Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 208 (2005) 416–434

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Numerical diffusion in the FCT algorithm, revisited

Junhui Liu [a,*], Elaine S. Oran [b], Carolyn R. Kaplan [b]

[a] *Berkeley Research Associates, Inc., P.O. Box 852, Springfield, VA 22150, USA*
[b] *Laboratory for Computational Physics and Fluid Dynamics, US Naval Research Laboratory, Washington, DC 20375, USA*

Received 29 September 2004; received in revised form 20 January 2005; accepted 8 February 2005
Available online 4 May 2005

Abstract

Numerical diffusion in a flux-corrected transport (FCT) algorithm embedded in a Navier–Stokes solver (TINY3D) has been analytically and numerically studied for flows where density variations can be neglected. It is found that numerical diffusion can be analytically expressed in a form similar to that of viscous diffusion. The effective total viscosity can be written as an effective viscosity which is the sum of the physical and numerical viscosities. A low-Mach-number laminar boundary-layer flow is used to test the analytical model of numerical diffusion. A series of simulations, in which the amount of numerical diffusion is varied, show results consistent with predictions of boundary-layer theory when the effective total viscosity is used. The minimum required numerical viscosity to meet the linear stability condition and the lower and upper limits of the cell Reynolds number are also derived.

Keywords: Numerical diffusion; Flux-corrected transport; Cell Reynolds number

1. Introduction

Numerical diffusion is present and often required in all fluid dynamics simulations. When viscous diffusion is not sufficient to ensure stability, numerical diffusion can be used to stabilize algorithms. One example of this use of numerical diffusion is to solve numerical problems with shock waves, since shock-capturing is particularly sensitive to numerical instabilities. Solving this type of problem has been the focus of substantial efforts in the past 50 years. Initial efforts to develop algorithms with good shock-capturing properties introduced "artificial" viscosities or diffusion, such as the work by Von Neumann and Richtmyer [1], Lax and Wendroff [2], and MacCormack and Baldwin [3]. In other approaches, numerical diffusion is introduced implicitly, and one of these approaches is *flux limiting*, which allows the use of higher-order

[*] Corresponding author. Tel.: +1 202 767 6590; fax: +1 202 767 4798.
E-mail addresses: jhliu@lcp.nrl.navy.mil (J. Liu), oran@lcp.nrl.navy.mil (E.S. Oran), ckplan@lcp.navy.mil (C.R. Kaplan).

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.02.030

algorithms and still maintains stability. Flux-limiting is based on adjusting the fluxes going in and out of computational cells, with the objective of ensuring that no unphysical local minima or maxima are introduced by the numerical convection algorithm. It is basic to a class of nonlinear monotone (positivity preserving) algorithms such as FCT, PPM, higher-order Gudonov, and certain TVD methods. The introduction of flux-limiting revolutionized the computation of the continuity equations and therefore of shocks waves and compressible flow.

Although numerical diffusion is useful for stabilizing the algorithms, excessive numerical diffusion, either explicit or implicit, may produce misleading results in some applications. Therefore, it is important to quantify the effects of numerical diffusion, especially in those cases where physical viscosity plays a major role, such as boundary-layer flows and mixing problems. Numerical diffusion and its effects have been studied and reviewed extensively in the past by a number of authors, such as Hirsch [4], Anderson [5], Tannehill et al. [6], Oran and Boris [7], and Wang and Richards [8].

Flux-corrected transport (FCT) [9,10] algorithms are high-order, conservative, monotone, positivity-preserving methods used to solve coupled conservation equations, such as those in the Euler and Navier–Stokes equations. Numerical diffusion in FCT algorithms has been previously investigated [11,12]. Shear layer calculations by Book et al. [11] were used to study numerical diffusion in a FCT algorithm by varying the antidiffusion coefficient. Grinstein and Guirguis [12] used a similar FCT algorithm to study numerical diffusion in inviscid simulations of a low-Mach-number mixing layer. They proposed a numerical viscosity model to quantify the numerical diffusion in their algorithm and also investigated the effects on numerical diffusion of grid resolution, grid-aspect ratio, and the free-stream velocity ratio. Their work showed that the numerical diffusion in the shear flow computed using FCT could emulate the effects of the physical viscosity.

The purpose of this paper is to evaluate numerical diffusion in the LCPFCT algorithm [13], the standard version of FCT, as it is embedded in the Navier–Stokes solver, TINY3D. This is a Navier–Stokes code that uses FCT combined with solutions of diffusive-transport terms that are modeled explicitly using a second-order finite-volume representation. In this paper, numerical diffusion has been studied by looking at an equivalent partial differential equation for the FCT algorithm and investigating those terms that contribute to numerical diffusion. Hirt [14] and Warming and Hyett [15] used a similar approach, to which they referred as a ‘‘modified equation approach’’, to study the instability problem and the nature of both dissipative and dispersive errors. Recently, truncation errors introduced by flux limiters have been used to mimic subgrid-stress models in large-eddy simulations, an example of which is the monotonically integrated large-eddy simulation (MILES) [16–20], where models calibrating the effect of the flux limiter on the subgrid stresses have been investigated. The goal of this paper, however, is different from that of MILES in that here, global numerical diffusion is the focus of the study, rather than the diffusion introduced by the flux limiter.

The plan of this paper is to describe FCT, present the equivalent partial differential equation for this algorithm, derive the numerical viscosity, and then derive analytical expressions for numerical diffusion of multidimensional problems. The test problem, a low-Mach-number laminar flow over a flat plate, was chosen for its simplicity and the availability of exact theoretical solutions. We first test TINY3D by performing a simulation in which there is negligible numerical diffusion. We then carry out simulations with various amounts of numerical diffusion to test the analytical model of numerical diffusion. The stability condition and the cell Reynolds number are discussed along with a discussion of effects of the FCT flux limiter and the anisotropy of numerical viscosity.

2. FCT and an equivalent PDE

We start with a one-dimensional continuity equation that has the form

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho U}{\partial x} = 0, \tag{1}$$

where U is the local convection velocity and ρ is a convected quantity. A FCT solution for this type of equation consists of three major steps (more information concerning this algorithm can be found in [13]):

(1) Central difference the continuity equation in a conserved form:

$$\rho_i^T = \rho_i^o - \frac{1}{2}\left[\epsilon_{i+1/2}(\rho_{i+1}^o + \rho_i^o) - \epsilon_{i-1/2}(\rho_i^o + \rho_{i-1}^o)\right], \tag{2}$$

where $\epsilon_{i+1/2} = U_{i+1/2}\dfrac{\Delta t}{\Delta x}$ is the local Courant number. The quantity Δx is the grid size and Δt is the time step. The subscript i is the location of the center of the computational cell in the x-direction, and superscript o denotes the quantity at time step $n - 1$. A uniform grid is assumed for simplicity. This finite-difference equation can be rewritten as

$$\rho_i^T = a_i^T\rho_{i-1}^o + b_i^T\rho_i^o + c_i^T\rho_{i+1}^o \tag{3}$$

with

$$\begin{cases} a_i^T = \frac{1}{2}\epsilon_{i-1/2}, \\ b_i^T = 1 - \frac{1}{2}(\epsilon_{i+1/2} - \epsilon_{i-1/2}), \\ c_i^T = -\frac{1}{2}\epsilon_{i+1/2}. \end{cases} \tag{4}$$

Since Eq. (2) is not a linearly stable algorithm, the following steps add numerical diffusion to stabilize this algorithm.

(2) Add numerical diffusion to stabilize the algorithm:

$$\tilde{\rho}_i = \rho_i^T + v_{i+1/2}^f(\rho_{i+1}^o - \rho_i^o) - v_{i-1/2}^f(\rho_i^o - \rho_{i-1}^o). \tag{5}$$

(3) Add antidiffusion to limit the amount of numerical diffusion:

$$\rho_i^n = \tilde{\rho}_i - \mu_{i+1/2}^f(\rho_{i+1}^T - \rho_i^T) + \mu_{i-1/2}^f(\rho_i^T - \rho_{i-1}^T). \tag{6}$$

A flux-limiting technique is used at this stage to assure the monotonicity of the solution by adjusting the amount of antidiffusion to avoid the generation of new maxima or minima in the solution. The final algorithm then becomes

$$\rho_i^n = d_i\rho_{i-2}^o + e_i\rho_{i-1}^o + f_i\rho_i^o + g_i\rho_{i+1}^o + h_i\rho_{i+2}^o \tag{7}$$

with

$$\begin{cases} d_i = -\mu_{i-1/2}^f a_{i-1}^T, \\ e_i = (1 + \mu_{i-1/2}^f + \mu_{i+1/2}^f)a_i^T - \mu_{i-1/2}^f b_{i-1}^T + v_{i-1/2}^f, \\ f_i = (1 + \mu_{i-1/2}^f + \mu_{i+1/2}^f)b_i^T - \mu_{i-1/2}^f c_{i-1}^T - \mu_{i+1/2}^f a_{i+1}^T - (v_{i-1/2}^f + v_{i+1/2}^f), \\ g_i = (1 + \mu_{i-1/2}^f + \mu_{i+1/2}^f)c_i^T - \mu_{i+1/2}^f b_{i+1}^T + v_{i+1}^f, \\ h_i = -\mu_{i+1/2}^f c_i^T, \end{cases} \tag{8a}$$

where quantities v^f and μ^f are the added diffusion and antidiffusion coefficients. If a time-step split method is used [13], ϵ is evaluated at the time level $n + 1/2$, for which an extra predictor step is needed to obtain ϵ at time level $n + 1/2$. If Eq. (4) is substituted into Eq. (8a), those coefficients can be rewritten as

$$\begin{cases} d_i = -\frac{1}{2}\mu^f_{i-1/2}\epsilon_{i-3/2}, \\ e_i = \frac{1}{2}\epsilon_{i-1/2} + \frac{1}{2}\mu^f_{i-1/2}(2\epsilon_{i-1/2} - \epsilon_{i-3/2}) + \frac{1}{2}\mu^f_{i+1/2}\epsilon_{i-1/2} + (\nu^f - \mu^f)_{i-1/2}, \\ f_i = 1 - \frac{1}{2}(\epsilon_{i+1/2} - \epsilon_{i-1/2}) + \frac{1}{2}\mu^f_{i-1/2}(2\epsilon_{i-1/2} - \epsilon_{i+1/2}) \\ \qquad - \frac{1}{2}\mu^f_{i+1/2}(2\epsilon_{i+1/2} - \epsilon_{i-1/2}) - (\nu^f - \mu^f)_{i-1/2} - (\nu^f - \mu^f)_{i+1/2}, \\ g_i = -\frac{1}{2}\epsilon_{i+1/2} - \frac{1}{2}\mu^f_{i+1/2}(2\epsilon_{i+1/2} - \epsilon_{i+3/2}) - \frac{1}{2}\mu^f_{i-1/2}\epsilon_{i+1/2} + (\nu^f - \mu^f)_{i+1/2}, \\ h_i = \frac{1}{2}\mu^f_{i+1/2}\epsilon_{i+3/2}. \end{cases} \tag{8b}$$

Because a finite-difference approximation introduces residual errors, a partial differential equation equivalent to the algorithm is not exactly the same as the original continuity equation. Residual errors come from various sources, such as numerical diffusion and numerical dispersion. Similar to the approach used in [14,15,18], the equivalent partial differential equation for the final FCT algorithm shown in Eq. (7) can be obtained by expanding the convected quantity ρ in terms of the Taylor series expansions in both time and space. The result of doing this produces

$$\begin{aligned} \frac{\partial \rho_i}{\partial t} + \frac{\partial (\rho U)_i}{\partial x} = 0 &+ \left(d_i + e_i + f_i + g_i + h_i - 1 + \frac{\partial \epsilon_i}{\partial x}\Delta x\right)\frac{1}{\Delta t}\rho_i + (-2d_i - e_i + g_i + 2h_i + \epsilon_i)\frac{\Delta x}{\Delta t} \\ &\times \frac{\partial \rho_i}{\partial x} + (2^2 d_i + e_i + g_i + 2^2 h_i)\frac{\Delta x^2}{2!\Delta t}\frac{\partial^2 \rho_i}{\partial^2 x} - \frac{\Delta t}{2!}\frac{\partial^2 \rho_i}{\partial t^2} + (-2^3 d_i - e_i + g_i + 2^3 h_i) \\ &\times \frac{\Delta x^3}{3!\Delta t}\frac{\partial^3 \rho_i}{\partial^3 x} - \frac{\Delta t^2}{3!}\frac{\partial^3 \rho_i}{\partial t^3} + (2^4 d_i + e_i + g_i + 2^4 h_i)\frac{\Delta x^4}{4!\Delta t}\frac{\partial^4 \rho_i}{\partial^4 x} - \frac{\Delta t^3}{4!}\frac{\partial^4 \rho_i}{\partial t^4} \\ &+ \mathcal{O}(\Delta x^5, \Delta t^4). \end{aligned} \tag{9a}$$

Taylor series expansions can be further applied to the algorithm coefficients to complete this equivalent partial differential equation, although it may not be applicable in some regions where the flux limiter is activated,

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial \rho U}{\partial x} = 0 &+ \left(\frac{\partial(\nu^f - \mu^f)}{\partial x} + \left(\frac{\partial\left(3\mu^f - \frac{1}{8}\right)\partial\epsilon/\partial x}{\partial x} - \frac{\partial \mu^f}{\partial x}\frac{\partial \epsilon}{\partial x}\right)\Delta x\right)\frac{\Delta x^2}{\Delta t}\frac{\partial \rho}{\partial x} \\ &+ \left(2(\nu^f - \mu^f) + \left(\frac{\partial(2\mu^f\epsilon - \frac{1}{2}\epsilon)}{\partial x} + 4\mu^f\frac{\partial \epsilon}{\partial x}\right)\Delta x\right)\frac{\Delta x^2}{2!\Delta t}\frac{\partial^2 \rho}{\partial x^2} - \frac{\Delta t}{2!}\frac{\partial^2 \rho}{\partial t^2} \\ &+ \left((6\mu^f - 1)\epsilon + \frac{\partial(\nu^f - \mu^f)}{\partial x}\Delta x\right)\frac{\Delta x^3}{3!\Delta t}\frac{\partial^3 \rho}{\partial x^3} - \frac{\Delta t^2}{3!}\frac{\partial^3 \rho}{\partial t^3} \\ &+ \left(2(\nu^f - \mu^f) + \left(\frac{\partial(8\mu^f\epsilon - \frac{1}{2}\epsilon)}{\partial x} + 16\mu^f\frac{\partial \epsilon}{\partial x}\right)\Delta x\right)\frac{\Delta x^4}{4!\Delta t}\frac{\partial^4 \rho}{\partial x^4} - \frac{\Delta t^3}{4!}\frac{\partial^4 \rho}{\partial t^4} + \mathcal{O}(\Delta x^4, \Delta t^4). \end{aligned} \tag{9b}$$

When U is constant, and

$$\begin{aligned} \nu^f &= \frac{1}{6} + \frac{\epsilon^2}{3}, \\ \mu^f &= D\left(\frac{1}{6} - \frac{\epsilon^2}{6}\right) \end{aligned} \tag{10}$$

with $D = 1$ are used, where D controls the amount of antidiffusion, the only residual terms left are those associated with the derivatives of the fourth order and above. Boris and Book [10] showed that this algorithm has a fourth-order phase error. It is, however, second order in space and first order in time.

3. Numerical diffusion

The numerical diffusion of the FCT algorithm shown in Eq. (7) arises from the following terms by combining the first and the second derivatives in Eq. (9b):

$$\frac{\partial}{\partial x}\left[\left(2(\nu^f-\mu^f)+\left(\frac{\partial(2\mu^f\epsilon-\frac{1}{2}\epsilon)}{\partial x}+4\mu^f\frac{\partial\epsilon}{\partial x}\right)\Delta x\right)\frac{\partial\rho}{\partial x}\right]\frac{\Delta x^2}{2!\Delta t}-\frac{\Delta t}{2!}\frac{\partial^2\rho}{\partial t^2}. \tag{11}$$

The remaining terms associated with the first-order derivative $\frac{\partial\rho}{\partial x}$ contribute to the second-order phase error when the convection velocity is not a constant. When Eq. (9b) is substituted into Eq. (11) and the high-order terms are neglected, the term that contributes to numerical diffusion becomes

$$\frac{\partial}{\partial x}\left[\left((\nu^f-\mu^f)-\epsilon^2/2+\left(\left(3\mu^f-\frac{1}{4}\right)\frac{\partial\epsilon}{\partial x}+\epsilon\frac{\partial\mu^f}{\partial x}\right)\Delta x\right)\frac{\partial\rho}{\partial x}\right]\frac{\Delta x^2}{\Delta t}. \tag{12}$$

The numerical viscosity can then be defined as

$$\nu_{\text{num}}\approx\left((\nu^f-\mu^f)-\epsilon^2/2+\left(\left(3\mu^f-\frac{1}{4}\right)\frac{\partial\epsilon}{\partial x}+\epsilon\frac{\partial\mu^f}{\partial x}\right)\Delta x\right)\frac{\Delta x^2}{\Delta t}. \tag{13}$$

If ν^f and μ^f are taken from Eq. (10), ν_{num} can be written as

$$\nu_{\text{num}}\approx\frac{1}{6}\left((1-D)(1-\epsilon^2)+\left(\frac{(4D-3)}{2}\frac{\partial\epsilon}{\partial x}+\frac{\partial D\epsilon}{\partial x}-\epsilon^2\left(4D\frac{\partial\epsilon}{\partial x}+\frac{\partial D\epsilon}{\partial x}\right)\right)\Delta x\right)\frac{\Delta x^2}{\Delta t}. \tag{14}$$

If $\epsilon^2\ll 1$, ν_{num} can be further simplified as

$$\nu_{\text{num}}\approx\frac{1}{6}\left((1-D)+\left(\frac{(4D-3)}{2}\frac{\partial\epsilon}{\partial x}+\frac{\partial D\epsilon}{\partial x}\right)\Delta x\right)\frac{\Delta x^2}{\Delta t}. \tag{15}$$

This numerical viscosity can be divided into two parts: a linear part, $\frac{1}{6}(1-D)\frac{\Delta x^2}{\Delta t}$, and a part related to the velocity gradient, $\frac{1}{6}(\frac{(4D-3)}{2}\frac{\partial U}{\partial x}+\frac{\partial D\cdot U}{\partial x})\Delta x^2$. The linear part is strictly dissipative if $|D|<1$. The velocity-gradient part depends on the sign of the velocity gradient. Both parts are proportional to Δx^2, and the linear part is inversely proportional to Δt, whereas the velocity-gradient part is independent of Δt. If the numerical viscosity is dominated by the linear contribution, numerical viscosity increases linearly with the inverse of Δt. This means that smaller Δt introduces more numerical diffusion. On the other hand, if the velocity-gradient part dominates, this viscosity behaves nonlinearly.

In addition to the velocity-gradient effect, the flux limiter itself is another factor introducing nonlinear behavior into the numerical viscosity. As mentioned above, the flux limiter is used to adjust the antidiffusion to ensure monotonicity and preserve positivity. This adjustment is equivalent to reducing D in some locations to avoid generating new maxima or minima. Where the convected variables are not monotonic, D becomes zero and there is much larger numerical diffusion introduced at these locations. To distinguish the contribution of the flux limiter from the global numerical diffusion, μ^f can be rewritten in a way similar to that used in [16]:

$$\mu^f=\mu^f_c[1-g(\rho,x)], \tag{16}$$

where μ^f_c is the antidiffusion coefficient applied in smooth regions. The function $g(\rho,x)\in|0,1|$ has a value of zero if the flux limiter is not activated , and a value of one if the flux limiter is fully activated when antidiffusion is completely turned off.

Similarly, we can write $D=D_c[1-g(\rho,x)]$ if $\mu^f_c=D_c(\frac{1}{6}-\frac{\epsilon^2}{6})$ is used, where D_c is often called the *mask coefficient*. Numerical viscosity in Eq. (15) can then be rewritten as

$$\nu_{\text{num}}\approx\nu_{\text{num}}|_{\text{sm}}+\nu_{\text{num}}|_{\text{fl}}, \tag{17}$$

where subscripts "sm" stands for smooth and "fl" for flux limiter. The smooth part has the form as

$$\nu_{\text{num}}|_{\text{sm}} \approx \frac{1}{6}\left((1-D_c) + \frac{3(2D_c-1)}{2}\frac{\partial \epsilon}{\partial x}\Delta x\right)\frac{\Delta x^2}{\Delta t}, \tag{18}$$

and the contribution of the flux limiter can be written accordingly if the Taylor series expansions can be applied to the coefficients in Eq. (9a),

$$\nu_{\text{num}}|_{\text{fl}} \approx \frac{1}{6}\left(D_c g(\rho,x) - \left(3D_c g(\rho,x)\frac{\partial \epsilon}{\partial x} + \epsilon\frac{\partial D_c g(\rho,x)}{\partial x}\right)\Delta x\right)\frac{\Delta x^2}{\Delta t}. \tag{19a}$$

If such expansions are not applicable, however, the original discretized form should be used for the contribution from the flux limiter,

$$\nu_{\text{num}}|_{\text{fl}} \approx \left[\frac{1}{2}(\mu_c)_{i+1/2}g_{i+1/2} + \frac{1}{2}(\mu_c)_{i-1/2}g_{i-1/2} - (\mu_c)_{i+1/2}g_{i+1/2}\epsilon_{i+3/2} + (\mu_c)_{i-1/2}g_{i-1/2}\epsilon_{i-3/2}\right]\frac{\Delta x^2}{\Delta t}. \tag{19b}$$

The smooth part of the numerical diffusion exists in all regions, but the contribution from the flux limiter is expected to have a spatially intermittent behavior, since the flux limiter is only activated in the selected regions. For flows where local maxima and minima are not frequent, such as in most of the laminar flows, numerical viscosity can be represented by Eq. (18). Since a lower-order algorithm is used when the flux limiter is activated, the global accuracy can be maintained by having enough resolution in a calculation so that the area where the flux limiter is activated is limited to a small percentage of the computational volume. This consideration is especially important for flows where local maxima and minima are frequent, such as turbulent or transient flows.

Since direction splitting is used in LCPFCT [13], the evaluation of the numerical diffusion in multidimensional simulations is similar to that of the one-dimensional convection equation. Although there are extra residual errors introduced by discretizing source and viscous diffusion terms, numerical diffusion is still similar to that shown in Eq. (12) because those extra terms are not associated with the second derivative of the convected variable. Therefore, numerical diffusion in the density, momentum, and energy equations can be written as

$$\begin{aligned}
&\text{density}: && \frac{\partial}{\partial x}\left(\nu^u_{\text{num}}\frac{\partial \rho}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu^v_{\text{num}}\frac{\partial \rho}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu^w_{\text{num}}\frac{\partial \rho}{\partial z}\right),\\
&x\text{-momentum}: && \frac{\partial}{\partial x}\left(\nu^u_{\text{num}}\frac{\partial \rho u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu^v_{\text{num}}\frac{\partial \rho u}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu^w_{\text{num}}\frac{\partial \rho u}{\partial z}\right),\\
&y\text{-momentum}: && \frac{\partial}{\partial x}\left(\nu^u_{\text{num}}\frac{\partial \rho v}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu^v_{\text{num}}\frac{\partial \rho v}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu^w_{\text{num}}\frac{\partial \rho v}{\partial z}\right),\\
&z\text{-momentum}: && \frac{\partial}{\partial x}\left(\nu^u_{\text{num}}\frac{\partial \rho w}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu^v_{\text{num}}\frac{\partial \rho w}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu^w_{\text{num}}\frac{\partial \rho w}{\partial z}\right),\\
&\text{energy}: && \frac{\partial}{\partial x}\left(\nu^u_{\text{num}}\frac{\partial E}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu^v_{\text{num}}\frac{\partial E}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu^w_{\text{num}}\frac{\partial E}{\partial z}\right),
\end{aligned} \tag{20}$$

where

$$\begin{aligned}
\nu^u_{\text{num}} &\approx \left((\nu^f-\mu^f) - \frac{1}{2}(\epsilon^u)^2 + \left(\left(3\mu^f-\frac{1}{4}\right)\frac{\partial \epsilon^u}{\partial x} + \epsilon^u\frac{\partial \mu^f}{\partial x}\right)\Delta x\right)\frac{\Delta x^2}{\Delta t}, \quad \epsilon^u = \frac{u\Delta t}{\Delta x},\\
\nu^v_{\text{num}} &\approx \left((\nu^f-\mu^f) - \frac{1}{2}(\epsilon^v)^2 + \left(\left(3\mu^f-\frac{1}{4}\right)\frac{\partial \epsilon^v}{\partial y} + \epsilon^v\frac{\partial \mu^f}{\partial y}\right)\Delta y\right)\frac{\Delta y^2}{\Delta t}, \quad \epsilon^v = \frac{v\Delta t}{\Delta y},\\
\nu^w_{\text{num}} &\approx \left((\nu^f-\mu^f) - \frac{1}{2}(\epsilon^w)^2 + \left(\left(3\mu^f-\frac{1}{4}\right)\frac{\partial \epsilon^w}{\partial z} + \epsilon^w\frac{\partial \mu^f}{\partial z}\right)\Delta z\right)\frac{\Delta z^2}{\Delta t}, \quad \epsilon^w = \frac{w\Delta t}{\Delta z}.
\end{aligned} \tag{21}$$

This numerical viscosity has three components ($\nu^u_{\mathrm{num}}, \nu^v_{\mathrm{num}}, \nu^w_{\mathrm{num}}$) which are functions of μ^f, ν^f, velocity gradients, the grid size, and the time step. To understand results of numerical solutions of the Navier–Stokes equations, the amount of numerical diffusion should be quantified. Based on the current FCT algorithm, it can be quantified by using Eqs. (20) and (21). If numerical diffusion is not negligible, it should be taken into account along with the viscous diffusion in analyzing simulation results.

For flows where the density and energy variations are negligible, viscous diffusion can be expressed as

$$\begin{aligned} &x\text{-momentum}: \quad \mu\left[\frac{\partial^2 u}{\partial^2 x}+\frac{\partial^2 u}{\partial^2 y}+\frac{\partial^2 u}{\partial^2 z}\right],\\ &y\text{-momentum}: \quad \mu\left[\frac{\partial^2 v}{\partial^2 x}+\frac{\partial^2 v}{\partial^2 y}+\frac{\partial^2 v}{\partial^2 z}\right],\\ &z\text{-momentum}: \quad \mu\left[\frac{\partial^2 w}{\partial^2 x}+\frac{\partial^2 w}{\partial^2 y}+\frac{\partial^2 w}{\partial^2 z}\right], \end{aligned} \tag{22}$$

where $\mu = \nu\rho$. If the numerical viscosity is isotropic and constant, i.e., $\nu^u_{\mathrm{num}} = \nu^v_{\mathrm{num}} = \nu^w_{\mathrm{num}} \equiv \nu$, the numerical diffusion shown in Eq. (20) has a similar form to that in Eq. (22). Therefore, at least for flows where the density and energy variations are negligible, numerical diffusion can be modeled in a fashion similar to that of viscous diffusion. From this, two conclusions can be drawn:

(i) The effective total viscosity can be evaluated as the sum of the physical and numerical viscosities.
(ii) Numerical diffusion can be constructed to emulate viscous diffusion.

It is always important to minimize the numerical diffusion, and Eq. (21) can be used as a guide to keep the numerical viscosity much smaller than the physical viscosity. For some applications, however, when implementing viscous diffusion is difficult or expensive, Eq. (21) can be also used to construct the required numerical viscosity to substitute for the physical viscosity. For highly compressible flows, numerical diffusion can be quantified similarly. However, whether or not it can be used to substitute for viscous diffusion requires further investigation, since the terms for viscous diffusion for compressible flows are much more complex.

4. Boundary-layer simulations

Here we present calculations of a laminar low-Mach-number boundary-layer formation on a flat plate. The problem is solved using TINY3D, which combines LCPFCT [13] and a finite-volume implementation of the viscous terms. Fig. 1 shows the schematic diagram of the computation set-up, and the computation parameters are summarized in Table 1. The quantity M_∞ is the free-stream Mach number, and Re_{ref} is the Reynolds number per centimeter. M_∞ is chosen as 0.1, so that compressibility is negligible. L = 45 cm is the length of the flat plate, which is placed at 5 cm away from the inflow plane. The slip boundary condition is used for the section between the inflow plane and the flat plate. Since the Reynolds number at the end of the plate is approximately 300,000, the flow is laminar. The spanwise velocity is set to zero in the code, because the problem is a two-dimensional flow.

4.1. Evaluation of numerical diffusion

From classical boundary-layer theory [21], the boundary thickness is proportional to the square root of the dynamic viscosity ν. The displacement thickness for a laminar boundary layer in the low-Mach-number region is [21]

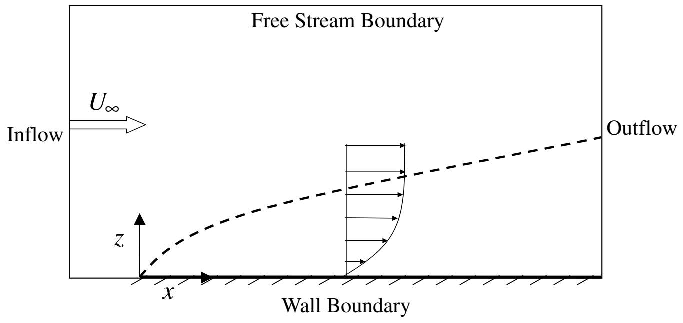

Fig. 1. The schematic diagram of a boundary-layer flow on a flat plate. U_∞ is the inflow velocity. Free stream boundary: zero-gradient conditions; inflow boundary: subsonic inflow conditions; outflow boundary: subsonic outflow conditions; wall boundary: adiabatic and no-slip conditions.

Table 1
Parameters for boundary-layer simulations on a flat plate

M_∞	Δz	Δx	$Z_{\max}$	L	Re_{ref}
0.1	0.02 cm	0.5 cm	2 cm	45 cm	6275

$$\delta_1 = 1.7208\sqrt{\frac{\nu x}{U_\infty}}. \tag{23}$$

The effective viscosity includes contributions from both viscous diffusion and numerical diffusion. For viscous diffusion, the dynamic viscosity coefficient used in the simulations is

$$\nu_{\text{phys}} = 0.631\ \text{cm}^2/\text{s}. \tag{24}$$

Since the smallest grid size is in the z-direction, the global Courant number is $\epsilon = \frac{\Delta t}{\Delta z}(U_\infty + a)$, where U_∞ is the magnitude of the incoming velocity and a is the sound speed. Because Eq. (10) is used in LCPFCT [13] and $\epsilon^2 \ll 1$ for all cases discussed here, the numerical viscosity can be computed from Eq. (15). In addition, since this problem is a laminar boundary-layer flow where few maxima and minima occur, the flux-limiter contribution is expected to be negligible, and the smooth part of the numerical viscosity, Eq. (18), can be used. Thus, the components of the numerical viscosity are

$$z\text{-direction}: \quad \nu^w_{\text{num}} \approx \frac{1}{6}\left[(1-D_c) + \frac{3(2D_c-1)}{2}\frac{\partial \epsilon^w}{\partial z}\Delta z\right]\frac{(U_\infty + a)}{\epsilon}\Delta z, \tag{25}$$

$$x\text{-direction}: \quad \nu^u_{\text{num}} \approx \frac{1}{6}\left[(1-D_c) + \frac{3(2D_c-1)}{2}\frac{\partial \epsilon^u}{\partial x}\Delta x\right]\frac{(U_\infty + a)}{\epsilon}\frac{\Delta x^2}{\Delta z}. \tag{26}$$

The local Courant numbers are $\epsilon^u = \frac{u\Delta t}{\Delta x}$ and $\epsilon^w = \frac{w\Delta t}{\Delta z}$. Since the velocity gradient $\partial u/\partial x$ is negligible for a flat plate boundary-layer problem, its contribution to ν^u_{num} can be neglected. In addition, because $\partial w/\partial z$ is negligible due to the constraint of the continuity equation, its contribution to ν^w_{num} can be also neglected. Therefore, only the linear part of the numerical viscosity is considered:

$$z\text{-direction}: \quad \nu^w_{\text{num}} \approx \frac{(1-D_c)}{6}\frac{(U_\infty + a)}{\epsilon}\Delta z, \tag{27}$$

$$x\text{-direction}: \quad \nu^u_{\text{num}} \approx \frac{(1-D_c)}{6}\frac{(U_\infty + a)}{\epsilon}\frac{\Delta x^2}{\Delta z}. \tag{28}$$

Combining Eqs. (27) and (28), and using a same D_c for all directions, we have

$$\frac{\nu^u_{\text{num}}}{\nu^w_{\text{num}}} \approx \left(\frac{\Delta x}{\Delta z}\right)^2. \tag{29}$$

Thus, the ratio of the components of numerical viscosity is proportional to the second power of the grid aspect ratio. Here the grid aspect ratio $\Delta x/\Delta z = 25$, resulting in ν^u_{num} with a magnitude of 625 times that of ν^w_{num} and producing a numerical viscosity far from isotropic. The viscous diffusion in this type of boundary-layer flow, however, is dominated by diffusion in the vertical direction, and the diffusion in the streamwise direction is negligible. Therefore, anisotropic numerical viscosity is acceptable for such simulations, and only the numerical viscosity in the vertical direction is needed to quantify the contribution of numerical diffusion to the boundary layer growth.

4.2. The total effective viscosity

As shown in Eqs. (25) and (26), numerical diffusion is negligible when $D_c = 1$, so we use this value in the test case to simplify the validation of TINY3D. This calculation is denoted as Case I in the paper. For all of the simulations, $a = 39{,}590$ cm/s. Figs. 2 and 3 show the computed and theoretical [21] velocity profiles at $x = 40.5$ cm and the friction coefficient C_f, where C_f is defined as

$$C_f = \frac{\mu \frac{\partial U}{\partial z}\big|_{\text{wall}}}{\frac{1}{2}\rho U_\infty^2}.$$

The agreement between computation and theory is excellent. The difference between theoretical and computed values of C_f at the leading edge is due to the fact that boundary-layer theory is not valid near that location.

We choose $D_c = 0.999$ for cases that are carried out for analyzing numerical diffusion. In all of the calculations, the grid size and the Mach number are kept constant. Based on these conditions,

$$\nu^w_{\text{num}} \approx \frac{0.145}{\epsilon} \text{ cm/s}, \tag{30}$$

which shows that the vertical component of the numerical viscosity is inversely proportional to Courant number ϵ. The total dynamic viscosity can be obtained by calculating the sum of the physical viscosity (from Eq. (24)) and the numerical viscosity (from Eq. (30)):

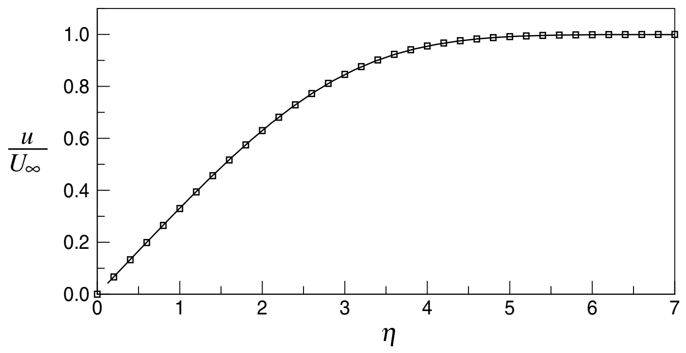

Fig. 2. Normalized velocity profiles as a function of the dimensionless coordinate η for Case I at $x = 40.5$ cm, where $\eta = z\sqrt{\frac{U_\infty}{\nu x}}$. Solid line: viscous simulation with $\epsilon = 0.125$ and $D_c = 1.0$; square: prediction of boundary-layer theory.

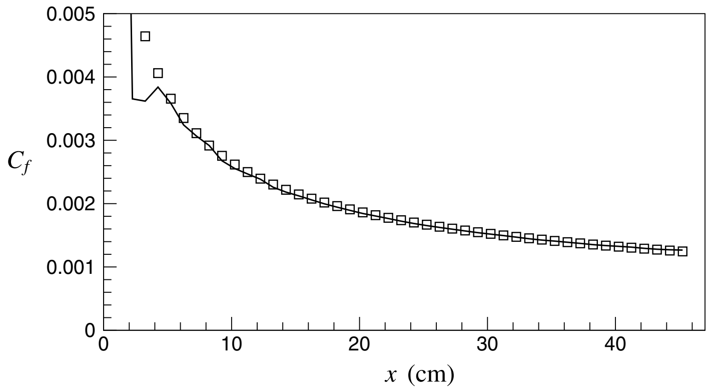

Fig. 3. Skin friction as a function of the streamwise coordinate x for Case I. Solid line: viscous simulation with $\epsilon = 0.125$ and $D_c = 1.0$; square: prediction of boundary-layer theory.

$$\nu_{\text{total}} \approx \left(\frac{0.145}{\epsilon} + 0.631\right) \text{ cm/s}. \tag{31}$$

We control the amount of numerical diffusion by varying the Courant number, which is changed by adjusting the time-step in the simulations.

Table 2 summarizes the information of the five cases we have carried out. The ratios of the computed and the theoretical boundary thicknesses are shown in this table. Quantity δ_1 is the displacement thickness computed from simulations, and $(\delta_1)_{\text{total}}$ and $(\delta_1)_{\text{phys}}$ are the theoretical predictions using either the effective total viscosity or the physical viscosity:

$$(\delta_1)_{\text{total}} = 1.7208\sqrt{\frac{(\nu_{\text{phys}} + \nu_{\text{num}})x}{U_\infty}}, \tag{32}$$

$$(\delta_1)_{\text{phys}} = 1.7208\sqrt{\frac{(\nu_{\text{phys}})x}{U_\infty}}. \tag{33}$$

As mentioned above, there is negligible numerical diffusion in Case I, which is our test case. In Cases II–IV, the total viscous diffusion has both physical and numerical contributions, and the amount of numerical diffusion is comparable to that of physical diffusion. For example, numerical diffusion is more than three times the physical diffusion in Case II. In Case V, the viscous contribution is turned off, and the boundary layer growth is exclusively introduced by the numerical diffusion in the computation. The comparisons between δ_1 and $(\delta_1)_{\text{total}}$ are very good, which is consistent with the analysis in Section 3 that the effective total

Table 2
Comparison between displacement thicknesses computed from simulation results and those predicted by boundary-layer theory using both effective total viscosity and physical viscosity

Cases	ϵ	NS/Euler	D_c	ν_{num} (cm^2/s)	ν_{total} (cm^2/s)	$\frac{\nu_{\text{num}}}{\nu_{\text{phys}}}$	$\frac{\delta_1}{(\delta_1)_{\text{total}}}$	$\frac{\delta_1}{(\delta_1)_{\text{phys}}}$
I	0.125	NS	1.0	0.0	0.631	0.0	1.0	1.0
II	0.0625	NS	0.999	2.32	2.95	3.68	1.02	2.20
III	0.125	NS	0.999	1.16	1.79	1.84	1.02	1.71
IV	0.25	NS	0.999	0.58	1.21	0.92	1.00	1.39
V	0.125	Euler	0.999	1.16	1.16	N/A	1.02	N/A

viscosity for flows where density variation is negligible can be modeled as the sum of the physical viscosity and the numerical viscosity. On the other hand, if only the physical viscosity is used to calculate the theoretical displacement thickness, as shown in Eq. (33), the difference between δ_1 and $(\delta_1)_{\rm phys}$ can be very large, and it increases as ϵ decreases. Since the solution changes as ϵ changes when numerical viscosity is not included, it can appear as though the simulations are not giving a unique solution.

Velocity profiles are shown in Figs. 4a–4d, where both the physical viscosity and the effective total viscosity are used to compute the dimensionless coordinate $\eta = z\sqrt{\frac{U_\infty}{\nu x}}$. Again, comparisons between the simulations and theory are very good if the effective total viscosity is used. If only the physical viscosity is used, however, the difference between the numerical results and the predicted values is large, similar to what is shown in the results of the displacement thickness discussed above. In addition, when numerical viscosity is used to analyze the results, as shown in Fig. 4d, the inviscid simulation compares well with the prediction of boundary-layer theory. This agrees with our conclusion in Section 3 that viscous diffusion can be emulated by numerical diffusion for flows where density variation is negligible.

5. Stability analysis

In the previous sections, we have developed an analytical model for the numerical viscosity in the FCT algorithm and have it tested using laminar boundary-layer simulations. In this section, we will evaluate the maximum allowed mask coefficient and the minimum required numerical viscosity for a given problem. In addition, we will also evaluate the upper limit on the cell Reynolds number and the corresponding maximum allowed grid resolution for a given amount of antidiffusion.

If the function defined in Eq. (16) has a top-hat profile across a cell, the FCT algorithm (7) can be rewritten in a way similar to that used in [16]:

$$\rho_i^n = L^h(\rho_i^o) + g(\rho_i^o, x_i)[L^l(\rho_i^o) - L^h(\rho_i^o)], \tag{34}$$

where superscripts h and l denote "high-order" and "low-order", respectively. The function $L^h(\rho_i^o)$ is the base algorithm, which is a high-order algorithm applied in smooth regions where the flux limiter is not activated. The function $L^l(\rho_i^o)$ is a lower-order algorithm with properties of unconditional stability and monotonicity. It is applied when the flux limiter is fully activated. In the case of a constant convection velocity, the full algorithm shown in Eq. (34) has an amplification factor for a wave number κ:

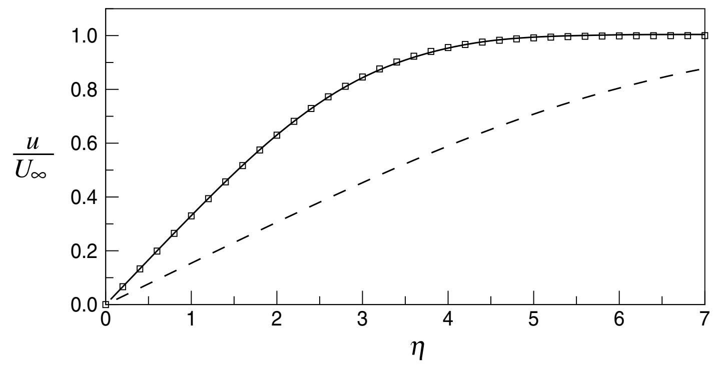

Fig. 4a. Normalized velocity profiles as a function of the dimensionless coordinate η for Case II ($\epsilon = 0.0625$ and $D_c = 0.999$) at $x = 40.5$ cm. Solid line: viscous simulation using the effective total viscosity to calculate η; dashed line: viscous simulation using physical viscosity to calculate η; square: prediction of boundary-layer theory.

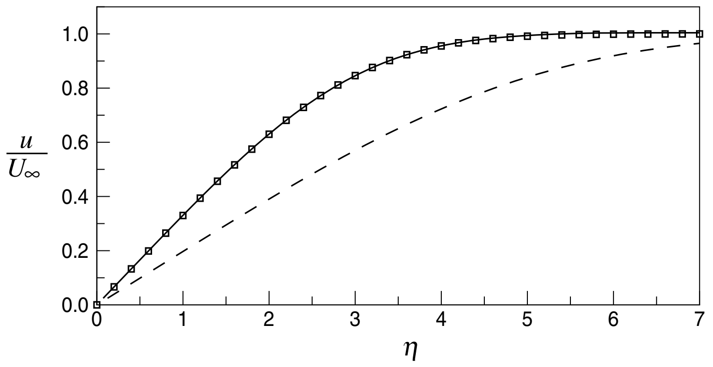

Fig. 4b. Normalized velocity profiles as a function of the dimensionless coordinate η for Case III ($\epsilon = 0.125$ and $D_c = 0.999$) at $x = 40.5$ cm. Solid line: viscous simulation using the effective total viscosity to calculate η; dashed line: viscous simulation using physical viscosity to calculate η; square: prediction of boundary-layer theory.

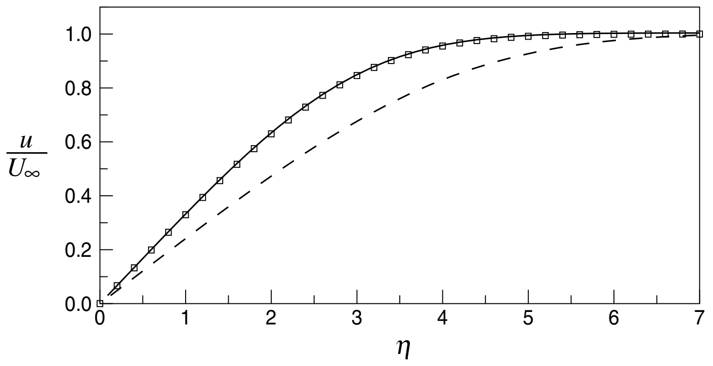

Fig. 4c. Normalized velocity profiles as a function of the dimensionless coordinate η for Case IV ($\epsilon = 0.25$ and $D_c = 0.999$) at $x = 40.5$ cm. Solid line: viscous simulation using the effective total viscosity to calculate η; dashed line: viscous simulation using physical viscosity to calculate η; square: prediction of boundary-layer theory.

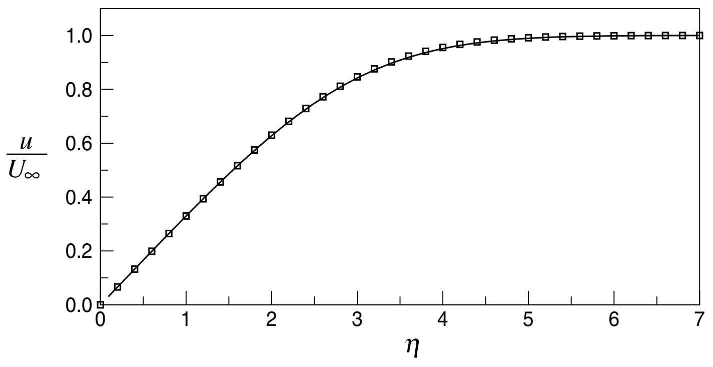

Fig. 4d. Normalized velocity profiles as a function of the dimensionless coordinate η for Case V ($\epsilon = 0.125$ and $D_c = 0.999$) at $x = 40.5$ cm. Solid line: inviscid simulation using numerical viscosity to calculate η; square: prediction of boundary-layer theory.

$$A(\beta) = A^h(\beta) + \mu_c^f G(\beta), \tag{35}$$

where $\beta = \kappa\Delta x$ and Δx is the grid size. The parameter μ_c^f is the antidiffusion coefficient applied in smooth regions, $A^h(\beta)$ is the amplification factor introduced by algorithm $L^h(\rho_i^o)$, and $\mu_c^f G(\beta)$ is the contribution from the flux limiter. Functions $A^h(\beta)$ [11] and $G(\beta)$ are

$$A^h(\beta) = 1 - 2(v^f - \mu_c^f)(1-\cos\beta) - \mathrm{I}\epsilon\sin\beta(1 + 2\mu_c^f(1-\cos\beta)), \tag{36}$$

$$G(\beta) = -2\frac{g(\beta) * \{\rho^o(\beta)[(1-\cos\beta)(1-\mathrm{I}\epsilon\sin\beta)]\}}{\rho^o(\beta)}, \tag{37}$$

respectively, where I indicates $\sqrt{-1}$, $*$ is the convolution operator, and $g(\beta)$ and $\rho^o(\beta)$ are the Fourier transforms of $g(\rho_i^o, x_i)$ and ρ_i^o, respectively.

If a diffusion term is added to Eq. (1) and is discretized by a second-order central algorithm, the amplification factor $A(\beta)$ becomes

$$A(\beta) = 1 - 2\left[v^f - \mu_c^f + \frac{\epsilon}{Re_{\Delta x}}\right](1-\cos\beta) - \mathrm{I}\epsilon\sin\beta(1 + 2\mu_c^f(1-\cos\beta)) + \mu_c^f G(\beta), \tag{38}$$

where U is the convection velocity and $Re_{\Delta x} = u\Delta x/\, v_{\text{phys}}$ is the cell Reynolds number. If Eq. (10) is used for v^f and μ^f, the stability condition in Eq. (38) is controlled by ϵ, $Re_{\Delta x}$, D_c, and $G(\beta)$. Thus, for a specified $G(\beta)$ and ϵ, the stability condition can be expressed as either

$$D_c \leqslant D_c^{\max}, \quad \text{given } Re_{\Delta x},$$

where $D_c^{\max}$ is the maximum allowed mask coefficient, or

$$Re_{\Delta x}^{\min} \leqslant Re_{\Delta x} \leqslant Re_{\Delta x}^{\max}, \quad \text{given } D_c,$$

where $Re_{\Delta x}^{\min}$ and $Re_{\Delta x}^{\max}$ are the lower and upper limits of the cell Reynolds number. Conditions for $Re_{\Delta x}$ reflect those for Δx if U and v_{phys} are fixed.

The function $G(\beta)$ defined in Eq. (37) depends on the distributions of the convected variables, the choice of flux limiter, the Courant number, and the grid resolution. Therefore, the stability condition is problem-dependent, and there may not be a universal analytical solution to describe this condition. Even for a simple problem, such as the convection of a sine or cosine wave, tedious analytical and numerical work may be required to evaluate the effects of the flux limiter on the stability condition. On the other hand, the stability condition for $L^h(\rho_i^o)$ can be easily and precisely defined, and the result given below will mainly focus on $L^h(\rho_i^o)$.

Fig. 5a and Fig. 6a show $D_c^{\max}$ as a function of ϵ and Δx, respectively, for a given U and v_{phys}, when the flux limiter is not used. The data are only shown for the values of ϵ required to maintain positivity, $\epsilon \leqslant 0.5$ [13]. As shown in Eq. (18), the smooth part of the numerical viscosity can be rewritten as $v_{\text{num}}|_{\text{sm}} \approx (1 - D_c)$ $U\Delta x/6\epsilon$, indicating that there is a minimum numerical viscosity corresponding to a $D_c^{\max}$. This minimum value of $v_{\text{num}}|_{\text{sm}}$ is shown in Figs. 5b and 6b. The quantity $D_c^{\max}$ is only a function of ϵ for inviscid simulations, whereas in viscous simulations, it is also a function of v, Δx, and U. We expect, however, that the presence of the flux limiter will alter the distribution of $D_c^{\max}$ to some extent, and, even for inviscid simulations, it will also introduce some dependence of Δx and of other parameters.

Without considering the flux limiter, and for a given amount of antidiffusion or a given value of D_c , the lower and upper limits of $Re_{\Delta x}$ can be derived from the stability condition:

$$Re_{\Delta x}^{\min} = \frac{\epsilon}{\frac{1}{2} - (v^f - \mu_c^f)}, \tag{39}$$

$$Re_{\Delta x}^{\max} = \frac{\epsilon}{H(\beta)|_{\max} - (v^f - \mu_c^f)}, \tag{40}$$

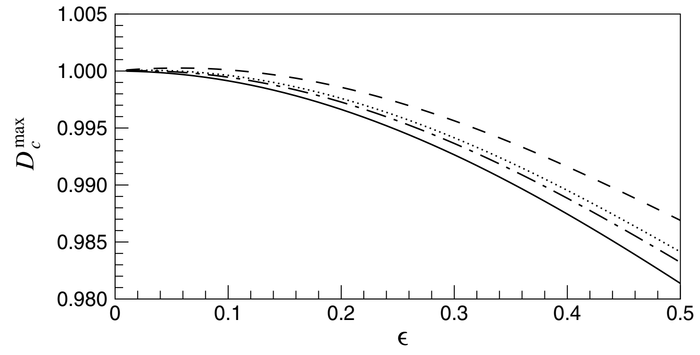

Fig. 5a. Maximum allowed mask coefficient $D_c^{\max}$ for the base algorithm as a function of the Courant number ϵ. Solid line: inviscid simulation; other lines: viscous simulations with $\nu_{\text{phys}} = 0.631\ \text{cm}^2/\text{s}$ and $U = 39590$ cm/s; dashed line: $\Delta x = 0.01$ cm; dotted line: $\Delta x = 0.02$ cm; dashed–dotted line: $\Delta x = 0.03$ cm.

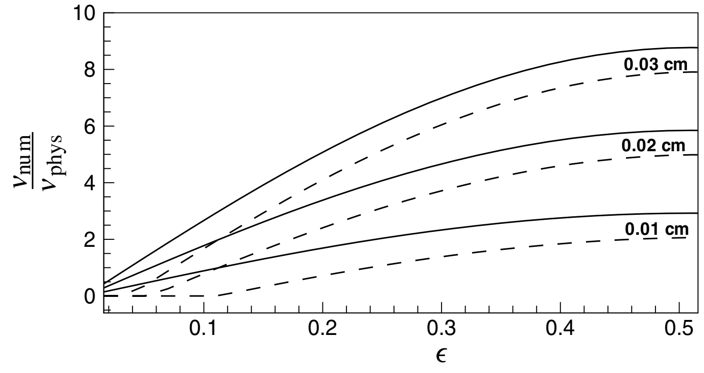

Fig. 5b. Minimum required numerical viscosities scaled by physical viscosity for the base algorithm as a function of the Courant number ϵ. $\Delta x = 0.01$, 0.02, and 0.03 cm are plotted for both inviscid (solid lines) and viscous cases (dashed lines). Physical viscosity and the convection velocity are $\nu_{\text{phys}} = 0.631\ \text{cm}^2/\text{s}$ and $U = 39590$ cm/s, respectively.

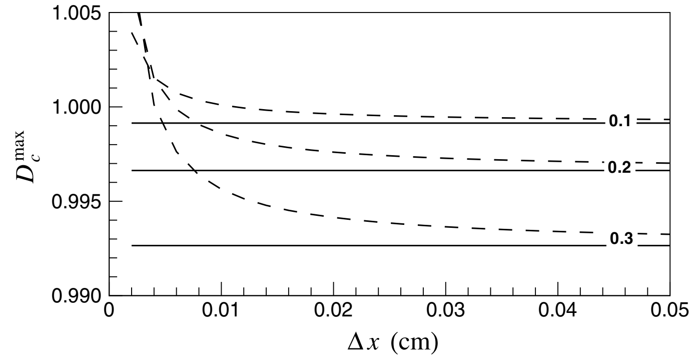

Fig. 6a. Maximum allowed mask coefficient $D_c^{\max}$ for the base algorithm as a function of the grid size Δx. Courant number $\epsilon = 0.1$, 0.2, and 0.3 are plotted for both inviscid (solid lines) and viscous cases (dashed lines). Parameters used for the viscous cases are $\nu_{\text{phys}} = 0.631\ \text{cm}^2/\text{s}$ and $U = 39590$ cm/s.

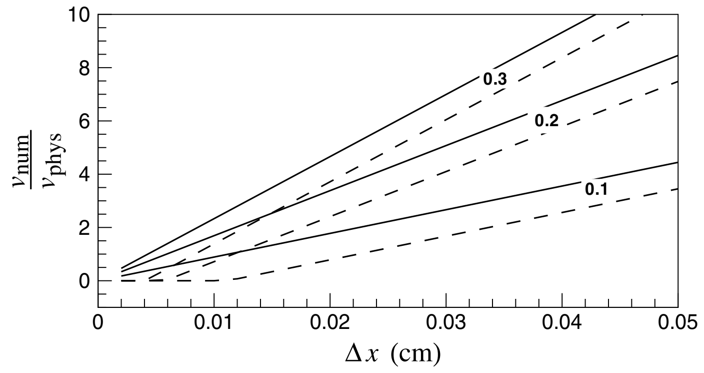

Fig. 6b. Minimum required numerical viscosities scaled by physical viscosity for the base algorithm as a function of the grid size Δx. Courant number ϵ = 0.1, 0.2, and 0.3 are plotted for both inviscid (solid lines) and viscous cases (dashed lines). Physical viscosity and the convection velocity are $\nu_{\text{phys}} = 0.631$ cm^2/s and $U = 39590$ cm/s, respectively.

where

$$H(\beta) = \frac{1 - \sqrt{1 - [\epsilon \sin\beta(1 + 2\mu_c^f(1 - \cos\beta))]^2}}{2(1 - \cos\beta)}. \tag{41}$$

Eq. (40) is valid if $(\nu^f - \mu_c^f) < H(\beta)|_{\max}$, when numerical diffusion is not sufficient to stabilize the algorithm. On the other hand, if $(\nu^f - \mu_c^f) \geqslant H(\beta)|_{\max}$, the algorithm is stable and $Re_{\Delta x}^{\max} = \infty$, since numerical diffusion itself is large enough to stabilize the algorithm. The minimum value of $H(\beta)|_{\max}$ is $\epsilon^2/2$, which occurs when no antidiffusion is present. Quantities $Re_{\Delta x}^{\min}$ and $Re_{\Delta x}^{\max}$ in Eqs. (39) and (40) depend on both ϵ and the amount of antidiffusion. Fig. 7a shows $Re_{\Delta x}^{\max}$ as a function of ϵ for selected values of D_c. The wave number corresponding to $H(\beta)|_{\max}$ around $\beta = 0.83$ for $D_c = 1$. Either decreasing ϵ or D_c alleviates the constraint of the stability condition, since both ways introduce more numerical diffusion. In addition, $Re_{\Delta x}^{\max}$ with a large ϵ is less sensitive to the magnitude of D_c. Fig. 7b shows the maximum grid size corresponding to $Re_{\Delta x}^{\max}$ when the physical viscosity and the sound speed of the current boundary-layer simulations are used.

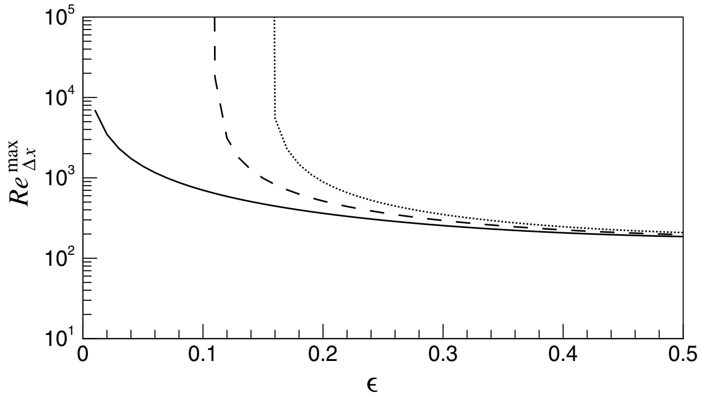

Fig. 7a. Upper limits of the cell Reynolds number for the base algorithm as a function of the Courant number ϵ. Solid line: $D_c = 1.0$; dashed line: $D_c = 0.999$; dotted line: $D_c = 0.998$.

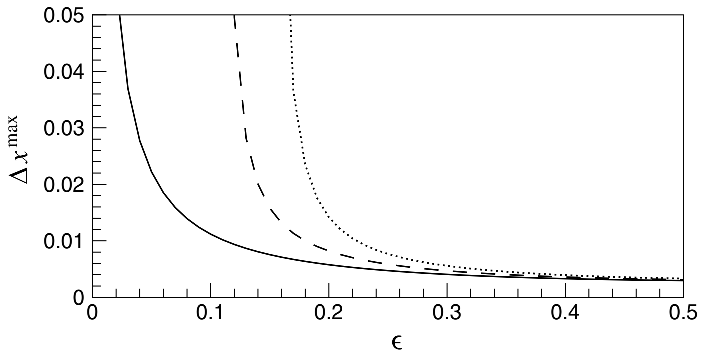

Fig. 7b. Maximum allowed grid size for the base algorithm as a function of the Courant number ϵ. Solid line: $D_c = 1.0$; dashed line: $D_c = 0.999$; dotted line: $D_c = 0.998$. Physical viscosity and the convection velocity are $\nu_{\text{phys}} = 0.631\ \text{cm}^2/\text{s}$ and $U = 39590\ \text{cm/s}$, respectively.

The lower limit $Re_{\Delta x}^{\min}$ is fortunately small: its value is less than 2 for $\epsilon \leqslant 0.5$. On the other hand, the restriction imposed by $Re_{\Delta x}^{\max}$ in Eq. (40) is difficult to meet. For example, as shown in Figs. 5b and 6b, Case IV with $\Delta x = 0.02$ cm requires $D_c < 0.9977$, resulting a numerical viscosity more than three times that of the physical viscosity in the vertical direction. On the other hand, if $D_c = 0.999$, Case IV requires $\Delta x < 0.0058$ cm, and Case I needs $\Delta x < 0.009$ cm, as shown in Fig. 7b. Thus, either a large amount of numerical diffusion or a very fine mesh is required to meet the stability condition of $L^h(\rho_i^o)$. This restriction is overkill since the upper limit is governed by the most unstable modes. Those unstable modes have large wave numbers and may only occur at a few locations or a few times. Since the stability of the most unstable modes can be controlled locally by activating the flux limiter, the stability condition of the full algorithm (34) will be less severe than that of $L^h(\rho_i^o)$. Further study of $G(\beta)$ is needed, however, to assess quantitatively the modification on the stability condition by the flux limiter.

6. Discussion

As shown in Eqs. (18) and (19), the flux limiter will contribute to the numerical diffusion. In addition, as shown in Eq. (21), the anisotropy of numerical viscosity will also affect simulations if there are large velocity gradients in more than one direction. The analysis in previous sections of this paper focused primarily on cases for which only the smooth part of the linear portion of the numerical viscosity is important, and a constant mask coefficient can be used in all directions. In this section, however, we will briefly discuss the effects of the flux limiter and of the anisotropy of numerical viscosity on numerical diffusion.

6.1. Effect of the flux limiter on numerical diffusion

Flux limiters were originally proposed to maintain the monotonicity of the convected variables and to improve stability condition of the algorithm. Although a flux limiter allows larger grid sizes to be used without introducing a large amount of numerical diffusion globally, it does introduce a large amount of numerical diffusion to a few locations where the flux limiter is activated. If we were to define a distribution of numerical viscosity in space, we would expect some large spikes in the distribution. The frequency of those spikes depends on the flow field (such as the occurrence of the local maxima or minima), grid resolution, and the choice of flux limiter. Since a lower-order algorithm is used when the flux limiter is activated, it is important to use sufficient grid resolutions to keep the frequency of those spikes relatively low. This will maintain the global accuracy and reduce the contribution of flux limiter to the overall numerical diffusion.

For the laminar boundary-layer problem studied here, the effect of the flux limiter is expected to be small, since local maxima and minima only appear in very few locations. This is consistent with our observation that the smooth part of the numerical viscosity model shown in Eq. (18) is a good estimate of numerical diffusion.

Currently, the numerical diffusion introduced by flux limiters is used to represent the subgrid-stress models in large-eddy simulations [16–20]. Margolin and Rider [20] designed implicit subgrid-stress models based on the assumption that the algorithm does not have much dissipation on resolved scales, but is strongly dissipative when the solution is unresolved. Since numerical diffusion in FCT can be made small in regions where the flux limiter is absent and large where the flux limiter is activated, FCT is ideally suited for the implicit subgrid-stress modeling. Because flux-limiting is a nonlinear process, it redistributes the energy of the convected variables among a range of scales, which is especially true in the region at large wave numbers. It is not always guaranteed, however, that the contribution from a flux limiter can accurately represent the physics governing subgrid scales and can be used as an appropriate model for subgrid stresses. Fureby and Grinstein [16] mentioned that the implicit subgrid stress model must be able to mimic the subgrid stress turbulence in terms of providing accurate inertial subrange and satisfying the near-wall requirement. Therefore, it is important to study and understand the performance of flux limiters in wave space, such as the evaluation of $G(\beta)$. This would allow us to develop flux limiters that more accurately mimic subgrid-stress turbulence.

6.2. *Effect of anisotropy of numerical viscosity*

We have shown that an anisotropic numerical viscosity is acceptable for problems such as boundary-layer flows, where the velocity gradient is significant in only one direction. It is not appropriate, however, for flows where there are large velocity gradients in more than one direction, such as in mixing-layer problems with vortex roll-up. Previous inviscid simulations of mixing layers [12] showed that, if there are vortices forming in the flow, a simulation with $\Delta x = 2.5\Delta z$ showed considerably larger numerical diffusion than that with $\Delta x = \Delta z$. The high-strain region was diffused more in the x-direction in the simulation with $\Delta x = 2.5\Delta z$. This observation is consistent with what we find in Eq. (29). That is, numerical diffusion is very sensitive to the grid size, and the ratio between the vertical and the streamwise numerical viscosity is roughly proportional to the second power of the grid-aspect ratio when D_c is constant. Thus, the streamwise numerical diffusion in a simulation with $\Delta x = 2.5\Delta z$ is roughly 6.25 times that of a simulation with $\Delta x = \Delta z$.

Since the anisotropy of numerical viscosity is very sensitive to the grid-aspect ratio, if numerical diffusion cannot be kept small, it is important to adjust D_c to construct an isotropic numerical viscosity for flows where there are large velocity gradients in more than one direction. As shown in Eqs. (25) and (26), if the contributions to the numerical viscosity by the local velocity gradients are kept small (for example, by using a small time step), an isotropic viscosity would only require a different constant in each direction based on the grid-aspect ratios. If those contributions from the velocity gradients cannot be kept small, however, a variable D_c should be used to balance the gradient effects.

7. Conclusions

Numerical diffusion in the flux-corrected transport algorithm [1,2], as implemented in LCPFCT [13] and embedded the Navier–Stokes solver TINY3D, has been studied analytically and numerically for problems in which physical diffusion plays an important role. The equivalent partial differential equation for the FCT algorithm was derived, and an analytical model for the numerical viscosity was formulated. We demonstrated that, for flows with small density variations, numerical diffusion can be modeled in a way similar to that of viscous diffusion. The total effective viscosity can be evaluated as the sum of physical and numerical viscosities, and viscous diffusion can be emulated by numerical diffusion. A series of low-Mach-number laminar boundary-layer simulations has been carried out to test the analytical model of the effective total viscosity, and the results agree very well with the predictions.

In addition to the development and the testing of the numerical viscosity model, we also used the stability condition to derive the minimum required numerical viscosity, the upper and lower limits of the cell Reynolds number, and the maximum allowed grid size. We found that, without the flux limiter, either a large amount of numerical diffusion or a very fine mesh is required to meet the stability condition. The flux limiter improves the stability condition and reduces the need of introducing large numerical diffusion globally. In addition, the effect of the anisotropy of the numerical viscosity is also discussed. It is important to use an isotropic numerical viscosity if the velocity gradients are large in more than one direction.

Although the analysis presented in this paper focuses on a standard version of FCT, the general methodology used in this paper could be applied to algorithms that have a similar structure to what is in FCT. For example, for users of PPM and TVD algorithms, this approach could give useful information on the numerical diffusion inherent in the algorithms.

Acknowledgments

The authors wish to express their gratitude to Dr. Jay P. Boris and Dr. Fernando F. Grinstein for valuable discussions and the gracious inputs. Also, the computational resources provided by the DoD High Performance Computing Modernization Program Office are greatly acknowledged.

References

[1] J. Von Neumann, R.D. Richtmyer, A method for the numerical calculations of hydrodynamical shocks, J. Math. Phys. 21 (1950).
[2] P.D. Lax, B. Wendroff, Systems of conservation laws, Comm. Pure Appl. Math. 13 (1960) 217.
[3] R.W. MacCormack, B.S. Baldwin, A numerical method for solving the Navier–Stokes equations with application to shock-boundary layer interaction, AIAA 75-1.
[4] C. Hirsch, Numerical Computation of Internal and External Flows vols. 1, 2, John Wiley, 1991.
[5] J.D. Anderson Jr., Computational Fluid Dynamics – The Basics with Applications, MaGraw-Hill, 1995.
[6] J.C. Tannehill, D.A. Anderson, R.H. Pletcher, Computational Fluid Dynamics and Heat Transfer, Taylor & Francis, 1997.
[7] E.S. Oran, J.P. Boris, Numerical Simulation of Reactive Flow, Cambridge University Press, 2001.
[8] Z. Wang, B.E. Richards, High resolution schemes for steady flow computation, J. Comput. Phys. 97 (1991) 53.
[9] J.P. Boris, D.L. Book, Flux-corrected transport I: SHASTA a fluid transport algorithm that works, J. Comput. Phys. 11 (1973) 38.
[10] J.P. Boris, D.L. Book, Solution of the continuity equation by the method of flux-corrected transport, Meth. Comput. Phys. 16 (1976) 85.
[11] D.L. Book, C. Li, G. Patnaik, F.F. Grinstein, Quantifying residual numerical diffusion in flux-corrected transport algorithms, J. Sci. Comput. 6 (1991) 323.
[12] F.F. Grinstein, R.H. Guirguis, Effective viscosity in the simulation of spatially evolving shear flows with monotonic FCT models, J. Comput. Phys. 101 (1992) 165.
[13] J.P. Boris, A.M. Landsberg, E.S. Oran, J.H. Gardner, LCPFCT – A flux corrected transport algorithm for solving generalized continuity equations, NRL Memorandum Report 6410-93-7192, Naval Research Laboratory, Washington, DC, 1993.
[14] C.W. Hirt, Heuristic stability theory for finite difference equations, J. Comput. Phys. 2 (1968) 339.
[15] R.F. Warming, B.J. Hyett, The modified equation approach to the stability and accuracy analysis of finite-difference methods, J. Comput. Phys. 14 (1974) 159.

[16] C. Fureby, F.F. Grinstein, Monotonically integrated large eddy simulation of free shear flows, AIAA J. 37 (1999) 544.
[17] C. Fureby, F.F. Grinstein, Large eddy simulation of high-Reynolds-number free and wall-bounded flows, J. Comput. Phys. 181 (2002) 68.
[18] L.G. Margolin, W.J. Rider, A rationale for implicit turbulence modeling, Int. J. Numer. Meth. Fluids 39 (2002) 821.
[19] W.J. Rider, L.G. Margolin, From numerical analysis to implicit subgrid turbulence modeling, AIAA-2003-4101, 16th AIAA CFD Conference, Orlando, FL, June 2003.
[20] L.G. Margolin, W.J. Rider, The design and construction of implicit subgrid scale models, Int. J. Numer. Meth. Fluids 47 (2005) 1173.
[21] H. Schlichting, Boundary-Layer Theory, McGraw-Hill, New York, 1987.