INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids **31**: 1121–1139 (1999)

AN ADAPTIVE LEAST-SQUARES METHOD FOR THE COMPRESSIBLE EULER EQUATIONS

F. TAGHADDOSI, W.G. HABASHI*, G. GUÈVREMONT AND D. AIT-ALI-YAHIA

CFD Laboratory, Department of Mechanical Engineering, Concordia University, 1455 de Maisonneuve Blvd. W., ER 301, Montréal, Québec, Canada, H3G 1M8

SUMMARY

An adaptive least-squares finite element method is used to solve the compressible Euler equations in two dimensions. Since the method is naturally diffusive, no explicit artificial viscosity is added to the formulation. The inherent artificial viscosity, however, is usually large and hence does not allow sharp resolution of discontinuities unless extremely fine grids are used. To remedy this, while retaining the advantages of the least-squares method, a moving-node grid adaptation technique is used. The outstanding feature of the adaptive method is its sensitivity to directional features like shock waves, leading to the automatic construction of adapted grids where the element edge(s) are strongly aligned with such flow phenomena. Using well-known transonic and supersonic test cases, it has been demonstrated that by coupling the least-squares method with a robust adaptive method shocks can be captured with high resolution despite using relatively coarse grids. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: least-squares method; artificial viscosity; Euler equations; grid adaption

1. INTRODUCTION

The efficient solution of the compressible Euler equations has been a focus of attention of computational fluid dynamics (CFD) researchers for more than a decade. This is due to the fact that in many problems of practical interest (especially high Reynolds number external flows) the flow field away from the solid boundaries is essentially inviscid, and therefore can be modeled by these equations. Moreover, it is a well-known fact among the CFD users that developing a robust Euler flow solver is the preliminary step in building an efficient Navier–Stokes solver.

So far, many different approaches have been adopted in developing numerical schemes to solve the compressible Euler equations. Using the ideas of upwind schemes in the finite difference method, Brooks and Hughes [6] introduced the Streamline Upwind Petrov–Galerkin (SUPG) method, in which the weight function is modified by adding a perturbation to the standard Galerkin test function. The added perturbation creates an upwind effect by weighting the upstream nodes within the elements more heavily than the downstream nodes. Hughes and Tezduyar [13] generalized the SUPG method to first-order hyperbolic systems. The shock capturing ability of the method was later improved by adding non-linear operators to the perturbation [5,14].

* Correspondence to: Department of Mechanical Engineering, Concordia University, 1455 de Maisonneuve Blvd. West, ER 301, Montréal, Québec, Canada, H3G 1M8.

CCC 0271–2091/99/231121–19$17.50

Received 10 September 1996
Revised 20 March 1998

Another approach was proposed by Baruzzi *et al.* [3], where the Laplacian of dependent variables was added to the continuity and momentum equations. The amount of artificial viscosity was then controlled by a single parameter as the coefficient of the Laplacians. They later extended this first-order artificial viscosity method to second-order [4].

Löhner *et al.* [20] used a method based on writing the non-symmetric first-order differential operators along characteristics to achieve a self-adjoint form, and then applied the Galerkin method for the solution of a system of hyperbolic equations. Oden *et al.* [25] used the Taylor–Galerkin method for the solution of the Euler equations in the supersonic regime, along with the flux-corrected-transport approach to avoid non-physical oscillations in the solution.

Another method is based on the least-squares weighted residual method. The method has very good stability properties due to its minimization nature, and has been applied for the solution of a variety of problems [17,19,29]. As one of the earliest efforts in this field, one can mention the technique presented by Polk and Lynn [27] for the solution of unsteady gas dynamic equations, with elements that are constructed in both space and time. Another space–time finite element scheme was presented by Nguyen and Reynen [23], and was applied to the solution of convection-dominated problems in one and two dimensions. They showed that by extending the least-squares formulation to the time domain, highly accurate and stable results can be obtained up to very large grid Peclet numbers.

Fletcher [11] used the least-squares method to solve the Euler equations for subcritical compressible flows. The special feature of his method was to represent groups of variables rather than single variables. Bruneau *et al.* [7] used a rather similar method to study the vortical phenomena created by the subsonic and supersonic flow over a flat plate at different angles of attack.

Application of the least-squares method to a governing equation of the general form: $\mathscr{L}(\phi) = f$ leads to the favorable result of a symmetric and positive definite coefficient matrix, if $\mathscr{L}$ is a first-order differential operator. If $\mathscr{L}$ is a higher-order operator, however, this property is completely lost during the integration-by-parts, and moreover, elements with higher-order continuity requirements, e.g. C^1 must be employed. Lynn and Arya [21] proposed to break down the higher-order system to its first-order counterpart as a way of eliminating this disadvantage. This idea was used by Carey and Jiang [8] to formulate an algorithm for the general system of first-order partial differential equations (PDEs) using the least-squares finite element method. An analysis of the method for the wave equation was performed by Carey and Jiang [9]. Jiang and Carey [16] also used the least-squares method for the solution of 2D compressible Euler equations. Lefebvre *et al.* [19] applied a similar least-squares method for the compressible Euler equations. They also adopted an adaptive refinement strategy based on the least-squares residual as the error estimator, in a similar fashion to that proposed by Jiang and Carey [15].

The least-squares finite element method is easy to implement, is naturally diffusive, contains no free parameter(s), is stable thus allowing equal-order interpolation of all variables, and more importantly, results in a symmetric and positive definite system of algebraic equations. This latter property allows only half the matrix to be stored, thus reducing the memory requirements by nearly 50%; a very desirable advantage, especially for large-scale problems where storage limitations are a major concern. Also, the resulting system can be solved very efficiently by the conjugate gradient (CG) iterative method. Application of other finite element methods, such as the Galerkin [3] or the SUPG [6] lead to non-symmetric matrices, where the efficient iterative methods used for their solution are variants of the optimal CG method, attempting to imitate its properties [2]. Nevertheless, they are not as robust as the CG method in terms of convergence and stability properties, storage requirements, and CPU time.

The artificial viscosity produced by the least-squares method does not allow discontinuities such as shock waves to be sharply resolved, unless an impractically fine grid is used. This is one of the main reasons why very little work has been done so far in applying the least-squares method to the Euler equations, despite its various and obvious advantages [16,19]. To alleviate this problem to a very large extent, the least-squares method is combined with a directionally-adaptive method, which uses an edge-based error estimate on quadrilateral grids. The error of the numerical solution is estimated through its second derivatives and the resulting Hessian tensor is used to define a Riemannian metric. An improved mesh movement strategy, based on a spring analogy with no orthogonality constraints, is introduced to equidistribute the lengths of the edges of the elements in the defined metric. This leads to a simple and efficient nodal redistribution algorithm, offering a greater range of grid-point displacement.

The present work, therefore, is an attempt to investigate the least-squares finite element method for compressible flows in the transonic and supersonic regimes with shocks, and improve its performance and accuracy via an adaptive grid method, and a preconditioned CG iterative solver.

2. LEAST-SQUARES FORMULATION

The Euler equations can be written as a first-order system in terms of primitive variables as

$$\mathscr{A}_t \frac{\partial \mathbf{Q}}{\partial t} + \mathscr{A}_x \frac{\partial \mathbf{Q}}{\partial x} + \mathscr{A}_y \frac{\partial \mathbf{Q}}{\partial y} = 0, \tag{1}$$

where $\mathbf{Q} = (\rho, u, v, p)^T$ is the vector of primitive variables and ρ is the density, (u, v) the Cartesian velocity components and p the pressure. Here, $\mathscr{A}_t = \mathbf{I}$, where $\mathbf{I}$ denotes the identity matrix and, $\mathscr{A}_x$ and $\mathscr{A}_y$ are the Jacobian matrices.

By discretizing the unsteady term using backward differences

$$\frac{\partial \mathbf{Q}}{\partial t} \approx \frac{\mathbf{Q}^{n+1} - \mathbf{Q}^n}{\Delta t}, \tag{2}$$

and linearizing the system (1) using the Newton method, equation (1) can be written as

$$\mathscr{L} \Delta \mathbf{Q}^{n+1} = -\mathbf{f}, \tag{3}$$

with

$$\mathscr{L} = \mathscr{A}_x^n \frac{\partial}{\partial x} + \mathscr{A}_y^n \frac{\partial}{\partial y} + \left(\frac{\mathscr{A}_t}{\Delta t} + \mathscr{A}^n \right); \qquad \mathbf{f} = \left(\mathscr{A}_x^n \frac{\partial \mathbf{Q}^n}{\partial x} + \mathscr{A}_y^n \frac{\partial \mathbf{Q}^n}{\partial y} \right).$$

Defining the residual vector as $\mathscr{R} = \mathscr{L} \Delta \mathbf{Q}^{n+1} + \mathbf{f}$, the minimization of the least-squares functional

$$\mathbf{I}(\mathbf{Q}^{n+1}) = \frac{1}{2} \int_\Omega \mathscr{R}^T \mathscr{R} \, \mathrm{d}\Omega, \tag{4}$$

yields the following weak form

$$\int_\Omega (\mathscr{L}\mathbf{W})^T (\mathscr{L} \Delta \mathbf{Q}^{n+1} + \mathbf{f}) \mathrm{d}\Omega = \mathbf{0}, \tag{5}$$

where $\mathbf{W}$ is the weight function. Since the least-squares method is a minimization problem and is therefore not subject to the LBB condition [17], equal-order interpolation can be used for all variables. Introducing the finite element approximation

$$\mathbf{Q}^{n+1} \approx \mathbf{Q}_h^{n+1} = \sum_{j=1}^{n_e} \mathbf{N}_j \mathbf{Q}_j^{n+1}, \tag{6}$$

where n_e is the number of nodes per element and $\mathbf{N}_i = N_i \mathbf{I}$ is the element shape function, and substituting it into the weak form results in the linear algebraic equations

$$[\mathbf{K}]\{\Delta\mathbf{Q}\} = -\{\mathbf{R}\}, \tag{7}$$

where $\Delta\mathbf{Q} = (\Delta\rho, \Delta u, \Delta v, \Delta p)^T$ is the global vector of unknowns. The global coefficient matrix $[\mathbf{K}]$ and the right-hand-side vector $\{\mathbf{R}\}$ are obtained by summing up the element matrices and vectors respectively

$$\mathbf{K}_{ij}^e = \int_{\Omega^e} (\mathscr{L}\mathbf{N}_i)^T (\mathscr{L}\mathbf{N}_j) \mathrm{d}\Omega; \qquad \mathbf{r}_i^e = \int_{\Omega^e} (\mathscr{L}\mathbf{N}_i)^T \mathbf{f}\, \mathrm{d}\Omega. \tag{8}$$

The integrals are evaluated using Gauss–Legendre quadrature.

The global coefficient matrix $[\mathbf{K}]$ has the important property of being symmetric and positive definite. The implications are two-fold: first, only half of the matrix is calculated and stored, resulting in savings in the calculation time and memory storage requirements; second, very simple iterative solvers such as the classical CG algorithm, with preconditioning, can be used for the solution of the linear system.

For the Euler equations (3), the numerical viscosity is inherent in the least-squares formulation and is proportional to the time step as demonstrated in Reference [9].

2.1. Boundary conditions

For the Euler equations, the number of boundary conditions to be imposed on the domain is determined by the theory of characteristics. In the finite element method, the numerical boundary conditions are naturally imposed through the finite element shape functions, and therefore no special treatments are required.

The flow-tangency boundary condition at solid boundaries can be imposed in one of two ways:

1. *Co-ordinate rotation*. The velocity components at solid wall nodes are rotated from the global co-ordinate system to the local co-ordinate frame, where the flow-tangency condition is easily imposed by setting the normal velocity component to zero ([28], Chapter 4). Since, due to discrete representation of the computational domain, the angle at a typical wall node A (Figure 1) can have two different values, it is necessary to assign a unique angle for such nodes. To accomplish this, a length-weighted average of the angles from the two adjacent wall edges can be used

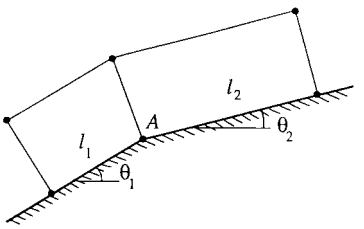

Figure 1. Discontinuous angle at wall nodes.

$$\theta_A = \frac{l_1\theta_2 + l_2\theta_1}{l_1 + l_2}. \tag{9}$$

The angles θ_1 and θ_2 are multiplied by l_2 and l_1 respectively, since the element with the shorter edge will approximate $\vec{v}_A$ better than the element with the longer edge.

2. *Penalty method.* The flow-tangency boundary condition at a solid wall can be imposed by adding the penalty term [19]

$$\mathbf{I}_w(\mathbf{Q}^{n+1}) = \frac{\alpha}{2}\int_{\Gamma_w} (\vec{v}\cdot\vec{n})^2 \, d\Gamma, \tag{10}$$

to the least-squares functional (4). Here, Γ_w, is the solid wall boundary, $\vec{v}$ is the velocity vector, $\vec{n}$ is the unit normal vector pointing outwards from the domain, and α is the penalty weight.

3. GRID ADAPTATION

The accuracy of the results obtained on a uniform grid can be significantly improved by adapting the grid to the solution. This requires first to estimate the error of a given flow variable on the coarse grid, and then the use of an adaptive strategy to equidistribute the error over the entire domain.

The employed adaptive method is based on the work of Ait-Ali-Yahia *et al.* [1], and uses a moving-node or *r*-method as the adaptive strategy. It is a very recent and important development which significantly improves the grid efficiency compared with traditional isotropic methods. It uses second derivatives of a given flow variable as the error estimator, compared with first derivatives used by most other adaptive methods, hence providing a more meaningful measure for the approximation error.

The sensitivity of the error estimator to directional flow phenomena, such as shock waves and boundary layers, is achieved by calculating the error on the element edges rather than the commonly used approaches based on nodes or on elements. When combined with the moving-node scheme, it leads to a dramatic re-alignment of the element edges with such directional structures. As a result, the shocks will be very thin and very sharply captured.

3.1. Edge-based error estimate

For the 1D element shown in Figure 2, where the scalar variable σ is assumed to vary linearly, the interpolation error can be estimated as the difference between a quadratic interpolation $\hat{\sigma}$ and the actual linear one. Using linear and quadratic shape functions to express σ and $\hat{\sigma}$ respectively, it can be shown that the error E is

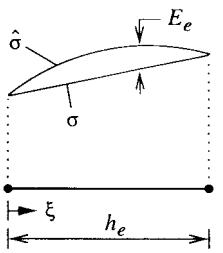

Figure 2. Approximation error in a 1D element.

$$E_e = \hat{\sigma} - \sigma = \frac{\xi}{2}(h_e - \xi)\left|\frac{\mathrm{d}^2\hat{\sigma}}{\mathrm{d}x^2}\right|_e, \tag{11}$$

where ξ is the local element co-ordinate and h_e the element length. A measure of the overall error in the element is then considered to be the root-mean-square value of E_e [26]

$$E_e^{\mathrm{RMS}} = \left(\int_0^{h_e} \frac{E_e^2}{h_e}\,\mathrm{d}\xi\right)^{1/2} = \frac{1}{\sqrt{120}} h_e^2 \left|\frac{\partial^2\hat{\sigma}}{\mathrm{d}x^2}\right|_e. \tag{12}$$

If we define an optimal mesh as one for which the error is equidistributed over elements, the following should hold for each element

$$h_e^2\left|\frac{\mathrm{d}^2\hat{\sigma}}{\mathrm{d}x^2}\right|_e = C, \tag{13}$$

where C is a positive constant. The second derivative in Equation (12) is based on $\hat{\sigma}$, which is linear, causing a problem. An approximation to this derivative is obtained by calculating the second derivative of the numerical solution, i.e. $\mathrm{d}^2\sigma/\mathrm{d}x^2$, by employing a recovery process based on a weighted residual method [1].

The above methodology can be extended to a 2D element based on the fact that each edge of a 2D element can be considered as a 1D element. The second derivative is now taken with respect to a unit vector $\mathbf{e}$

$$\frac{\partial^2\sigma}{\partial\mathbf{e}^2} = \mathbf{e}^T H \mathbf{e}, \tag{14}$$

where H is the Hessian matrix of σ which replaces the second derivative in equation (12)

$$H = \begin{bmatrix} \dfrac{\partial^2\sigma}{\partial x^2} & \dfrac{\partial^2\sigma}{\partial x\,\partial y} \\ \dfrac{\partial^2\sigma}{\partial y\,\partial x} & \dfrac{\partial^2\sigma}{\partial y^2} \end{bmatrix}, \tag{15}$$

the second derivatives are calculated for all nodes throughout the domain. The Hessian matrix can be decomposed into the form

$$H = R\Lambda R^T, \tag{16}$$

where Λ is the diagonal matrix of eigenvalues and R the matrix of eigenvectors. The matrix R can be interpreted as a rotation with angle α which the eigenvector corresponding to the smaller eigenvalue λ_1 makes with the x-axis. Since the error should be made positive, the Hessian matrix is reconstructed by replacing Λ with $|\Lambda|$ in (16)

$$\bar{H} = R|\Lambda|R^T \quad \text{or} \quad \bar{H} = SS^T, \tag{17}$$

with $S = R\sqrt{|\Lambda|}$ considered a transformation stretching the element in the direction of the principal axes of $\bar{H}$.

Once $\bar{H}$ is calculated for all the nodes, with assumed linear variation throughout the domain, the error (a scalar) on each edge is obtained from

$$e(\ell) = \int_{\xi_1}^{\xi_2} \sqrt{\mathbf{r}^T(\xi)\bar{H}(\xi)\mathbf{r}(\xi)}\,\mathrm{d}\xi, \tag{18}$$

where $\mathbf{r}(\xi)$ is the parametric representation of the edge ℓ. This edge-based error estimate is then used as input for the moving-node mechanism to reposition the nodes, and accordingly, the elements.

It should be noted that although $e(\ell)$ is a scalar, it represents the 'directional' error of the edge.

3.2. Moving-node strategy

To equidistribute the error over the entire domain, a moving-node method based on a spring analogy is used. This method is applicable to both structured and unstructured grids, although here it is used for structured grids. In this approach, the mesh is interpreted as a network of springs, where each edge is considered to be a fictitious spring with its stiffness representing the measure of the error (Figure 3). The equilibrium of forces in this network will then determine the movement of each node. This idea was first introduced by Gnoffo [12]. Nakahashi and Diewert [22] later complemented his work by incorporating a grid-orthogonality constraint, and applied it in a finite differences context. In the finite element method, however, grid orthogonality is not essential, and a simpler approach can be adopted.

In the spring network of Figure 3, the new position of point i is determined by minimizing the potential energy of the system at that node

$$P_i = \sum_j (\mathbf{x}_i - \mathbf{x}_j)^2 k_{ij}, \tag{19}$$

where $\mathbf{x}$ is the position vector. The stiffness of the interconnected springs, k_{ij}, is defined as

$$k_{ij} = \frac{e(\ell)}{\|\mathbf{x}_i - \mathbf{x}_j\|} = \frac{e(\mathbf{x}_i \cdot \mathbf{x}_j)}{\|\mathbf{x}_i - \mathbf{x}_j\|}, \tag{20}$$

in which $\|\cdot\|$ denotes the Euclidean norm. Minimizing (19) with respect to $\mathbf{x}_i$ gives the following equation

$$\sum_j (\mathbf{x}_i^{m+1} - \mathbf{x}_j^{m+1}) k_{ij}^{m+1} = 0, \tag{21}$$

which expresses the equilibrium of forces in the local spring network at the present adaptive iteration $m+1$. By lagging $\mathbf{x}_j$ and k_{ij} in equation (21), the correction to the model can be written as

$$\Delta \mathbf{x}_i = \frac{\sum_j (\mathbf{x}_j^m - \mathbf{x}_i^m) k_{ij}^m}{\sum_j k_{ij}^m}. \tag{22}$$

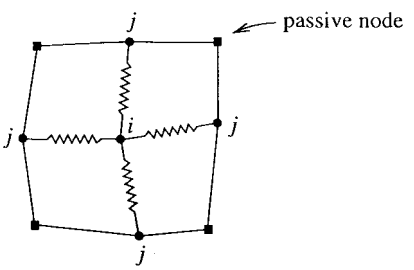

Figure 3. Local spring network corresponding to node i.

The new position of $\mathbf{x}_i$ is then calculated from

$$\mathbf{x}_i^{m+1} = \mathbf{x}_i^m + \omega \Delta \mathbf{x}_i, \tag{23}$$

where ω is a relaxation parameter.

The iteration process (23) is applied to all nodes in the domain in order to adapt the mesh to the solution. Boundary nodes can also move in the same way as internal nodes, but they are then reprojected onto the boundary to maintain the geometric integrity of the domain. The moving-node scheme is applied to grid points in a sweeping manner. The reason is to allow to check the quality of each newly oriented element during the mesh movement, and thus avoid formation of elements with a negative or nearly zero Jacobian. To obtain an appropriate adapted mesh, the number of adaptive iterations per adaptive cycle is chosen to be in the range $200 \sim 400$.

The adaptive method uses the solution of one of the scalar variables to adapt the mesh, and then interpolates the solution on the new mesh. The adapted mesh and the interpolated results are then used as initial data for the least-squares code to obtain a more accurate solution. Each mesh adaptation followed by the least-squares solution is called one adaptive cycle.

4. SOLUTION METHOD

The resulting system of linear algebraic equations

$$Ax = b, \tag{24}$$

is solved iteratively using the CG method. The coefficient matrix is symmetric and positive definite, therefore making the CG method the most optimal choice.

To improve the convergence properties of the iterative solver, preconditioning is applied using a diagonal (Jacobi) preconditioner. The elements of the preconditioner matrix M are considered to be the L_1-norm of the row elements of the coefficient matrix

$$m_{ii} = \sum_{j=1}^{n} |a_{ij}|.$$

The effect of the Jacobi preconditioner is to make A more diagonally-dominant, which in turn leads to a more uniform distribution of eigenvalues. This preconditioner is the simplest and the cheapest since inverting a diagonal matrix is very straightforward, memory requirements are limited to a vector of length n and its application only needs n multiplications.

For the CG method to have guaranteed convergence, the coefficient matrix should preserve the symmetry and positive definite property after preconditioning. As a result, applying the preconditioner matrix M from either left or right would be inappropriate, since $M^{-1}A$ and AM^{-1} are not generally symmetric nor positive definite, even when both M and A are. The remedy is to split M into two matrices

$$M = EE^T, \tag{25}$$

and use split preconditioning

$$(E^{-1}AE^{-T})(E^T x) = E^{-1}b, \tag{26}$$

where $E^{-1}AE^{-\mathrm{T}}$ is symmetric and positive definite.

The convergence test used to halt the iterative process is

$$\|r_k\| < \epsilon_{\text{rel}}\|b\| + \epsilon_{\text{abs}}, \tag{27}$$

where $\|r_k\|$ is the L_2-norm of residual: $r_k = b - Ax_k$, and ϵ_{rel} and ϵ_{abs} are relative and absolute tolerances respectively.

5. NUMERICAL RESULTS

Four test cases are examined to study the ability of the least-squares method in capturing shocks, and the robustness of the grid adaptation technique. The test cases are chosen to cover the transonic and supersonic flow regimes. The flow variable used for grid adaptation in all cases is the pressure.

5.1. Shock-reflection problem

The first test case is the reflection of a shock from a solid wall, as shown in Figure 4. The domain is discretized uniformly with 60×20 bilinear rectangular elements. The boundary conditions are

$$\text{inlet} \begin{cases} \rho = 1.0 \\ u = 2.9 \\ v = 0.0 \\ p = 0.7143 \end{cases} \qquad \text{upper boundary} \begin{cases} \rho = 1.7 \\ u = 2.6193 \\ v = 0.5063 \\ p = 1.5282 \end{cases}$$

On the lower boundary, the no-penetration boundary condition is imposed, and the exit boundary is left free. The values at the upper boundary are used as the initial guess.

The effect of the inherent viscosity, controlled by the time step, on the shock resolution is shown in Figure 5, where the time step has changed from 0.05 to 0.1. As expected (Figure 6), oscillations near the shock start growing after reducing the artificial viscosity, and the shocks become more smeared by increasing it. Figure 7 shows the effect of preconditioning on the convergence rate of the CG iterative solver. Here the relative and absolute tolerances are set to $\epsilon_{\text{rel}} = 10^{-6}$ and $\epsilon_{\text{abs}} = 10^{-14}$ respectively. It is clear from the figure that using the simple Jacobi preconditioner has led to a two orders of magnitude drop in the solver residual, and consequently, in about a 30% reduction in the number of iterations.

In an attempt to reduce the solution time, the same test case was run using $\epsilon_{\text{rel}} = 10^{-2}$ and $\epsilon_{\text{abs}} = 10^{-7}$ (one order of magnitude less than the global tolerance). The converged solution was found to be identical to the solution before reducing tolerances up to the fourth decimal point, indicating the accuracy of the results. This, however, led to about a 50% reduction in the overall CPU time, which is quite significant.

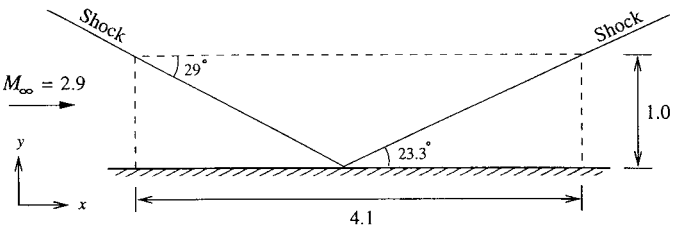

Figure 4. Reflection of a shock from a solid wall: description of the problem.

Using the reduced tolerances and $\Delta t = 0.1$, the results are adapted through five cycles. The artificial viscosity is reduced beginning at the third cycle by reducing the time step to 0.05. The initial level of artificial viscosity was high for the size of the grid near the shock, and it must be reduced for better shock resolution. Figure 8 shows the pressure contours and the grids after each adaptation. The improvements in the shock resolution after adaptation is quite evident in Figure 9.

5.2. Supersonic channel flow

The second test case is that of supersonic flow over a 4% circular arc bump placed in a channel. The height of the channel and its length, ahead and after the bump, are equal to the chord length. The boundary conditions at inlet are:

$$\rho = 1.0, \quad u = 1.65, \quad v = 0.0, \quad p = 0.7143.$$

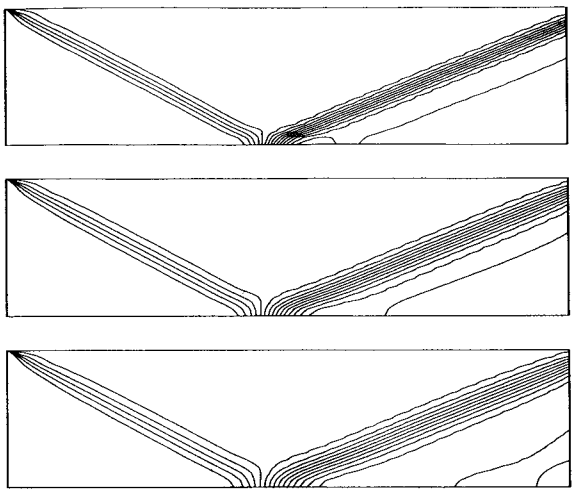

Figure 5. Pressure contours for different time steps; from top to bottom: $\Delta t = 0.05$, 0.1, 0.15.

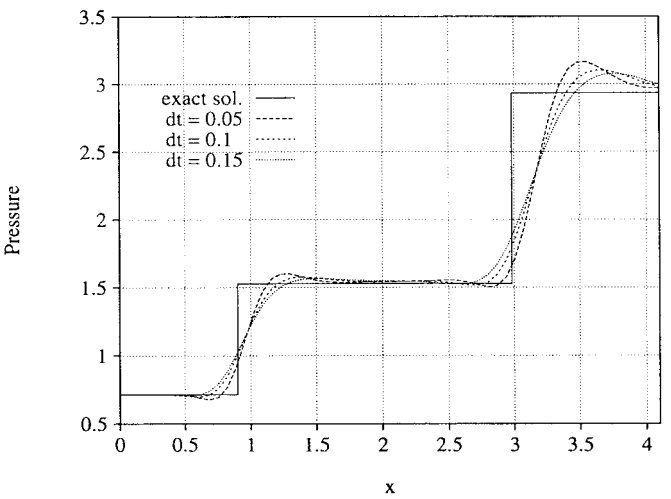

Figure 6. Pressure distribution for different time steps at $y = 0.5$, showing smearing and oscillations on non-adapted mesh.

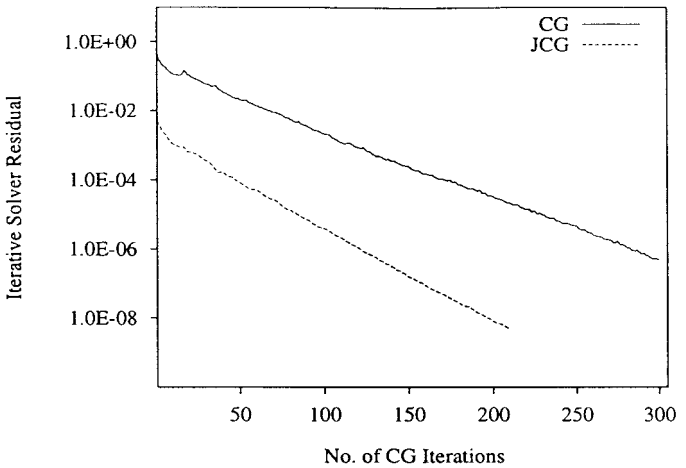

Figure 7. Convergence history of the iterative solver with Jacobi preconditioner at 30th global iteration, $\Delta t = 0.1$, shock-reflection problem.

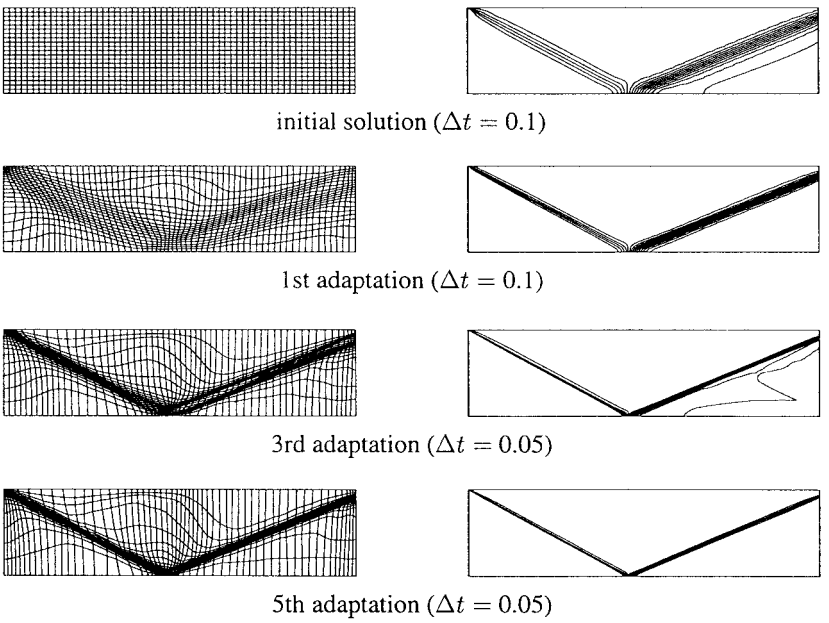

Figure 8. Evolution of the grid and solution during adaptation (pressure contours).

On walls the flow tangency boundary condition is imposed, and the exit boundary is left free. The grid consists of 64×16 uniformly distributed bilinear rectangular elements, with 16 elements in the y-direction, 22 elements on the bump, and 21 elements on each side of it.

The pressure contours for $\Delta t = 0.1$ are shown in Figure 10. The leading- and trailing-edge shocks, as well as the interaction of the trailing-edge shock with the reflected shock, are qualitatively well-captured, although they are rather smeared. The results are obtained using Jacobi preconditioned CG method with the tolerances set to: $\epsilon_{\text{rel}} = 10^{-2}$ and $\epsilon_{\text{abs}} = 10^{-7}$.

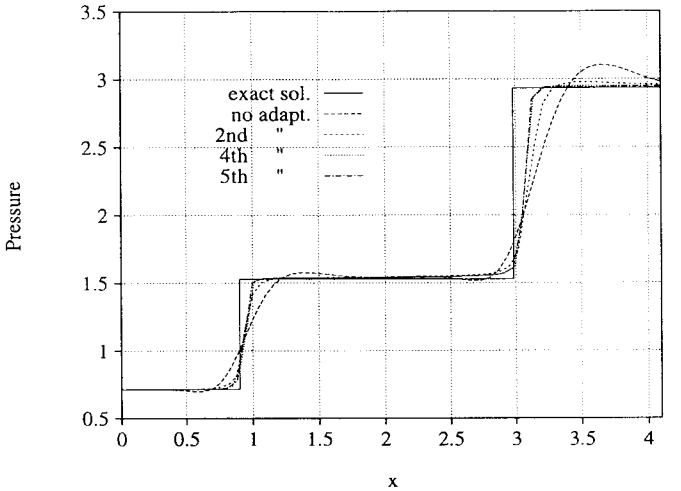

Figure 9. Pressure distribution at $y = 0.5$ before and after adaptation.

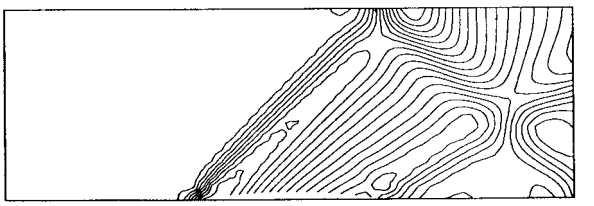

Figure 10. Pressure contours for supersonic channel flow on non-adapted grid, $\Delta t = 0.1$.

The adapted grids and solutions after five cycles of adaptation are shown in Figure 11, where the artificial viscosity is reduced after the second cycle. All the shocks, especially the strongest leading-edge shock, are quite sharply captured. Figure 12 shows the improvement in shocks resolution, where the leading-edge shock is quite sharp, and the trailing-edge and the reflected shock profiles are close to vertical. Magnification of the grid near the leading-edge (Figure 13) shows how the elements are re-oriented to be aligned with the shock, creating very high aspect ratio elements.

The final adapted solution is compared with the published results of Eidelman *et al.* [10] and Jiang *et al.* [18] in Figure 14. Since these authors have not used an adaptive method, the shock is more smeared in their results. Overall, the adapted least-squares method shows good agreement except for the oscillations at the leading- and trailing-edges, which are due to the use of a non-conservative formulation.

5.3. Transonic channel flow

The third test case is the transonic flow in a channel with a 10% circular arc bump. The grid consists of bilinear rectangular elements, with 32 elements uniformly distributed on the bump, and 16 elements on each side of it, centrally clustered with respect to the bump. The height of the channel is divided into 16 elements which are clustered toward the bottom wall, with the minimum element size on the wall equal to 0.03125. The Mach number at inlet is $M_\infty = 0.675$,

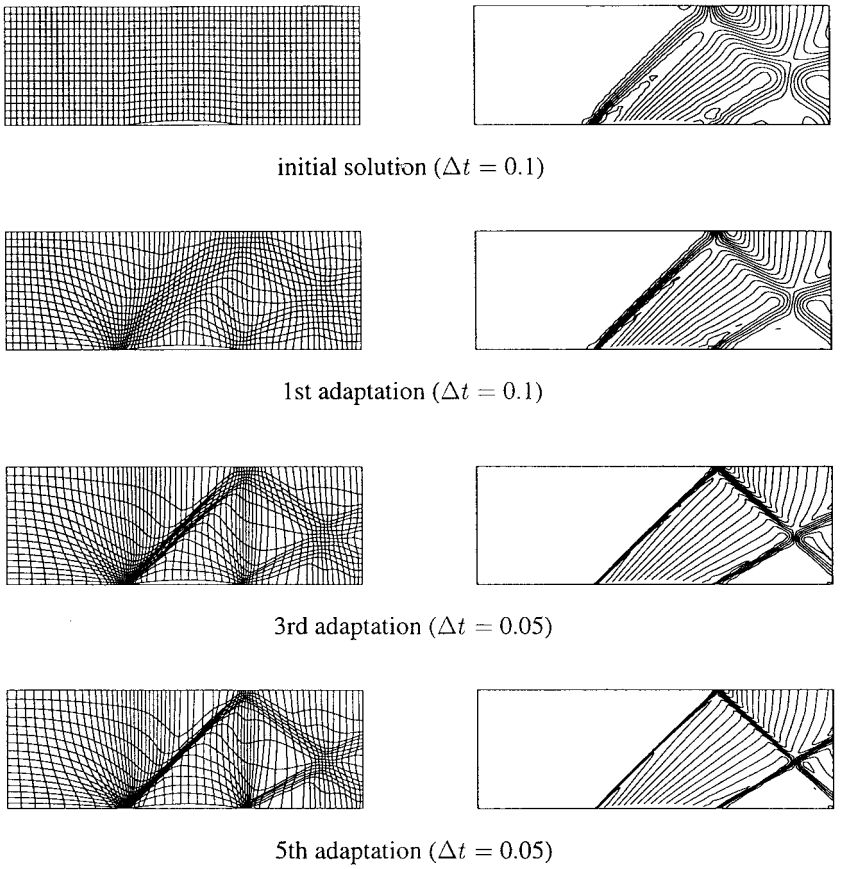

Figure 11. Evolution of the grid and solution during adaptation (pressure contours).

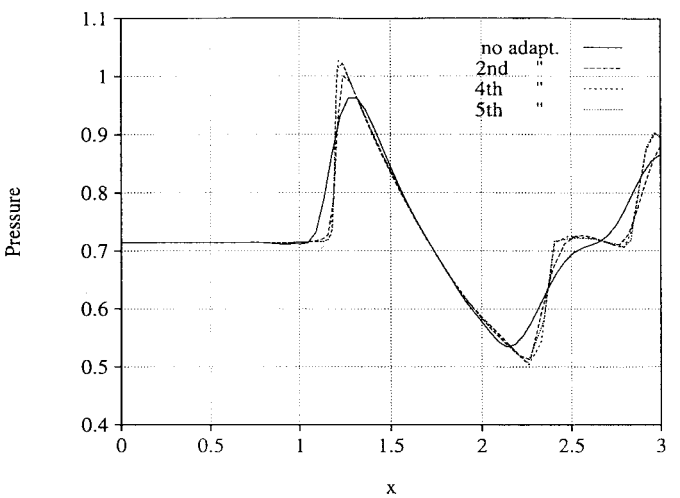

Figure 12. Pressure distribution at $y = 0.2$ before and after adaptation.

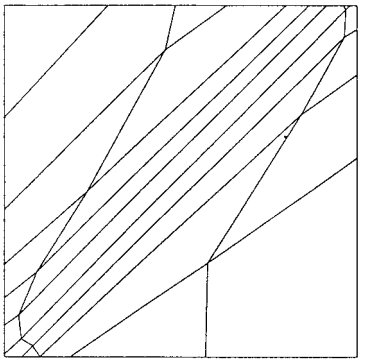

Figure 13. Magnification of the grid in the leading-edge shock region.

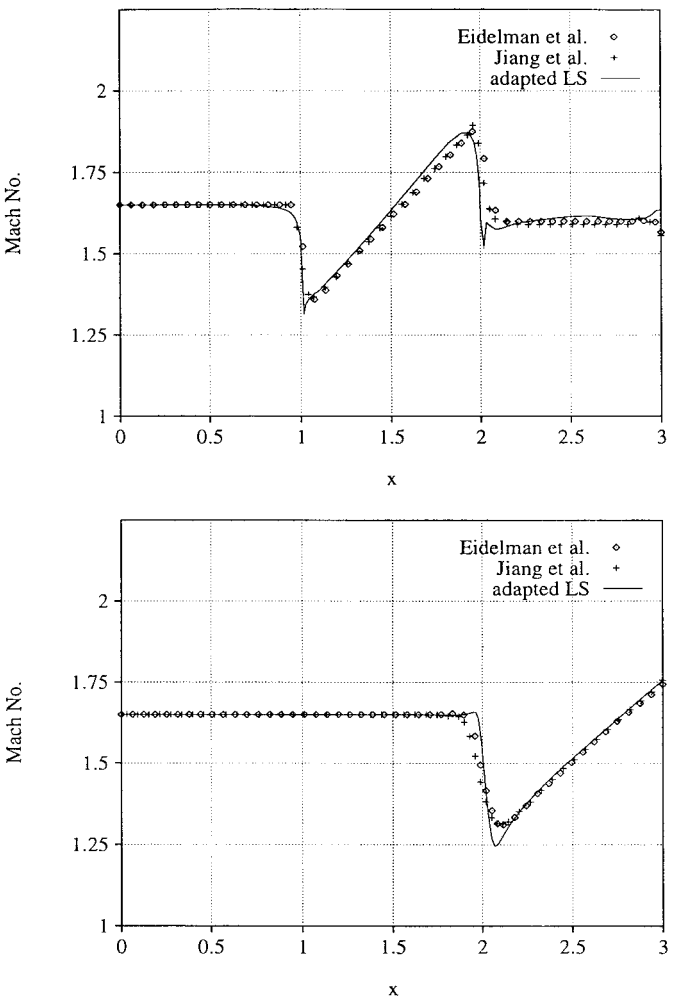

Figure 14. Mach number distribution on the lower and upper walls; comparison with the published data.

which is large enough to create a supersonic pocket on the bump followed by a shock. The inlet and exit boundary conditions are:

$$\text{inlet: } \rho = 1.0, \quad u = 0.675, \quad v = 0.0; \quad \text{exit: } p = 1.5282.$$

On the lower and upper walls, the no-penetration boundary condition is imposed.

The initial solution and adapted results are shown in Figure 15. The time step, which is initially set to 0.3, is reduced to 0.1 at the final adaptive cycle. Figure 16 demonstrates the change in the Mach number distribution on the walls before and after adaptation.

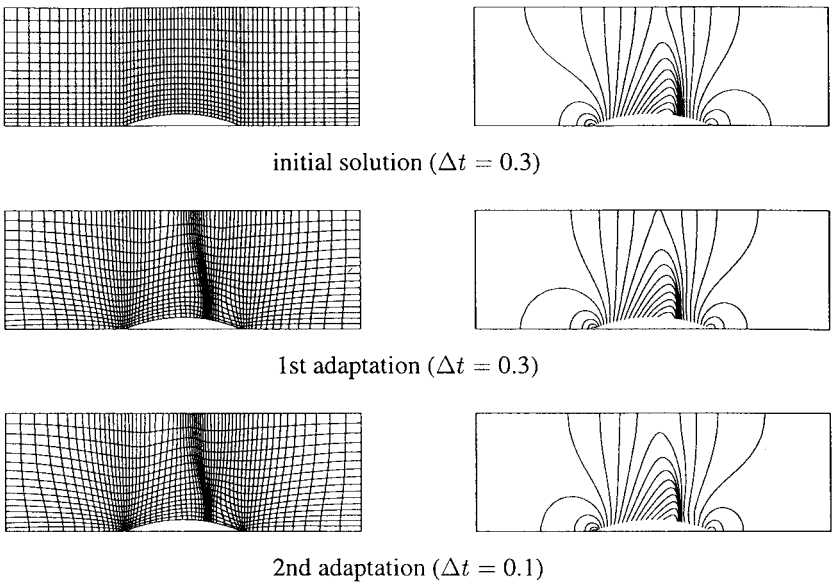

Figure 15. Original and adapted solutions and grids.

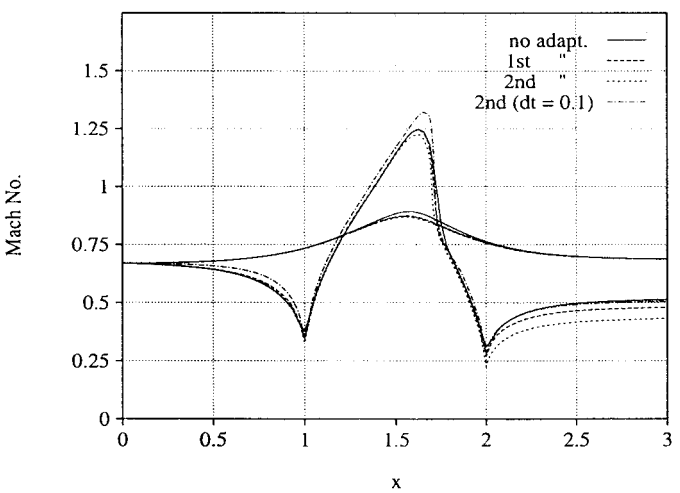

Figure 16. Mach number distribution on the lower and upper walls before and after adaptation; transonic channel flow.

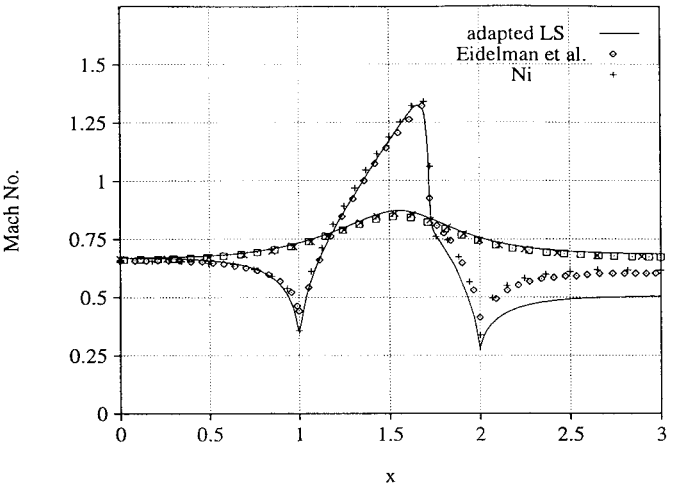

Figure 17. Mach number distribution on the lower and upper wall; comparison with the published data.

The adapted results are compared with those of Ni [24] and Eidelman *et al.* [10] in Figure 17. The position of the shock is approximately at 72% of the chord, and the maximum Mach number is 1.323, both in very good agreement with these results. The deviation is, however, in the Mach number distribution after the shock. Figure 18 shows the Mach contours for the final adapted case. It is seen that there is a viscous effect near the lower wall (limited to the first row of the elements) similar to a boundary layer. This viscous layer has little effect on the solution before the shock. After the shock, however, the flow becomes rotational near the wall and hence adversely affects this layer, leading to underestimation of the Mach number.

Figure 19 shows that using larger Δt, i.e. more artificial viscosity, worsens the situation both ahead and after the shock. This is more evident from Figure 16, where the Mach number profile behind the bump is elevated after reducing Δt in the second adaptive cycle. The figure also shows that as the shock becomes stronger, thus generating more vorticity, the tail branch of the Mach number profile is reduced.

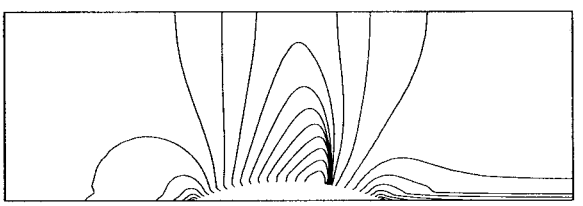

Figure 18. Iso-mach lines for the second adapted solution, $\Delta t = 0.1$.

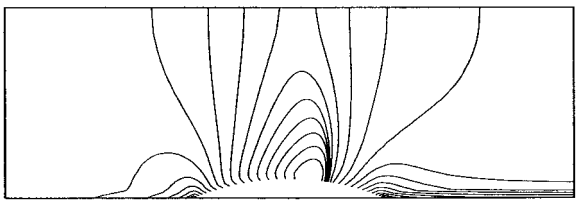

Figure 19. Iso-mach lines for the second adapted solution, $\Delta t = 0.3$.

5.4. Supersonic flow over a blunt body

As the last test case, the supersonic flow over a blunt body is examined. This is to show that both attached and detached shock waves can be clearly captured by the least-squares method.

The grid for this case is shown in Figure 20(a), which consists of 32×52 uniformly distributed bilinear rectangular elements. The flow conditions at inlet are

$$\rho = 1.0, \quad u = 3.0, \quad v = 0.0, \quad p = 0.7143.$$

Flow tangency conditions are imposed on the walls and no boundary conditions are specified at the supersonic exit.

The initial and adapted grids and solutions are shown in Figure 20. The initial solution is perfectly symmetric about the centerline. The detached shock is very strong at the nose and becomes weaker further downstream. After three cycles of adaptation, the thick shock of the initial solution is converted to a very sharp shock, as shown in Figure 20(d). This is more clearly evident in Figure 21, which shows the effect of adaptation on the shock along the centerline.

For the initial and the first two adapted solutions, a constant time step of $\Delta t = 0.02$ was used. For the third adaptation, the artificial viscosity was reduced by decreasing Δt to 0.01. As a result, the shock was much sharper and with very little dissipation.

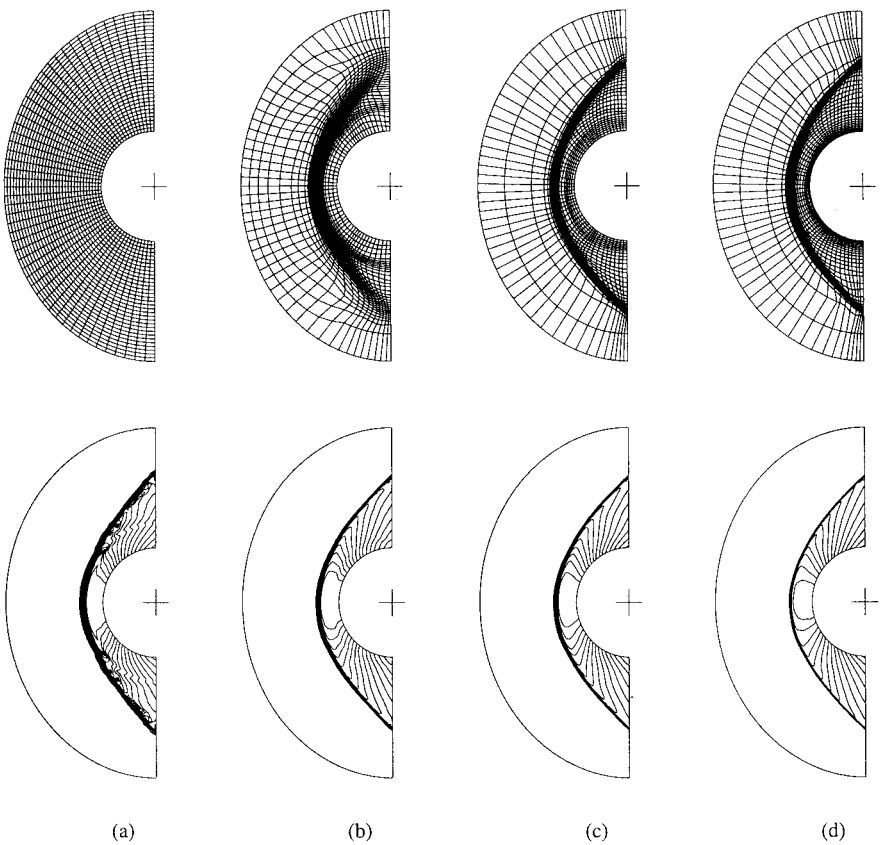

Figure 20. Initial and adapted solutions and grids (pressure contours).

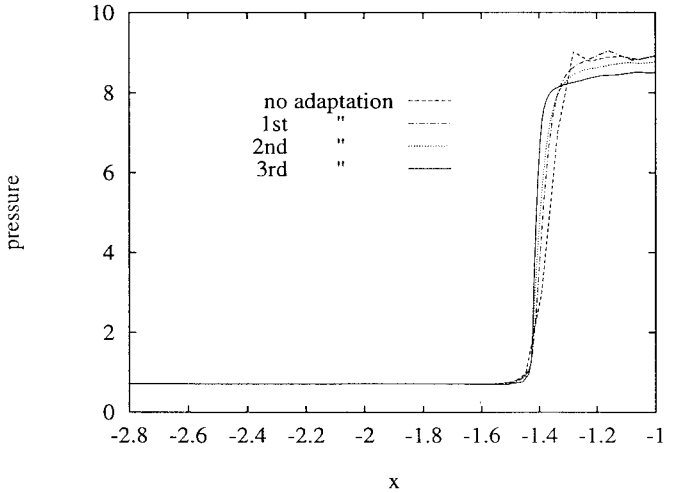

Figure 21. Pressure distribution at centerline before and after adaptation.

6. CONCLUSIONS

The least-squares finite element method is combined with a robust grid adaptation technique to solve the Euler equations for three supersonic and one transonic test problems. The quality of the numerical results indicate the remarkable performance of the adaptive method, and demonstrate its superiority to many existing techniques. This is quite evident from the final adapted grids, especially for the shock-reflection and supersonic channel problems, where the elements are strongly aligned with the shocks.

The presence of oscillations after the leading- and trailing-edge shocks in the supersonic channel case, and the slightly misplaced shock position in the shock-reflection problem is anticipated to be due to using a primitive-variable formulation. As a result, using the least-squares method with conservative variables is recommended.

Overall, the combination of the least-squares method with the directionally-adaptive method has produced promising results, despite using an artificial viscosity mechanism without any free parameters and relatively coarse grids.

REFERENCES

1. D. Ait-Ali-Yahia, W.G. Habashi, A. Tam, M.-G. Vallet and M. Fortin, 'A directionally-adaptive methodology using an edge-based error estimate on quadrilateral grids', *Int J. Numer. Methods Fluids*, **23**, 673–690 (1996).
2. O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, 1994.
3. G.S. Baruzzi, W.G. Habashi and M.M. Hafez, 'Finite element solutions of the Euler equations for transonic external flows', *AIAA J.*, **29**, 1886–1893 (1991).
4. G.S. Baruzzi, W.G. Habashi and M.M. Hafez, 'A second-order finite element method for the solution of the transonic Euler and Navier–Stokes equations', *Int. J. Numer. Methods Fluids*, **20**, 671–693 (1995).
5. G.J. Le Beau, S.E. Ray, S.K. Aliabadi and T.E. Tezduyar, 'SUPG finite element computation of compressible flows with the entropy and conservation variables formulations', *Comput. Methods Appl. Mech. Eng.*, **104**, 397–422 (1993).
6. A.L. Brooks and T.J.R. Hughes, 'Streamline upwind Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations', *Comput. Methods Appl. Mech. Eng.*, **32**, 199–259 (1982).
7. Ch.H. Bruneau, J. Laminie and J.J. Chattot, 'Computation of 3D vortex flows past a flat plate at incidence through a variational approach of the full steady Euler equations', *Int. J. Numer. Methods Fluids*, **9**, 305–323 (1989).

8. G.F. Carey and B.-N. Jiang, 'Least-squares finite element method and preconditioned conjugate gradient solution', *Int. J. Numer. Methods Eng.*, **24**, 1283–1296 (1987).
9. G.F. Carey and B.-N. Jiang, 'Least-squares finite elements for first-order hyperbolic systems', *Int. J. Numer. Methods Eng.*, **26**, 81–93 (1988).
10. S. Eidelman, P. Colella and R.P. Shreeve, 'Application of the Godunov method and its second-order extension to cascade flow modeling', *AIAA J.*, **22**, 1609–1615 (1984).
11. C.A.J. Fletcher, 'A primitive variable finite element formulation for inviscid compressible flow', *J. Comput. Phys.*, **33**, 301–312 (1979).
12. P.A. Gnoffo, 'A finite-volume, adaptive grid algorithm applied to planetary entry flowfields', *AIAA J.*, **21**, 1249–1254 (1983).
13. T.J.R. Hughes and T.E. Tezduyar, 'Finite element methods for first-order hyperbolic systems with particular emphasis on the compressible Euler equations', *Comput. Methods Appl. Mech. Eng.*, **45**, 217–284 (1984).
14. T.J.R. Hughes, M. Mallet and A. Mizukami, 'A new finite element formulation for computational fluid dynamics: II. Beyond SUPG', *Comput. Methods Appl. Mech. Eng.*, **54**, 341–355 (1986).
15. B.-N. Jiang and G.F. Carey, 'Adaptive refinement for least-squares finite elements with element-by-element conjugate gradient solution', *Int. J. Numer. Methods Eng.*, **24**, 569–580 (1987).
16. B.-N. Jiang and G.F. Carey, 'Least-squares finite element methods for compressible Euler equations', *Int. J. Numer. Methods Fluids*, **10**, 557–568 (1990).
17. B.-N. Jiang and L.A. Povinelli, 'Least-squares finite element method for fluid dynamics', *Comput. Methods Appl. Mech. Eng.*, **81**, 13–37 (1990).
18. Y. Jiang, C.P. Chen and H.M. Shang, 'A new pressure–velocity coupling procedure for inviscid and viscous flows at all speeds', *Proceedings of the Fourth Int. Sym. on Computational Fluid Dynamics*, Vol. 1, University of California, Davis, September 9–12, 1991, pp. 545–550.
19. D. Lefebvre and J. Peraire, 'Finite element least squares solution of the Euler equations using linear and quadratic approximations', *Int. J. Comp. Fluid Dyn.*, **1**, 1–23 (1993).
20. R. Löhner, K. Morgan and O.C. Zienkiewicz, 'The solution of non-linear hyperbolic equation systems by the finite element method', *Int. J. Num. Meth. Fluids*, **4**, 1043–1063 (1984).
21. P.P. Lynn and S.K. Arya, 'Use of the least squares criterion in the finite element formulation', *Int. J. Num. Meth. Eng.*, **6**, 75–88 (1973).
22. K. Nakahashi and G.S. Diewert, 'Self-adaptive-grid method with application to airfoil flow', *AIAA J.*, **25**, 513–520 (1987).
23. H. Nguyen and J. Reynen, 'A space–time least-square finite element scheme for advection–diffusion equations', *Comp. Meth. Appl. Mech. Eng.*, **42**, 331–342 (1984).
24. R.-H. Ni, 'A multiple grid scheme for solving the Euler equations', *AIAA J.*, **20**, 1565–1571 (1982).
25. J.T. Oden, T. Strouboulis and Ph. Devloo, 'Adaptive finite element methods for high-speed compressible flows', *Int. J. Num. Meth. Fluids*, **7**, 1211–1228 (1987).
26. J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz, 'Adaptive remeshing for compressible flow computations', *J. Comp. Phys.*, **72**, 449–466 (1987).
27. J.F. Polk and P.P. Lynn, 'A least squares finite element approach to unsteady gas dynamics', *Int. J. Num. Meth. Eng.*, **12**, 3–10 (1978).
28. J.N. Reddy, *An Introduction to the Finite Element Method*, McGraw-Hill, New York, 1993.
29. L.Q. Tang and T.T.H. Tsang, 'An efficient least-squares finite element method for incompressible flows and transport processes', *Int. J. Comp. Fluid Dyn.*, **4**, 21–39 (1995).

