AIAA JOURNAL
Vol. 45, No. 1, January 2007

Flux-Corrected Transport Algorithms for an Adaptively Refined Cartesian Mesh

Takanobu Ogawa*
Seikei University, Tokyo 180-8633, Japan
and
Elaine S. Oran†
Naval Research Laboratory, Washington, D.C. 20375

DOI: 10.2514/1.23587

The flux-corrected transport algorithm was implemented on an adaptively refined Cartesian mesh. The error at a mesh-refinement boundary, investigated using Fourier analysis, showed low accuracy when a wave moves from a coarse to a fine mesh. This was improved by using a linear interpolation between neighbor cells for evaluating convective fluxes. For extension to multidimensions, the flux-corrected transport limiter was modified to take into account information from all the cells facing a mesh-refinement boundary. A series of computations show that the present method simulates a shock wave passing through a mesh-refinement boundary without causing any unphysical waves. Methods for estimating the gain of adaptive mesh refinement, based on available computer memory or computational time, are discussed.

I. Introduction

FLUX-CORRECTED transport (FCT) is a nonlinear, monotone, conservative high-resolution algorithm for solving the continuity equation and sets of generalized continuity equations [1–4]. It is usually used as a basis for solving the Euler or Navier–Stokes equations, often with additional source terms for material or energy exchange. As such, FCT has been applied to a wide range of flows from laminar to turbulent, reacting to nonreacting, subsonic to supersonic.

FCT is a multistep method. First, the provisional solution is obtained with a convective and a diffusive flux. The convective flux is based on the high-order method such as the central-difference scheme. The diffusive flux must be dissipative enough to guarantee the positivity and monotonicity in the provisional solution. In the second step, an antidiffusive flux is calculated and used to remove excessive numerical diffusion in the provisional solution. This flux is limited to prevent it from generating new maxima or minima, or accentuating already existing extrema. Finally, the monotone high-resolution solution is obtained with the limited antidiffusive flux. This flux-limiting procedure is the key factor in the FCT algorithm.

Even though FCT is a high-resolution method, adaptive mesh-refinement (AMR) techniques are needed to solve equations representing problems with a wide range of spatial scales. For example, when we use a fixed nonadaptive mesh to capture steep gradients traveling over a computational domain, a large portion of the domain is covered with small computational cells where such resolution is not needed. In Cartesian-mesh AMR, a rectangular mesh is locally subdivided into finer computational cells where higher resolution is needed. The changes in mesh size are usually a factor of 2, which is considered a large change. Thus, there are two aspects to the AMR approach that have to be discussed: how to apply a numerical algorithm on this mesh, and how to ensure accuracy at a mesh-refinement boundary.

A numerical algorithm needs modification at a mesh-refinement boundary where a coarse mesh borders a fine mesh. One approach is to treat an adaptive Cartesian mesh as an unstructured mesh in the way the flux is calculated at the cell interface. This is widely used in compressible flow solvers that use a MUSCL-type variable reconstruction and limiters developed for unstructured grids to obtain physical variables and calculate the flux at a cell face [5]. Another approach is the "auxiliary node" concept originally proposed by Ferziger and Peric for unstructured grids [6–8]. In this approach, physical variables on a coarse mesh are determined by interpolation. The flux can be calculated using these variables in the same way as was done on the original structured mesh.

At the mesh-refinement boundary, typical AMR mesh sizes change by a factor of 2. This abrupt change in cell size affects the accuracy of a numerical scheme. Attention should be paid to monotonicity, as well as accuracy at a mesh-refinement boundary. It is known that shock-capturing methods generate spurious waves when a shock wave passes through a mesh-refinement boundary [9,10]. Berger et al. [9] explain that this spurious wave is caused by errors in shock-capturing methods, which contain $O(1)$ errors that do not cancel each other if the mesh size changes. This problem can be avoided by either capturing all shocks and contact discontinuities [10] or all shock waves strong enough to generate unacceptable oscillations [9] with the finest grid. Although this remedy essentially suppresses the oscillations, it is inefficient to track all the discontinuities including those moving away from the region of interest. This important problem has not been addressed in recent literature.

In this paper, we describe a method for extending FCT for use on an adaptively refined Cartesian mesh. Our focus is the compressible Euler equations. First, we describe the basic principles of the method and investigate the error of FCT analytically and numerically at a one-dimensional mesh-refinement boundary. This leads to a more accurate way to interpolate the fluxes and still maintain monotonicity. Then, the method is extended to multidimensions. At a mesh-refinement boundary, the physical variables are interpolated based on the auxiliary node concept, and the original one-dimensional FCT limiter is modified when a cell shares a border with more than one neighboring cell. We then show a series of examples that test monotonicity at the mesh-refinement boundary.

Presented as Paper 882 at the 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, 9–12 January 2006; received 2 March 2006; revision received 18 July 2006; accepted for publication 4 August 2006.

*Associate Professor, Department of Mechanical Engineering, 3-3-1 Kichijoji-Kitamachi, Musashino-shi. Member AIAA.

†Senior Scientist, Laboratory for Computational Physics and Fluid Dynamics, 4555 Overlook Avenue SW. Fellow AIAA.

The data structure used to organize the adaptive Cartesian mesh strongly affects efficiency and programming complexity. The tree data is a reasonable choice, because its hierarchical structure is suitable for a recursively refined mesh and its regularity in data alignment reduces programming complexity. In an ordinary tree data structure, traversing a tree is used to access neighbor cells instead of storing pointers to neighbor cells. Although memory overhead can be minimized with this method, the disadvantage of traversing a tree is the CPU overhead. In this work, we used the fully threaded tree (FTT) developed by Khokhlov [10]. In FTT, a neighbor cell is accessed by a pointer and traversing a tree is not necessary. Memory overhead for the pointers is small in FTT, because the pointer to a neighbor cell is shared with subdivided computational cells.

There is always a question about whether an adaptive method is really worth the effort compared to a fixed mesh, especially when we consider the added cost, complexity of programming and use, and developing complexity in certain flows that requires uniform mesh resolution. This can be addressed in several ways. Here we present quantitative estimates of the advantages of an adaptive mesh, and we also show how this applies to a two- and three-dimensional time-dependent calculation.

II. Background

The governing equations are the compressible Euler equations in Cartesian coordinates.

$$\frac{\partial}{\partial t}\int_V \rho \,\mathrm{d}V + \int_S \rho \boldsymbol{v}\,\mathrm{d}\boldsymbol{S} = 0 \tag{1}$$

$$\frac{\partial}{\partial t}\int_V \rho \boldsymbol{v}\,\mathrm{d}V + \int_S \rho \boldsymbol{v}\times\boldsymbol{v}\,\mathrm{d}\boldsymbol{S} = -\int_S p\boldsymbol{I}\,\mathrm{d}\boldsymbol{S} \tag{2}$$

$$\frac{\partial}{\partial t}\int_V E\,\mathrm{d}V + \int_S E\boldsymbol{v}\,\mathrm{d}\boldsymbol{S} = -\int_S p\boldsymbol{v}\,\mathrm{d}\boldsymbol{S} \tag{3}$$

where E is the total energy per volume and p is the pressure. For an ideal gas,

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho \boldsymbol{v}\cdot\boldsymbol{v} \tag{4}$$

where γ is the specific heat ratio. We have written them in integral form to discretize them in the finite-volume formulation.

A. Adaptive Cartesian Mesh and the Fully Threaded Tree

Figure 1a shows a sample adaptive Cartesian mesh. A computational cell is locally and isotropically subdivided where higher resolution is needed. The governing equations are discretized with a finite-volume method in which a flux is evaluated at a cell interface and the physical variables are defined at the center of cells. Although we used a square grid for this exposition, the algorithm described below can be applied to a rectangular mesh with any arbitrary aspect ratio.

The FTT has been used to organize the adaptive mesh [10]. Figure 1b shows the FTT data structure corresponding to the mesh in Fig. 1a. Each cell data consists of relevant physical variables and a pointer to its children cells, and each child cell has a pointer to its parent. This linkage between children and parent cells (indicated with solid arrows in Fig. 1b) forms a tree data structure. When a cell is subdivided to create children, eight cells (four cells in two dimensions) are generated and are usually in consecutive memory space. Thus, FTT handles subdivided sibling cells together as a basic unit of data, which is called an *Oct* in the original paper [10]. For fast access to neighbor cells, FTT gives an *Oct* six (four in two dimensions) pointers to its neighbor *Oct*s. These neighbor pointers do not point directly to the neighbor *Oct* itself, but to its parent cell. This is shown with broken arrows in Fig. 1b. If the neighbor pointer points directly to the neighbor *Oct*, the neighbor relation needs to be recalculated when that neighbor *Oct* is removed. The indirect access to the neighbor *Oct* via its parent cell eliminates this recalculation of the neighbor relation. Keeping neighbor pointers would decrease the memory efficiency of a tree data structure. In FTT, however, the neighbor pointers are shared by the sibling cells in the *Oct*, and the memory overhead per cell is reduced by a factor of 8 (four in two dimensions). Because of these properties, FTT can access neighbors quickly and uses memory efficiently.

We add two constraints on mesh refinement [10], as shown in Fig. 2. The difference in cell levels difference cannot be more than one between neighbor cells sharing a cell face and between diagonal neighbor cells. These constraints contribute to programming simplicity and lead to a gradual change in the mesh size, which reduces the truncation error in the interpolation described later.

B. Flux-Corrected Transport

FCT was designed to solve the one-dimensional continuity equation in conservative form,

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho v}{\partial x} = 0 \tag{5}$$

This equation is discretized on a one-dimensional finite-volume mesh, shown in Fig. 3. The suffix $i + 1/2$ denotes the cell face between the cell i and $i + 1$. $A_{i+1/2}$ is the cross-sectional area of the cell face $i + 1/2$ and is a constant A in this case. V_i is the volume of the cell i, and $V_i = A\Delta x_i$.

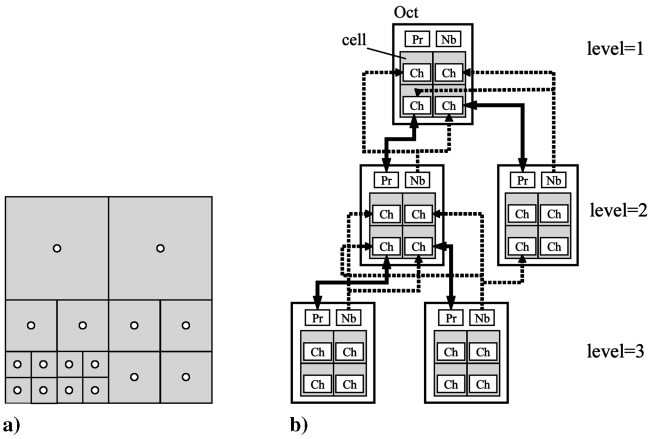


Fig. 1 a) Adaptively refined Cartesian mesh. Physical variables are defined at the center of a cell. b) FTT corresponding to the mesh a). *Oct* is a basic unit of data in FTT, and it contains subdivided sibling cells and pointers to organize the tree. Each cell has physical variables and a pointer *Ch* to its child *Oct*. *Pr* is a pointer to a parent cell of an *Oct*, and the linkage between a parent cell and a child *Oct* is indicated with a solid arrow. *Nb*s are pointers to neighbor cells and it points to the parent cell of the neighbor cells, as indicated with a broken arrow. The generation of a cell is denoted as *level*.

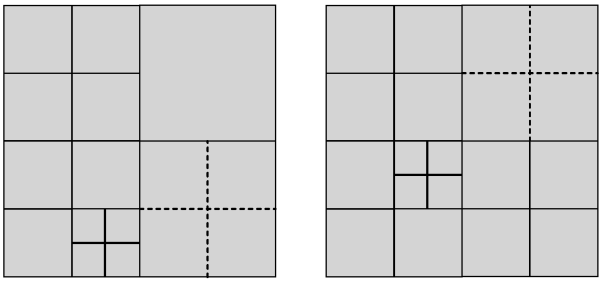


Fig. 2 Constraints on mesh refinement. More than one level of difference is not allowed a) between neighbor cells sharing a cell face and b) between diagonally neighboring cells. When mesh subdivision (indicated with thick lines) violates these constraints, the neighbor cell that is two levels coarser than the newly generated cells is split, as indicated by broken lines.

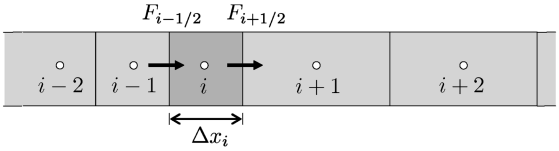


Fig. 3 Test mesh in one dimension.

The first FCT step is to find a higher-order convected solution ρ^T,

$$\rho_i^T V_i = \rho_i^0 V_i - (F_{i+1/2}^t - F_{i-1/2}^t) \tag{6}$$

where F^t is the convective flux defined as

$$F_{i+1/2}^t = \Delta t A_{i+1/2} \rho_{i+1/2}^0 v_{i+1/2}^0 \tag{7}$$

In the latest version of FCT, LCPFCT [11], the algebraic average is used to define the variable at the cell interface. This corresponds to the central-difference scheme,

$$\rho_{i+1/2} = \tfrac{1}{2}(\rho_i + \rho_{i+1}), \qquad v_{i+1/2} = \tfrac{1}{2}(v_i + v_{i+1}) \tag{8}$$

Next, we find the provisional low-order solution $\tilde{\rho}$ by adding a diffusive flux F^d to the convected solution ρ^T,

$$\tilde{\rho}_i V_i = \rho_i^T V_i + (F_{i+1/2}^d - F_{i-1/2}^d) \tag{9}$$

The diffusive flux F^d assures positivity and monotonicity of $\tilde{\rho}$, and is defined as

$$F_{i+1/2}^d = \nu_{i+1/2} V_{i+1/2} (\rho_{i+1}^0 - \rho_i^0) \tag{10}$$

where $V_{i+1/2} = (V_i + V_{i+1})/2$. Note that F^d is based on the initial state of the solution ρ^0, not on the convected solution ρ^T.

An antidiffusive flux is defined using the convected solution ρ^T,

$$F_{i+1/2}^a = \mu_{i+1/2} V_{i+1/2} (\rho_{i+1}^T - \rho_i^T) \tag{11}$$

The diffusion and the antidiffusion coefficient, ν and μ, respectively, are defined as

$$\nu_{i+1/2} = \tfrac{1}{6} + \tfrac{1}{3}\epsilon_{i+1/2}^2 \tag{12}$$

$$\mu_{i+1/2} = \tfrac{1}{6} - \tfrac{1}{6}\epsilon_{i+1/2}^2 \tag{13}$$

where ϵ is the Courant number at the cell face,

$$\epsilon_{i+1/2} = \frac{1}{2}\left(\frac{1}{V_i} + \frac{1}{V_{i+1}}\right) v_{i+1/2} A_{i+1/2} \Delta t \tag{14}$$

These coefficients reduce the relative phase error in convection on a uniform mesh to fourth order [4] and have been used in LCPFCT [11].

The antidiffusive flux is limited so that it does not generate new unphysical maxima or minima, and not accentuate existing extrema. The following flux limiter gives the corrected antidiffusive flux $F^{a'}$ that meets this monotonicity requirement:

$$F_{i+1/2}^{a'} = S_{i+1/2} \max[0, \min\{|F_{i+1/2}^a|, S_{i+1/2} V_{i+1} \Delta\tilde{\rho}_{i+3/2}, S_{i+1/2} V_i \Delta\tilde{\rho}_{i-1/2}\}] \tag{15}$$

where $\Delta\tilde{\rho}_{i+1/2}$ is the difference in the provisional solution $\tilde{\rho}$ between the cell i and the cell $i+1$,

$$\Delta\tilde{\rho}_{i+1/2} = \tilde{\rho}_{i+1} - \tilde{\rho}_i \tag{16}$$

Finally, the high-order solution at the next time step is obtained with the corrected antidiffusive flux,

$$\rho_i^n = \tilde{\rho}_i - \frac{1}{V_i}(F_{i+1/2}^{a'} - F_{i-1/2}^{a'}) \tag{17}$$

III. FCT at Mesh-Refinement Boundaries

The behavior of FCT at mesh-refinement boundaries is investigated by focusing on one-dimensional problems and deriving the error at the boundary. The effect of the interpolation at a cell interface is discussed. Then, we describe extension to multi-dimensions on AMR and apply it to a two-dimensional mesh-refinement boundary.

A. Errors of FCT at Mesh-Refinement Boundaries

Consider the one-dimensional mesh-refinement boundary shown in Fig. 4. A uniform coarse mesh is adjacent to a uniform fine mesh at the cell i. We limit mesh refinement to one level for simplicity. The mesh size of the coarse cell can be written as $s\Delta x$, where Δx is the cell size of the fine mesh and s is a stretch ratio. In this study, s is restricted to be two. In a nonuniform mesh, the truncation error depends on the cell position. Here we focus on the error at the cell i which faces the mesh-refinement boundary.

The actual equation solved by the finite-volume descretization of FCT can be derived by expanding ρ in Eqs. (6–17) in terms of a Taylor series [12]. To see the effect of the mesh nonuniformity on truncation errors, we assume that the velocity is constant and uniform, and that the antidiffusive flux is turned off. Then, we obtain the following modified equation at the cell i in Fig. 4 for $s = 2$:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho U}{\partial x} = \overbrace{-\frac{1}{4}U\frac{\partial \rho}{\partial x} - \frac{5}{16}U\frac{\partial^2 \rho}{\partial x^2}\Delta x}^{\text{from the convective flux}} + \underbrace{\left(\frac{9}{4}\nu_{i+1/2} - \nu_{i-1/2}\right)\frac{\partial \rho}{\partial x}\Delta x}_{\text{from the diffusive flux}} + O(\Delta t, \Delta x^2) \tag{18}$$

The right side of Eq. (18) shows the lowest-order errors affected by the mesh nonuniformity. The first two terms on the right side are generated from the convective flux of FCT. The first term of them is a zeroth order error, which makes the scheme inconsistent. The second term of them gives a first-order numerical dissipation or antidissipation depending on the direction of the wave propagation,

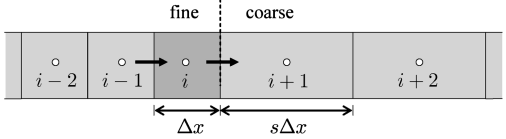


Fig. 4 A one-dimensional mesh-refinement boundary.

that is, the sign of U. The diffusive flux, which gives no phase shift on a uniform mesh, introduces the third term, a first-order dispersive error.

The inconsistency results from the simple arithmetic average Eq. (8). This form has been used in LCPFCT for a wide range of applications on many types of nonuniform meshes. In the literature, the general problem of loss of accuracy due to mesh stretching has been discussed in terms of the magnitude of a truncation error. Here we investigate the source of the error in more detail. By using Fourier analysis, we show how and why the error appears in a numerical solution on a nonuniform mesh. For the mesh-refinement boundary of Fig. 4, the analysis can be carried out using the general Fourier expansion of a continuous function, as was done by Ikeda et al. [13]. A function can be written as

$$\rho_j^m = \hat{\rho}^m e^{ik_m x_j}$$

at a discrete location with a Fourier series. Substituting this relation into Eqs. (6–17), we obtain the amplification factor $|G|$ and the phase angle ϕ for one time step, defined as

$$|G| = \left|\frac{\rho^n}{\rho^0}\right| \tag{19}$$

$$\phi = \tan^{-1}\frac{\mathrm{Im}(\rho^n/\rho^0)}{\mathrm{Re}(\rho^n/\rho^0)} \tag{20}$$

Figures 5–7 show the results of Fourier analysis for the convected solution Eq. (6), the low-order solution Eq. (9), and the final solution of FCT Eq. (17), respectively. In this analysis, the FCT limiter is turned off everywhere. In the low-order solution, the convected solution ρ^T is replaced with the initial solution ρ^0 to show the effect of the diffusive flux Eq. (10). Thus, the low-order solution used in Fourier analysis is

$$\tilde{\rho}_i V_i = \rho_i^0 V_i + (F_{i+1/2}^d - F_{i-1/2}^d) \tag{21}$$

The amplification factor and the phase angle at the cell i are shown in these figures as a function of the nondimensional wave number $\beta = k\Delta x$. The velocity is uniform and constant, and the Courant number is $\epsilon = \pm 1/2$, where $\epsilon > 0$ indicates the mesh stretches in the direction of convection, and for $\epsilon < 0$ the mesh contracts. We denote $\epsilon > 0$ values as ϵ^s indicating that the mesh stretches, and $\epsilon < 0$ values as ϵ^c indicating that the mesh contracts. The phase angle ϕ is compared with the exact phase angle ϕ_e, which is given as $-\beta\epsilon$ in this case. Values of $|G|$ and ϕ for a uniform mesh ($s = 1$) are shown as dashed lines.

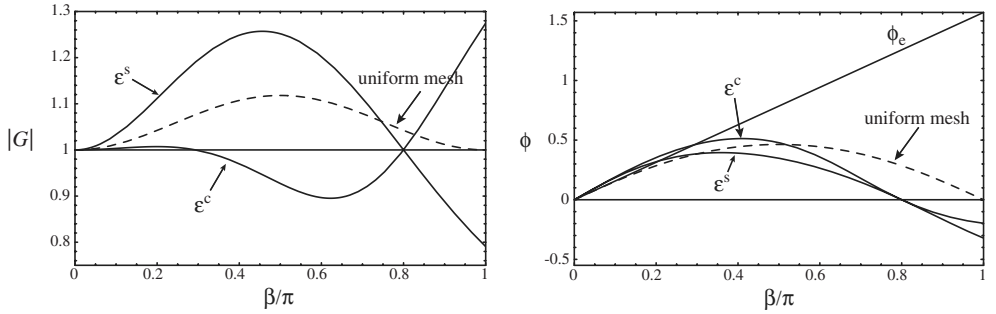


Fig. 5 Amplification factor $|G|$ and phase angle ϕ of the convective solution, Eq. (6), at cell i for $s = 2$ and $\epsilon = \pm 1/2$. A mesh stretches for ϵ^s ($\epsilon > 0$) and contracts for ϵ^c ($\epsilon < 0$).

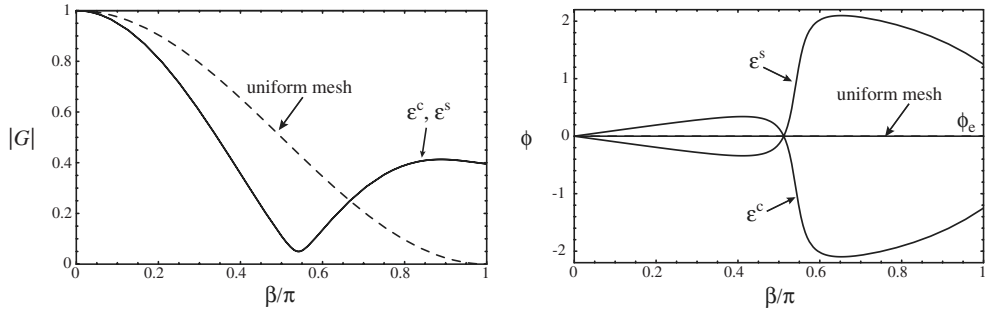


Fig. 6 Amplification factor $|G|$ and phase angle ϕ of the low-order solution, Eq. (21), at cell i for $s = 2$ and $\epsilon = \pm 1/2$. A mesh stretches for ϵ^s ($\epsilon > 0$) and contracts for ϵ^c ($\epsilon < 0$).

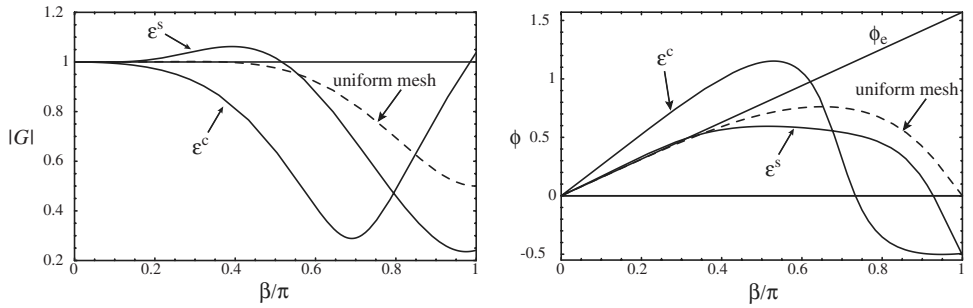


Fig. 7 Amplification factor $|G|$ and phase angle ϕ of FCT Eq. (17) at cell i for $s = 2$ and $\epsilon = \pm 1/2$. For ϵ^s ($\epsilon > 0$), a mesh stretches, and for ϵ^c ($\epsilon < 0$), a mesh contracts. The FCT limiter Eq. (15) is turned off.

In Fig. 5, the amplification factor for the convected solution shows that the convective flux introduces the damping error at around $\beta/\pi = 0.6$ for a contracting mesh ϵ^c, and the antidamping error $|G| > 1$ at around $\beta/\pi = 0.5$ for a stretching mesh ϵ^s. This is caused by the second-derivative term from the convective flux in Eq. (18), which acts as numerical dissipation or antidissipation depending on velocity direction. Figure 5 also shows that the convective flux gives a phase delay for a high wave number for both ϵ^c and ϵ^s.

On a uniform mesh, the diffusive flux has the minimum amplification factor at the highest wave number and gives no phase shift. This is affected by the mesh nonuniformity, as shown in Fig. 6. The diffusive flux at the mesh-refinement boundary gives a strong damping error at around $\beta/\pi = 0.55$ for both ϵ^c and ϵ^s, and generates a large phase gain for ϵ^s and delay for ϵ^c at the wave number $\beta/\pi > 0.5$.

The combination of the convective flux and the diffusive flux reduces the error due to the mesh nonuniformity for ϵ^s, but intensifies it for ϵ^c. This is demonstrated by the result of Fourier analysis for the final solution of FCT shown in Fig. 7. For ϵ^s, the damping error of the diffusive flux reduces the excessive amplification factor of the convective flux at an intermediate wave number, and the phase delay error of the convective flux is weakened by the phase gain error of the diffusive flux at a high wave number. Thus, both the amplification factor and the phase angle for ϵ^s shown in Fig. 7 become close to those of a uniform mesh. For ϵ^c, however, both the damping error and the phase delay error of the convective flux are strengthened by the diffusive flux. As a result, the amplification factor has a small minimum peak around $\beta/\pi = 0.7$, and the phase angle has a large delay error for $\beta/\pi > 0.6$. This leads to the large deviations from the result for a uniform mesh.

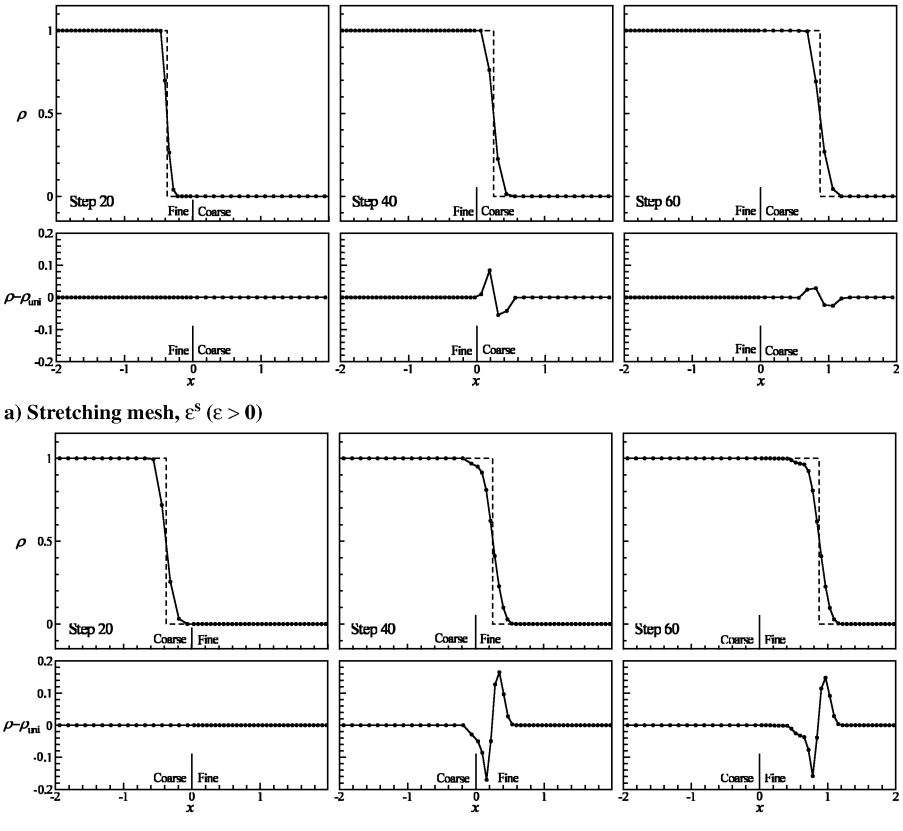
Computational tests were performed to examine the effect of the mesh nonuniformity at the mesh-refinement boundary. First, we simulated a scalar wave moving through the mesh-refinement boundary by solving the equation

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho v}{\partial x} = 0$$

where the velocity v is unity and constant, and the CFL number is $1/2$. The wave is a step function initially located at $x = -1$, and it moves through the mesh-refinement boundary at $x = 0$. Figure 8a shows the time series of the wave moving from the fine mesh to the coarse mesh ϵ^s ($\epsilon > 0$), and Fig. 8b shows the wave moving from the coarse mesh to the fine mesh ϵ^c ($\epsilon < 0$). The deviation from the solution with a uniform mesh ($\rho - \rho_{\text{uni}}$) is also shown.

For ϵ^s, for which the error is close to that of a uniform mesh, the wave passes through the mesh-refinement boundary without deforming. The deviation from the uniform mesh shows that the wave is slightly sharper than that of the uniform coarse mesh, because it is solved with the fine mesh before entering the coarse mesh. On the other hand, for ϵ^c, the solution becomes more diffusive than that of the uniform fine mesh, because the wave moves from the coarse mesh. In addition to the numerical dissipation introduced by the coarse mesh, the large damping error at the mesh-refinement boundary reduces the wave amplitude when the wave front passes the mesh-refinement boundary. This leads to the wave deformation, which can be observed as a hollow behind the wave front at about $x = -0.1$ of step 40 and $x = 0.6$ of step 60 in Fig. 8b.

The results for a propagating shock wave are presented in Fig. 9. The shock wave is initially located at $x = -10$ and propagates at Mach 10. The CFL number is $1/2$. For ϵ^s, there is no wave deformation or monotonicity violation. The self-steepening property of a shock wave reduces the diffusion introduced in the coarse mesh, and the deviation from the result of the uniform mesh is smaller than the scalar wave case. For ϵ^c, however, a large error is introduced at the mesh-refinement boundary, as suggested by the error analysis described above. The wave front is strongly damped when the shock moves over the mesh-refinement boundary at step 25. This leads to



a) Stretching mesh, ϵ^s ($\epsilon > 0$)

b) Contracting mesh, ϵ^c ($\epsilon < 0$)

Fig. 8 Computational results for a one-dimensional scalar wave moving through a mesh-refinement boundary at $x = 0$. ρ is the solution, and $\rho - \rho_{\text{uni}}$ is the deviation from the result of a uniform mesh. The CFL number is $1/2$; a) from a fine mesh to a coarse mesh, ϵ^s ($\epsilon > 0$); b) from a coarse mesh to a fine mesh, ϵ^c ($\epsilon < 0$). Dashed lines show the exact solution.

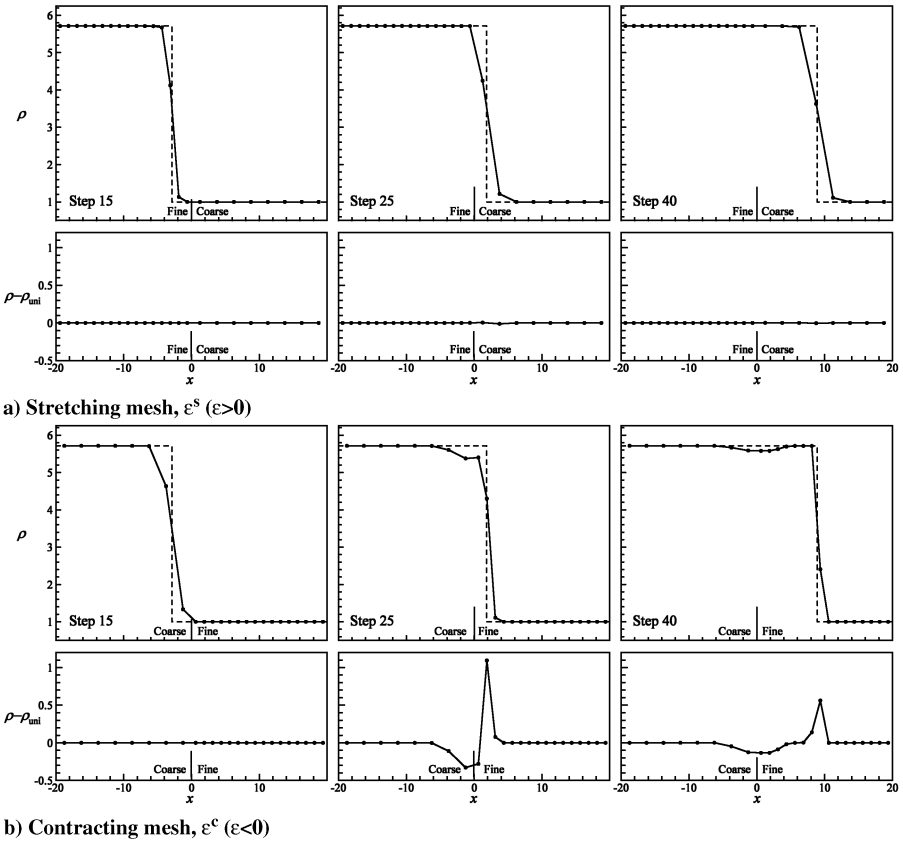


Fig. 9 Computational results for a one-dimensional shock wave moving at $M_s = 10.$ through a mesh-refinement boundary at $x = 0.$ ρ is the density, and $\rho - \rho_{\text{uni}}$ is the deviation from the result of a uniform mesh. The CFL number is 1/2; a) stretching mesh, ϵ^s ($\epsilon > 0$); b) contracting mesh, ϵ^c ($\epsilon < 0$). Dashed lines show the exact solution.

the wave deformation and generates a spurious oscillation behind the shock.

Fourier analysis in Fig. 7 shows that there is an unstable mode for ϵ^s at about $\beta/\pi = 0.4$. The numerical results, however, show no numerical instability. Although the FCT limiter Eq. (15) is turned off in the Fourier analysis, it is used in the actual computation and may suppress the instability at the mesh-refinement boundary.

B. Improvement of FCT Flux Calculation

The major error in the convective flux Eq. (7) at the mesh-refinement boundary comes from the algebraic average Eq. (8). This error can be reduced by introducing a linear interpolation of the cell face values,

$$\rho^0_{i+1/2} = \frac{1}{1+s}(s\rho^0_i + \rho^0_{i+1}), \qquad v_{i+1/2} = \frac{1}{1+s}(sv_i + v_{i+1}) \tag{22}$$

The modified equation with this linearly interpolated convective flux at the cell i in Fig. 4 becomes

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho U}{\partial x} = \underbrace{-\frac{1}{8}U\frac{\partial^2 \rho}{\partial x^2}\Delta x}_{\text{from the convective flux}} + \underbrace{\left(\frac{9}{4}\nu_{i+1/2} - \nu_{i-1/2}\right)\frac{\partial \rho}{\partial x}\Delta x}_{\text{from the diffusive flux}} + O(\Delta t, \Delta x^3) \tag{23}$$

Introducing linear interpolation removes the zeroth order error in Eq. (18), and, thus, the scheme becomes consistent. The numerical dissipation term from the convective flux is also reduced by a factor of more than two. Fourier analysis of the convective flux with the linear interpolation presented in Fig. 10 shows this improvement in accuracy. The amplification factors for both ϵ_s and ϵ_c become close to that of the uniform mesh.

We now check the positivity condition for the FCT with the linearly interpolated convective flux. The low-order solution Eq. (9)

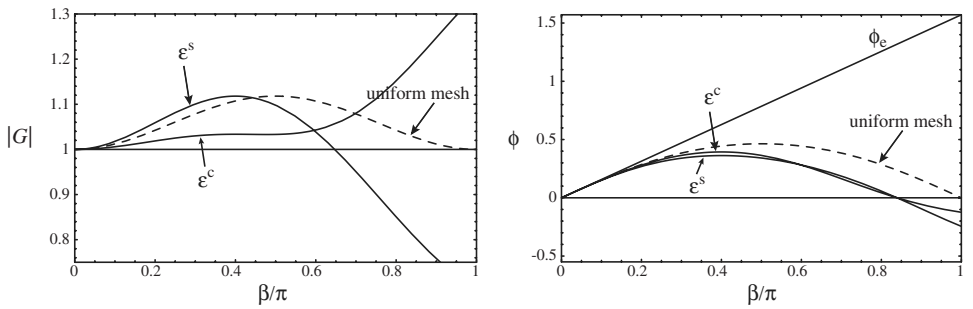


Fig. 10 Amplification factor $|G|$ and phase angle ϕ of the convected solution Eq. (6) with the linearly interpolated convective flux at cell i for $s = 2$ and $\epsilon = \pm 1/2$. A mesh stretches for ϵ^s ($\epsilon > 0$) and contracts for ϵ^c ($\epsilon < 0$).

with the linear interpolation Eq. (22) can be written as

$$\tilde{\rho}_i = \rho_i^0 - (F^t_{i+1/2} - F^t_{i-1/2}) + (F^d_{i+1/2} - F^d_{i-1/2}) = a_i\rho^0_{i-1} + b_i\rho^0_i + c_i\rho^0_{i+1} \tag{24}$$

where

$$a_i = \nu_{i-1/2} + \tfrac{1}{2}\epsilon_{i-1/2},$$
$$b_i = 1 - \tfrac{2}{3}\epsilon_{i+1/2} + \tfrac{1}{2}\epsilon_{i-1/2} - \tfrac{3}{2}\nu_{i+1/2} - \nu_{i-1/2}, \tag{25}$$
$$c_i = \tfrac{3}{2}(\nu_{i+1/2} - \tfrac{2}{9}\epsilon_{i+1/2})$$

Positivity requires that a_i, b_i, and c_i be nonnegative [4] to assure positive $\tilde{\rho}_i$. Equation (13) meets this requirement, and therefore the diffusive flux Eq. (10) of the original FCT can be used with the linearly interpolated convective flux.

A higher-order interpolation can be used to construct the cell face values in Eq. (22). In that case, the diffusive flux should be also modified so that the positivity condition is satisfied. In this study, we use a linear interpolation to improve the accuracy while keeping the original diffusive flux.

Figure 11 shows the result of Fourier analysis for the final solution of FCT with the linearly interpolated convective flux. The accuracy has now improved for both ϵ^s and ϵ^c. For ϵ^c, the minimum peak of the amplification factor at around $\beta/\pi = 0.7$ increases, and thus the damping error is reduced. For ϵ^s, introducing linear interpolation removes the unstable mode that the original FCT has at about $\beta/\pi = 0.4$, though the scheme becomes slightly dissipative over a wide range of a wave number. An unstable mode appears for ϵ^c at the highest wave number. This unstable mode, however, has no effect on an actual computation for ϵ^c, because a wave enters the mesh-refinement boundary from a coarse mesh, where no wave exists at $\beta/\pi > 0.5$. The phase angle is also improved for both ϵ^s and ϵ^c. Although there is a large phase delay error at $\beta/\pi > 0.5$ for ϵ^c, this error has little effect on the solution for the same reason mentioned above.

The accuracy improvement from the linear interpolation is examined with the same test problems as in the previous section. Results for the scalar wave and the shock wave problems are shown in Figs. 12 and 13, respectively. For ϵ^c, the results of both the scalar and the shock problem are similar to those by the original FCT shown in Figs. 8a and 9a, and no wave deformation or spurious wave is observed. Though FCT with a linear interpolation becomes slightly diffusive at the mesh-refinement boundary, it does not affect the computational result.

The accuracy is remarkably improved for ϵ^c. In the scalar wave problem, the wave front is not deformed at the mesh-refinement boundary because of the reduced damping error, as shown in Fig. 12b. In the shock problem, there is greater improvement in accuracy. The shock moves through the mesh-refinement boundary without generating the spurious wave observed in Fig. 9b, and the deviation ($\rho - \rho_{\text{uni}}$) is enormously reduced. For both ϵ^s and ϵ^c, the shock front can be captured within few mesh points after it has passed the mesh-refinement boundary.

C. Multidimensional Mesh-Refinement Boundaries

1. Direction-Splitting FCT

Here we extend FCT on AMR to multidimensions. Figure 14 shows a sample two-dimensional mesh. The indices j and k denote a cell next to cell i in the x and y directions, respectively. The symbol $\bar{j}$ denotes the cells at the upper side of cell i, and $\underline{j}$ is at the lower side.

Direction splitting is straightforward [14]. The procedures from Eqs. (6–17) are applied, in turn, to convective derivatives of multidimensional transport equations in each spatial direction. At a mesh-refinement boundary, however, two additional procedures are needed.

First, the FCT limiter needs to be modified where two cells (four cells in three dimensions) share a cell face with a larger cell. At the mesh-refinement boundary shown in Fig. 15, the FCT limiter, Eq. (15), can be written as

$$F^{a'}_{i\underline{j}} = S_{i\underline{j}} \max[0, \min(|F^a_{i\underline{j}}|, S_{i\underline{j}} V_i \Delta\tilde{\rho}_{\bar{j}i}, S_{i\underline{j}} V_{\underline{j}} \Delta\tilde{\rho}_{\underline{j}\underline{j}'})] \tag{26}$$

where $\underline{j}'$ indicates the lower side of the cell $\underline{j}$, and $\Delta\tilde{\rho}_{\bar{j}i} = \tilde{\rho}_{\bar{j}} - \tilde{\rho}_i$. The difference in the physical variables between the neighboring cells, $\Delta\tilde{\rho}_{\underline{j}\underline{j}'}$, is defined as $\tilde{\rho}_{\underline{j}} - \tilde{\rho}_{\underline{j}'_1}$ or $\tilde{\rho}_{\underline{j}} - \tilde{\rho}_{\underline{j}'_2}$ for the mesh in Fig. 15. To assure monotonicity of the solution, we must take the minimum of these possible $\Delta\tilde{\rho}_{\underline{j}\underline{j}'}$s. Thus, we define $\Delta\tilde{\rho}_{\underline{j}\underline{j}'}$ as

$$\Delta\tilde{\rho}_{\underline{j}\underline{j}'} = \min(\tilde{\rho}_{\underline{j}} - \tilde{\rho}_{\underline{j}'_1}, \ldots, \tilde{\rho}_{\underline{j}} - \tilde{\rho}_{\underline{j}'_m}) \tag{27}$$

where $m = 2^d$, and d is the number of dimensions.

Second, we need an interpolation at a mesh-refinement boundary to construct fluxes. In this study, the auxiliary node concept is applied. For example, the cell $\underline{j}$ is finer, as shown in Fig. 16; the physical values are linearly interpolated at the auxiliary node i' using those at the cell i and the cell $\underline{k}$. Then, the flux $F_{i'\underline{j}}$ can be calculated with the cell i' and $\underline{j}$ in the same way as is done on a one-dimensional mesh.

2. Two-Dimensional Tests

Now consider a Mach 10 shock propagating obliquely through a two-dimensional mesh-refinement boundary. For a similar problem, Berger et al. [9] observed a violation of monotonicity using their numerical method. Figure 17 shows the initial conditions and the computed results for a shock moving both ways, from a coarse to a fine mesh and from a fine to a coarse mesh. Contours of density along the fine mesh next to the mesh-refinement boundary show that, in both cases, the shock is captured by few mesh points without any significant unphysical oscillations. Thus, even in a multidimensional problem, FCT with linear interpolation for convected flux can resolve a sharp discontinuity while ensuring monotonicity at a mesh-refinement boundary.

It might seem unlikely that a situation such as that shown in Fig. 17 would actually occur if the mesh adaptation were done properly. Nevertheless, even with a good mesh-adaptation technique, it can happen. For example, suppose a shock is partially weakened by an impinging expansion wave, then there might not be a fine mesh on the weakened portion of the shock unless some threshold in an error

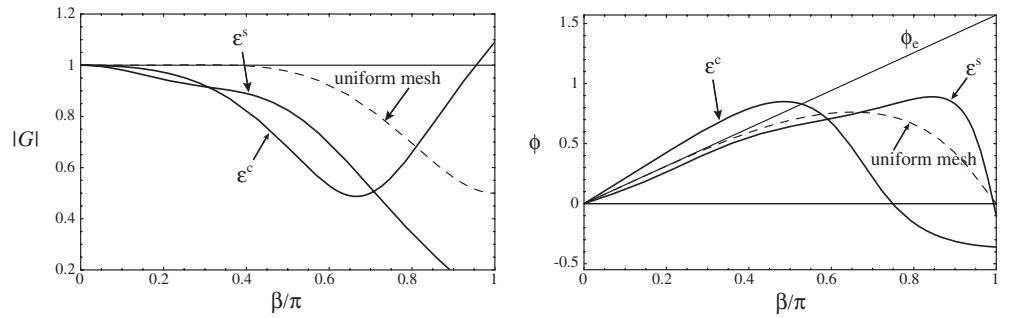
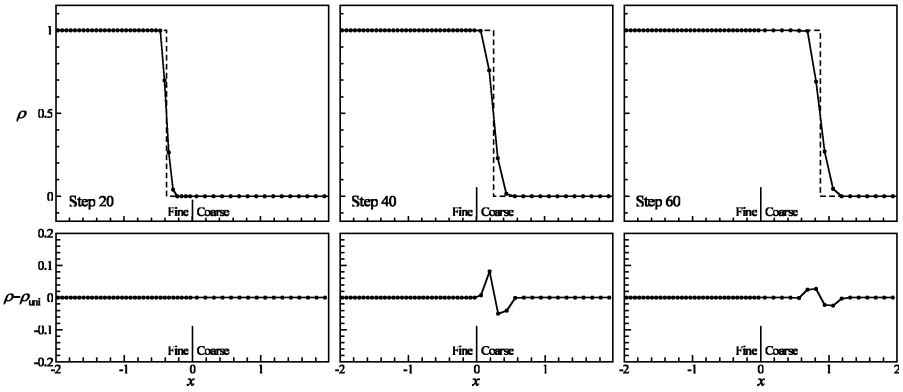
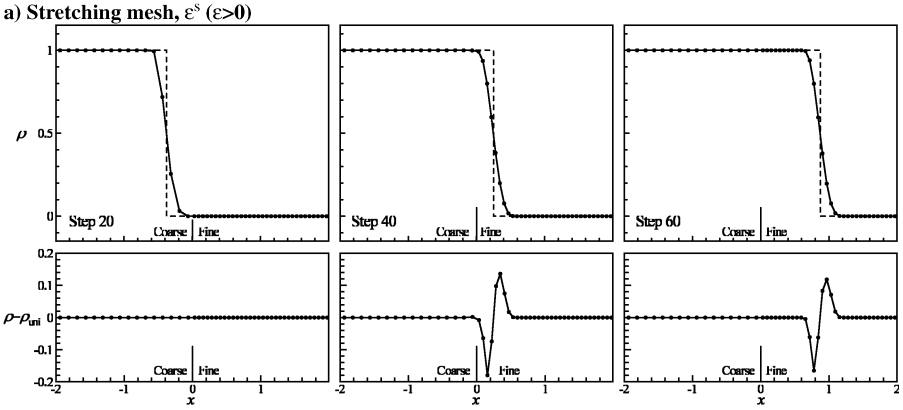


Fig. 11 Amplification factor $|G|$ and phase angle ϕ of FCT Eq. (17) at cell i for $s = 2$ and $\epsilon = 1/2$. Linear interpolation is used for the convective flux Eq. (7). A mesh stretches for ϵ^s ($\epsilon > 0$) and contracts for ϵ^c ($\epsilon < 0$). The FCT limiter Eq. (15) is turned off.

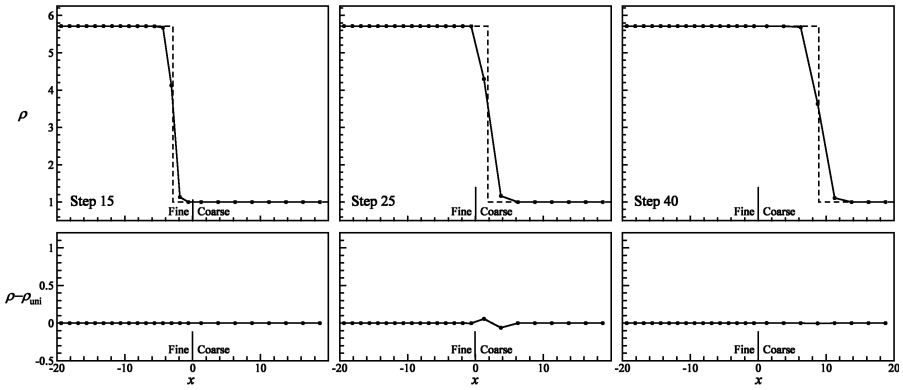


a) Stretching mesh, ε^s ($\varepsilon>0$)

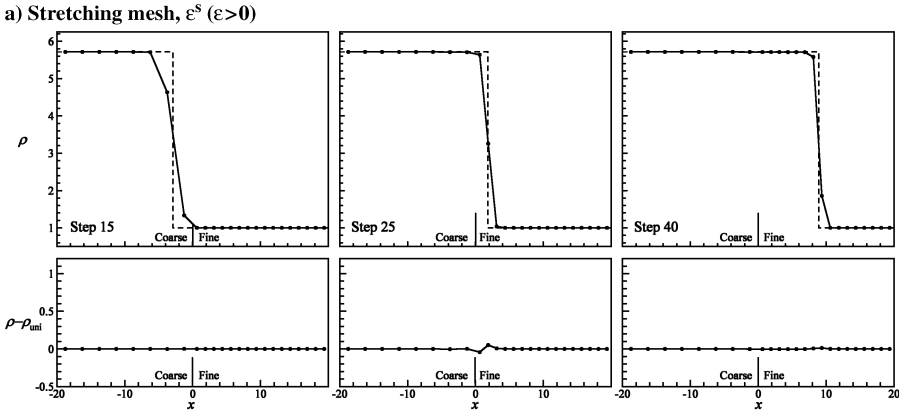


b) Contracting mesh, ε^c ($\varepsilon<0$)

Fig. 12 Computational results for a one-dimensional scalar wave moving through a mesh-refinement boundary at $x = 0$. Linear interpolation is used for the convective flux. ρ is the solution, and $\rho - \rho_{\text{uni}}$ is the deviation from the result of a uniform mesh. The CFL number is 1/2; a) stretching mesh, ϵ^s ($\epsilon > 0$); b) contracting mesh, ϵ^c ($\epsilon < 0$). Dashed lines show the exact solution.



a) Stretching mesh, ε^s ($\varepsilon>0$)



b) Contracting mesh, ε^c ($\varepsilon<0$)

Fig. 13 Computational results for a one-dimensional shock wave moving at $M_s = 10$ through a mesh-refinement boundary at $x = 0$. Linear interpolation is used for the convective flux. ρ is the density, and $\rho - \rho_{\text{uni}}$ is the deviation from the result of a uniform mesh. The CFL number is 1/2; a) stretching mesh, ϵ^s ($\epsilon > 0$); b) contracting mesh, ϵ^c ($\epsilon < 0$).

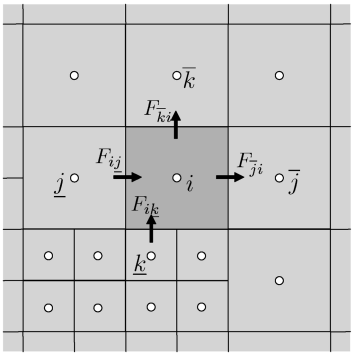


Fig. 14 Two-dimensional adaptive Cartesian mesh.

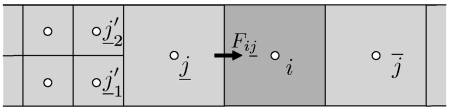


Fig. 15 A mesh-refinement boundary in two dimensions.

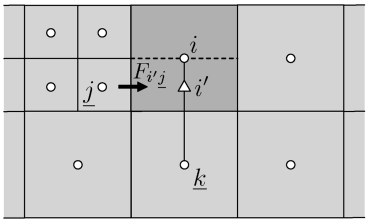


Fig. 16 Interpolation at a mesh-refinement boundary.

estimator is exceeded. This could lead to a problem such as that shown in Fig. 17.

IV. Quantitative Estimates of the Advantages of the Adaptive Mesh Refinement

In this section, we present a quantitative estimate of the benefits of using an adaptive mesh. The benefit is evaluated in terms of the *gain*, defined here as the ratio of the finest spatial resolution on an adaptive mesh to what would be needed in a uniform mesh. The "finest spatial resolution" is defined as the smallest mesh size available *when the same amount of a computer memory* is used for both an adaptive mesh and a uniform mesh, and *when the same CPU time* is used for both meshes. Here we derive an expression for the gain of using adaptive mesh refinement and evaluate it for sample two- and three-dimensional computations.

A. Parameters for an Adaptive-Mesh Computation

The parameters for an adaptive-mesh computation are denoted with the subscript a. The maximum cell level l for mesh refinement is usually a user-prescribed value and is constant. The size of the finest cell used on the adaptive mesh δX_a is

$$\delta X_a = \frac{L}{2^l} \tag{28}$$

where L is the size of a computational domain. Figure 18a shows a sample adaptive mesh with the finest cells of size δX_a. The total number of cells in the adaptive mesh N_a is time dependent when mesh adaptation takes place dynamically.

If the computational domain is discretized only with the finest cells, like the uniform mesh shown in Fig. 18b, the total number of uniform cells N_A, is

$$N_A = \left(\frac{L}{\delta X_a}\right)^d \tag{29}$$

where d is the number of dimensions. This value is constant and the maximum number of cells for the cell size δX_a. The value of N_a is usually smaller than N_A. The ratio of N_a to N_A, R, measures the extent to which the finest cells are used in the adaptive mesh,

$$R = \frac{N_a}{N_A} (\le 1) \tag{30}$$

B. Estimation of Gain for Fixed Memory Size

Now we determine the finest mesh size available on a uniform mesh for the same amount of a computer memory as the adaptive-mesh computation uses. Figure 18c illustrates such a uniform mesh. If the adaptive-mesh and uniform-mesh computations use the same amount of computer memory, the total number of cells on the uniform mesh is

$$N_m = WN_a \tag{31}$$

where N_m is the number of cells in the uniform mesh, and W is the memory overhead of the adaptive-mesh solver compared to that of the uniform-mesh solver. The cell size for this uniform mesh is

$$\delta X_m = \frac{L}{N_m^{1/d}} \tag{32}$$

The ratio of the spatial resolution δX_m to δX_a is a measure of the amount of high spatial resolution that can be obtained using the adaptive-mesh solver compared to the uniform mesh. We denote this value of gain as G_m, which can be written as

$$G_m = \frac{\delta X_m}{\delta X_a} = \frac{1}{W^{1/d}} \left(\frac{N_A}{N_a}\right)^{\frac{1}{d}} = (WR)^{-\frac{1}{d}} \tag{33}$$

This gain depends on R and the memory overhead of the adaptive-mesh solver. The memory overhead degrades the gain by a factor of $W^{1/d}$.

C. Estimation of Gain for Fixed CPU Time

Next, we assume that the same amount of CPU time is available for the adaptive-mesh and the uniform-mesh computations. Then the total CPU time ΔT for a complete computation can be written as

$$\Delta T = c_a N_a s_a = c_c N_c s_c \tag{34}$$

The subscript c is a parameter for the uniform-mesh computation, and the variable c denotes the CPU time per step per cell. It can be expressed as

$$c_a = W' c_c \tag{35}$$

where W' is the CPU time overhead of the adaptive-mesh solver.

The quantities s_a and s_c in Eq. (34) are some number of time steps for the adaptive-mesh and the uniform-mesh computations, respectively, and are defined as

$$s_a = \frac{\Delta T}{\delta t_a}, \qquad s_c = \frac{\Delta T}{\delta t_c} \tag{36}$$

Using Eqs. (34–36), we obtain the following relation:

$$W' \frac{N_a}{\delta t_a} = \frac{N_c}{\delta t_c} \tag{37}$$

If the same CFL number is used for both computations,

$$\frac{\delta t_a}{\delta X_a} = \frac{\delta t_c}{\delta X_c} \tag{38}$$

Using Eqs. (37) and (38), we obtain

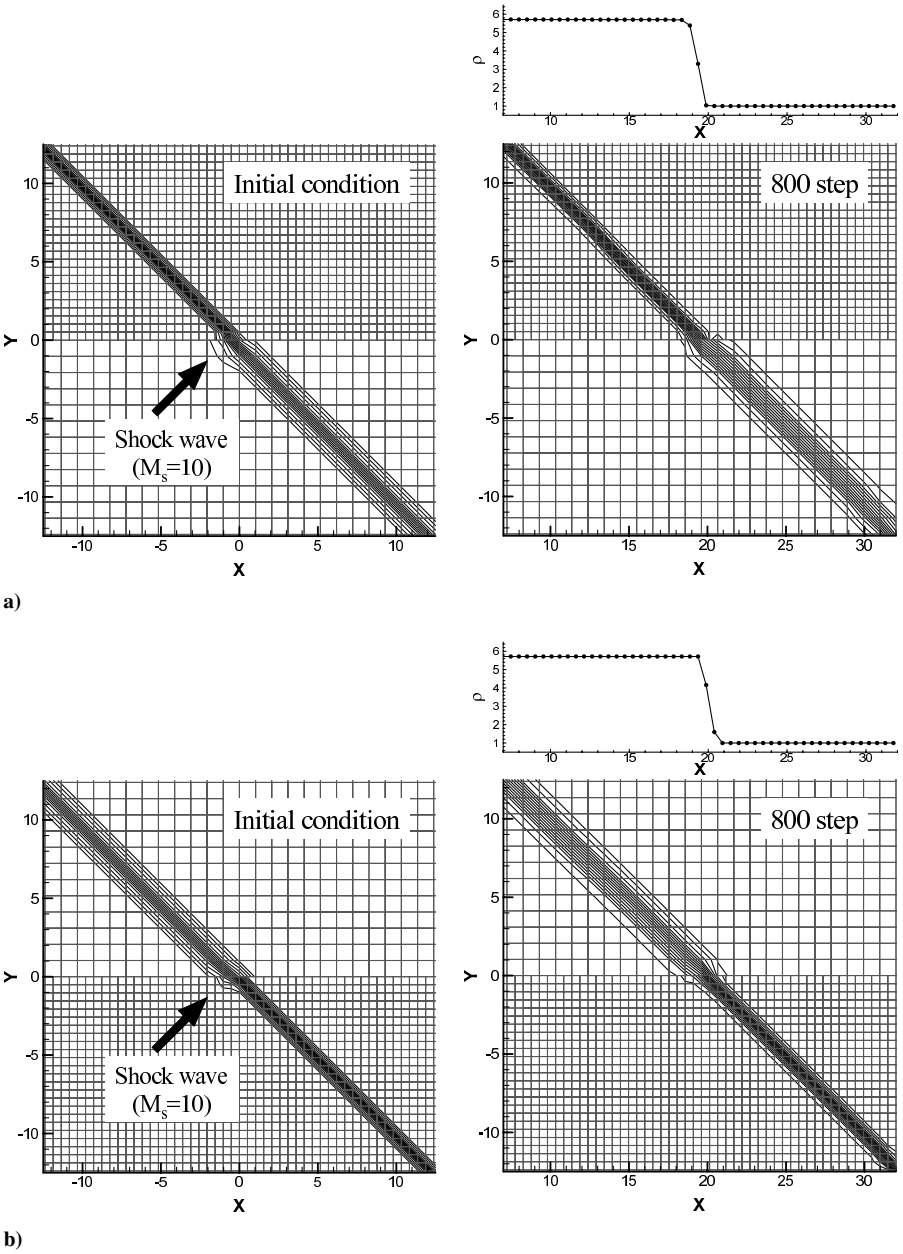


Fig. 17 Density contours of a two-dimensional shock wave passing through the mesh-refinement boundary a) from the coarse mesh to the fine mesh and b) from the fine mesh to the coarse mesh. The CFL number is 0.25. Density profiles along the fine mesh next to the mesh-refinement boundary are also plotted.

$$\frac{\delta X_a}{\delta X_c} = W' \frac{N_a}{N_c} = W' N_a \left(\frac{\delta X_c}{L}\right)^d \tag{39}$$

The ratio of the spatial resolution δX_c to δX_a is the gain in spatial resolution for fixed CPU time and is denoted as G_c. Substituting Eq. (29) into Eq. (39) gives G_c as

$$G_c = \frac{\delta X_c}{\delta X_a} = \left(\frac{1}{W'}\frac{N_A}{N_a}\right)^{\frac{1}{d+1}} = (W'R)^{-\frac{1}{d+1}} \tag{40}$$

This gain again depends only on R and the overhead of the adaptive-

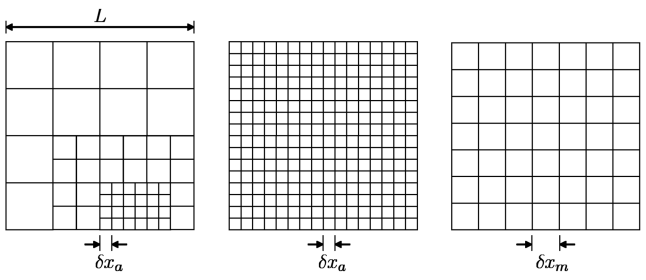


Fig. 18 Mesh sizes compared on adaptive and uniform meshes. a) An adaptive mesh with the finest mesh size δx_a; b) a uniform mesh consisting of only the finest cells; c) a uniform mesh that requires the same amount of computer memory as the adaptive mesh, mesh size δx_m.

mesh solver W'. The time step restriction of CFL condition changes d in the exponent of R in Eq. (33) into $d + 1$.

D. Gain for Two Test Problems

Figure 19 shows the two-dimensional computational configuration. The density is unity and the pressure is $1/\gamma$ everywhere, where γ is 1.4, and the speed of the uniform flow over the cavity is Mach number 1.6. Initially, the flow inside the cavity is stationary and the shear layer over the cavity is slightly perturbed. The computation was done with five levels of mesh refinement and the size of the finest cell is 4.7×10^3. Shock waves and strong velocity gradients are detected and resolved with the finest allowed cells. We used the shock and the gradient indicator developed by Khokhlov [10]. Figure 19 also shows the initial mesh distribution, in which the finest cells are along the shear layer over the cavity. Direction splitting is used with CFL number 0.5.

Figures 20a–20c show vorticity and density contours, and cell distributions at time steps 1000, 2250, and 19,000, respectively. In the beginning (Fig. 20a), the shear layer rolls up and generates vortices, around which small shock waves are formed. Then, shock waves begin to form at both the front and the rear ends of the cavity (Fig. 20b). At a later time, these shock waves become fully developed and small shocks along the shear layer disappear (Fig. 20c). The cluster of small cells shows the refinement at the shear layer and shock waves. The finest cells resolve the vortex motion and the shock waves.

One problem with visualizing the results of these calculations is that some of the contour lines are disconnected at mesh-refinement boundaries, as indicated with the arrows in the density contour in Fig. 20a. This occurs because values of bordering fine and coarse cells are doubly defined, and interpolating physical values could give different values to these cell faces at the same position in the postprocessing stage. This results in discontinuous contour lines at a mesh-refinement boundary. The actual solution, however, does not have an unphysical discontinuity.

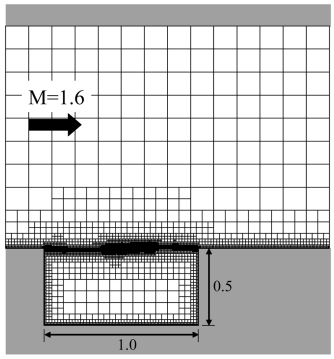
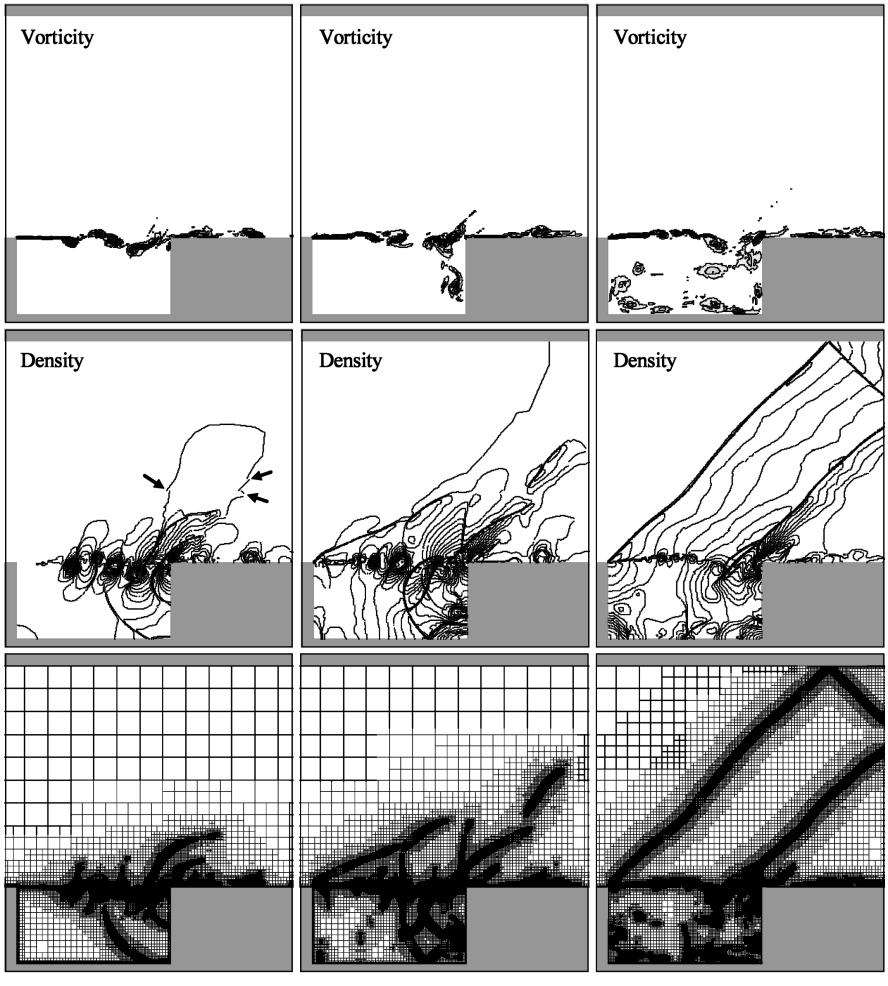


Fig. 19 Computational configuration and initial cell distribution for two-dimensional supersonic flow over a cavity.

In the three-dimensional computation, we used the same flow conditions as in the two-dimensional case, except that now the cavity has a finite width of unity in the spanwise direction, and the mesh refinement has only four levels. The size of the finest cell is 9.4×10^3. Figures 21a–21d show cell distributions and density contours in the center plane of the cavity and three-dimensional



a) step 1000 **b) step 2250** **c) step 1900**

Fig. 20 Two-dimensional inviscid supersonic flow at Mach 1.6 over a cavity. Direction splitting is used with CFL number 0.5. Computation is done with five levels of mesh refinement.

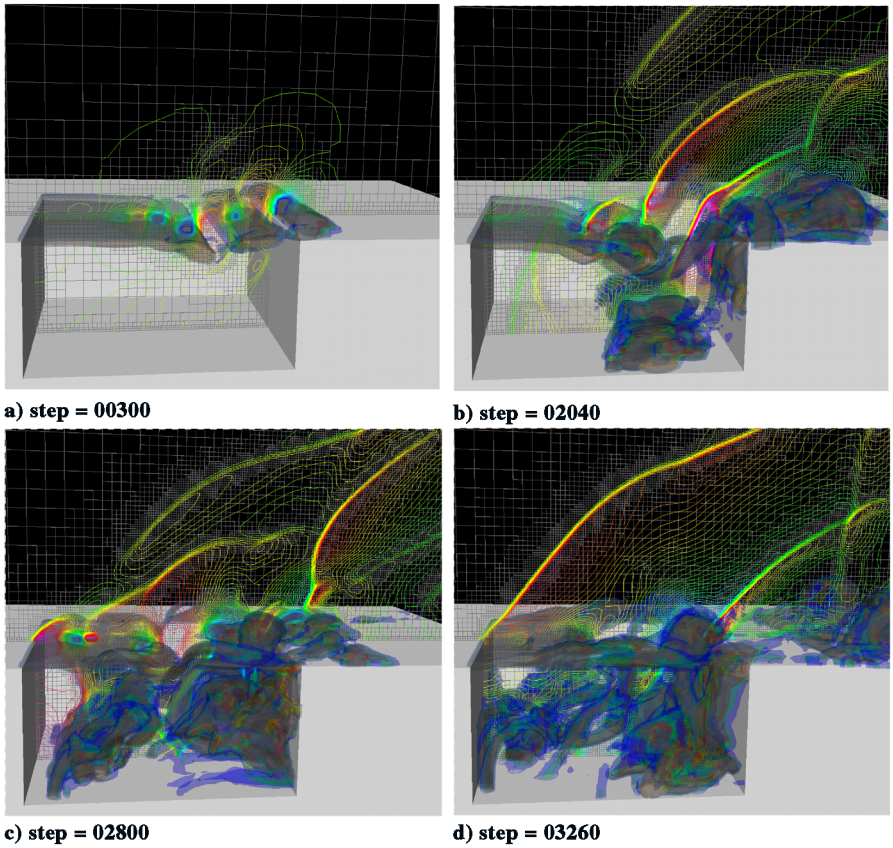


Fig. 21 Three-dimensional inviscid supersonic flow at Mach 1.6 over a cavity. The cavity has a length of unity in the spanwise direction. Direction splitting is used with the CFL number 0.5. Computation is done with four levels of mesh refinement. Cell distributions and density contours in the center plane of the cavity and isosurfaces of vorticity magnitude are shown.

isosurfaces of vorticity magnitude at a sequence of times. This computation shows the same phenomena observed in the two-dimensional computation, that is, roll up of vortices, formation of small shock waves around the vortices, and shock waves at the edges of the cavity. Vorticity isosurfaces show that three-dimensional vortex structures appear as the shear layer rolls up, and that these become fine vortex tubes tangled inside the cavity.

E. Evaluation of the Gain

Here we assume for simplicity that there is no overhead in the adaptive-mesh solver, that is, $W = W' = 1$. Then, the gain of an adaptive mesh, Eqs. (33) and (40), solely depends on R, which is a problem-dependent parameter. If the finest cells are used in most of the computational domain, R approaches unity, and the adaptive-mesh computation loses its advantage. The parameter that controls the mesh adaptation also affects the gain, and unnecessary cells should be minimized to keep R small.

We apply this gain estimation to the computational results in the previous section. Figure 22 shows histories of R, G_m, and G_c of the two-dimensional computation. Initially, the finest cell covers only the shear layer that is yet to roll up, as shown in Fig. 19, and the value of $R = 0.03$ is small. The gain G_m is about 6, which means an adaptive-mesh solver has 6 times higher resolution than a uniform mesh for fixed memory size. As time progresses, cells are refined where the vortices and the shock waves emerge, and R increases. Meanwhile, the gain decreases with the increase of R. Then, when the shock waves fully develop at the front and the rear ends of the cavity at around 5000 time steps, R stops increasing and reaches about 0.16. We finally obtain the gain $G_m = 2.5$ and $G_c = 1.9$ in this particular problem.

Histories of R, G_m, and G_c of the three-dimensional computation are shown in Fig. 23. The value of R reaches about 0.16, which is close to that of the two-dimensional result. The gain, however,

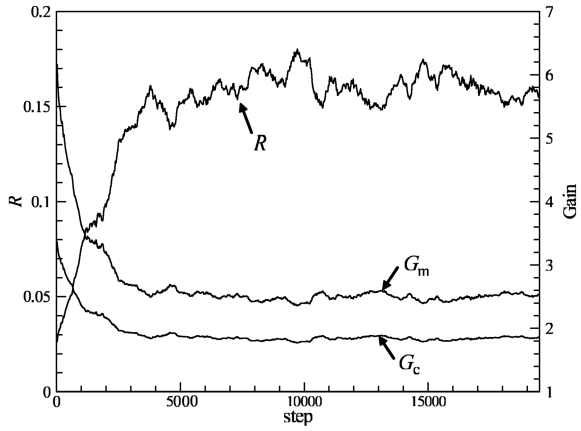


Fig. 22 Time histories of the ratio R, and the gain in spatial resolution for fixed memory size G_m and for fixed CPU time G_c in the two-dimensional simulation of a supersonic cavity flow.

is lower than the two-dimensional result, and it finally becomes $G_m = 1.9$ and $G_c = 1.6$. This is because the absolute values of the exponents in the gain G_m, Eq. (33) and G_c, Eq. (40) decrease for three dimensions.

The ratio R and the gains, G_m and G_c, strongly depend on a flowfield. In this cavity flow, the size of the computational domain is relatively small compared to the size of the cavity, and the cavity has many fine cells because of the vorticities inside the cavity. Thus, this problem is not so favorable for AMR, though the gain is still nearly two. A problem such as a shock traveling in a large domain gives a greater gain. For example, in a simulation of a flame propagating in a narrow channel [15], the gain G_m is on the order of 10 while the flame is propagating through the channel. To obtain a greater gain, it is

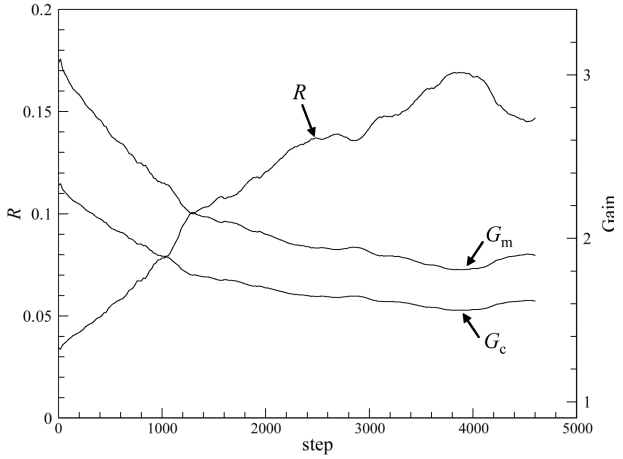


Fig. 23 Time histories of the ratio R, and the gain in spatial resolution for fixed memory size G_m and for fixed CPU time G_c in the three-dimensional simulation of a supersonic cavity flow.

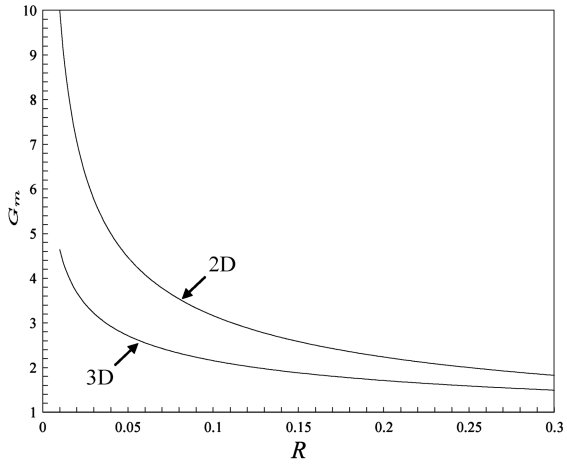


Fig. 24 The gain for the fixed memory size G_m against R in two and three dimensions.

important to know how the gain depends on the ratio R shown in Fig. 24. The gain rapidly decreases with R and then slowly approaches to unity from $R \simeq 0.1$. This suggests that the ratio R should be roughly less than 0.1 for an efficient AMR computation.

V. Conclusions

The FCT algorithm was implemented on an adaptively refined Cartesian mesh. An error analysis at a mesh-refinement boundary showed that accuracy is low when a wave moves from a coarse to a fine mesh, whereas accuracy is close to that of a uniform mesh for a wave moving from a fine to a coarse mesh. This low accuracy gives rise to wave deformation and monotonicity violation at a mesh-refinement boundary. We improved the accuracy by replacing the algebraic averaging in the convective flux with a linear interpolation. With this interpolation, the scheme becomes consistent at mesh-refinement boundaries, and the positivity condition of FCT is still valid. Numerical tests show that FCT with the linearly interpolated convective flux removes the wave deformation and maintains monotonicity at a mesh-refinement boundary. Particularly, in a shock problem, FCT with linear interpolation gives a result almost identical to that of a uniform mesh.

FCT on an adaptive mesh was extended to multidimensions by direction splitting. For assuring monotonicity, the FCT limiter is modified to take into account information of all the cells facing at a mesh-refinement boundary. The two-dimensional test problems show that AMR-FCT captures a shock wave propagating through a mesh-refinement boundary without causing any unphysical waves. Simulations of a two- and a three-dimensional supersonic cavity flow show that vortex structures and shock waves are well defined with an adaptive mesh.

We also presented a way to estimate the gain of using an adaptive mesh compared with a nonadaptive mesh. Two ways were used to define the gain, one based on computer memory and the other based on CPU time. The results show that the gain depends solely on the ratio of the finest cells, that is, the extent to which the finest cells are used in the adaptive mesh. We derived the quantitative relation between this ratio and the gain for Cartesian-mesh AMR. The ratio can be also used to indicate the efficiency of using a general adaptive mesh based on the h-refinement technique. The two- and three-dimensional computational results were used to demonstrate how the gains of adaptive mesh refinement can be quantified.

A change in mesh size by a factor of 2 can have a significant effect on a numerical scheme. When a scheme is applied to an AMR mesh, the error at a mesh-refinement boundary should be carefully investigated. As we showed, accuracy at a mesh-refinement boundary depends on the wave direction. Thus, for example, if we simulate a phenomenon of interest with a fine mesh, and a wave comes from a coarse mesh and interacts with the phenomenon, we should pay more attention to the error for the incoming wave direction. For the schemes designed based on a Taylor expansion, the generalized Fourier analysis is useful to investigate and improve the accuracy of the scheme.

Acknowledgments

This study was done as a funded research project of Japan's New Energy and Industrial Technology Development Organization (NEDO) and by the Office of Naval Research. The authors wish to express their gratitude to J. P. Boris, C. R. DeVore, G. Patnaik, V. N. Gamezo, and J. Liu for valuable discussions and suggestions.

References

[1] Boris, J., and Book, D., "Flux-Corrected Transport I: SHASTA—A Fluid Transport Algorithm That Works," *Journal of Computational Physics*, Vol. 11, No. 1, 1973, pp. 38–69.

[2] Boris, J. P., Book, D. L., and Hain, K., "Flux-Corrected Transport II: Generalization of the Method," *Journal of Computational Physics*, Vol. 18, No. 3, 1975, pp. 248–283.

[3] Boris, J. P., and Book, D. L., "Flux-Corrected Transport III: Minimal-Error FCT Algorithms," *Journal of Computational Physics*, Vol. 20, No. 4, 1976, pp. 397–431.

[4] Boris, J. P., and Book, D. L., "Solution of the Continuity Equation by the Method of Flux-Corrected Transport," *Methods in Computational Physics*, Vol. 16, Chap. 3, Academic Press, New York, 1976, pp. 85–129.

[5] De Zeeuw, D., and Powell, K. G., "An Adaptively Refined Cartesian Mesh Solver for the Euler Equations," *Journal of Computational Physics*, Vol. 104, No. 1, 1993, pp. 56–68.

[6] Ham, F. E., Lien, F. S., and Strong, A. B., "A Cartesian Grid Method with Transient Anisotropic Adaptation," *Journal of Computational Physics*, Vol. 179, No. 2, 2002, pp. 469–494.

[7] Ferziger, J. H., and Peric, M., *Computational Methods for Fluid Dynamics*, Springer–Verlag, Berlin, 1997.

[8] Popinet, S., "Gerris: A Tree-based Adaptive Solver for the Incompressible Euler Equations in Complex Geometries," *Journal of Computational Physics*, Vol. 190, No. 2, 2003, pp. 572–600.

[9] Berger, M. J., and Collela, P., "Local Adaptive Mesh Refinement for Shock Hydrodynamics," *Journal of Computational Physics*, Vol. 82, No. 1, 1989, pp. 64–84.

[10] Khokhlov, A. M., "Fully Threaded Tree Algorithms for Adaptive Refinement Fluid Dynamics Simulations," *Journal of Computational Physics*, Vol. 143, No. 2, 1998, pp. 519–543.

[11] Boris, J. P., Landsberg, A. M., Oran, E. S., and Gardner, J. H., "LCPFCT—Flux-Corrected Transport Algorithm for Solving Generalized Continuity Equations," *NRL Memorandum*, Vol. 93–7192, 1993.

[12] Liu, J., Oran, E. S., and Kaplan, C. R., "Numerical Diffusion in the FCT Algorithm, Revisited," *Journal of Computational Physics*, Vol. 208, No. 2, 2005, pp. 416–434.

[13] Ikeda, T., and Durbin, P. A., "Mesh Stretch Effects on Convection in Flow Simulations," *Journal of Computational Physics*, Vol. 199, No. 1,

2004, pp. 110–125.
[14] Oran, E. S., and Boris, J. P., *Numerical Simulation of Reactive Flow*, 2nd ed., Cambridge Univ. Press, Cambridge, England, U.K., 2001.
[15] Gamezo, V. N., Ogawa, T., and Oran, E. S., "Numerical Simulations of Flame Propagation and DDT in Obstructed Channels Filled with Hydrogen-Air Mixture," *Proceedings of the Combustion Institute*, Vol. 31, 2006.

D. Gaitonde
Associate Editor