International Journal of Computational Fluid Dynamics
Vol. 22, Nos. 1–2, January–February 2008, 3–9

Taylor & Francis
Taylor & Francis Group

Role of coordinates in computational fluid dynamics

W.H. Hui[a]*, J.J. Hu[b] and K.M. Shyue[c]

[a]*Division of Mechanics, Research Centre for Applied Sciences Academia Sinica, Taipei, Taiwan;* [b]*Department of Information Management, Shu-Te University, Kaohsiung County, Taiwan;* [c]*Department of Mathematics, National Taiwan University, Taipei, Taiwan*

(*Received January 2007; final version received 19 June 2007*)

Computational fluid dynamics uses large scale numerical computation to solve problems of fluid flow. It turns out that the numerical solution for a given flow depends on the coordinates (grid) used to compute the flow. The commonly used Eulerian and Lagrangian coordinate systems both have advantages and drawbacks. In this paper, we first discuss the role of coordinates in computational fluid dynamics regarding the questions of: (a) conservation form partial differential equations; (b) numerical resolution of contact discontinuities; (c) grid generation; and (d) grid orthogonality. We then introduce a unified coordinate system which combines the advantages of both Eulerian and Lagrangian system and beyond, while avoiding their drawbacks. Examples include a transonic flow past an airfoil and a two-fluids flow with shocks.

Keywords: unified coordinates; automatic grid generation; Eulerian coordinates; multi-fluids flow; Godunov scheme

1. Eulerian versus Lagrangian coordinates

The starting point in CFD is the physical conservation law

$$\frac{\partial}{\partial t}\int_{\Omega(t)} E\,\mathrm{d}\Omega = -\int_{\partial\Omega(t)} \vec{F}\cdot\vec{n}\,\mathrm{d}S. \tag{1}$$

In Eulerian coordinates, which are fixed in space, Equation (1) can be written as partial differential equations (PDE) in conservation form. Thus, for a γ-law perfect gas we have

$$\frac{\partial E_e}{\partial t} + \frac{\partial F_e}{\partial x} + \frac{\partial G_e}{\partial y} = 0, \tag{2}$$

$$E_e = [\rho, \rho u, \rho v, \rho e]^{\mathrm{T}},$$

$$F_e = [\rho u, \rho u^2 + p, \rho uv, u(\rho e + p)]^{\mathrm{T}},$$

$$G_e = [\rho v, \rho uv, \rho v^2 + p, v(\rho e + p)]^{\mathrm{T}}.$$

In (2), u and v are the x- and y-components of fluid velocity, and

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

Shock-capturing computation is therefore easy to apply in the Eulerian system, but it tends to smear contact discontinuities badly. Furthermore, for computing flow past a body, which is a central problem in CFD, it is necessary to generate a body-fitted grid prior to flow computation.

By contrast, in Lagrangian coordinates, contact discontinuities can be resolved sharply. However, the coordinates are flow dependent, hence Equation (1) cannot easily be written in conservation PDE form (except in the special case of one-dimensional flow). This complicates the computation, for instance, a stagger grid is needed, causing numerical diffusion. Moreover, the grid deforms with the fluid, causing inaccuracy and even breakdown of computation.

2. The 'Optimal' coordinate system

Can we have a coordinate system that combines the advantages of the Eulerian and Lagrangian, while avoiding their drawbacks? Such a system would be 'optimal' in some sense (whether or not a system is optimal depends on the criteria, which are necessarily subjective). Specifically, we want the coordinate system to possess the following properties:

(1) conservation PDE form exists, as in Eulerian;
(2) contact discontinuities are sharply resolved, as in Lagrangian;
(3) grid can be automatically generated to fit given body shapes;
(4) grid is orthogonal; and
(5) grid is uniform, etc.

*Corresponding author. Email: whhui@ust.hk

ISSN 1061-8562 print/ISSN 1029-0257 online

DOI: 10.1080/10618560701737112
http://www.informaworld.com

We shall show that the unified coordinate system introduced in the next section possesses these properties.

3. The unified coordinate system

For simplicity, we consider 2D flow and introduce a unified coordinate system (λ, ξ, η) (Hui *et al.* 1999, Hui and Kudriakov 2001) via the transformation from Eulerian system (t, x, y)

$$\begin{cases} \mathrm{d}t = \mathrm{d}\lambda \\ \mathrm{d}x = U\,\mathrm{d}\lambda + A\,\mathrm{d}\xi + L\,\mathrm{d}\eta \\ \mathrm{d}y = V\,\mathrm{d}\lambda + B\,\mathrm{d}\xi + M\,\mathrm{d}\eta \end{cases} \tag{3}$$

From (3), we get

$$\frac{D_{\mathbf{Q}}}{Dt}\begin{pmatrix} \xi \\ \eta \end{pmatrix} = 0, \tag{4}$$

where

$$\frac{D_{\mathbf{Q}}}{Dt} \equiv \frac{\partial}{\partial t} + \mathbf{Q}\cdot\nabla_{\vec{x}}.$$

So, the coordinates (ξ, η), and hence computational cells, move with velocity $\mathbf{Q} = (U, V)$. It includes the Eulerian as a special case when $\mathbf{Q} = \mathbf{0}$ and Lagrangian when $\mathbf{Q} = \mathbf{q} = (u, v)$.

Under the transformation, (2) becomes

$$\frac{\partial E}{\partial \lambda} + \frac{\partial F}{\partial \xi} + \frac{\partial G}{\partial \eta} = 0, \tag{5}$$

where

$$E = \begin{pmatrix} \rho J \\ \rho J u \\ \rho J v \\ \rho J e \\ A \\ B \\ L \\ M \end{pmatrix}, \quad F = \begin{pmatrix} \rho X \\ \rho X u + pM \\ \rho X v - pL \\ \rho X e + p(uM - vL) \\ -U \\ -V \\ 0 \\ 0 \end{pmatrix},$$

$$G = \begin{pmatrix} \rho Y \\ \rho Y u - pB \\ \rho Y v + pA \\ \rho Y e + p(vA - uB) \\ 0 \\ 0 \\ -U \\ -V \end{pmatrix}.$$

Here, $J = AM - BL$, $X = (u - U)M - (v - V)L$ and $Y = (v - V)A - (u - U)B$. The first four equations in (5) are the physical conservation laws. But since they involve the coefficients A, B, L and M of the transformation, they are not closed. It is, therefore, necessary and sufficient to append the time evolution of these coefficients, i.e. the last four equations, to make the system closed. These four equations are the compatibility conditions of the transformation and are called geometric conservation laws.

4. Grid movement

Since there are two arbitrary functions, U and V, we prescribe two requirements.

(A) Coordinate lines $\eta =$ consant shall be material lines of fluid particles, meaning

$$\frac{D_{\mathbf{q}}\eta}{Dt} = 0. \tag{6}$$

Together with the first of (4), we get

$$(v - V)A = (u - U)B. \tag{7}$$

4.1 Observations

(a) Contact lines, being material lines, must coincide with coordinate lines and, therefore, can be resolved sharply.
(b) As the body surface is a material line, condition (7) guarantees that the unified grid is automatically a body-fitted grid at all times. This provides the basis for automatic grid generation.
(c) A material interface (including a free surface) corresponds to $\eta =$ constant and thus can be resolved sharply.
(d) $\eta(x, y, t)$ is a level set function; hence there is no need to introduce an extra function when using the level set method.

All these make the unified coordinate approach particularly suitable and useful for problems of multi-fluids flow and flow–solid interactions, etc.

(B) Grid angles and hence grid orthogonality shall be preserved during the λ marching computation. Thus, we have

$$\frac{\partial}{\partial\lambda}\cos^{-1}\left[\frac{\nabla\xi\cdot\nabla\eta}{|\nabla\xi||\nabla\eta|}\right] = \frac{\partial}{\partial\lambda}\cos^{-1}\left[\frac{AL+BM}{\sqrt{A^2+B^2}\sqrt{L^2+M^2}}\right]=0. \tag{8}$$

After eliminating V from (7) and using the geometric conservation laws, (8) becomes an ordinary differential equation for U

$$\frac{\partial U}{\partial\eta}+P(\eta;\lambda,\xi)U=Q(\eta;\lambda,\xi), \tag{9}$$

$$P(\eta;\lambda,\xi)=\frac{S^2}{T^2J}\left(A\frac{\partial B}{\partial\xi}-B\frac{\partial A}{\partial\xi}\right)-\frac{L}{AJ}\left(A\frac{\partial B}{\partial\eta}-B\frac{\partial A}{\partial\eta}\right)$$

$$Q(\eta;\lambda,\xi)=\frac{S^2A}{T^2J}\left(B\frac{\partial u}{\partial\xi}-A\frac{\partial v}{\partial\xi}\right)+\frac{L}{J}\left(A\frac{\partial v}{\partial\eta}-B\frac{\partial u}{\partial\eta}\right)+uP(\eta;\lambda,\xi),$$

$$S^2=L^2+M^2,\quad T^2=A^2+B^2.$$

We can specify any initial data for U at $\eta=$ const.

5. Computation procedure

The special case of steady supersonic flow was successfully studied in Hui and Hu (2006), and here we are concerned with steady flow which may have subsonic regions. Flows of this type are computed by marching in time λ until a steady state is reached.

The computation procedure for uniform flow past a body is illustrated by a Mach 0.8 steady air flow ($\gamma=1.4$) past a NACA 0012 airfoil as follows.

Initialisation stage – automatic generation of body-fitted grid in a computational window. Given the grid sizes, Δx and Δy, and the number of cells, $M\times N$, in the window (we use $M=200$ and $N=100$ in the example).

Step 1. Begin with a column of N orthogonal cells, representing the given uniform flow in the x-direction (Figure 1(a)). This gives the initial values of (A, B, L, M) = (1, 0, 0, 1). We also take (U, V) = (u, v) initially.

Step 2. Compute the solution to Equation (5) by marching in time λ, using dimensional splitting: splitting into two 1D systems in $\lambda-\xi$ and $\lambda-\eta$, each of them is solved using the standard Godunov/MUSCL scheme with the minmod limiter. (Details are as follows. To update the solution from time n to time $n+1$: (a) solve the first four equations (the physical conservation laws) of Equation (5) for (ρ, p, u, v) keeping A, B, L, M, U and V at time n level, (b) use this updated values together with a specified initial data to solve Equation (9) for U and then Equation (7) for V at time $n+1$ and (c) use these updated values of U and V to update (A, B, L, M) at time level $n+1$ by integrating the geometric conservation laws. At all outer boundaries of the computational region at every time step, we apply the characteristic boundary conditions.) After one time step $\Delta\lambda$, this column of cells moves to the right by $U\Delta\lambda$.

Step 3. After several time steps when the initial column of cells has moved to the right by a distance equal to Δx, add one new column of cells on the left that is identical to the initial column.

Step 4. Repeat this process of adding cell columns on the left of the computational region until the leading column meets the body surface (Figure 1(b)) we then impose the boundary condition of zero normal velocity on the body surface.

Step 5. Continue this process until after the columns of cells cover the whole body surface and further downstream, when we have M columns of cells in the window (Figure 1(c)–(e)).

This completes the initialisation stage, and we now have a body-fitted grid (Figure 1(e),(f)) and a flow field around the airfoil in the window. The computed grid is seen orthogonal, as predicted. It is also fairly uniform in the x-direction, because in solving Equation (9) we have specified uniform data for U at $\eta=$ constant The associated flow field computed (e.g. surface pressure in Figure 1(g)) is, however, only a very rough approximation to the correct one [see, e.g. the potential solution of Hafez *et al.* (1984)], partly because it has not reached the steady state and partly because the downstream boundary conditions used at the transient times, e.g. in Figure 1(a)–(d), are obviously incorrect as the computational regions at those times are not the full window.

To progress further, one could use the body-fitted orthogonal grid generated so far to perform an Eulerian computation with the associated flow field as an initial solution. This can be easily done by putting $U=V=0$ (without solving Equation (9)) during the subsequent iterations towards a steady state. In this way, the unified coordinate approach plays the role of grid generation for Eulerian computation.

An alternative and better way is to continue the unified coordinate computation to proceed to the *Main stage – iteration with flow-adjusted grids*.

Step 6. To iterate the solution towards a steady state, whenever we add a new column of cells on the left of the window we also simultaneously delete the

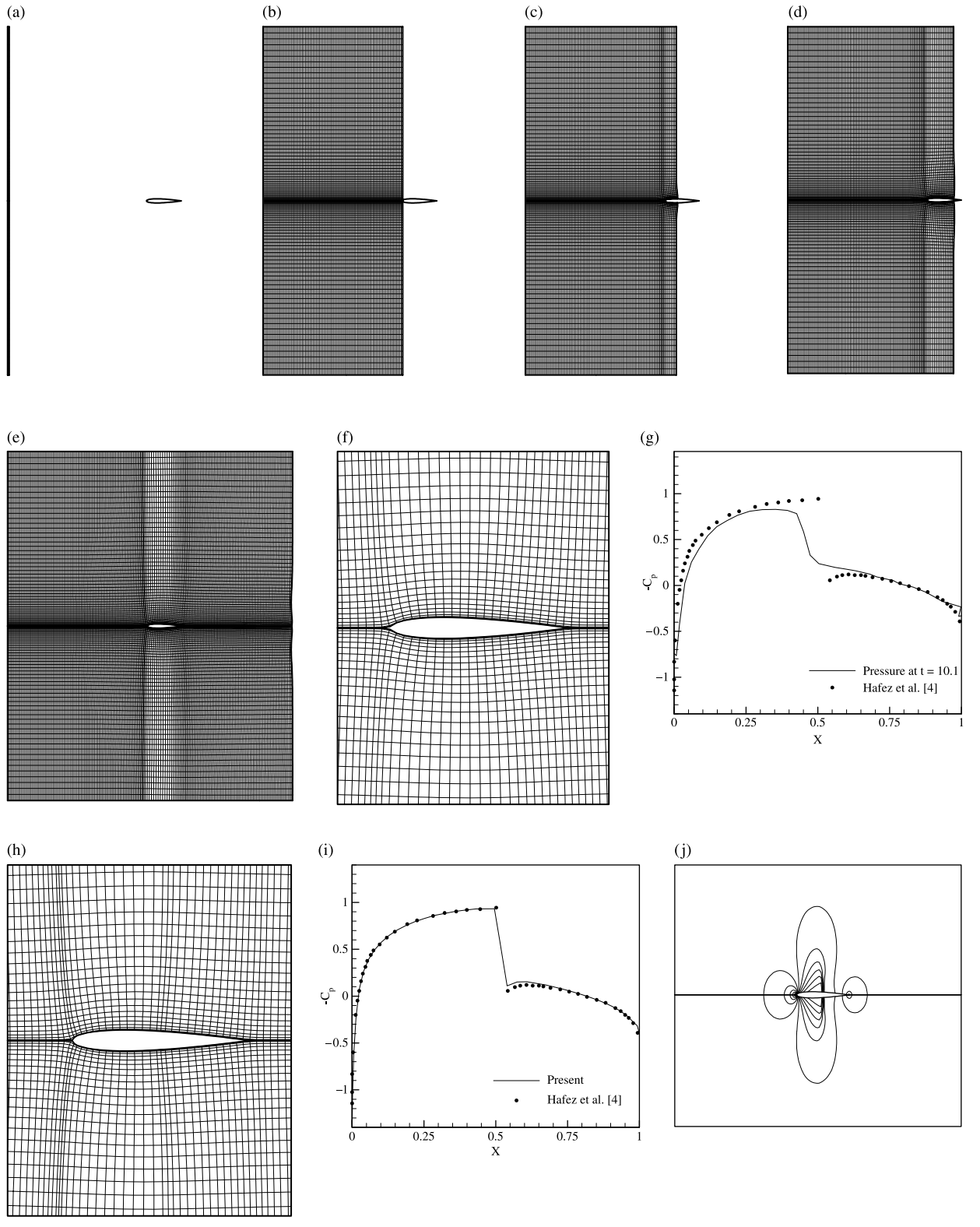

Figure 1. Computed results for a Mach 0.8 steady flow past a NACA 0012 airfoil, showing flow-generated grids at different times, surface pressure and pressure contours. $\gamma = 1.4$: (a) initial grid at $t = 0.0$, (b) flow-generated grid at $t = 4.0$, (c) flow-generated grid at $t = 4.6$, (d) flow-generated grid at $t = 4.9$, (e) flow-generated grid at $t = 10.1$, (f) blow-up grid at $t = 10.1$, (g) surface pressure at $t = 10.1$, (h) flow-adjusted blow-up grid at $t = 50.0$, (i) surface pressure at $t = 50.0$ and (j) pressure contours at $t = 50.0$.

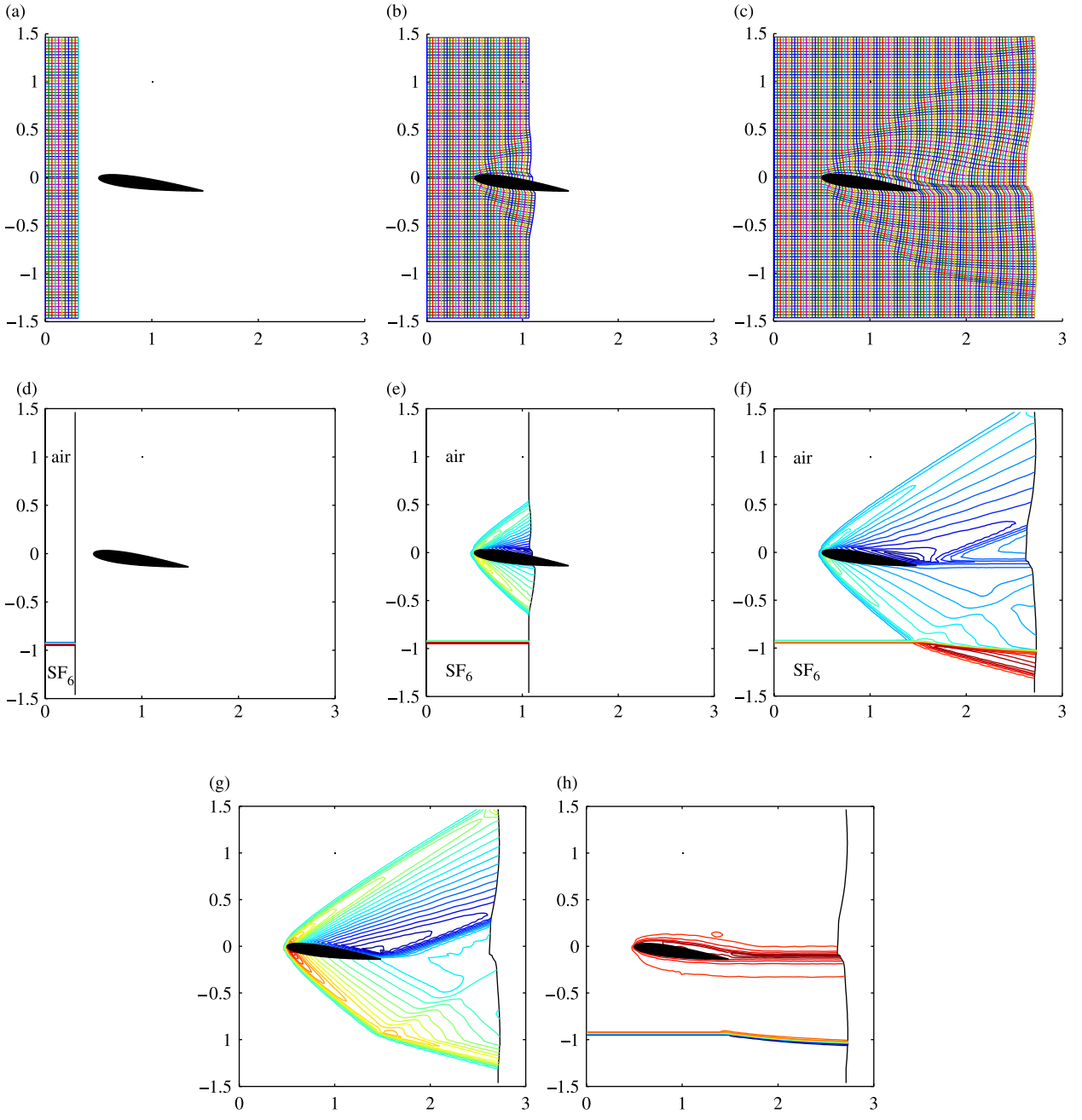

Figure 2. Sample computation of a Mach 2.2 steady flow past a NACA 0012 airfoil at angle of attack of 8° and a horizontal air–SF_6 material interface. The flow-generated grid, and the contours of the density, pressure and entropy are shown at three different times: (a) flow-generated grid, $t = 0.128$, (b) flow-generated grid, $t = 1.04$, (c) flow-generated grid, $t = 4.0$, (d) density contours, $t = 0.128$, (e) density contours, $t = 1.04$, (f) density contours, $t = 4.0$, (g) pressure contours, $t = 4.0$ and (h) entropy contours, $t = 4.0$. (All colour figures available online in colour.)

right-most column of cells from the computation window, thus keeping the window in the same size. At the same time, we improve the solution by using the information of the flow field at every time step, e.g. the surface pressure gradient, to adjust the initial data of U at $\eta =$ constant in solving (9) so that the grid is refined in regions of high pressure gradient (Figure 1(h)). We note that this flow-adjusted refined grid remains orthogonal. The computed surface pressure distribution is shown in Figure 1(i), which is much better than that in Figure 1(g) and is in good agreement with the potential flow computation of

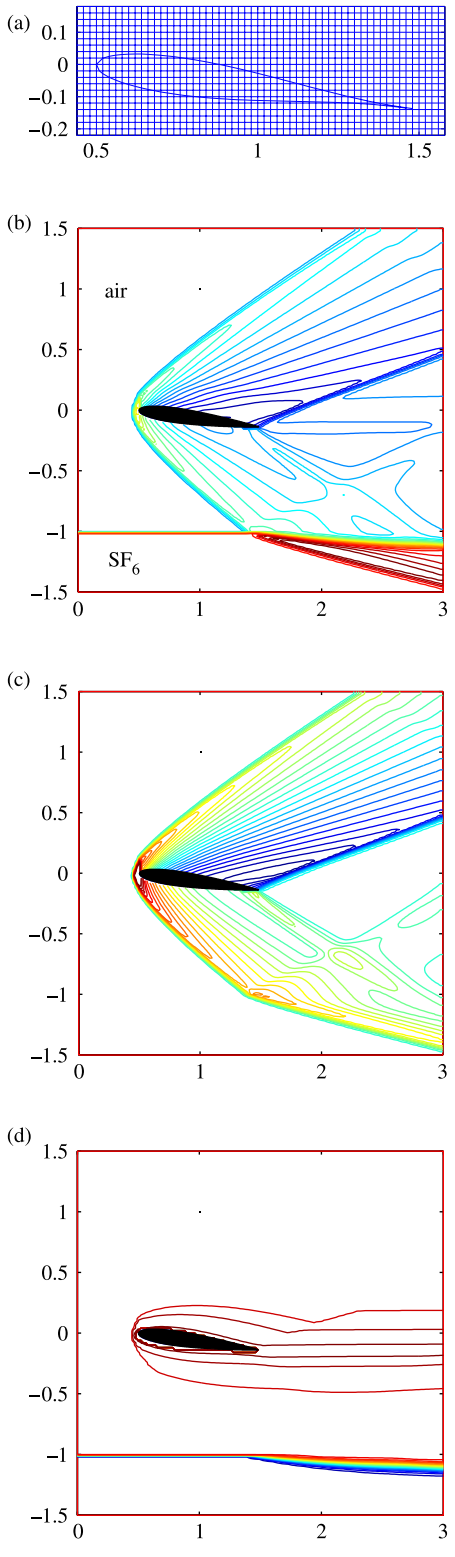

Figure 3. Eulerian computation of a Mach 2.2 steady flow past a NACA 0012 airfoil at angle of attack of 8° and a horizontal air–SF_6 material interface: (a) local Eulerian grid, (b) density contours, (c) pressure contours and (d) entropy contours.

Hafez *et al.* (1984). The pressure contours at the same time is shown in Figure 1(j).

The flow-adjusted grid in Figure 1(h) looks similar to those obtained by the grid re-distribution method in Eulerian computation. But there are differences: grid re-distribution requires generating another grid at every time step by solving an elliptic equation, and conservation properties in the interpolations of geometry and flow variables between the two grids must be ensured. These issues do not arise in our flow-adjusted grid approach because we need only one grid; the only modification is to specify the initial data for U at $\eta =$ constant by using the pressure gradient information.

In terms of computing time, most of the CPU time is used in solving the Riemann problems for the physical conservation laws, which is the same as in Eulerian computation. However, additional times are needed to solve Equation (9) and to update the geometric variables (A, B, L, M). These typically increase the CPU time by 5–10%. (This can also be reduced if in step 6 after the flow-adjusted grid is well established we put $U = V = 0$ so that there is no need to solve Equation (9) subsequently.) On the other hand, grid re-distribution in Eulerian computation by solving an elliptic equation increases CPU time. What is most important is that Eulerian computation always needs generating a body-fitted grid prior to flow computation, and this can be time-consuming.

6. Examples

Example 1 showing a transonic flow computation is already given above. Example 2 shows a sample computation for a Mach 2.2 steady flow over a NACA 0012 airfoil at an angle of attack of 8° and a horizontal air–SF_6 material. Figure 2(a)–(c) shows the flow-generated grids at different times, whereas Figure 2(d)–(h) shows the computed contours of density, pressure and entropy at those times. The interface between air and SF_6 was initially identified with a particular value of η and it remained so, because a contact line coincides with a coordinate line $\eta =$ constant in the unified coordinate system. The computation was straight forward, with no special treatment; yet the results are better than the corresponding Eulerian ones (Figure 3(a)–(d)), in particular the interface and the slip line behind the airfoil are resolved sharper.

7. Conclusion

This paper demonstrates that coordinate system plays an important role in computational fluid dynamics. It further

shows that the unified coordinate system is superior to the Eulerian and the Lagrangian system.

References

Hafez, M.M., Osher, S. and Whitlow, W., 1984. Improved finite different schemes for transonic potential calculations. *AIAA 22nd Aerospace Sciences Meeting*, AIAA paper 84-0092.

Hui, W.H. and Hu, J.J., 2006. Space-marching gridless computation of steady supersonic/hypersonic flow. *International Journal of Computational Fluid Dynamics*, 20, 55–59.

Hui, W.H. and Kudriakov, S., 2001. A unified coordinate system for solving the two-dimensional Euler equations. *Journal of Computational Physics*, 172, 235–260.

Hui, W.H., Li, P.Y. and Li, Z.W., 1999. A unified coordinate system for solving the two-dimensional Euler equations. *Journal of Computational Physics*, 153, 596–637.