as follows. In the next section, we described the main algorithm in detail, including the discretization of Euler equations, the new regularization for the Jacobian matrix, the linear multigrid method with the revised block LU-SGS iteration as the smoother, the reconstruction and the limiter adopted, and some standard technique details. In Section 3, we presented some numerical examples to illustrate the robustness and efficiency of our algorithm. Some concluding remarks were made in the final section.

2. Discretization and algorithm

We consider the compressible inviscid Euler equations in two space dimensions:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = 0, \tag{2.1}$$

where $\mathbf{U}$ and $\mathbf{F}$ are the vectors of the conservative variables and the inviscid flux, respectively. For ideal flows, its components are as

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} \rho u & \rho v \\ \rho u^2 + p & \rho uv \\ \rho uv & \rho v^2 + p \\ u(E+p) & v(E+p) \end{bmatrix},$$

where ρ is the density, $\mathbf{u} = (u, v)$ is the velocity vector, p is the pressure, and E is the total energy. The closure of the system is achieved by the equation of state

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2),$$

where $\gamma = 1.4$ is the ratio of specific heats for air, which is mainly composed of diatomic molecules: nitrogen and oxygen. For the stationary solution, we are solving the system (2.1) without the time derivative term; thus the equations under consideration are

$$\nabla \cdot \mathbf{F}(\mathbf{U}) = 0. \tag{2.2}$$

In our algorithm, the extensively used artificial time-derivative regularization method was replaced by a residual dependent regularization. Therefore, we will discretize (2.2) directly instead of using (2.1).

In this paper, the domain Ω of the Euler equations (2.1) is $\mathbb{R}^2 \backslash \Omega_c$, where Ω_c is the area occupied by the airfoil, i.e., the body of the aircraft in 2D case. Due to the unboundedness of the solution domain, we adopted the current popular strategy, which is to solve the equations in a bounded domain as $\Omega \bigcap \{|\mathbf{x}| < R\}$, where R is a certain times of the airfoil chord length, and made use of the far field vortex correction technique to remedy the error introduced by the abrupt domain truncation. Therefore, before the end of the paper, we took the problem domain Ω directly as the truncated bounded domain with the truncation radius R in terms of the chord length.

2.1. Finite volume discretization

Let $\mathscr{T}$ be a partition of Ω with its cells denoted as $T \in \mathscr{T}$, that $\bar{\Omega} = \bigcup_{T \in \mathscr{T}} \bar{T}$, and any two different cells have no common interior parts. The intersection of two different cells $\bar{T}_j$ and $\bar{T}_k$ can be either an edge e_{jk}, a vertex in the partition or empty set. The unit out normal on the edge e_{jk}, pointing from T_j to T_k is denoted as $\mathbf{n}_{jk}$. Let h_T denote the diameter of cell T. After applying the Green's formula on a cell T_j for (2.2), we have

$$\oint_{\partial T_j} \mathbf{F}(\mathbf{U}_h) \cdot \mathbf{n} \, ds = 0, \tag{2.3}$$

where $\mathbf{n}$ is the unit out normal of T_j. The numerical solution $\mathbf{U}_h$ is approximated on every cell by a polynomial. By taking the cells in the partition as the control volumes, the flux in the integral form (2.3) of (2.2) was approximated as

$$\oint_{\partial T_j} \mathbf{F}(\mathbf{U}_h) \cdot \mathbf{n} \, ds \approx \sum_{e_{jk} \in \partial T_j} \int_{e_{jk}} \bar{\mathbf{F}}(\mathbf{U}_j, \mathbf{U}_k) \cdot \mathbf{n}_{jk} dl, \tag{2.4}$$

where $\bar{\mathbf{F}}$ is the numerical flux. Several numerical fluxes have been verified to be effective for the transonic flow problem, including CUSP flux [9], HLLC flux [1] and Lax-Wendroff flux, in our numerical experiments.

2.2. Newton-iteration and regularization

The system (2.4) is nonlinear and the corresponding Newton-iteration can be written as

$$\sum_{e_{jk}\in\partial T_j}\int_{e_{jk}}\bar{\mathbf{F}}^{(n)}\cdot\mathbf{n}_{jk}dl+\sum_{e_{jk}\in\partial T_j}\int_{e_{jk}}\left(\frac{\partial\bar{\mathbf{F}}^{(n)}}{\partial\mathbf{U}_j}\delta\mathbf{U}_j^{(n)}\right)\cdot\mathbf{n}_{jk}dl + \sum_{e_{jk}\in\partial T_j}\int_{e_{jk}}\left(\frac{\partial\bar{\mathbf{F}}^{(n)}}{\partial\mathbf{U}_k}\delta\mathbf{U}_k^{(n)}\right)\cdot\mathbf{n}_{jk}dl=0, \tag{2.5}$$

where

$$\bar{\mathbf{F}}^{(n)}=\bar{\mathbf{F}}\left(\mathbf{U}_j^{(n)},\mathbf{U}_k^{(n)}\right)\quad\text{and}\quad\mathbf{U}_j^{(n+1)}=\mathbf{U}_j^{(n)}+\tau_j\delta\mathbf{U}_j^{(n)},$$

τ_j is a relaxation parameter on cell T_j. In our finite volume implementation, the increment $\delta\mathbf{U}^{(n)}$ in the Newton-iteration is piecewise constant on every cell. Since the Jacobian matrix in the Newton-iteration (2.5) can be singular, the iteration is in fact unable to be carried out directly. The approach to make the iteration (2.5) work is to add a regularization term to the Jacobian matrix. Generally, the regularization is formulated as an artificial time derivative term

$$\int_{T_j}\frac{\delta\mathbf{U}_j^{(n)}}{\Delta t_j}dx,$$

which is added to the left-hand side of (2.5), where Δt_j is the artificial local time step. The artificial local time step Δt_j is often given as

$$CFL\times h_{T_j}/\lambda^+,$$

where CFL is the local CFL number, h_{T_j} is the local mesh grid size and $\lambda^+=|\mathbf{u}|+c$, with c the speed of sound, i.e., the maximal local wave propagation speed on T_j. The formula is the same as the one for time-dependent conservation laws except that the CFL number here is often chosen quite large. For time-dependent conservation laws, the time stepping length is chosen to make the computation stable; while in solving the stationary solution, the main concern is to keep the iterative procedure convergent and to accelerate the convergence speed. Numerical experiences showed that the CFL number can be larger when the solution is closer to the steady state locally. Furthermore, with a larger CFL number, the convergence can often be accelerated. Based on these observations and noticing that the local residual

$$\mathbf{R}_j^{(n)}\triangleq\sum_{e_{jk}\in\partial T_j}\int_{e_{jk}}\bar{\mathbf{F}}^{(n)}\cdot\mathbf{n}_{jk}dl$$

can quantify how close the solutions are to the steady state ones, we directly use the l^1 norm of $\mathbf{R}_j^{(n)}$ as the regularization term to the Jacobian matrix. Consequently, our regular-

ized version of (2.5) is of the form

$$\alpha\left\|\mathbf{R}_j^{(n)}\right\|_{l^1}\delta\mathbf{U}_j^{(n)}+\sum_{e_{jk}\in\partial T_j}\int_{e_{jk}}\left(\frac{\partial\bar{\mathbf{F}}^{(n)}}{\partial\mathbf{U}_j}\delta\mathbf{U}_j^{(n)}\right)\cdot\mathbf{n}_{jk}dl$$
$$+\sum_{e_{jk}\in\partial T_j}\int_{e_{jk}}\left(\frac{\partial\bar{\mathbf{F}}^{(n)}}{\partial\mathbf{U}_k}\delta\mathbf{U}_k^{(n)}\right)\cdot\mathbf{n}_{jk}dl=-\mathbf{R}_j^{(n)}, \tag{2.6}$$

where the parameter α is often set as 2 in the numerical examples presented in the next section.

Now we can write the Newton-iteration algorithm as follows:

Algorithm 2.1: [Newton-iteration]

1. Input $\mathbf{U}^{(0)}$ as the initial guess, and set $n=0$;
2. Use an approximate solver for the linear system (2.6) to get a $\delta\mathbf{U}^{(n)}$;
3. Update $\mathbf{U}^{(n+1)}$ by $\mathbf{U}^{(n)}+\tau\delta\mathbf{U}^{(n)}$;
4. Reconstruct $\mathbf{U}^{(n+1)}$ using its cell mean values to get a piecewise polynomial expression on each cell;
5. Check if the residual $\mathbf{R}^{(n+1)}$ is small enough: if yes, stop; otherwise, set $n:=n+1$ and goto step 2.

The steps not clear yet in the above algorithm are the approximate solver for the linear system (2.6) and the reconstruction procedure, which will be clarified in Sections 2.3 and 2.4 below.

2.3. Linear multigrid solver for the Jacobian matrix

A linear multigrid solver is mainly composed of two ingredients: the projection operator and the smoother. To construct the projection operator, we utilized the idea of aggregated multigrid method as in [3] and reference therein to generate a sequence of coarse meshes. We denote the original partition on Ω as $\mathscr{T}_0=\mathscr{T}$. The mesh aggregation algorithm can give a sequence of coarse meshes denoted as $\mathscr{T}_m$, $m=1,2,\cdots$, in which every cell $T_{m,j}$ is the union of some cells in mesh $\mathscr{T}_{m-1}$, i.e.,

$$T_{m,j}=\bigcup_{k\in\mathscr{I}(m,j)}T_{m-1,k},$$

where $\mathscr{I}(m,j)$ is the set of the indices of cells in $\mathscr{T}_{m-1}$ of which $T_{m,j}$ is composed.

We reformulated (2.6) into matrix form, added a subscript 0 to the variables to denoted the mesh level and removed the superscript (n) for simplicity as

$$\sum_k\mathbf{A}_{0,jk}\delta\mathbf{U}_{0,j}=-\mathbf{R}_{0,j}, \tag{2.7}$$

where

$$\mathbf{A}_{0,jj} = \alpha \left\| \mathbf{R}_j^{(n)} \right\|_{l^1} + \sum_{e_{jk} \in \partial T_j} \int_{e_{jk}} \frac{\partial \bar{\mathbf{F}}^{(n)}}{\partial \mathbf{U}_j} \cdot \mathbf{n}_{jk} dl,$$
$$\mathbf{A}_{0,jk} = \sum_{e_{jk} \in \partial T_j} \int_{e_{jk}} \frac{\partial \bar{\mathbf{F}}^{(n)}}{\partial \mathbf{U}_k} \cdot \mathbf{n}_{jk} dl, \qquad j \neq k,$$
$$\mathbf{R}_{0,j} = \mathbf{R}_j^{(n)}, \qquad \delta \mathbf{U}_{0,j} = \delta \mathbf{U}_j^{(n)}.$$

Then the projected linear system on mesh level m from level $m-1$ is as

$$\sum_k \mathbf{A}_{m,jk} \delta \mathbf{U}_{m,j} = -\mathbf{R}_{m,j}, \tag{2.8}$$

where

$$\mathbf{A}_{m,jk} = \sum_{\xi \in \mathscr{I}(m,j)} \sum_{\eta \in \mathscr{I}(m,k)} \mathbf{A}_{m-1,\xi\eta},$$
$$\mathbf{R}_{m,j} = \sum_{k \in \mathscr{I}(m,j)} \left(\mathbf{R}_{m-1,k} + \sum_{\xi} \mathbf{A}_{m-1,k\xi} \delta \mathbf{U}_{m-1,\xi} \right).$$

The corrections obtained from the mesh level m, $\delta \mathbf{U}_{m,j}$ are added back to the solution on the mesh level $m-1$ as

$$\delta \mathbf{U}_{m-1,k} \leftarrow \delta \mathbf{U}_{m-1,k} + \delta \mathbf{U}_{m,j}, \qquad \text{if } k \in \mathscr{I}(m,j).$$

The smoother adopted for our multigrid method is the block LU-SGS iteration, which can be formulated into two symmetric loops as

1. for $T_{m,j} \in \mathscr{T}_m$, loop for j increasingly

$$\delta \mathbf{U}_{m,j} \leftarrow \mathbf{A}_{m,jj}^{-1} \left(-\mathbf{R}_{m,j} - \sum_{k \neq j} \mathbf{A}_{m,jk} \delta \mathbf{U}_{m,k} \right);$$

2. for $T_{m,j} \in \mathscr{T}_m$, loop for j decreasingly

$$\delta \mathbf{U}_{m,j} \leftarrow \mathbf{A}_{m,jj}^{-1} \left(-\mathbf{R}_{m,j} - \sum_{k \neq j} \mathbf{A}_{m,jk} \delta \mathbf{U}_{m,k} \right).$$

The V-cycle type iteration was adopted in the implementation of our linear multigrid solver after carrying out extensive numerical experiments for efficiency comparison. The smoother was symmetrically applied before the projection and after the coarse grid corrections. It is known the multigrid iteration here, as the inner iterations of the complete Newton-iteration in Algorithm 2.1, should be used for only a few steps in every Newton-iteration step. As the result of the balance between the inner iterations and the outer iterations, the number of multigrid iteration steps was set to be 2 or 3 in our computations.

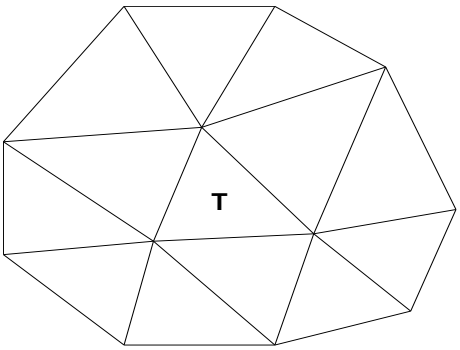

Figure 1: The cell patches $\widetilde{\omega}_T$ used for the least square reconstruction, which is composed by all the cells in the partition have at least one common vertex with T.

2.4. Reconstruction and limiter

As mentioned above, the step 3 in Algorithm 2.1 only updated the cell mean values of the solutions. In order to obtain second-order accuracy solutions, reconstruction of the numerical solutions based on the updated cell mean values into piecewise linear solutions was applied. The reconstruction techniques have been extensively considered for the solver of the stationary solution since the methods used for the time dependent problems, especially those highly diffusive reconstruction, generally destroy the convergence of the Newton-iteration. We followed the least square reconstruction method (see [21] and references therein) which was designed for the steady solution solver. The details of this method are briefly described here for completeness. Let $\mathcal{N}(T)$ denote the set of control volume T's vertices, the least square reconstruction for the gradient on cell T is taken on the cell patch

$$\widetilde{\omega}_T := \bigcup_{\mathcal{N}(T)\cap\mathcal{N}(T')\neq\emptyset} T'. \tag{2.9}$$

See Fig. 1 for the profile of $\widetilde{\omega}_T$. Let $P(\mathbf{x})$ denote the linear function will be reconstructed on $\widetilde{\omega}_T$, $\mathbf{Q} = \nabla_{\mathbf{x}} P$ and P_0 is the cell mean value of $P(\mathbf{x})$ on T. For triangle cell, the value of a linear function at the barycenter $\mathbf{x}_0$ of T is the same as the cell mean. Then, $\mathbf{Q}$ is obtained as the optimal solution of the following problem

$$\arg\ \min_{\mathbf{Q}} \sum_{\forall T_k\in\widetilde{\omega}_T} \left\| \frac{\left(P_k - P_0 - \mathbf{Q}\cdot(\mathbf{x}_k - \mathbf{x}_0)\right)}{\left|\mathbf{x}_k - \mathbf{x}_0\right|} \right\|_2^2, \tag{2.10}$$

where P_k is the mean value of $P(\mathbf{x})$ on T_k.

From the formation of the least square reconstruction, it can be seen that some numerical oscillations will be introduced, especially around the discontinuities of the solutions. The numerical oscillation should be removed from the solution, since it not only induces some non-physical data into the numerical solution, but also impedes the convergence of the iteration. To remove the numerical oscillations from the reconstructed solutions, a certain type of limiter should be applied, such as the ones used in the high order discontinuous Galerkin method for conservation laws. A limiter with moderate diffusive effects

should be chosen to take account of the convergence of the iterative procedure as well as removing the numerical oscillations. In [20], the Venkatakrishnan type limiter was proposed to reconstruct the gradients of the numerical solutions to reduce the adverse effects of the least square reconstruction. A modified version of the Venkatakrishnan limiter [21], which is robuster than the original one, was adopted in our implementation.

2.5. Some technical details

Local mesh refinement. Appearance of discontinuities in the stationary solutions is the basic character of the transonic flows. The capacity to resolve such discontinuities is one of the key criterions as effective judgement of the numerical method. Since the locations of the discontinuities are not known beforehand, it is impossible to get satisfactory numerical solutions on a single set of mesh grids. Hence mesh adaptivity techniques should be considered. The basic idea of mesh adaptivity is to add, remove, or redistribute grid points according to the numerical solutions obtained on the mesh in hand, so that the grid points were concentrated in regions with complex solution structure. The adaptive mesh method we used is the so-called h-method that the mesh is modified by local refinement and coarsening. The gradient of the pressure was used as the indicator of mesh adaptation heuristically. More precisely, on the control volume T_i, the indicator I is calculated as

$$I_i = \sum_{\forall e_{ik} \in \partial T_i} \frac{\left|p_i - p_k\right|}{\left|\mathbf{x}_i - \mathbf{x}_k\right|} |e_{ik}|, \tag{2.11}$$

where $\mathbf{x}_i$ and $\mathbf{x}_k$ are the barycenters of T_i and T_k, and p_i and p_k are the cell mean pressure on T_i and T_k, respectively. The procedure of the mesh adaptation was as

1. Solve the stationary solutions on current mesh using the solver proposed in Section 2;
2. Calculate the indicator using (2.11);
3. Adapt the mesh based on the indicator and go to 1.

Far field boundary conditions for subsonic flow. Due to the abrupt truncation of the solution domain, the far field boundary conditions are essential to the well-posedness, convergence and accuracy of the problem. For well-posedness it is necessary to impose additional boundary conditions on the artificial boundary to reduce the negative influence of the reflection of any outgoing disturbances back into the computational region [3]. Let subscripts ∞ and e denote free-stream values and values from the interior cells adjacent to the boundary, respectively. Let u_n and $c = \sqrt{\gamma p/\rho}$ be the velocity component normal to the boundary and the speed of sound. The one dimensional Riemann invariants along the out normal vector of the boundary are

$$R_1 = u_n + \frac{2c}{\gamma - 1}, \quad R_2 = u_n - \frac{2c}{\gamma - 1}, \tag{2.12}$$

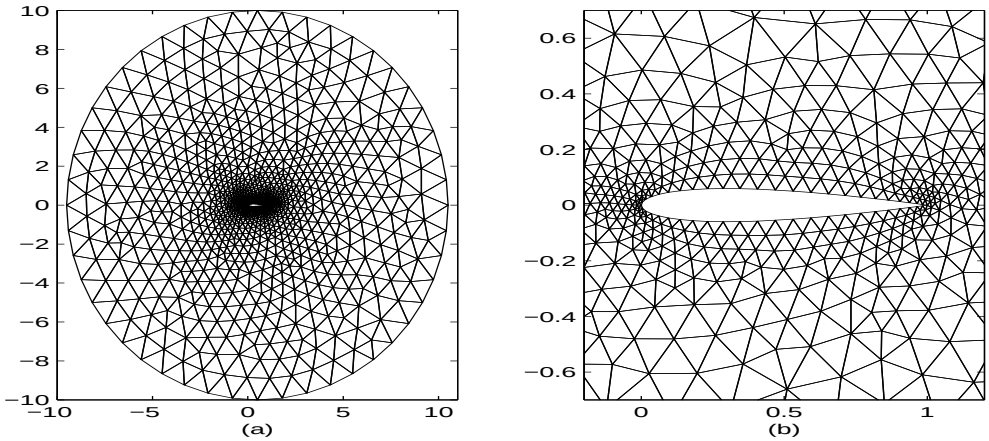

Figure 2: Initial mesh grids generated using Easymesh around NACA 0012 airfoil, containing 2402 triangular cells. By setting suitable parameters in the input file for Easymesh, the mesh generated can be finer around the airfoil body. (a) The entire grid; (b) Grid near the body of the airfoil.

which are correspond to the eigenvalues $\lambda_1 = u_n - c$ and $\lambda_2 = u_n + c$. By assuming the far field flow is subsonic ($|u_n| < c$), the far field boundary conditions can be classified into the following category:

1. At the inflow boundary $u_n < 0$,
   $\lambda_1 < 0$, $\lambda_2 > 0$, $R_1 = R_{1\infty}$, $R_2 = R_{2e}$, $S = S_\infty$, $u_t = u_{t\infty}$,
   where S is the entropy, $S = p/\rho^\gamma$ and u_t is the tangential velocity along the boundary.

2. At the outflow boundary $u_n > 0$,
   $\lambda_2 > 0$, $\lambda_1 < 0$, $R_1 = R_{1\infty}$, $R_2 = R_{2e}$, $S = S_e$, $u_t = u_{te}$.

Far field vortex correction. As suggested by Usab and Murman [19], the components of the free-stream conditions should be corrected after every iteration. The details of the far field vortex correction can be found in [3].

3. Numerical results

In this section, we presented a sequence of numerical examples to illustrate the robustness and efficiency of the algorithm described above. In all the numerical examples below, the parameters in the algorithms were not changed at all. More precisely, we always set $\alpha = 2$, $\tau = 1$ and the smoothing steps in the multigrid solver is 2.

We first tested our algorithm on quasi-uniform meshes, and then on local adaptive meshes. It was interesting to see that for three free-stream conditions with essential differences, our algorithm converged without difficulties. At last, the numerical results on different geometric configurations were presented.

All the computations were carried out on a Pentium IV desktop computer with core speed 3.0G, and Linux as its operating system. The codes were based on the adaptive finite element package AFEPack [15] developed by R. Li and W.-B. Liu.

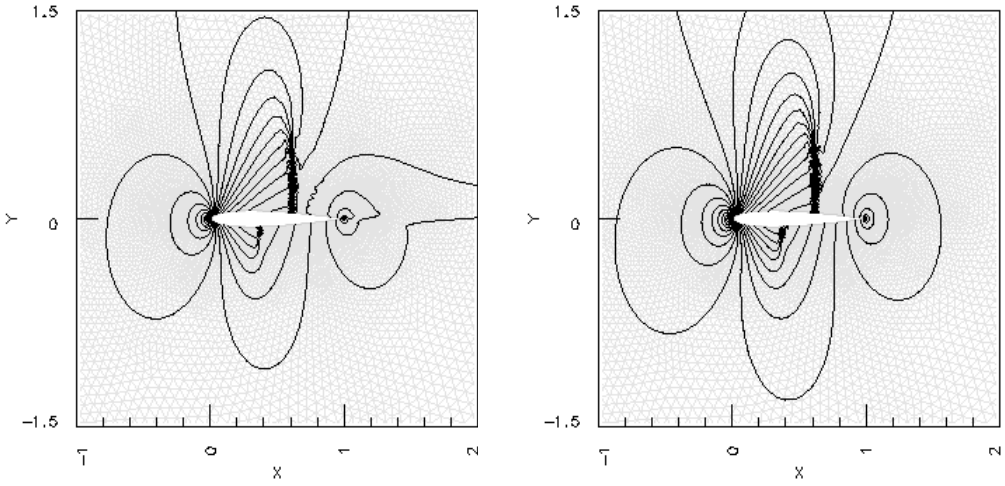

Figure 3: Mach number contours (left) and pressure contours (right) of NACA 0012 airfoil at free-stream condition with Mach number = 0.8, attack angle = 1.0° on quasi-uniform mesh.

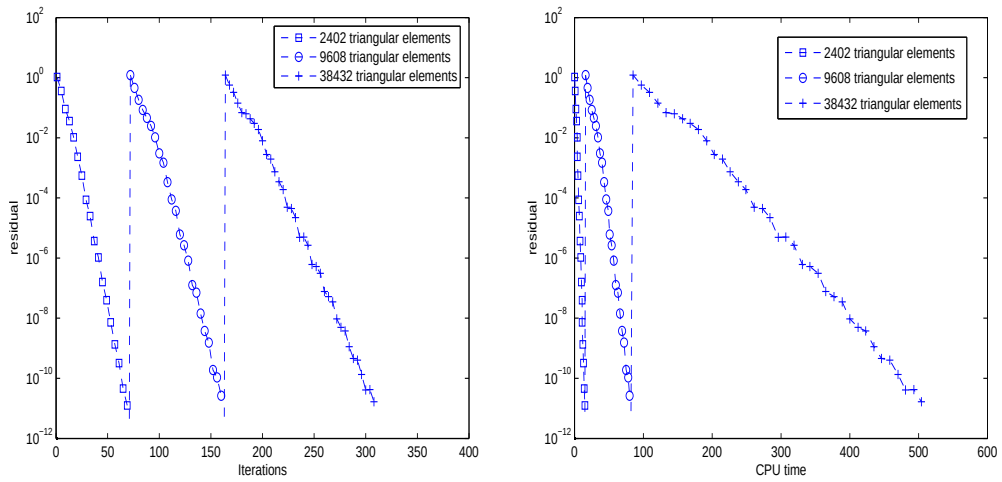

Figure 4: Convergence history in term of iterations (left) and CPU seconds (right) of NACA 0012 airfoil at free-stream condition with Mach number = 0.8, attack angle = 1.0° through two uniform mesh refinement. The solver has the similar behaviors on these meshes.

3.1. On quasi-uniform meshes

We first tested our algorithm on the quasi-uniform meshes by using the NACA 0012 airfoil as the geometric configuration. The truncation of the domain was set at the location of $R = 10 \times$ *chord length*. We carried out the computations on a uniform refined mesh series, with the free-stream Mach number 0.8 and 1.0 attach angle. Fig. 3 plots the contours of Mach number and pressure on the finest mesh used. The background mesh was plotted in Fig. 2. With the mesh generation software Easymesh [16], the initial mesh can be finer around the airfoil body. The convergence history for all three computations were plotted in Fig. 4, from which it can be seen that the behaviors of our solver on all these uniform

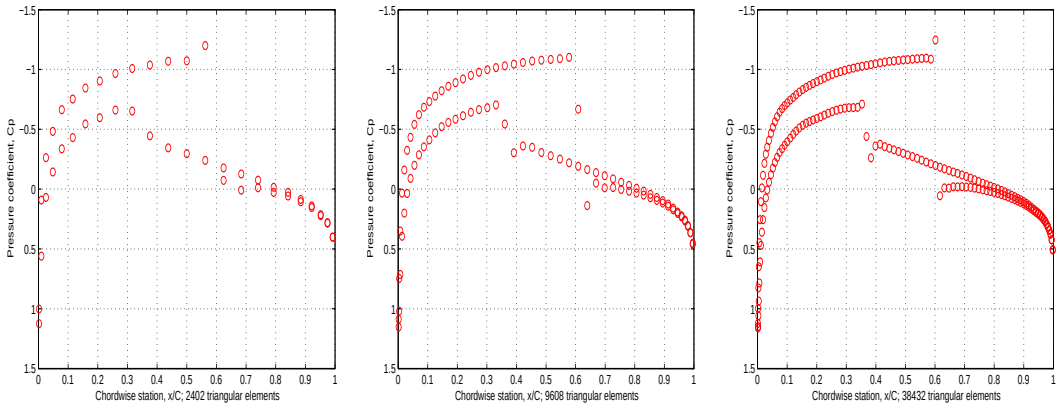

Figure 5: Pressure coefficient on the airfoil surface of NACA 0012 airfoil at free-stream condition with Mach number = 0.8, attack angle = 1.0° on three continuously uniform refined meshes based on a quasi-uniform mesh.

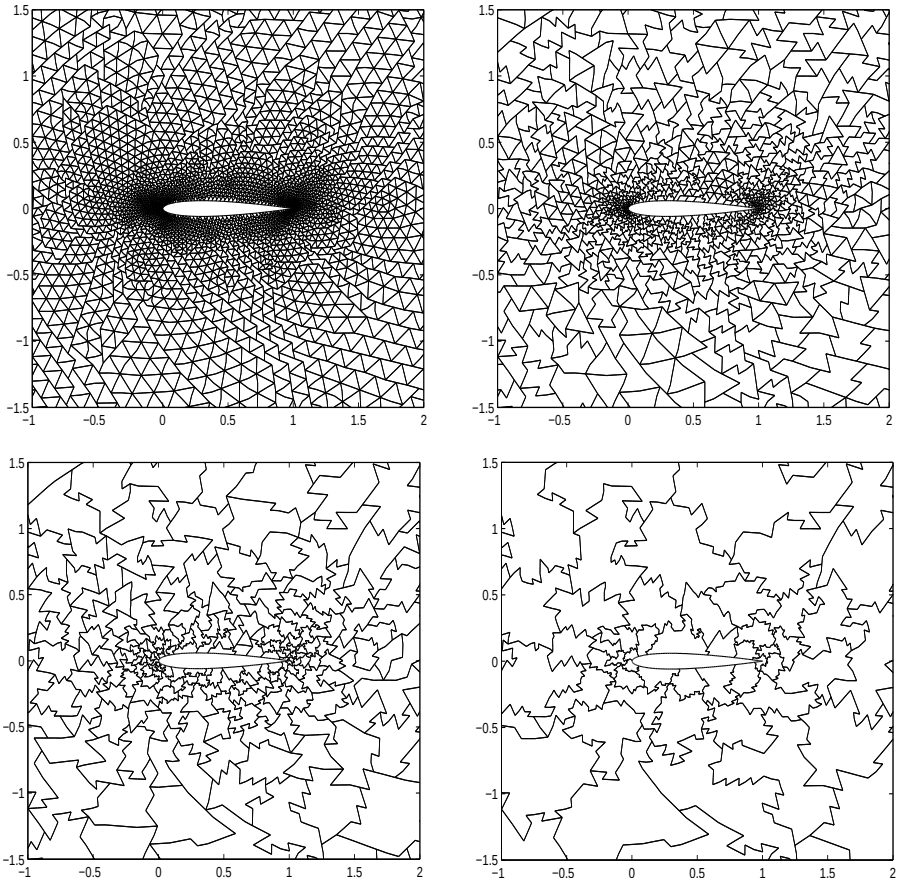

Figure 6: Element patches near airfoil body of NACA 0012 generated by our aggregation codes from a quasi-uniform mesh. The figures are the element patches generated on four continuous levels, with free-stream condition of Mach number = 0.85 and attack angle = 1.0°.

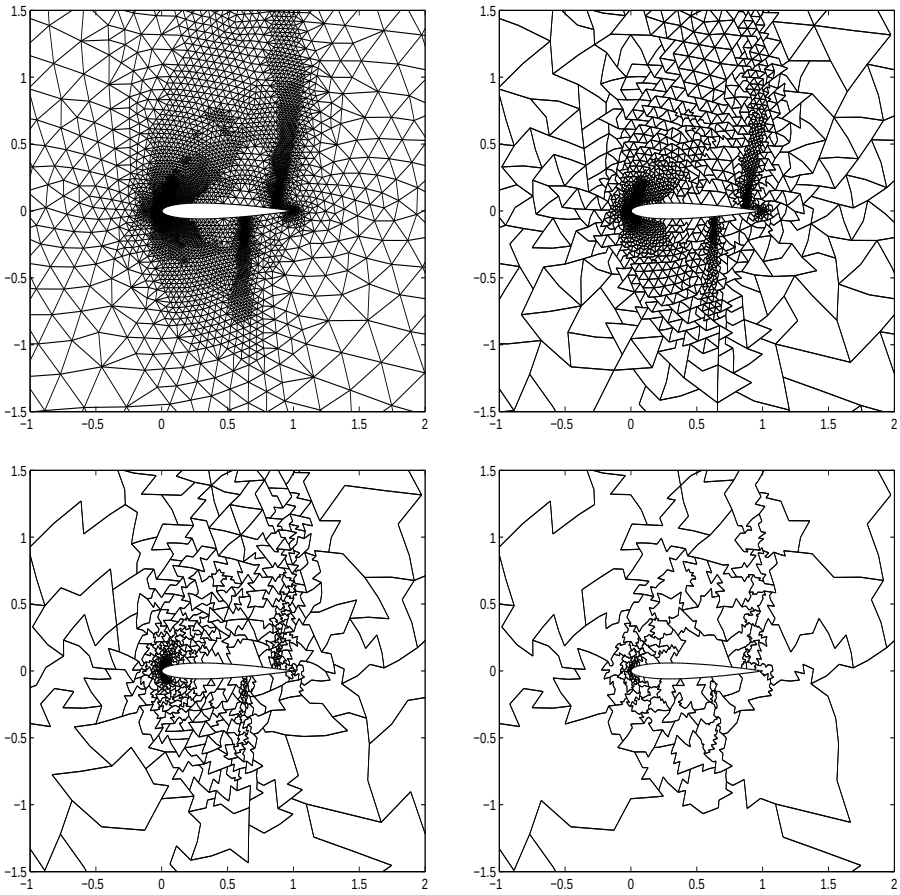

Figure 7: Same as Fig. 6, except from the local adaptive mesh obtained.

refined meshes are similar. Pressure coefficients obtained were plotted in Fig. 5 as an evidence of the numerical convergence of our method.

In Fig. 6, we plotted the coarse meshes generated sequentially by our implementation of aggregation codes. It can be seen that our codes work well on the quasi-uniform meshes.

3.2. On local adaptive meshes

The following examples are the numerical tests of our algorithm on local adaptive meshes. The mesh adaptation method used was the one implemented in AFEPack. After the adaptation indicator is calculated, AFEPack can provide the local adaptive meshes automatically. We set a different free-stream condition from the previous subsection to obtain more extensive numerical results. The geometric configuration was NACA 0012 again, while the free-stream Mach number now was 0.85 with attach angle as 1.0.

In Fig. 8, we plotted the contours of Mach number (left) and pressure (right) on each of the local adaptive meshes, which were obtained by two times of local adaptive mesh refinement. The mesh sequence generated by our mesh aggregation codes were plotted in

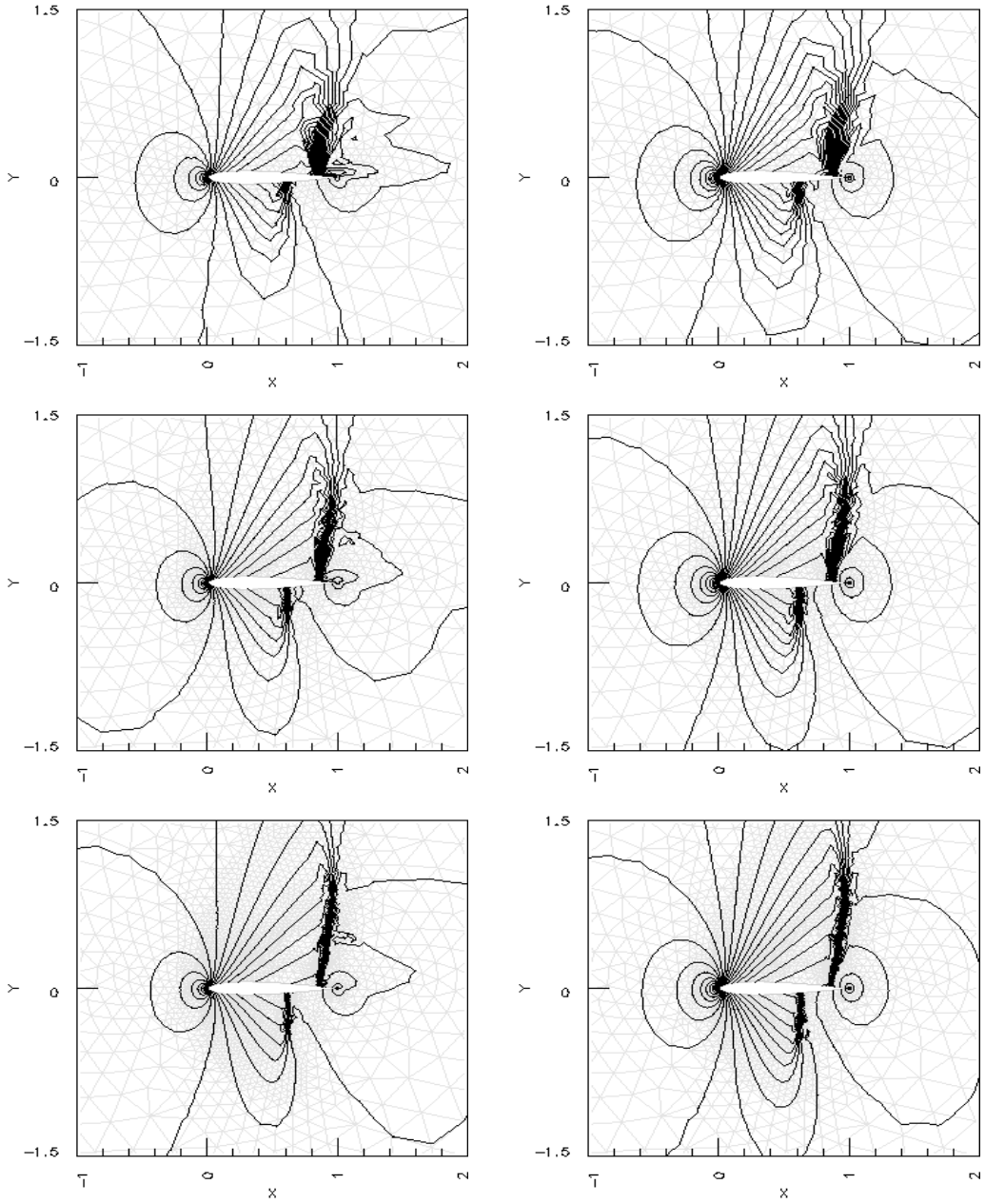

Figure 8: The contours of Mach number (left column) and the pressure (right column) on three continuous local adaptive meshes, with the free-stream condition with Mach number = 0.85, attack angle = 1.0°. The background of the figures were the obtained local adaptive meshes.

Fig. 7. The coarse meshes generated here had the similar quality as the ones generated from the quasi-uniform mesh in Fig. 6.

It can be observed from Fig. 7 that mesh grids concentrated near the region where the shocks appear, thus the shocks in the solutions were better resolved on the local adaptive

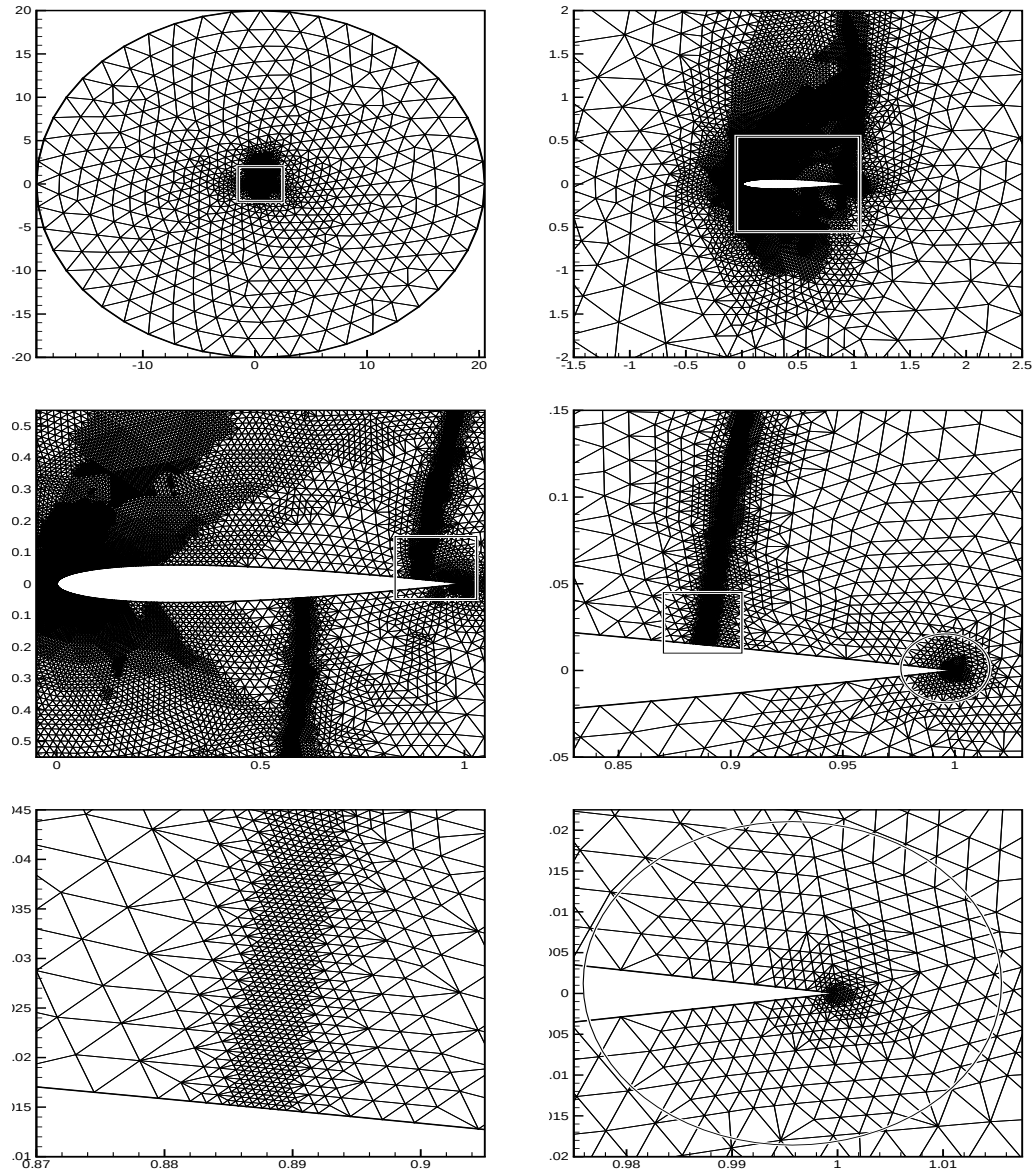

Figure 9: Local structure details of the adaptive meshes, obtained in a computation for NACA 0012 airfoil with the free-stream condition with Mach number = 0.85, attack angle = 1.0°. The top left figure is the entire mesh. In the first and the second rows, the marked part of the previous figure is zoomed in and shown in the next figure sequentially. The figures in the last row are two different parts marked in the middle right figure near the upper shock (marked in a rectangle) and the end point of the airfoil (marked in a circle).

mesh. Higher quality shocks were captured by our numerical solutions on the mesh with more steps of local adaptive refinement as in Fig. 8. In Fig. 9, we plotted a sequentially zoomed local mesh structure to show the flexibility and the ability of the adaptive mesh in resolving local discontinuities. Finally, in Fig. 10, the convergence history on continuous

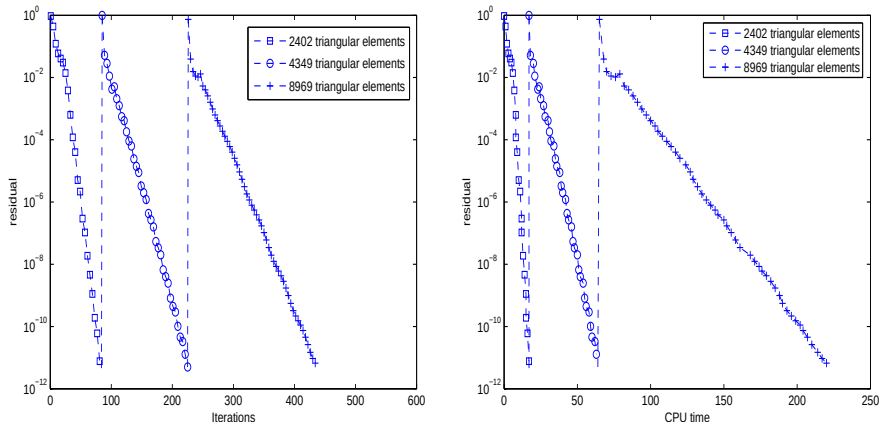

Figure 10: The convergence history in term of iterations (left) and CPU seconds (right) of the NACA 0012 airfoil computation on three continuous local adaptive meshes, with the free-stream condition with Mach number = 0.85, attack angle = 1.0°.

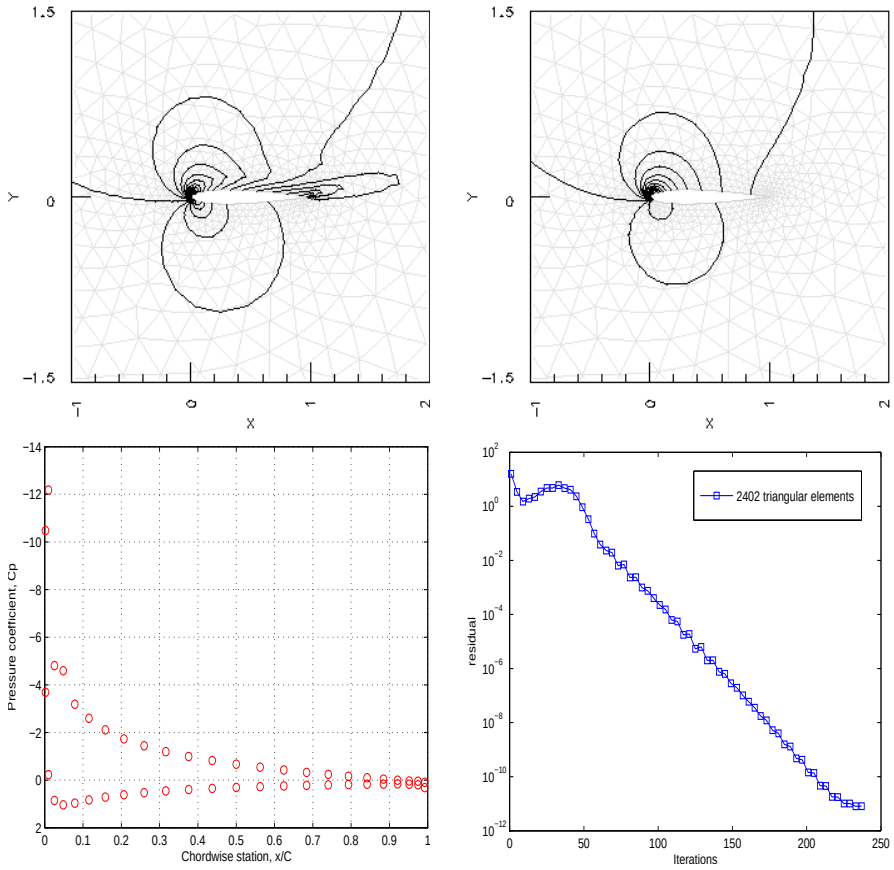

Figure 11: The contours of Mach number (top left) and pressure (top right), and pressure coefficient (bottom left) and the convergence history (bottom right) for the NACA 0012 airfoil with free-stream condition with Mach number = 0.3 and attack angle = 15°.

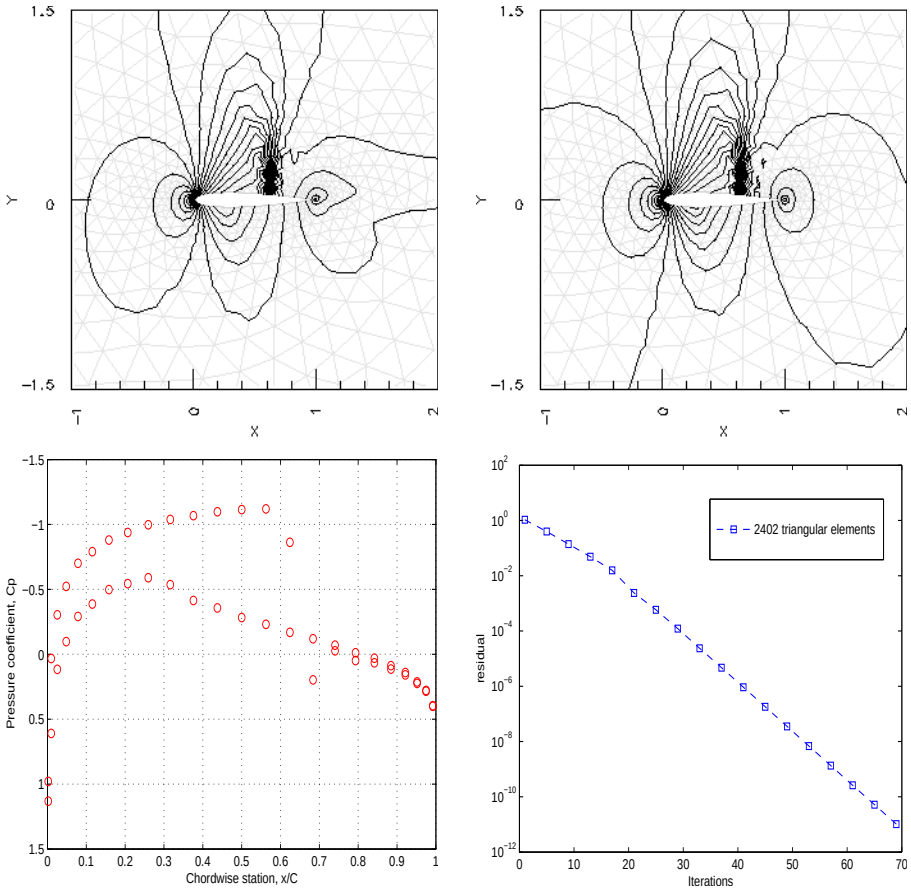

Figure 12: Same as Fig. 11, except with free-stream condition with Mach number = 0.8 and attack angle = 1.25°.

local adaptive meshes were plotted. It can be seen that the algorithm behaved quite stable on all these meshes.

3.3. On different free-stream configurations

In this subsection, our solver was tested on three very different free-stream configurations:

- Low free-stream Mach number as 0.3 and a big attack angle as 15°;
- Moderate free-stream Mach number as 0.8 and attack angle as 1.25°;
- High free-stream Mach number as 0.99 and attack angle as 0.2°.

The contours of Mach number and pressure of the numerical solutions, the pressure coefficient and the convergence history in terms of iteration steps were plotted in Figs. 11-13, respectively. From the convergence history for these three free-stream configurations, it can be seen that the first case was the most difficult one to converge. But for all three cases, our algorithm got converged results without changing the parameters involved.

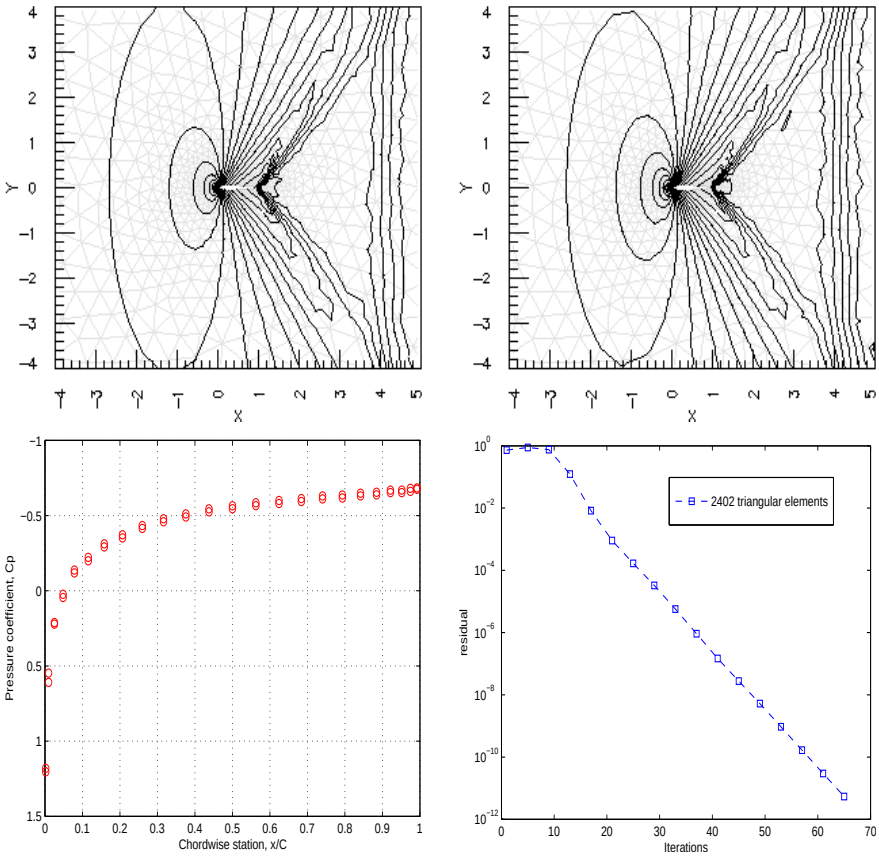

Figure 13: Same as Fig. 11, except with free-stream condition with Mach number = 0.99 and attack angle = 0.2°.

3.4. On different geometric configurations

Other than the computations on the NACA 0012 airfoil, we tested in this subsection the geometric configurations on the RAE 2822 airfoil and a combined configuration using two NACA 0012 airfoils.

For the RAE 2822 case, the free-stream condition is Mach number 0.75 and attack angle 3°. The contours of Mach number and pressure of the numerical solutions, the pressure coefficient and the convergence history in term of iteration steps were plotted in Fig. 14. It can be seen that our algorithm works well for this example.

For the geometric configuration using two NACA 0012 airfoils, we set the truncation radius $R = 20 \times$ *chord length* since for this case the support of the two airfoils occupied about double size of the domain occupied by the one airfoil case. This computation was carried out on local refined meshes to check the robustness of our algorithm. We plotted the contours of Mach number and pressure of the numerical solutions, the pressure coefficient and the convergence history in terms of iteration steps in Fig. 15. Though the flow field was more complex than that of the one airfoil case, our algorithm can again solve this example efficiently.

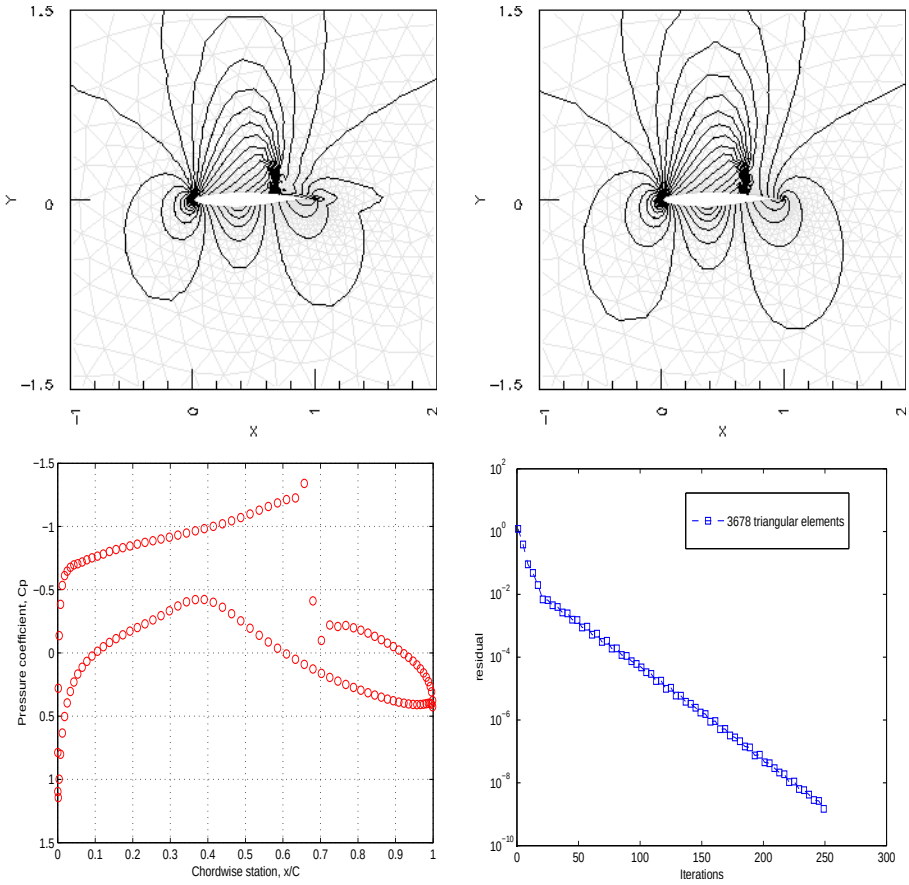

Figure 14: The contours of Mach number (top left) and pressure (top right), and pressure coefficient (bottom left) and the convergence history (bottom right) for the RAE 2822 airfoil with free-stream condition with Mach number=0.75, attack angle=1.0°.

4. Concluding remarks

In this paper, we devised an effective and robust algorithm for steady Euler equations on unstructured grids, which is a Newton-iteration method using a multigrid algorithm to solve the linearized Jacobian matrix. We developed a local residual dependent regularization for the linearized Jacobian matrix. The block LU-SGS iteration was adopted as the smoother of the multigrid algorithm. The numerical examples demonstrated the efficiency and the robustness of our algorithm.

The algorithm proposed above is now being extended for the second-order reconstruction and for the three spatial dimension problems. Some preliminary numerical experiments indicate that the algorithm can work quite well for both cases. Similar convergence history can be obtained as shown in this paper. Those results will be reported elsewhere.

Taken the developing of the algorithm as a beginning step, our long-range research object is on the aerodynamics design. The aerodynamics design can be studied using the optimal control theory. Currently, we have deduced the discretized optimal condition for

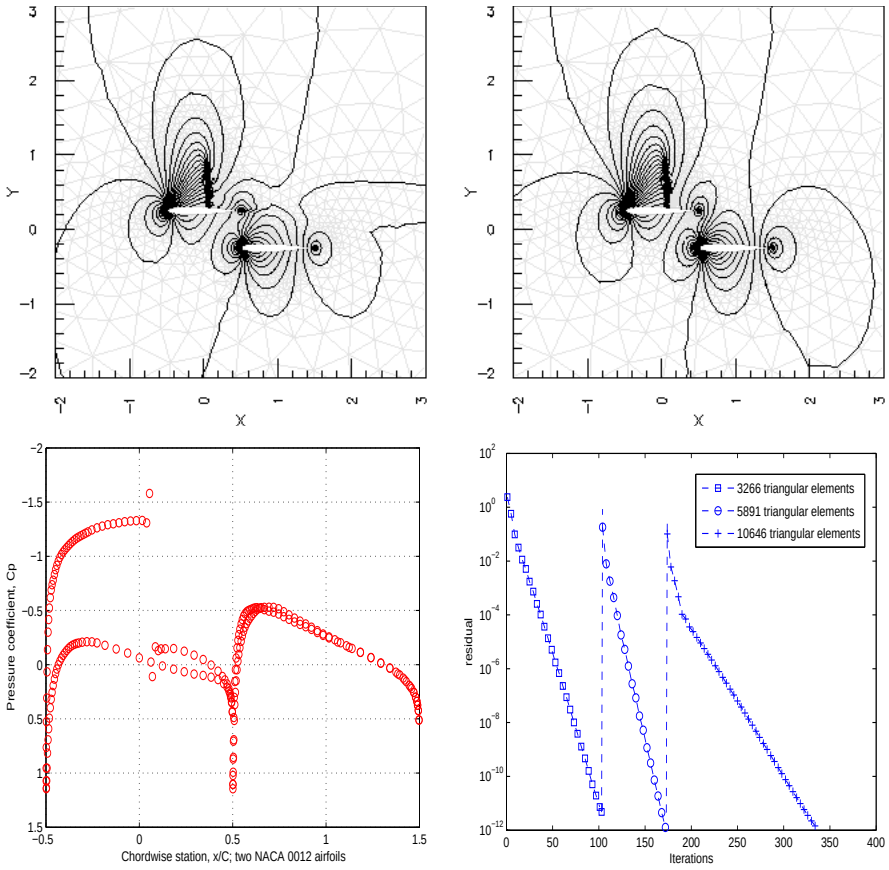

Figure 15: Same as Fig. 14, except for two closely set NACA0012 airfoils.

the finite volume discretization adopted in this paper. The solver of the optimal control problem is under implementation.

Acknowledgments The research of Li was subsidized by the National Basic Research Program of China under the grant 2005CB321701. During the preparation of this paper, partial support from Beijing University of Aeronautics and Astronautics went to Wang and Li. We are grateful to Prof Zhijian Wang for his professional advices and for sharing some of his codes with us. This research was pushed by Prof Tao Tang and Prof Weinan E with the expectation of some contributions to this very important application area in China.

References

[1] P. Batten, M. A. Leschziner, and U. C. Goldberg, *Average-state Jacobians and implicit methods for compressible viscous and turbulent flows*, J. Comput. Phys., 137 (1997), pp. 38–78.

[2] J. Blazek, *Investigation of the implicit LU-SSOR scheme*, DLR Research Report, 93-51, 1993.

[3] J. Blazek, *Computational Fluid Dynamics: Principles and Applications*, Elsevier Science Publication, 2005.

[4] G. V. CANDLER, M. J. WRIGHT, AND J. D. MCDONALD, *A data-parallel LU-SGS method for reacting flows*, AIAA, Aerospace Sciences Meeting and Exhibit, 10-13, January 1994.
[5] R. F. CHEN AND Z. J. WANG, *Fast block lower-upper symmetric Gauss-Seidel scheme for arbitrary grids*, AIAA J., 38(12) (2000), pp. 2238–2245.
[6] A. JAMESON, *Iterative solution of transonic over airfoil and wings including at mach 1*, Commun. Pure Appl. Math., 27 (1974), pp. 283–309.
[7] A. JAMESON, *Solution of the Euler equations for two dimensional transonic flow by a multigrid method*, Appl. Math. Comput., 13 (1983), pp. 327–356.
[8] A. JAMESON, *Multigrid Algorithms for Compressible Flow Calculations*, volume 1228 of Lecture Notes in Mathematics, Springer Verlag, 1985.
[9] A. JAMESON, *Analysis and design of numerical schemes for gas dynamics 1: Artificial diffusion,upwind biasing, limiters and their effect on accuracy and multigrid convergence*, Int. J. Comput. Fluid Dyn., 4 (1994), pp. 171–218.
[10] A. JAMESON AND T. J. BAKER, *Solution of the Euler equations for complex configurations*, AIAA Paper, 83-1929, 1983.
[11] A. JAMESON, W. SCHMIDT, AND E. TURKEL, *Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes*, AIAA Paper, 81-1259, January 1981.
[12] A. JAMESON AND E. TURKEL, *Implicit scheme and LU-decompositions*, Math. Comput., 37 (1981), pp. 385–397.
[13] B. VAN LEER, W. T. LEE, AND L. ROE, *Characteristic time-stepping or local preconditioning of the Euler equations*, AIAA Comput. Fluid Dyn. Conf., 1991.
[14] B. VAN LEER AND P. L. ROE, *Local preconditioning of the Euler equations and its numerical applications*, Algorithmic Trends in Computational Fluid Dynamics, 1993.
[15] R. LI AND W. B. LIU, `http://circus.math.pku.edu.cn/AFEPack`.
[16] B. NICENO, `http://www-dinma.univ.trieste.it/nirftc/research/easymesh/`.
[17] H. RIEGER AND A. JAMESON, *Solution of steady 3-D compressible Euler and Navier-Stokes equations by an implicit LU scheme*, AIAA Paper, 88-0619, 1988.
[18] R. F. TOMARO, W. Z. STRANG, AND L. N. SANKAR, *An implicit algorithm for solving time dependent flows on unstructured grids*, AIAA Paper, 97-0333, 1997.
[19] W. J. USAB AND E. M. MURMAN, *Embedded mesh solution of the Euler equation using a multiple-grid method*, AIAA Paper, 83-1946, 1983.
[20] V. VENKATAKRISHNAN, *Convergence to steady state solutions of the Euler equations on unstructured grids with limiters*, J. Comput. Phys., 118 (1995), pp. 120–130.
[21] Z. J. WANG, *A fast nested multi-grid viscous flow solver for adaptive Cartesian/Quad grids*, J. Numer. Mech. Fluids, 33 (2000), pp. 657–680.
[22] S. YOON AND A. JAMESON, *LU implicit schemes with multiple grids for the Euler equations*, AIAA J., 25(7) (1986), pp. 929–935.
[23] S. YOON AND A. JAMESON, *A multigrid LU-SSOR scheme for approximate Newton-iteration applied to Euler equations*, NASA-CR-19754, 1986.
[24] S. YOON AND A. JAMESON, *Lower-upper symmetric-Gauss-Seidel method for the Euler and Navier-Stokes equations*, AIAA J., 26 (1988), pp. 1025–1026.
[25] S. YOON AND D. KWAK, *Multigrid convergence of an implicit symmetric relaxation scheme*, AIAA Paper, 93-3357, 1993.