Journal of Computational Physics **157,** 371–403 (2000)
doi:10.1006/jcph.1999.6387, available online at http://www.idealibrary.com on IDEAL®

A Compact-Difference Scheme for the Navier–Stokes Equations in Vorticity–Velocity Formulation

Hubert L. Meitz and Hermann F. Fasel

Department of Aerospace and Mechanical Engineering, University of Arizona, 1130 North Mountain, P.O. Box 210119, Tucson, Arizona 85721-0119

Received February 10, 1998; revised June 10, 1999

This paper presents a new numerical method for solving the incompressible, unsteady Navier–Stokes equations in vorticity–velocity formulation. The method is applicable to spatial simulations of transitional and turbulent boundary layer flows. It is based on a compact-difference discretization of the streamwise and wall-normal derivatives in Cartesian coordinates. A Fourier collocation approach is used for the spanwise derivatives. Important new features of the numerical method are the use of nonequidistant differences in the wall-normal direction; the use of split-compact differences in the streamwise direction; a new, fast iteration for a semi-implicit time integration of the wall-normal diffusion terms; and an improvement of the buffer domain technique to prevent reflections of waves at the outflow boundary. Results of test calculations are presented to verify the improvements obtained by the use of these new techniques. © 2000 Academic Press

1. INTRODUCTION

The principal difficulty in obtaining numerical solutions to the incompressible Navier–Stokes equations is the fact that there is no evolution equation for the pressure p. Rather, the pressure serves as an instantaneous correction to the evolution equations for the velocities such that the continuity equation (zero divergence of the velocity) is satisfied everywhere in the flow field.

There are several distinct approaches to overcoming this difficulty. The first approach is known as the artificial compressibility method [5]. It uses an artificial compressibility parameter θ to couple the divergence of the velocity to a change of the pressure in pseudotime τ, thus turning the continuity equation into an evolution equation for the pressure. Typically, the solution procedure consists of integrating this system of hyperbolic equations in pseudotime until the divergence of the velocity has been reduced to the desired accuracy. The chief difficulty here lies in devising an iteration scheme that converges reasonably quickly without requiring excessive amounts of memory [25, 29]. Although this approach can be used to compute truly unsteady flows, it has mostly been applied to compute steady, incompressible, turbulent (i.e., Reynolds-averaged) flows over complicated geometries.

The second approach is known as the pressure correction or fractional step method [6]. Over the past 10 years, this approach has become by far the most popular numerical method for the solution of the incompressible Navier–Stokes equations. In this scheme, the integration over one timestep is split into a predictor step for the intermediate velocity that omits the pressure, a Poisson equation for a pressure-like quantity, and a final corrector step for the velocity to enforce conservation of mass. The details of the method vary between different implementations. Typically, the diffusion terms (at least in the wall-normal direction, say y) are integrated with an implicit scheme for numerical stability, while the nonlinear terms are integrated with an explicit scheme for efficiency. A major difficulty with fractional step methods is the specification of wall boundary conditions at the intermediate steps. Usually, ad hoc wall boundary conditions are derived by extrapolation of the velocities and pressure gradients from previous timesteps. While such an extrapolation is sufficient for numerical stability, it introduces large splitting errors into the integration scheme [24]. This significantly reduces the timestep necessary for numerical accuracy, canceling a good part of the advantage gained from the switch to an implicit time integration method. It is possible to construct schemes that overcome this problem and actually use the correct boundary conditions [14]. While these schemes avoid the large splitting errors of the conventional fractional step methods, they are very memory intensive [17, 21].

The third approach avoids the calculation of the pressure altogether by taking the curl of the momentum equations. This results in a set of evolution equations for the vorticity $\vec{\omega}$ (the curl of the velocity). These evolution equations are augmented by a set of elliptic equations relating the vorticity either to the velocities or to a stream function $\vec{\psi}$. A key advantage of the vorticity formulation is that, if properly implemented, the wall vorticity can be calculated with the full spatial and temporal accuracy of the numerical scheme. This is in contrast to the fractional step method, where the numerical accuracy can be substantially reduced near the wall [13]. In many applications, the wall vorticity is a paramount quantity that is essential for capturing the physics of viscous flows. In these cases, a vorticity method would allow for higher numerical accuracy with a given spatial and temporal discretization. Alternatively, it would allow a reduction of the number of gridpoints and timesteps to obtain the desired accuracy.

Vorticity methods are particularly attractive in two dimensions, where the number of variables can be reduced from three (u, v, p) to two (ω, ψ). In three dimensions, however, the number of variables actually increases, from four (u, v, w, p) to six $[(\omega_x, \omega_y, \omega_z, u, v, w)$ or $(\omega_x, \omega_y, \omega_z, \psi_x, \psi_y, \psi_z)]$. Another important drawback of the vorticity formulation is that there are no boundary conditions for the vorticity on a solid wall. This is inconsequential for flows without solid boundaries (e.g., jets, wakes, free shear layers). It can be, however, a serious impediment for calculations of boundary layer flows. One way around this problem, at least for simple geometries, is to use a fully explicit method for the time integration. This introduces another potential drawback in flows that require a very fine resolution in the wall-normal direction, namely, a severe restriction of the timestep due to numerical instability. For calculations of turbulent boundary layers, when the necessary spatial resolution near the wall becomes very fine, the timestep limit due to numerical stability may be substantially smaller than the timestep necessary for numerical accuracy. In these cases, an implicit scheme would be more desirable. Recent higher-order accurate finite difference schemes for the vorticity formulation are given in [10] for implicit time integration, [15] for explicit time integration, and [7–9] for compact differences.

The first researcher to successfully use the vorticity–velocity approach was Fasel (for references see [10]). He investigated the early (two-dimensional) stages of boundary layer transition. His method was second-order accurate in space and time, using finite differences for the spatial derivatives and a fully implicit scheme for the time integration. The resulting difference equations were solved with a direct method in the wall-normal (y) direction, with iteration in the streamwise (x) direction. This scheme coupled the implicit integration of the vorticity transport equation with the calculation of the velocities. The iteration loop of the difference equations was combined with the iterative calculation of the wall vorticity. At the outflow boundary, a radiation condition was imposed on the second derivative in x of all variables. This condition allowed waves with one specified streamwise wavenumber to pass through the outflow boundary without severe reflections.

The basic numerical method was extended to three dimensions in [10] to investigate the later stages of transition in a flat-plate boundary layer. The numerical scheme of [10] used fourth-order accurate finite differences in x and y and Fourier collocation in the spanwise direction z. The time integration was still carried out by a fully implicit scheme, with the radiation condition at the outflow boundary. In later studies, the implicit scheme was replaced by a fully explicit scheme, and a buffer domain was introduced to suppress disturbances before they could reach the outflow boundary [15].

The principal application of the current method is the direct numerical simulation of transition and turbulence in wall-bounded shear flows. While the fundamental equations are unchanged from those of [15], several new numerical techniques have been introduced that lead to substantial improvement of accuracy and speed. These techniques include nonequidistant differences in the wall-normal direction; split-compact differences in the streamwise direction; a new, fast iteration for a semi-implicit time integration of the wall-normal diffusion terms; and an improvement of the buffer domain technique to prevent reflections of waves at the outflow boundary.

In Section 2, the governing equations are presented. In Section 3, the numerical model is described in detail, including analyses of the new techniques listed above. In Section 4, results of several test calculations are presented to demonstrate the accuracy and convergence of the numerical method.

2. GOVERNING EQUATIONS

The governing equations are the incompressible, unsteady Navier–Stokes equations with constant density and viscosity. They consist of three momentum equations for the velocity components u, v, w in the streamwise (x), normal (y), and spanwise (z) directions, respectively,

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\,\vec{u} + \nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \vec{u}, \tag{1}$$

and the continuity equation (conservation of mass)

$$\nabla \cdot \vec{u} = 0. \tag{2}$$

In these equations, the velocities are normalized by the free-stream velocity U_∞. The spatial variables x, y, z are normalized by a reference length L, and the time t is normalized by U_∞/L. The global Reynolds number is defined as $\mathrm{Re} = U_\infty L/\nu$.

We define the vorticity as $\vec{\omega} = -\nabla \times \vec{u}$ (i.e., the negative curl of the velocity),

$$\omega_x = \frac{\partial v}{\partial z} - \frac{\partial w}{\partial y}, \qquad \omega_y = \frac{\partial w}{\partial x} - \frac{\partial u}{\partial z}, \qquad \omega_z = \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}. \tag{3}$$

Taking the curl of the momentum equations (1) eliminates the pressure gradient. Using the fact that both the velocity and the vorticity vectors are solenoidal, one obtains three vorticity transport equations for the streamwise (ω_x), normal (ω_y), and spanwise (ω_z) components of the vorticity:

$$\frac{\partial \omega_x}{\partial t} = -\frac{\partial a}{\partial y} + \frac{\partial c}{\partial z} + \frac{1}{\mathrm{Re}} \nabla^2 \omega_x \tag{4a}$$

$$\frac{\partial \omega_y}{\partial t} = \frac{\partial a}{\partial x} - \frac{\partial b}{\partial z} + \frac{1}{\mathrm{Re}} \nabla^2 \omega_y \tag{4b}$$

$$\frac{\partial \omega_z}{\partial t} = -\frac{\partial c}{\partial x} + \frac{\partial b}{\partial y} + \frac{1}{\mathrm{Re}} \nabla^2 \omega_z. \tag{4c}$$

The nonlinear terms resulting from convection and vortex stretching are

$$a = v\,\omega_x - u\,\omega_y \tag{5a}$$

$$b = w\,\omega_y - v\,\omega_z \tag{5b}$$

$$c = u\,\omega_z - w\,\omega_x. \tag{5c}$$

This formulation of the nonlinear terms follows the approach taken in [10]. It has the advantage of minimizing the number of Fourier transforms required for a pseudospectral computation of these terms.

From the definition of the vorticity, and again using the fact that both the velocity and the vorticity vectors are solenoidal, one obtains three equations for the velocity components

$$\nabla^2 v = \frac{\partial \omega_x}{\partial z} - \frac{\partial \omega_z}{\partial x} \tag{6a}$$

$$\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial z^2} = \frac{\partial \omega_y}{\partial x} - \frac{\partial^2 v}{\partial y \partial z} \tag{6b}$$

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial z^2} = -\frac{\partial \omega_y}{\partial z} - \frac{\partial^2 v}{\partial x \partial y}. \tag{6c}$$

When used together with an appropriate finite-difference discretization of the x, y derivatives (see Section 3), this formulation of the velocity equations does not require the vorticity values ω_x, ω_z at the wall for the calculation of the right-hand sides of Eqs. (6a)–(6c). The calculation of the wall vorticity will be discussed in Section 3.1.

The flow is assumed to be periodic in the spanwise direction z. In the calculations presented here, the flow is also taken to be symmetric w.r.t. $z = 0$. Therefore, the flow field is expanded in real Fourier cosine and sine series with K spanwise Fourier modes,

$$(u, v, \omega_z, b, c) = \sum_{k=0}^{K} (U_k, V_k, \Omega_{zk}, B_k, C_k) \cos(\gamma_k z) \tag{7a}$$

$$(w, \omega_x, \omega_y, a) = \sum_{k=1}^{K} (W_k, \Omega_{xk}, \Omega_{yk}, A_k) \sin(\gamma_k z), \tag{7b}$$

where the spanwise wavenumber is

$$\gamma_k = \frac{2\pi k}{\lambda_z}, \tag{8}$$

and λ_z is the spanwise wavelength of the lowest spanwise Fourier mode. Substitution of these expansions into the vorticity transport equations (4a), (4b), (4c) and the velocity equations (6a), (6b), (6c) yields the governing equations in Fourier space

$$\frac{\partial \Omega_{xk}}{\partial t} = -\frac{\partial A_k}{\partial y} - \gamma_k C_k + \frac{1}{\mathrm{Re}} \nabla_k^2 \Omega_{xk} \tag{9a}$$

$$\frac{\partial \Omega_{yk}}{\partial t} = \frac{\partial A_k}{\partial x} + \gamma_k B_k + \frac{1}{\mathrm{Re}} \nabla_k^2 \Omega_{yk} \tag{9b}$$

$$\frac{\partial \Omega_{zk}}{\partial t} = -\frac{\partial C_k}{\partial x} + \frac{\partial B_k}{\partial y} + \frac{1}{\mathrm{Re}} \nabla_k^2 \Omega_{zk} \tag{9c}$$

$$\frac{\partial^2 V_k}{\partial x^2} + \frac{\partial^2 V_k}{\partial y^2} - \gamma_k^2 V_k = \gamma_k \Omega_{xk} - \frac{\partial \Omega_{zk}}{\partial x} \tag{10a}$$

$$\frac{\partial^2 W_k}{\partial x^2} - \gamma_k^2 W_k = \frac{\partial \Omega_{yk}}{\partial x} + \gamma_k \frac{\partial V_k}{\partial y} \tag{10b}$$

$$\frac{\partial^2 U_k}{\partial x^2} - \gamma_k^2 U_k = -\gamma_k \Omega_{yk} - \frac{\partial^2 V_k}{\partial x \partial y}, \tag{10c}$$

where the Laplacian operator ∇^2 is transformed into

$$\nabla_k^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} - \gamma_k^2. \tag{11}$$

The nonlinear terms A_k, B_k, C_k of the vorticity transport equations are evaluated pseudospectrally, using fast Fourier transforms [28] to convert from Fourier space (x, y, k) to physical space (x, y, z) and back. To avoid aliasing errors, the values of a, b, c in physical space are calculated on $3/2K$ spanwise collocation points [23].

3. NUMERICAL MODEL

3.1. *Boundary Conditions*

The governing equations (9a)–(10c) are solved inside a rectangular integration domain $x_0 \leq x \leq x_{\max}$, $0 \leq y \leq y_{\max}$, with periodicity in the spanwise direction z. The computational domain is shown schematically in Fig. 1. The numerical method is used to simulate spatially developing, unsteady wall-bounded shear flows. Thus, fluid enters the computational domain at the inflow boundary at $x = x_0$ and exits at the outflow boundary at $x = x_{\max}$.

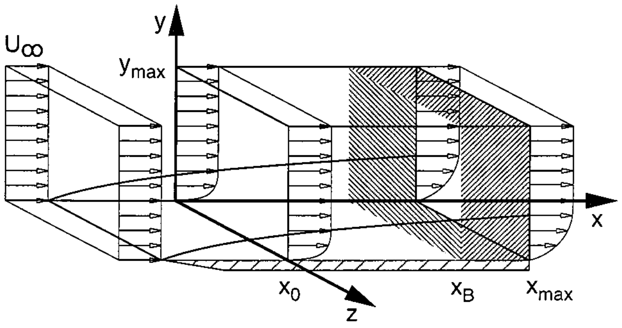

FIG. 1. Sketch of computational domain.

Inflow boundary conditions. At the inflow boundary at $x = x_0$, all velocity and vorticity components are specified. In addition, all x derivatives needed for the compact-difference approximations of the governing equations are also specified. Imposing derivatives at the boundaries may appear to be an overspecification of the problem. However, apart from solid walls and boundaries "at infinity," any computational boundary is necessarily a cut through the physical flow field. Consequently, the numerical boundary conditions specified at such a boundary should take into account the physics of the flow. The issue of proper boundary conditions for computational fluid dynamics has been hotly debated among fluid mechanics scholars, particularly in the context of simulations of transitional and turbulent flows. Bertolotti [4] argued that realizable boundary conditions for the Navier–Stokes equations should be a cut (e.g., at $x = x_0$) of a flow field that is itself a solution of the Navier–Stokes equations. Morkovin [22] called for environmentally realizable disturbances, i.e., for boundary conditions that can be causally linked to disturbances that occur in nature. At the heart of the matter lies a discrepancy between boundary conditions that are permissible in obtaining a mathematically well-posed problem and boundary conditions that are "physically meaningful." On the one hand, one may impose mathematically proper inflow boundary conditions that lead to a unique and numerically stable solution that cannot be physically realized in any experiment. An example of this type of boundary condition are the inflow conditions specified in certain numerical simulations of transient growth of disturbances in boundary layers [12]. On the other hand, if a flow is known to be a physically meaningful solution of the Navier–Stokes equation, then the derivatives of the relevant variables (velocity, vorticity) are also known. Thus, one could reasonably expect that the consistent specification of additional derivatives at the boundaries should not cause numerical problems. As an example, the parabolized stability equations (PSE) require inflow boundary conditions that specify, in fact, the dependent variables and their first two streamwise derivatives [3].

There is yet another point to consider when a disturbance at the inflow boundary leads to a transient in the flowfield: If reflected at the outflow boundary, such a transient can cause waves to be trapped inside the computational domain. At the very least, these waves will corrupt the solution for a long time; at worst, they might cause the numerical solution to grow without bounds. This underlines the need for a suitable damping region near the outflow boundary.

In the unsteady calculations presented in Section 4, the steady part of the flow at the inflow boundary is taken as the solution of the Blasius boundary layer equations; hence, all derivatives are known and can be specified in a consistent manner. Moreover, since the calculations are usually started with the Blasius solution as the initial condition, the flow at the inflow boundary is also consistent with the initial flow field. In these calculations, time-harmonic Tollmien–Schlichting waves are specified at the inflow boundary. Since these waves are valid solutions of the Navier–Stokes equations (or of some appropriate approximation, such as PSE) all derivatives can be consistently specified. However, if such a periodic solution is suddently imposed as an inflow boundary condition for an otherwise steady Blasius boundary layer, there will be an initial transient adjustment of the flow, until periodicity is attained. While this transient adjustment is a valid solution of the Navier–Stokes equation, it is unphysical, because it cannot be realized in an experiment. Thus, in this case, only the periodic results, after the initial transient, can be considered physically meaningful, since they can be reproduced in an experiment.

Wall boundary conditions. At the wall at $y = 0$, no-slip conditions are imposed on U_k and W_k, while V_k can be arbitrarily specified to model suction or blowing through the wall. In addition to prescribing V_k, $\partial V_k/\partial y = 0$ is imposed at the wall to ensure conservation of mass. This follows from the continuity equation (2).

A crucial aspect of the vorticity–velocity formulation is the fact that there are no proper boundary conditions for the vorticity; i.e., the vorticity values at the wall cannot be arbitrarily specified or computed from the vorticity transport equations (4a), (4b), (4c). Rather, they should be computed from the velocity fields to maintain consistency and ensure overall conservation of mass and zero-divergence of the vorticity field. The following relations are used to evaluate the vorticity at the wall $y = 0$:

$$\frac{\partial^2 \Omega_{xk}}{\partial x^2} - \gamma_k^2 \Omega_{xk} = -\frac{\partial^2 \Omega_{yk}}{\partial x \partial y} - \gamma_k \left(\frac{\partial^2 V_k}{\partial x^2} + \frac{\partial^2 V_k}{\partial y^2} - \gamma_k^2 V_k \right) \tag{12a}$$

$$\Omega_{yk} = 0 \tag{12b}$$

$$\frac{\partial \Omega_{zk}}{\partial x} = \gamma_k \Omega_{xk} - \left(\frac{\partial^2 V_k}{\partial x^2} + \frac{\partial^2 V_k}{\partial y^2} - \gamma_k^2 V_k \right). \tag{12c}$$

Equation (12a) is obtained by taking the x-derivative of the divergence of the vorticity $\partial/\partial x \nabla \cdot \vec{\omega}$ and eliminating the spanwise vorticity component via the z-derivative of Eq. (10a). Equation (12b) follows from the definition of the normal vorticity (3) together with the no-slip boundary conditions for the velocities at the wall. Given the normal velocity V_k and the normal vorticity Ω_{yk}, their derivatives can be computed at the wall $y = 0$. The streamwise vorticity Ω_{xk} is then computed by solving Eq. (12a). Once Ω_{xk} is known, Ω_{zk} is computed by integration of (12c), starting at the inflow boundary.

Free-stream boundary conditions. At the free-stream boundary at $y = y_{\max}$ the flow is assumed to be irrotational. This assumption is usually satisfied to machine precision in numerical calculations. Thus, all vorticity components and their derivatives are set to zero. A Robin boundary condition is specified for the disturbance velocity V_k,

$$\left. \frac{\partial V_k}{\partial y} \right|_{y_{\max}} = -\alpha_M V_k. \tag{13}$$

This condition imposes exponential decay $V_k \propto \exp(-\alpha_M y)$ of disturbances at the free stream. In the case of a Tollmien–Schlichting (TS)-wave, this exponential decay follows from linear stability theory, where α_M is the wavenumber of the TS-wave. For sufficiently large $y_{\max}$ the solution is quite insensitive to the value of α_M.

Outflow boundary conditions. In Eq. (10c) for the streamwise velocity U_k, a Neumann boundary condition is used based on the continuity equation (2) to ensure global conservation of mass:

$$\frac{\partial U_k}{\partial x} = -\frac{\partial V_k}{\partial y} - \gamma_k W_k. \tag{14}$$

In all other equations, the second derivatives in x are set to zero at the outflow boundary.

3.2. *x-Derivatives*

In the streamwise direction x the grid points are equally spaced from $i = 1$ at the inflow at $x = x_0$ to $i = m1$ at the outflow at $x = x_{\max}$. Hence, $x(i) = x_0 + (i - 1)\,\Delta x$. Inside, from $i = 2$ to $i = m1 - 1$, all x-derivatives are approximated with fourth-order compact differences, except for the derivative of the nonlinear terms, as discussed below.

x-Derivatives of nonlinear terms. The streamwise derivatives $\partial A_k/\partial x$ and $\partial C_k/\partial x$ in the vorticity transport equations (9b) and (9c) are approximated by split compact differences,

$$\begin{aligned}&\frac{1}{6}((2 - \mathrm{w}_c)\, f'_{i-1} + 4 f'_i + \mathrm{w}_c\, f'_{i+1})\\ &\quad = \frac{1}{6\Delta x}(-(5 - 2\mathrm{w}_c)\, f_{i-1} + 4(1 - \mathrm{w}_c)\, f_i + (1 + 2\mathrm{w}_c)\, f_{i+1})\\ &\quad\quad \times \left[-(1 - \mathrm{w}_c)\frac{1}{36}\frac{\partial^4 f}{\partial x^4}(\Delta x)^3 + \mathrm{w}_c\frac{1}{180}\frac{\partial^5 f}{\partial x^5}(\Delta x)^4 + \cdots\right],\end{aligned} \tag{15}$$

$$\begin{aligned}&\frac{1}{6}(\mathrm{w}_c\, f'_{i-1} + 4 f'_i + (2 - \mathrm{w}_c)\, f'_{i+1})\\ &\quad = \frac{1}{6\Delta x}(-(1 + 2\mathrm{w}_c)\, f_{i-1} - 4(1 - \mathrm{w}_c)\, f_i + (5 - 2\mathrm{w}_c)\, f_{i+1})\\ &\quad\quad \times \left[+(1 - \mathrm{w}_c)\frac{1}{36}\frac{\partial^4 f}{\partial x^4}(\Delta x)^3 + \mathrm{w}_c\frac{1}{180}\frac{\partial^5 f}{\partial x^5}(\Delta x)^4 + \cdots\right],\end{aligned} \tag{16}$$

where the subscript i is the index in x direction and w_c is a weighting factor between 0 (fully biased differences) and 1 (central compact differences). At consecutive substeps of the four-stage Runge–Kutta scheme (Fig. 2), the numerical scheme alternates between upwind-biased differences (Eq. (15)) and downwind-biased differences (Eq. (16)). For example, when upwind-biased differences are used to compute f'_0, f'_{ii}, downwind-biased differences are used to compute f'_i, f'_{iii}. Since the equations are nonlinear and coupled, the order of the biasing itself is reversed at every other time step to avoid any undesired overall biasing.

When the four-stage classical explicit Runge–Kutta scheme is used for the time integration, the biasing factor w_c is set to a suitable value between 0 and 1. When any other scheme is used for the time integration, the biasing factor is set to $\mathrm{w}_c = 1$; i.e., the derivatives are approximated by central compact differences.

The average of the two difference formulae (15) and (16) is the usual central compact difference formula for the first derivative. However, when used in this split form, they provide a much better approximation than the usual central difference formula.

First, note that the leading order terms of the truncation error of the two formulae are equal in magnitude and opposite in sign. Since they are used at consecutive substeps of the

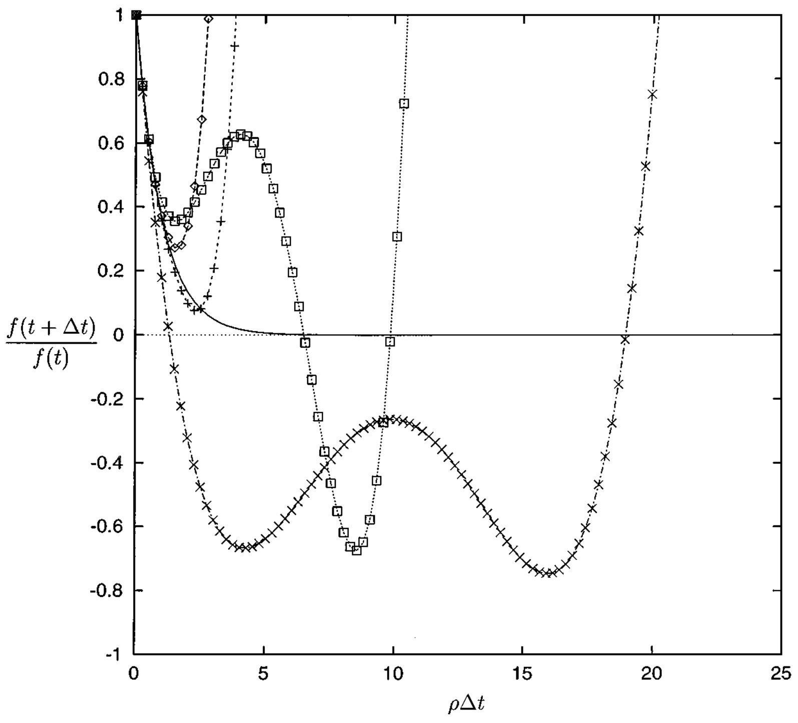

FIG. 2. Amplification $f(t+\Delta t)/f(t)$ after one Runge–Kutta step, plotted over $\rho\Delta t$. ($\times$) first order; ($\square$) second order; ($+$) third order; ($\diamond$) fourth order; (—) exact solution.

Runge–Kutta scheme, one can write the leading term of the truncation error as

$$\begin{aligned}
&-(1-\mathrm{w}_c)\frac{1}{36}\frac{\partial^4 f(t)}{\partial x^4}(\Delta x)^3 + (1-\mathrm{w}_c)\,\frac{1}{36}\,\frac{\partial^4 f(t+\Delta t)}{\partial x^4}(\Delta x)^3 \\
&= -(1-\mathrm{w}_c)\frac{1}{36}\frac{\partial^4 f(t)}{\partial x^4}(\Delta x)^3 + (1-\mathrm{w}_c)\frac{1}{36}\frac{\partial^4 f(t)}{\partial x^4}(\Delta x)^3 \\
&\quad + (1-\mathrm{w}_c)\frac{1}{36}\frac{\partial^5 f(t)}{\partial x^4 \partial t}(\Delta x)^3(\Delta t) + \mathcal{O}((\Delta x)^3(\Delta t)^2) \\
&\approx (1-\mathrm{w}_c)\frac{1}{36}\frac{\partial^5 f(t)}{\partial x^4 \partial t}(\Delta x)^3(\Delta t).
\end{aligned} \tag{17}$$

Hence, the method is still formally fourth-order accurate. A further understanding of this method can be gained by analyzing its dispersion relation. Consider the model equation

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = 0 \tag{18}$$

with periodic boundary conditions in x. One can then apply a Fourier transform in x to obtain

$$\frac{d\hat{f}}{dt} + \mathrm{i}\alpha U\hat{f} = 0, \tag{19}$$

where $\mathrm{i} = \sqrt{-1}$ and α is the streamwise wavenumber. The solution of this equation after one time step Δt is

$$\frac{\hat{f}(t+\Delta t)}{\hat{f}(t)} = \mathrm{e}^{-\mathrm{i}\alpha U \Delta t} \equiv \mathrm{e}^{-\mathrm{i}\omega\Delta t}, \tag{20}$$

where ω is the circular frequency.

The dispersion relation of the exact solution is

$$\frac{\omega}{\alpha} = U, \tag{21}$$

where α and ω are real numbers. Note that all waves have the same phase speed and that they are neither amplified nor damped. Also, the group velocity of the exact solution

$$c_g = \frac{d\omega}{d\alpha} = U \tag{22}$$

is independent of the wavenumber.

When Eq. (18) is integrated numerically, i.e., with finite differences in x and a Runge–Kutta method in t, the solution will be different. Instead of Eq. (20), we expect a solution of the form

$$\frac{\hat{f}(t+\Delta t)}{\hat{f}(t)} = \mathrm{e}^{-\mathrm{i}\bar{\alpha} U \Delta t} \equiv \mathrm{e}^{-\mathrm{i}\bar{\omega}\Delta t}. \tag{23}$$

The dispersion relation of the numerical scheme can be written as

$$\frac{\bar{\omega}(\alpha, \mathrm{CFL})\Delta t}{\mathrm{CFL}} = \alpha\Delta x, \tag{24}$$

where

$$\mathrm{CFL} = \frac{U\Delta t}{\Delta x} \tag{25}$$

is the Courant–Friedrichs–Levy number.

The modified frequency $\bar{\omega}$ is now generally complex and depends nonlinearly on the wavenumber and the CFL number. A positive imaginary part $\bar{\omega}_i$ corresponds to exponential damping of waves, in contrast to the properties of the exact solution. A negative imaginary part $\bar{\omega}_i$ corresponds to exponential growth, i.e., to numerical instability.

The weighting factor w_c in Eqs. (15) and (16) can be adjusted to provide "optimal" damping of numerical errors in the sense that grid-mesh oscillations with a wavenumber $\alpha = \pi/\Delta x$ are completely eliminated. For a given CFL number, this "optimum" value of w_c can be found from

$$\mathrm{w}_c = \frac{\mathrm{CFL} - \frac{1}{8}\sqrt{24 - \sqrt{192}}}{\mathrm{CFL}}. \tag{26}$$

This relation holds so long as the values of CFL and w_c are within the stability limits of the scheme. The stability boundary $\mathrm{CFL}_{\max}$ vs w_c is plotted in Fig. 3. For a given biasing factor w_c, CFL numbers above the curve will lead to numerical instability. Note that stronger biasing, i.e., lower w_c, will reduce the allowable timestep for a given spatial step.

In Fig. 4 the normalized imaginary part of the modified frequency, $\bar{\omega}_i\Delta t/\mathrm{CFL}$, is plotted vs the normalized wavenumber, $\alpha\Delta x$, for several fourth-order accurate schemes: standard five-point central differences, compact central differences, compact split differences with

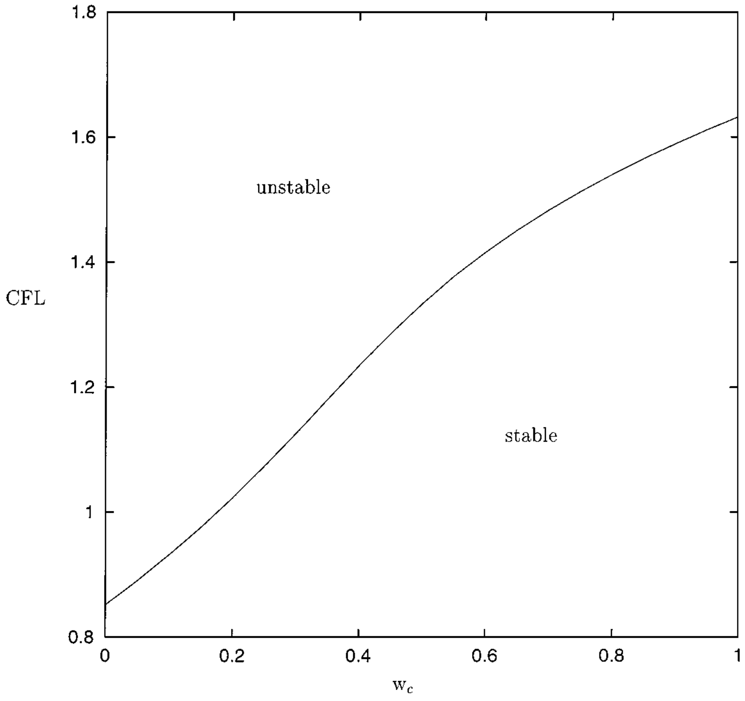

FIG. 3. Stability boundary $\mathrm{CFL}_{\max}$ vs weighting factor w_c of the fourth-order Runge–Kutta scheme with split-biased compact differences. The numerical scheme is stable for parameter combinations (CFL, w_c) below the curve $\mathrm{CFL}_{\max}$.

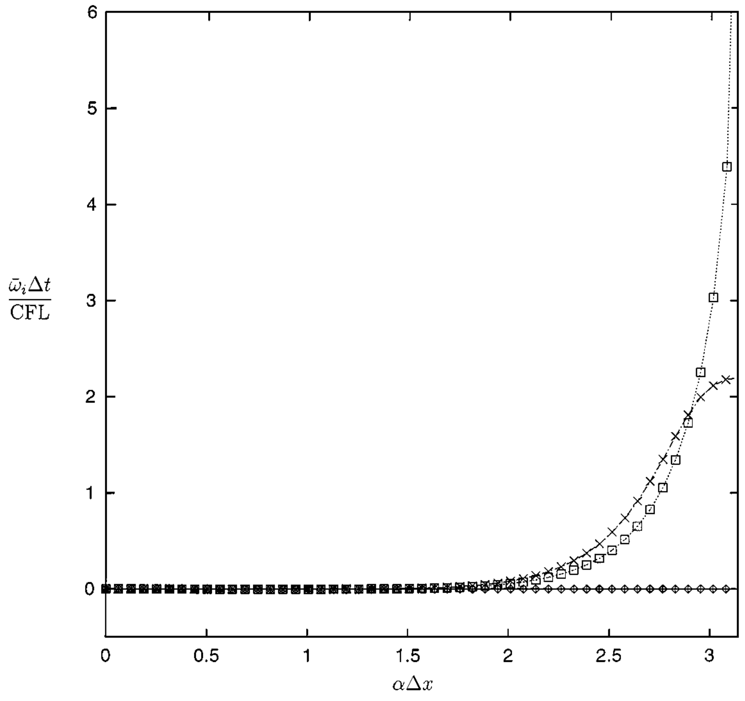

FIG. 4. Normalized imaginary part of modified frequency $\bar{\omega}_i \Delta t/\mathrm{CFL}$, plotted vs normalized wavenumber $\alpha\Delta x$. Time integration with the fourth-order Runge–Kutta scheme, spatial differentiation with the following fourth-order finite-difference schemes: (◇) standard central; (+) compact central; (□) weighted compact split; (×) compact split; (—) exact solution.

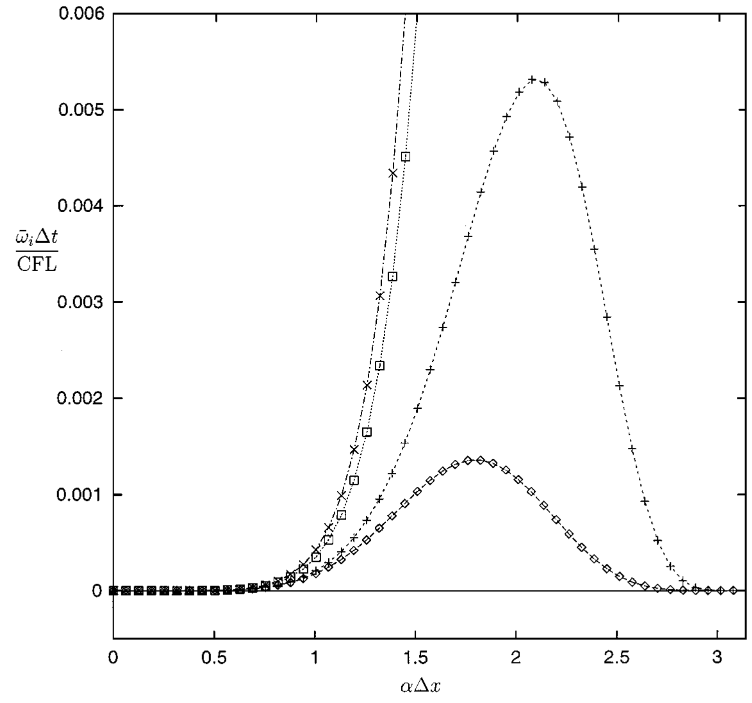

FIG. 5. Detail of normalized imaginary part of modified frequency $\bar{\omega}_i \Delta t/\mathrm{CFL}$, plotted vs normalized wavenumber $\alpha\Delta x$. Time integration with the fourth-order Runge–Kutta scheme, spatial differentiation with the following fourth-order finite-difference schemes: (◇) standard central; (+) compact central; (□) weighted compact split; (×) compact split; (—) exact solution.

$\mathrm{w}_c = 0$, and weighted compact split differences with $\mathrm{w}_c = 0.2$. In each case, the fourth-order Runge–Kutta scheme is used for time integration. The CFL number in this example is CFL $= 0.5$, and the optimal weighting factor according to Eq. (26) is $\mathrm{w}_c = 0.2$. Indeed, the two curves with split compact differences show strong damping of waves with wavenumbers $\alpha\Delta x > 2$, i.e., of waves with a resolution of fewer than three points per wavelength.

Figure 5 shows a more detailed view of the previous graph near the ordinate axis. Apparently, the split compact differences cause stronger damping of waves than the central differences. With a biasing factor of $\mathrm{w}_c = 0.2$, a wave with wavenumber $\alpha\Delta x = \pi/2$, i.e., with four grid points per wavelength and 8 time steps per period, loses 2.4% of its amplitude over each period. However, at a finer resolution with a wavenumber $\alpha\Delta x = 1$ (six points per wavelength, 12 steps per period), a wave loses less than 0.3% of its amplitude per period. Most importantly, these damping losses can be made arbitrarily small by reducing the CFL number, i.e., by reducing the time step Δt. Note also that the central difference formulae do not cause any damping at the highest wavenumbers.

The normalized real part of the modified frequency, $\bar{\omega}_r \Delta t/\mathrm{CFL}$, is plotted in Fig. 6. With standard central differences, the numerical solution is seen to depart from the correct solution for wavenumbers $\alpha\Delta x > 1$. It reaches a maximum at $\alpha\Delta x = 1.82$ and returns to zero for higher wavenumbers. This indicates that waves with a resolution of fewer than six points per wavelength have the wrong phase speed; they lag the correct solution. More importantly, the group velocity of these underresolved waves reaches zero at $\alpha\Delta x = 1.82$. This means that any numerical error at this wavenumber will not propagate at all. Worst of all,

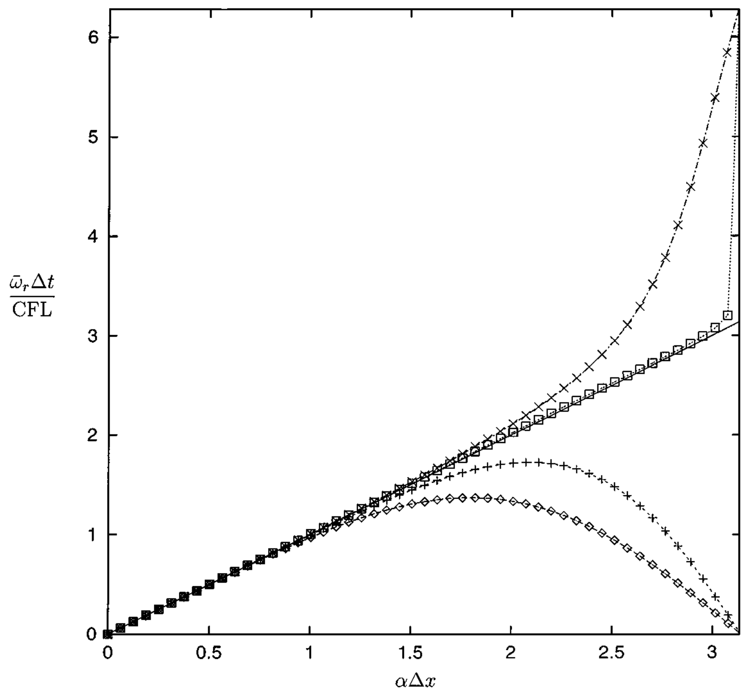

FIG. 6. Normalized real part of modified frequency $\bar{\omega}_r \Delta t/\mathrm{CFL}$, plotted vs normalized wavenumber $\alpha\Delta x$. Time integration with the fourth-order Runge–Kutta scheme, spatial differentiation with the following fourth-order finite-difference schemes: (◇) standard central; (+) compact central; (□) weighted compact split; (×) compact split; (—) exact solution.

the group velocity of the least resolved waves is negative; i.e., short-scale numerical errors will actually propagate upstream, with the shortest possible waves, grid-mesh oscillation with $\alpha\Delta x = \pi$, having the absolute largest (negative) group velocity. As shown in Fig. 5, these spurious waves are not damped at all. Since these properties are inherent in the spatial finite difference operator, only an increase in the number of spatial grid points can improve the accuracy of the solution. Thus, increasing the resolution of a physically meaningful wave from four gird points per wavelength to eight grid points per wavelength will clearly improve its accuracy. However, it will not affect any short scale numerical errors that may be caused by roundoff.

Matters are not much better for the central compact differences. On the positive side, the departure from the correct solution and the threshold of zero group velocity occur at higher wavenumbers, at $\alpha\Delta x > 1.5$ and $\alpha\Delta x = 2.07$, respectively. On the negative side, the group velocity of the shortest waves has a much larger (negative) value than in the case of central differences. Again, these waves are not damped.

In contrast, the split compact differences reproduce at least the correct sign of the group velocity; i.e., they do not cause upstream propagation of numerical errors. And with the "optimal" biasing factor of $\mathrm{w}_c = 0.2$, the phase and group velocities of all waves except for grid-mesh oscillations are very close to the correct values. Also, recall that in this case grid-mesh oscillations are completely damped.

In summary, the use of split compact differences can yield enormous improvements in accuracy over conventional compact (and standard) differences for short waves, i.e., for waves with 6 gridpoints per wavelength or less. On the other hand, there is no appreciable difference between the three methods (standard, compact, split-compact) for long waves with 10 or more gridpoints per wavelength, for a given CFL number. Since the computational effort for all three methods is about equal, the improved short-wave resolution comes at no extra cost.

3.3. *y-Derivatives*

In the wall-normal direction y an exponential stretching is used to cluster grid points near the wall [1, Eq. 5-216]:

$$y(j) = y_{\max}\frac{(\beta+1)-(\beta-1)\left(\frac{\beta+1}{\beta-1}\right)^{1-(j-1)/(m_y-1)}}{\left(\frac{\beta+1}{\beta-1}\right)^{1-(j-1)/(m_y-1)}+1}. \tag{27}$$

Here j is the index of the grid points in the y direction; i.e., $j=1$ is at the wall ($y=0$) and $j=m_y$ is at the free stream ($y=y_{\max}$), while β is a parameter to control the clustering of grid points. $\beta\to 1^+$ clusters all points at the wall, $\beta\to\infty$ distributes points on an equidistant grid. It is important to note that the grid stretching used here is not done by a coordinate transformation. Rather, the finite-difference approximations for the derivatives with respect to y are constructed for a nonequidistant grid. While this approach is tedious, it can yield higher accuracy than the traditional method of grid stretching by a coordinate transformation. Intuitively, this can be seen from the fact that, when a coordinate transformation is used, only one parameter (the metric) can be adjusted in a given finite-difference formula, while the technique used in this work allows the adjustment of all coefficients in the formula. For higher-order formulae with many coefficients, this should give a substantial improvement. This approach has been successfully used in aeroacoustics [11].

For example, the first (f') and second (f'') derivatives in the y-direction at a gridpoint j away from the boundaries are given by

$$\text{afdy}\, f'_{i-1} + \text{bfdy}\, f'_i + \text{cfdy}\, f'_{i+1} = \text{ardy}\, f_{i-1} + \text{brdy}\, f_i + \text{crdy}\, f_{i+1}, \tag{28}$$

where

$$\text{afdy} = \frac{r^3(r+1)}{2}, \qquad \text{dfdy} = \frac{r(r+1)^3}{2}, \qquad \text{cfdy} = \frac{r(r+1)}{2} \tag{29a}$$

$$\text{ardy} = -\frac{r^3(r+2)}{\Delta_j}, \qquad \text{brdy} = \frac{(r-1)(r+1)^3}{\Delta_j}, \qquad \text{crdy} = \frac{(2r+1)}{\Delta_j}, \tag{29b}$$

and

$$\text{afd2y}\, f''_{j-1} + \text{dfd2y}\, f''_j + \text{cfd2y}\, f''_{j+1} = \text{ard2y}\, f_{j-1} + \text{brd2y}\, f_j + \text{brd2y}\, f_{j+1}, \tag{30}$$

where

$$\text{afd2y} = -\frac{r(r^2-r-1)}{12}, \qquad \text{bfd2y} = \frac{(r+1)(r^2+3r+1)}{12}, \qquad \text{cfd2y} = \frac{r^2+r-1}{12} \tag{31a}$$

$$\text{ard2y} = \frac{r}{(\Delta_j)^2}, \qquad \text{brd2y} = -\frac{r+1}{(\Delta_j)^2}, \qquad \text{crd2y} = \frac{1}{(\Delta_j)^2}. \tag{31b}$$

In these equations $\Delta_j = y_j - y_{j-1}$, and $r = (y_{j+1}-y_j)/\Delta_j$.

To illustrate the benefits of specially constructed finite-difference approximations, consider the function

$$f(y) = \cos\left(10\frac{y}{y_{\max}}\right). \tag{32}$$

At $y = 0$, the first derivative is $df/dy = 0$. Typical grid parameters for the calculations in this work are $y_{\max} = 0.15$ and $m_y = 80$. Given the function values on the grid points, one can numerically calculate the derivative using the finite-difference formula

$$f_1' = \mathrm{hrdy}_1 f_1 + \mathrm{hrdy}_2 f_2 + \mathrm{hrdy}_3 f_3 + \mathrm{hrdy}_4 f_4 + \mathrm{hrdy}_5 f_5 + \mathrm{hrdy}_6 f_6, \tag{33}$$

where

$$\mathrm{hrdy}_1 = -\frac{h_2h_3h_4h_5 + h_1h_3h_4h_5 + h_1h_2h_4h_5 + h_1h_2h_3h_5 + h_1h_2h_3h_4}{h_1h_2h_3h_4h_5} \tag{34a}$$

$$\mathrm{hrdy}_2 = \frac{h_2h_3h_4h_5}{h_1(h_2-h_1)(h_3-h_1)(h_4-h_1)(h_5-h_1)} \tag{34b}$$

$$\mathrm{hrdy}_3 = -\frac{h_1h_3h_4h_5}{h_2(h_2-h_1)(h_3-h_2)(h_4-h_2)(h_5-h_2)} \tag{34c}$$

$$\mathrm{hrdy}_4 = \frac{h_1h_2h_4h_5}{h_3(h_3-h_1)(h_3-h_2)(h_4-h_3)(h_5-h_3)} \tag{34d}$$

$$\mathrm{hrdy}_5 = -\frac{h_1h_2h_3h_5}{h_4(h_4-h_1)(h_4-h_2)(h_4-h_3)(h_5-h_4)} \tag{34e}$$

$$\mathrm{hrdy}_6 = \frac{h_1h_2h_3h_4}{h_5(h_5-h_1)(h_5-h_2)(h_5-h_3)(h_5-h_4)} \tag{34f}$$

and $h_j = y_{j+1} - y_1$. In the limiting case $y_{j+1} - y_j = \Delta y = \text{const.}$, Eq. (33) reduces to

$$f_1' = \frac{1}{60\Delta y}(-137 f_1 + 300 f_2 - 300 f_3 + 200 f_4 - 75 f_5 + 12 f_6)\left[-\frac{1}{6}\frac{\partial^6 f}{\partial y^6}(\Delta y)^5 + \cdots\right]. \tag{35}$$

Alternatively, one could use Eq. (27) to define a coordinate transformation from the physical coordinate y to a mapped coordinate η, where

$$\eta = \frac{j-1}{m_y - 1} \tag{36}$$

such that the grid is equidistant in the mapped coordinate system. Here, $\Delta\eta = 1/(m_y - 1) = 0.012658$. The derivative can then be calculated as

$$\frac{df}{dy} = \frac{d\eta}{dy}\frac{df}{d\eta}, \tag{37}$$

where the transformation metric is given by

$$\frac{d\eta}{dy} = \frac{2\beta}{y_{\max}\left(\beta^2 - \left(1 - \frac{y}{y_{\max}}\right)^2\right)(\log(\beta+1) - \log(\beta-1))} \tag{38}$$

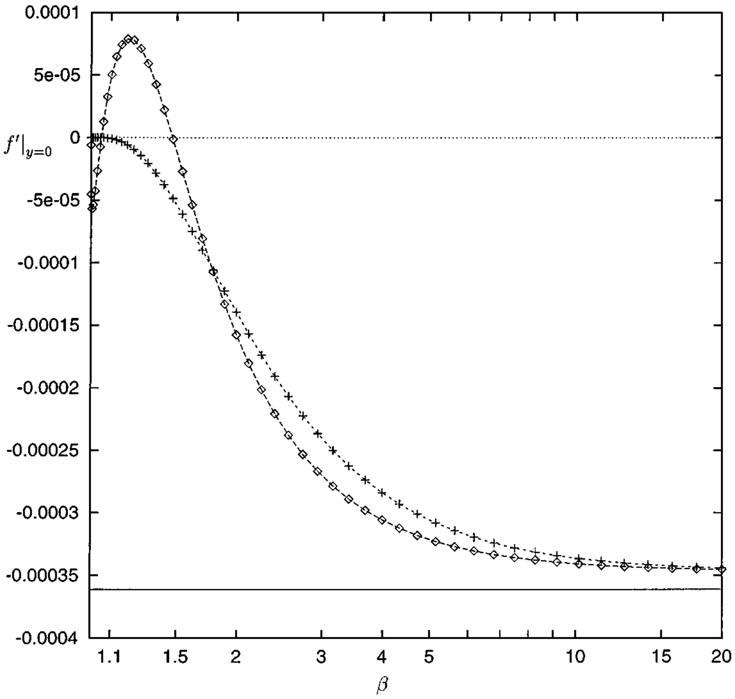

FIG. 7. Error of the numerical derivative d/dy of $f = \cos(10y/y_{\max})$ at the wall $y = 0$ with standard differences, plotted vs stretching parameter β. Calculation with 80 grid points. $(+)$ df/dy finite-difference derived for stretched grid; $(\diamond)$ df/dy coordinate transformation with finite difference for equidistant grid; (—) leading term of truncation error for equidistant grid.

and the finite-difference formula (35) can be used to calculate the derivative, with Δy replaced by $\Delta\eta$.

The results are plotted in Fig. 7. The first curve shows the numerical derivative computed from Eq. (33), plotted over the stretching parameter β. The second curve shows the numerical derivative computed according to Eq. (37) with the finite-difference coefficients from Eq. (35). For large values of β, as the grid approaches the limit of equidistant spacing, the two numerical results converge to an asymptotic limit. This limit is close to the leading term of the truncation error of Eq. (35), plotted as a straight line near the bottom of the graph. For smaller values of β, as the grid points become clustered near the wall, the accuracy of both numerical derivatives improves. However, while the error of Eq. (33) goes to zero as desired, the error of Eq. (37) does not. Rather, it oscillates about zero without reaching the proper limit.

To gain higher accuracy at the wall, one could also use a one-sided compact-difference approximation, such as

$$\text{qrdy}_1 f_1' + \text{qrdy}_2 f_2' + \text{qrdy}_3 f_3' + \text{qrdy}_4 f_4' = \text{prdy}_1 f_1 + \text{prdy}_2 f_2 + \text{prdy}_3 f_3 + \text{prdy}_4 f_4, \tag{39}$$

where

$$\text{qrdy}_1 = 1 \tag{40a}$$

$$\text{qrdy}_2 = \frac{h_2^2 h_3^2}{(h_2 - h_1)^2 (h_3 - h_1)^2} \tag{40b}$$

$$\mathrm{qrdy}_3 = \frac{h_1^2 h_3^2}{(h_2-h_1)^2(h_3-h_2)^2} \tag{40c}$$

$$\mathrm{qrdy}_4 = \frac{h_1^2 h_2^2}{(h_3-h_1)^2(h_3-h_2)^2} \tag{40d}$$

$$\mathrm{prdy}_1 = -\frac{2(h_2h_3+h_1h_3+h_1h_2)}{h_1h_2h_3} \tag{40e}$$

$$\mathrm{prdy}_2 = \frac{2h_2^2h_3^2\left(h_2h_3-2h_1h_3-2h_1h_2+3h_1^2\right)}{h_1(h_2-h_1)^3(h_3-h_1)^3} \tag{40f}$$

$$\mathrm{prdy}_3 = \frac{2h_1^2h_3^2\left(2h_2h_3-h_1h_3-3h_2^2+2h_1h_2\right)}{h_2(h_2-h_1)^3(h_3-h_2)^3} \tag{40g}$$

$$\mathrm{prdy}_4 = \frac{2h_1^2h_2^2\left(3h_3^2-2h_2h_3-2h_1h_3+h_1h_2\right)}{h_3(h_3-h_1)^3(h_3-h_2)^3} \tag{40h}$$

and $h_j = y_{j+1} - y_1$.

In the limiting case $y_{j+1} - y_j = \Delta y = \text{const.}$, Eq. (39) reduces to

$$\begin{aligned}&(3f_1' + 27f_2' + 27f_3' + 3f_4')\\ &\quad = \frac{1}{\Delta y}(-11f_1 - 27f_2 + 27f_3 + 11f_4)\left[+\frac{3}{140}\frac{\partial^7 f}{\partial y^7}(\Delta y)^6 + \cdots\right].\end{aligned} \tag{41}$$

In this case, the derivatives at the points $j = 2, 3, 4$ are known; thus, the derivative at $j = 1$ can be computed in a straightforward manner without solving a system of equations. The use of a coordinate transformation together with the equidistant formula (41) requires some care. The derivatives at $j = 1, 2, 3, 4$ must be scaled by the values of the transformation metric at these points, i.e.,

$$\begin{aligned}&\left(3\frac{d\eta}{dy}\bigg|_1 f_1' + 27\frac{d\eta}{dy}\bigg|_2 f_2' + 27\frac{d\eta}{dy}\bigg|_3 f_3' + 3\frac{d\eta}{dy}\bigg|_4 f_4'\right)\\ &\quad = \frac{1}{\Delta\eta}(-11f_1 - 27f_2 + 27f_3 + 11f_4).\end{aligned} \tag{42}$$

The leading term of the truncation error of Eq. (41) contains an odd derivative. Thus, to allow for a comparison of the numerical results with this term, the function used here is

$$f(y) = \sin\left(10\frac{y}{y_{\max}}\right). \tag{43}$$

The results of the compact-difference derivatives are plotted in Fig. 8. While the overall accuracy is two orders of magnitude better than that of the standard one-sided derivatives described above, the qualitative trend is the same. For large values of β, the error approaches the leading term of the truncation error for an equidistant grid. For small values of β, the error of Eq. (39) approaches zero, while the error of Eq. (42) does not.

These results confirm that, for higher-order differences, it is preferable to derive a finite-difference formula specifically for a stretched grid, rather than to use a coordinate transformation combined with an equidistant grid in computational space.

A full listing of all finite-difference approximations used is beyond the scope of this paper. The interested reader is referred to the listing given in [19].

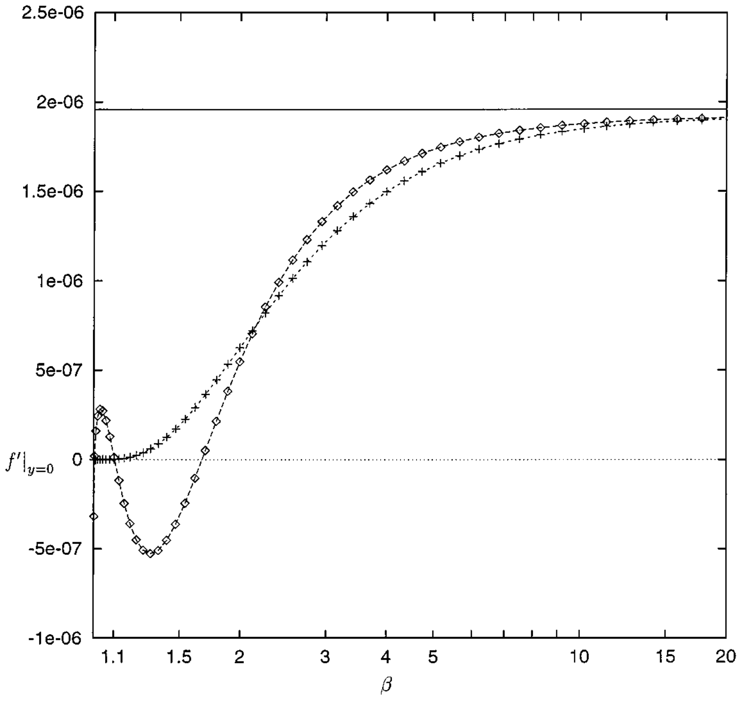

FIG. 8. Error of the numerical derivative d/dy of $f = \sin(10y/y_{\max})$ at the wall $y = 0$ with compact differences, plotted vs stretching parameter β. Calculation with 80 grid points. (+) df/dy finite-difference derived for stretched grid; (◇) df/dy coordinate transformation with finite difference for equidistant grid; (—) leading term of truncation error for equidistant grid.

3.4. *Solution of the Velocity Poisson Equations*

After the calculation of the vorticity, the normal velocity V_k is computed. At the outflow boundary, the second derivative in x on the left-hand side of Eq. (10a) is dropped, leaving an ordinary differential equation for V_k. This equation is discretized with compact differences in y. After solving for V_k at $i = m1$, these values are then used as a boundary condition for the solution inside the integration domain. To solve for the velocity inside the integration domain, Eq. (10a) is discretized with compact differences in x. Following Swarztrauber [27, 28], a Fourier sine transform is applied in x. This yields a set of ordinary differential equations in y for each Fourier sine component in x. These equations are then discretized with compact differences in y.

The Poisson equations (10b) for W_k and (10c) for U_k are also discretized with compact differences in x.

3.5. *Calculation of the Wall Vorticity*

For the calculation of the vorticity at the wall (Eqs. (12a) and (12c)), the Laplacian of V_k and the mixed derivative $\partial^2\Omega_{y_k}/\partial x \partial y$ are needed at the wall.

The Laplacian $\nabla_k^2 V_k$ at the wall can be computed from the Poisson equation for the V_k. At this stage in the calculation, V_k is known everywhere, and $\nabla_k^2 V_k \equiv \gamma_k \Omega_{xk} - \partial\Omega_{zk}/\partial x$ is known on the grid points $j = 2, 3, 4$. Thus, Eq. (10a) can be turned around to solve for $\nabla_k^2 V_k$ at the wall.

Since Ω_{yk} is zero at the wall, it would be straightforward to compute the derivative $\partial\Omega_{yk}/\partial y$ with one-sided differences, e.g., with Eq. (33). However, by using the compact differences (39), one can achieve substantially higher accuracy and also ensure a divergence-free vorticity vector at the wall. Taking the y-derivative of the divergence of the vorticity yields

$$\frac{\partial^2\Omega_{yk}}{\partial x\partial y} = -\frac{\partial^2\Omega_{xk}}{\partial x^2} + \gamma_k\frac{\partial\Omega_{zk}}{\partial x}. \tag{44}$$

Thus, $\partial^2\Omega_{yk}/\partial x\partial y$ can be computed on the grid points $j = 2, 3, 4$ with very high accuracy. Once the derivatives at these points are known, Eq. (39) can be used to solve for the derivative at the wall.

When $\partial^2\Omega_{yk}/\partial x\partial y$ and $\nabla_k^2 V_k$ at the wall have been calculated, Ω_{xk} can be computed by solving Eq. (12a). Finally, Ω_{zk} is computed by numerical integration of Eq. (12c), starting at the inflow boundary and marching downstream.

3.6. *Damping of Disturbances Near the Outflow Boundary*

The buffer domain technique is a very effective method for avoiding reflections of disturbance waves at the outflow boundary [15, 26]. Between $x = x_B$ and $x = x_{\max}$, the disturbance vorticity is gradually ramped down to zero using

$$f(\xi) = c(\xi)\, f_T(\xi), \tag{45}$$

where

$$\xi = \frac{x - x_{\max}}{x_B - x_{\max}}. \tag{46}$$

Here $f_T(\xi)$ is the vorticity as computed from the vorticity transport equation, before damping, $f(\xi)$ is the vorticity after damping, and $c(\xi)$ is a weighting function that varies smoothly from $c = 1$ at $\xi = 0$ to $c = 0$ at $\xi = 1$. The length of the buffer domain is $l_B = x_{\max} - x_B$. Kloker *et al.* [15] used a fifth-order polynomial for the weighting function $c(\xi)$ to ensure smooth first and second derivatives at the beginning and end of the damping,

$$c(\xi) = 1 - 6\xi^5 + 15\xi^4 - 10\xi^3. \tag{47}$$

This function is antisymmetric w.r.t. the midpoint of the buffer domain, i.e.,

$$c(1/2 + s) = 1 - c(1/2 - s), \qquad 0 < s < 1/2 \tag{48}$$

During the course of the calculation, this damping is performed at every stage of the Runge–Kutta time integration. The effect of applying this damping function n times can be written as

$$f(x) = c^n(x)\, f_T(x). \tag{49}$$

In Fig. 9, the function $c^n(\xi)$ is plotted vs ξ, for $n = 1$, $n = 50$, $n = 100$, and $n = 150$. While $c(\xi)$ varies smoothly between 1 and 0, repeated application of the damping causes a rapid

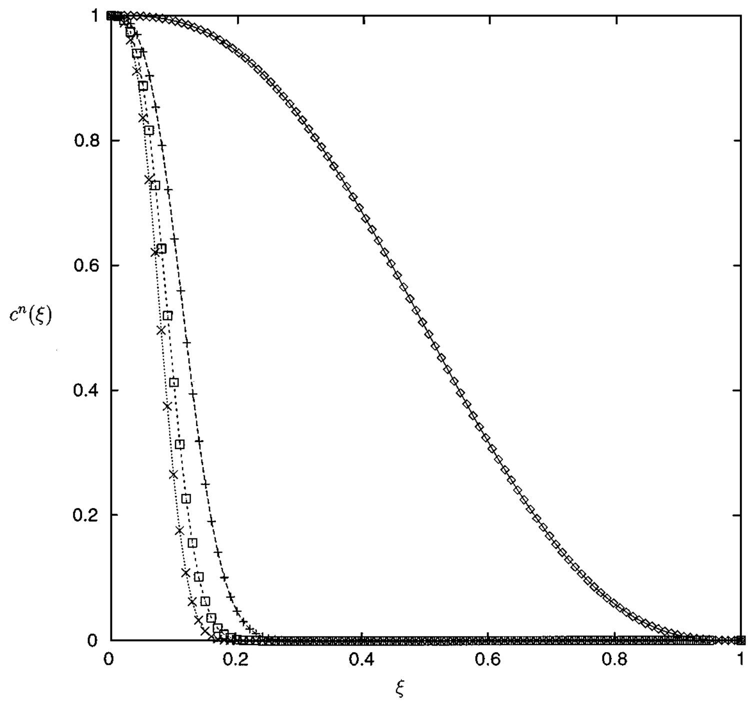

FIG. 9. Effect of applying the polynomial damping function $c(\xi) = 1 - 6\xi^5 + 15\xi^4 - 10\xi^3$ n times: (◇) $n = 1$; (+) $n = 50$; (□) $n = 100$; (×) $n = 150$.

drop at the beginning of the damping domain, as can be seen from the curves for larger n. This is clearly undesired, as such a sudden jump in the vorticity might act just like a boundary, causing reflections of waves. Since $f(\xi)$ in Eq. (45) consists of traveling waves, the dropoff due to repeated damping should be counterbalanced by the downstream propagation of these waves. In practice, the phase speed of waves in a boundary layer is on the order of one third of the free-stream speed. With a typical CFL number of 0.5, a wave travels one spatial step Δx in six time steps Δt. Thus, with the four-stage Runge–Kutta scheme, the damping function is applied 24 times for every step Δx a wave advances. The buffer domain extends typically over two wavelengths, e.g., $l_B = 40\Delta x$ for a wavelength $\lambda = 20\Delta x$. After one half period of $30\Delta t$, the wave has propagated $10\Delta x$ downstream into the buffer domain. After the damping has been applied $n = 120$ times, the wave has propagated a distance of $\xi = 0.25$. As seen from Fig. 9, this distance is not enough to counter the severe attenuation caused by the damping. Indeed, the figures in [15] show a very rapid change of the flow within the first few grid points of the buffer domain.

In some simulations of low-frequency free-stream vortices [20], the buffer domain technique with the damping function discussed above did not work. Waves were reflected from the junction at the upstream end of the buffer domain and destroyed the results inside the computational domain. Therefore, a new damping function was devised that took the convective nature of the flow into account:

$$c(\xi) = \mathrm{e}^{-\xi^4/10}(1 - \xi^{50})^4. \tag{50}$$

The constants in Eq. (50) were found through numerical experiments. The new damping

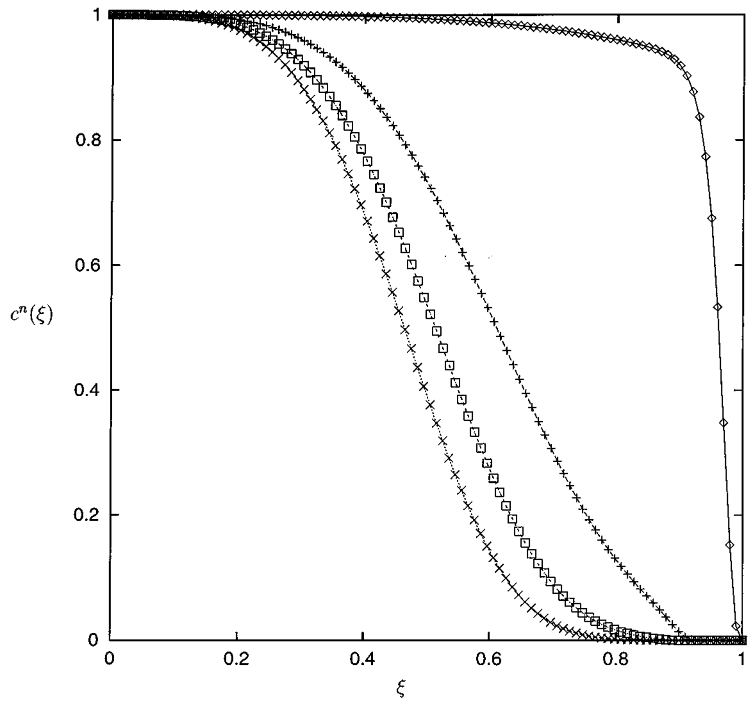

FIG. 10. Effect of applying the exponential damping function $c(\xi) = \exp(-x^4/10)(1-x^{50})^4$ n times: ($\diamond$) $n=1$; ($+$) $n=50$; ($\square$) $n = 100$; ($\times$) $n=150$.

function is plotted in Fig. 10, again for $n=1$, $n=50$, $n=100$, and $n=150$. While there is a steep dropoff near the end for $n=1$, the curves for higher values of n are much smoother than in the previous figure. This damping function has performed very well in calculations of many different unsteady flows.

However, one purpose of the present code is the calculation of three-dimensional steady flows to be used as base flows for subsequent unsteady calculations. For such flows, the normal buffer domain technique of ramping down the disturbance to zero near the outflow boundary may not be adequate, due to the elliptic nature of the steady flow. A solution to this problem is to use a weighted average of fourth-order compact differences and first-order upwind differences near the outflow boundary between $x = x_B$ and $x = x_{\max}$ when calculating the x-derivatives of the nonlinear terms in the vorticity-transport equations (9b), (9c),

$$\frac{\partial f}{\partial x} = c(\xi)\frac{\partial f(\xi)}{\partial x}_{\text{compact}} + (1 - c(\xi))\frac{\partial f(\xi)}{\partial x}_{\text{upwind}}, \tag{51}$$

where $f = A_k, C_k$, and the compact differences are calculated according to the method given in Section 3.2. The weighting function $c(\xi)$ used here is the same as used for the direct damping of the vorticity. This technique has worked exceedingly well even for very strong streamwise vortices [18], and it has no adverse effects on the flow upstream of the buffer domain. Hence, we have retained it in our code for all calculations.

3.7. *Filtering of the Vorticity in Streamwise Direction*

The numerical integration of Eq. (12c) along the wall introduces grid-mesh oscillation in x. When the streamwise derivatives of the nonlinear terms of the vorticity transport equations are computed by the split-compact differences (15) and (16), any grid-mesh oscillations are sufficiently damped. On the other hand, when the streamwise derivatives of the nonlinear terms are computed with central compact differences, those grid-mesh oscillations may grow and cause trouble. To suppress them, the vorticity components are filtered at each stage of the Runge–Kutta time integration. The filter used here is a five-point compact difference filter proposed by Lele [16, equations (C.2.1) and (C.2.10.b)].

3.8. *Time Integration*

In our code, two different methods can be used for the time integration of the vorticity-transport equations (9a)–(9c). The first is a four-stage explicit Runge–Kutta scheme which is very accurate, up to order $\mathcal{O}((\Delta t)^4)$. The second combines a three-stage explicit Runge–Kutta scheme with a semi-implicit Crank–Nicolson scheme for better numerical stability. This second scheme is accurate of order $\mathcal{O}((\Delta t)^2)$.

Four-stage explicit Runge–Kutta method. This method is based on the classical fourth-order Runge–Kutta method. However, the weighting of the intermediate stages in the final corrector step can be adjusted to increase the numerical stability of the scheme, in return for reducing its accuracy. The four stages of the integration over one timestep are

$$f_i = f_0 + \frac{\Delta t}{2} f_0' \tag{52a}$$

$$f_{ii} = f_0 + \frac{\Delta t}{2} f_i' \tag{52b}$$

$$f_{iii} = f_0 + \Delta t f_{ii}' \tag{52c}$$

$$f = f_{iv} = f_0 + \frac{\Delta t}{6}(a_{\mathrm{RK}} f_0' + b_{\mathrm{RK}} f_i' + c_{\mathrm{RK}} f_{ii}' + d_{\mathrm{RK}} f_{iii}'), \tag{52d}$$

where f denotes any vorticity component, Δt is the time step, f' is the right-hand side of the vorticity–transport equations, and the subscript 0 denotes the previous timestep. The weighting coefficients of the final corrector stage are given in Table I. A key feature of this family of Runge–Kutta integrators is the fact that all share the same intermediate steps. This allows us to use different orders for different terms of the same partial differential equation while maintaining consistency of the boundary conditions and of the nonlinear terms.

TABLE I
Coefficients of the Final Corrector Stage for the Explicit Runge–Kutta Scheme

Order	a_{RK}	b_{RK}	c_{RK}	d_{RK}
(Δt)	3.60897	2.04000	0.34206	0.00897
$(\Delta t)^2$	0.11	3.92	1.86	0.11
$(\Delta t)^3$	0.65	2.70	2.00	0.65
$(\Delta t)^4$	1	2	2	1

As an explicit method, this scheme is only conditionally stable. Many calculations of boundary layer flows require a very fine grid spacing in y near the wall. In these cases, the major stability restriction for the time step is due to the wall-normal diffusion terms $1/\mathrm{Re}\, \partial^2/\partial y^2(\Omega_{xk}, \Omega_{yk}, \Omega_{zk})$ of the vorticity–transport equations. For a numerical stability analysis, these terms can be modeled by the ordinary differential equation

$$\frac{df}{dt} = -\rho f, \tag{53}$$

where ρ is the largest eigenvalue of the finite-difference operator. For this model equation, the stability of a given numerical scheme depends only on the product $\rho\Delta t$. For the diffusion operator, ρ is dominated by its real part. Therefore, the fourth-stage corrector can be modified to allow for a larger real part of the eigenvalues, in return for reduced formal accuracy. In Fig. 2 the amplification $f(t+\Delta t)/f(t)$, obtained with the coefficients from Table I, after one timestep is plotted vs the product $\rho\Delta t$, for real $\rho \geq 0$. The method is stable if $|f(t+\Delta t)/f(t)| < 1$. These curves show that the lower-order schemes are much more stable than the fourth-order scheme. In practice, the second-order scheme is sufficiently accurate, while allowing for a timestep that is much larger than that allowed by the standard fourth-order scheme. The first-order scheme is useful for the calculation of steady flows, but is too dissipative for genuinely unsteady calculations. Hence, the wall-normal diffusion terms, which are most critical for stability, are integrated with the second-order scheme. All other terms are integrated with the fourth-order scheme.

At each Runge–Kutta stage, the calculation proceeds as follows:

1. Compute the right-hand side of the vorticity–transport equations (9a)–(9c). Split-compact differences with biasing are used to compute the streamwise derivatives $\partial A_k/\partial x$ and $\partial C_k/\partial x$.
2. Integrate the vorticity-transport equations over one substep, according to Eqs. (52a)–(52d).
3. If desired, taper the disturbance vorticity to zero near the outflow boundary.
4. If desired, filter the vorticity in the streamwise direction.
5. Solve the velocity–Poisson equations (10a)–(10c).
6. Solve Eqs. (12a) and (12c) to obtain the vorticity components Ω_{xk}, Ω_{zk} at the wall.

Three-stage Runge–Kutta/Crank–Nicolson method. For some calculations, the explicit schemes described above are still too restrictive; i.e., the maximum timestep allowed for numerical stability is much smaller than the time step necessary for numerical accuracy. In these cases, an implicit time integration scheme would be preferable, at least for the diffusion terms in the y direction. This raises the problem of boundary conditions: The implicit time integration of the wall-normal diffusion terms $1/\mathrm{Re}\, \partial^2/\partial y^2(\Omega_{xk}, \Omega_{yk}, \Omega_{zk})$ requires the specification of the vorticity at the wall, which is not known before the solution of the velocity–Poisson equations. This issue appears to be a major drawback of any vorticity formulation of the Navier–Stokes equations.

One way to deal with wall boundary conditions for an implicit scheme is to use an influence matrix method, similar to those proposed for the primitive variable fractional step method [14, 21]. This is equivalent to the numerical calculation of a Green's function. Unfortunately, the memory requirements of such an approach are very high.

An alternative approach is to iterate between the vorticity–transport equations and the elliptic equations (normal velocity and wall vorticity) until the vorticity at the wall has converged. This approach has been applied to the two-dimensional vorticity transport equations by Fasel, although in the context of a completely implicit scheme. In the present work, the implicit Crank–Nicolson scheme is used for the time integration of the y-diffusion terms only. All other terms of the vorticity–transport equations are integrated with an explicit three-stage Runge–Kutta scheme. The vorticity at the wall is computed by an iteration between the implicit part of the vorticity–transport equations and the Poisson equation for the normal velocity V_k. It is important to note that the iteration of the implicit wall-boundary conditions here is only necessary to improve the numerical stability of the scheme and does not affect the accuracy. This is in contrast to the fractional step method in primitive variables, where the problem of wall boundary conditions is a lack of accuracy and not of numerical stability.

The three stages of the Runge–Kutta scheme are

$$f_i = f_0 + \Delta t f_0' \tag{54a}$$

$$f_{ii} = f_0 + \Delta t f_i' \tag{54b}$$

$$f_{iii} = f_0 + \frac{\Delta t}{2}(f_0' + f_{ii}'). \tag{54c}$$

The equations for the Crank–Nicolson scheme at a grid point away from the boundaries are

$$\begin{aligned}
&\left(\frac{\Delta t}{2\,\mathrm{Re}}\mathrm{ard2y} - \mathrm{afd2y}\right) fn\Big|_{j-1} + \left(\frac{\Delta t}{2\,\mathrm{Re}}\mathrm{brd2y} - \mathrm{bfd2y}\right) f_n\Big|_{j} \\
&\quad + \left(\frac{\Delta t}{2\,\mathrm{Re}}\mathrm{crd2y} - \mathrm{cfd2y}\right) f_n\Big|_{j+1} \\
&= -\left(\frac{\Delta t}{2\,\mathrm{Re}}\mathrm{ard2y} + \mathrm{afd2y}\right) f_0\Big|_{j-1} - \left(\frac{\Delta t}{2\,\mathrm{Re}}\mathrm{brd2y} + \mathrm{bfd2y}\right) f_0\Big|_{j} \\
&\quad - \left(\frac{\Delta t}{2\,\mathrm{Re}}\mathrm{crd2y} + \mathrm{cfd2y}\right) f_0\Big|_{j+1} - \Delta t\,(\mathrm{afd2yrhs}_{n-1}|_{j-1} \\
&\quad + \mathrm{bfd2yrhs}_{n-1}|_j + \mathrm{cfd2yrhs}_{n-1}|_{j+1}),
\end{aligned} \tag{55}$$

where f denotes any vorticity component, j is the grid index in the y direction, Δt is the time step, Re is the Reynolds number, and rhs, is the explicit part of the vorticity transport equation without the y diffusion terms. The subscript n refers to the stage i, ii, iii of the Runge–Kutta scheme, and the subscript 0 denotes the previous timestep. The coefficients ard2y, brd2y, crd2y, afd2y, bfd2y, cfd2y at the gridpoint j are given by Eqs. (31a) and (31b) in Section 3.3.

At each stage of the Runge–Kutta scheme (54a)–(54c), the calculation proceeds as follows:

1. Compute the explicit right-hand side terms rhs of the vorticity–transport equations (9a)–(9c), excluding the wall-normal diffusion terms $1/\mathrm{Re}\, \partial^2/\partial y^2(\Omega_{xk}, \Omega_{yk}, \Omega_{zk})$. Here, central compact differences (without biasing) are used for the streamwise derivatives $\partial A_k/\partial x$ and $\partial C_k/\partial x$.

2. Compute the explicit wall-normal diffusion terms $\omega_{yy} = 1/\mathrm{Re}\,\partial^2/\partial y^2(\Omega_{xk}, \Omega_{yk}, \Omega_{zk})$ (cf. Eq. (30)).
3. Using the explicit right-hand side $f' = \mathrm{rhs} + \omega_{yy}$, advance the vorticity in time, according to Eqs. (54a)–(54c). These vorticity values are used to start the iteration in step 4.
4. Compute the wall-normal diffusion terms implicitly with the Crank–Nicolson scheme. The following iteration is used to compute the vorticity at the wall:
   (a) Using the previous vorticity values inside the domain, solve the V_k–Poisson equation (10a).
   (b) Solve Eqs. (12a) and (12c) to obtain the vorticity components Ω_{xk}, Ω_{zk} at the wall.
   (c) Using the wall vorticity values (after underrelaxation, see below) as boundary conditions, calculate the new vorticity values inside, using Eq. (55).
5. If desired, taper the disturbance vorticity to zero near the outflow boundary.
6. If desired, filter the vorticity in the streamwise direction.
7. Solve the velocity–Poisson equations (10a)–(10c).
8. Solve Eqs. (12a) and (12c) to obtain the vorticity components Ω_{xk}, Ω_{zk} at the wall.

For the iteration of the Crank–Nicolson scheme to converge, an underrelaxation must be used to update the vorticity. The vorticity values $f_n|_j$ used in step 4c above are relaxed as

$$f_n\bigg|_1 = \frac{l-1}{l_{\max}-1} f_{n,l}\bigg|_1 + \frac{l_{\max}-l}{l_{\max}-1} f_{n,l-1}\bigg|_1 \tag{56a}$$

$$f_n\bigg|_j = \frac{1}{2} f_{n,l}\bigg|_j + \frac{1}{2} f_{n,l-1}\bigg|_j, \tag{56b}$$

where j is the wall-normal grid point index, n is the stage of the Runge–Kutta scheme, l is the iteration level, and $l_{\max}$ is the total number of iterations. Note the gradual change of the relaxation factor for the wall vorticity in Eq. (56a). Of the many different relaxation schemes tested, this scheme proved to be the fastest and most robust one. In practice, six iterations are sufficient for convergence.

4. CODE VALIDATION

In this section, we present the results of several numerical calculations that demonstrate the accuracy and convergence of the numerical scheme. The best way to assess the accuracy of a numerical method is to compute a flow for which there is a known exact solution. By comparing the error from calculations for different stepsizes, one can calculate the formal accuracy of the overall method, as opposed to the formal accuracy of individual finite-difference approximation of different terms in the equations. Suppose the numerical error is dominated by the leading term of the truncation error of a Taylor series, i.e.,

$$\varepsilon = f_{\text{numerical}} - f_{\text{exact}} = cm^{-p} \equiv c\Delta^p, \tag{57}$$

where c is a constant, m is the number of steps Δ, and p is the accuracy of the numerical scheme in x, y, or t, respectively. Using Eq. (57) for two numerical solutions with different resolutions, m_1 and m_2, yields the accuracy p as

$$p = \frac{\log(\varepsilon_1/\varepsilon_2)}{\log(m_2/m_1)}. \tag{58}$$

Unfortunately, there are not many exact solutions of the Navier–Stokes equations suitable for such an accuracy analysis of the present code. However, even without an exact solution, it is still possible to estimate the convergence rate of the numerical scheme from Eq. (57). In this case, we require three numerical solutions f_0, f_1, and f_2 with different resolutions m_0, m_1, and m_2 to obtain an equation for p,

$$(f_1 - f_0)\big((m_0/m_2)^p - 1\big) = (f_2 - f_0)\big((m_0/m_1)^p - 1\big). \tag{59}$$

This equation must be solved numerically. In practice, p is usually a nonintegral number, and the formal accuracy is considered to be its integral part. Once p (and c) are known, one can use Richardson extrapolation to obtain an estimate for the truncation error $c\Delta^p$ and for the asymptotic solution as $\Delta \to 0$.

4.1. *Asymptotic Suction Flow*

One exact solution suitable for a validation of our numerical method is the asymptotic solution of a flow over a flat plate with zero pressure gradient and with uniform wall suction $v_s < 0$. In the present nondimensional variables, this solution of the Navier–Stokes equation is

$$u(y) = 1 - e^{v_s \text{ Re } y} \tag{60a}$$

$$v = v_s \tag{60b}$$

$$\omega_z = -v_s \text{ Re } e^{v_s \text{ Re } y}. \tag{60c}$$

This flow is of particular relevance, because the study of wall suction in laminar flow control is an important application for our Navier–Stokes code. To determine the accuracy of the numerical method, we compared the values of the wall vorticity from the two different calculations to the exact solution and used Eq. (58) to compute the convergence rate. The computational parameters used in the calculations and the results are given in Table II. These results show that the overall code is indeed fourth-order accurate in the y direction, even on a highly stretched grid.

4.2. *Tollmien–Schlichting Waves*

In this test, the Navier–Stokes code was used to compute the propagation and amplification of TS-waves in a Blasius boundary layer. At the inflow boundary, time-harmonic boundary

TABLE II
Computational Parameters and Results for Asymptotic Suction Flow

Reynolds number	$\text{Re} = 10^5$
Suction velocity	$v_s = -2 \times 10^{-3}$
Wall vorticity	$\omega_z(y = 0) = 200$ (exact solution)
Free-stream boundary	$y_{\max} = 0.15$
Grid stretching parameter	$\beta = 1.02$
Number of y-gridpoints	$m_{y,1} = 40$ (case 1)
	$m_{y,2} = 80$ (case 2)
Numerical error	$\varepsilon_1 = -1.589354 \times 10^{-2}$ (case 1)
	$\varepsilon_2 = -0.093644 \times 10^{-2}$ (case 2)
Convergence rate	$p = 4.1$

TABLE III
Computational Parameters Used in All TS-Wave Calculations

Inflow location	$x_0 = 1.6$
Begin of buffer domain	$x_B = 5.7$
Reynolds number	$\mathrm{Re} = 10^5$
Free-stream boundary	$y_{\max} = 0.15$
Grid stretching parameter	$\beta = 1.02$
Number of y gridpoints	$m_y = 80$
Number of spanwise Fourier modes	$K = 2$
Fundamental spanwise wavenumber	$\gamma_1 = 30$
Frequency of the TS-waves	$F = 2\pi f \nu / U_\infty^2 \times 10^4 = 1$
Inflow amplitudes of the TS-waves	$\hat{u}_{2-D} = 0.364529 \times 10^{-4}$ (2-D wave)
	$\hat{u}_{3-D} = 0.871989 \times 10^{-4}$ (3-D wave, $\gamma = 30$)

conditions were specified that corresponded to a superposition of one plane (2-D) TS-wave and one oblique (3-D) TS-wave. The computational parameters common to all TS-wave calculations presented here are given in Table III. All spatial dimensions are scaled by the reference length $L = 0.1$ M, and all velocities are scaled by the free-stream velocity $U_\infty = 15$ m/s. With these flow parameters, both TS-waves are initially damped after they enter the integration domain. They subsequently pass through both branches of the neutral stability curve and are damped again before they reach the buffer domain. Their amplitudes (maxima of $\hat{u}$ over y at each x-location) are plotted in Fig. 11. The wavelength of the 2-D TS-wave near the maximum amplitude is about $\lambda_{\mathrm{TS}} \approx 0.22$.

With this basic configuration, several calculations were performed, using different step sizes Δx and Δt, different time integration schemes, different biasing factors in the split-compact differences for the nonlinear terms, and different buffer domains lengths l_b and weighting functions. The grid spacing in the y direction was not changed in this test. The parameters of these different calculations are listed in Table IV. In this table, exp refers to the exponential damping function in Eq. (50), poly refers to the polynomial damping function in Eq. (47), RKCN denotes the three-stage Runge–Kutta/Crank–Nicolson method, and RK4-2 denotes the four-stage Runge–Kutta method with second-order accuracy for the y diffusion terms.

An issue of considerable importance is the measure of the error in these calculations. Traditionally, in linear stability theory as applied to TS-waves, the amplification rate α_I has been used to compare different prediction methods. However, α_I is only a local measure of amplitude growth and does not provide information about the global development of the waves, and hence about the total error. In this study, we have chosen to use the maximum u amplitude attained by the TS-waves as the quantity for comparison. This measure includes both the error due to the numerical treatment of the inflow boundary and the cumulative error from the propagation of the waves over 7 (3-D) and 16 (2-D) wavelengths, respectively. Hence, it is a better measure of the global error than, say, the maximum value of the growth rate.

The numerical results are given in Table V. In addition to the results from the individual calculations, we also include the results obtained by using Richardson extrapolation for $\Delta x, \Delta t \to 0$. These results are labeled "extrap." The formal accuracy of the scheme in the x direction was computed from the amplitudes of test cases X1 and X2. In spite of using formally fourth-order accurate difference approximations for all x-derivatives, the overall code is only third-order accurate in Δx. This is due to the fact that the right-hand sides of the velocity equations (10a)–(10c) and of the wall vorticity equations (12a), (12c) contain

TABLE IV
Computational Parameters for Different TS-Wave Calculations

Case	$\Delta x/\Delta x_0$	w_c	l_b	$c(\xi)$	$\Delta t/\Delta t_0$	Time integration
REF	1	1.0	0.95	exp	1	RKCN
X1	2	1.0	0.95	exp	1	RKCN
X2	3	1.0	0.95	exp	1	RKCN
T1	1	1.0	0.95	exp	4/3	RKCN
T2	1	1.0	0.95	exp	2	RKCN
SI	3	1.0	0.95	exp	1/2	RKCN
SC	3	1.0	0.95	exp	1/2	RK4-2
SM	3	0.5	0.95	exp	1/2	RK4-2
SS	3	0.0	0.95	exp	1/2	RK4-2
E35	1	1.0	0.35	exp	1	RKCN
E25	1	1.0	0.25	exp	1	RKCN
E15	1	1.0	0.15	exp	1	RKCN
P35	1	1.0	0.35	poly	1	RKCN
P25	1	1.0	0.25	poly	1	RKCN
P15	1	1.0	0.15	poly	1	RKCN

Note. $\Delta x_0 = 0.01$ (approx. 22 points/wavelength); $\Delta t_0 = 3.927 \times 10^{-3}$ (160 timesteps/period).

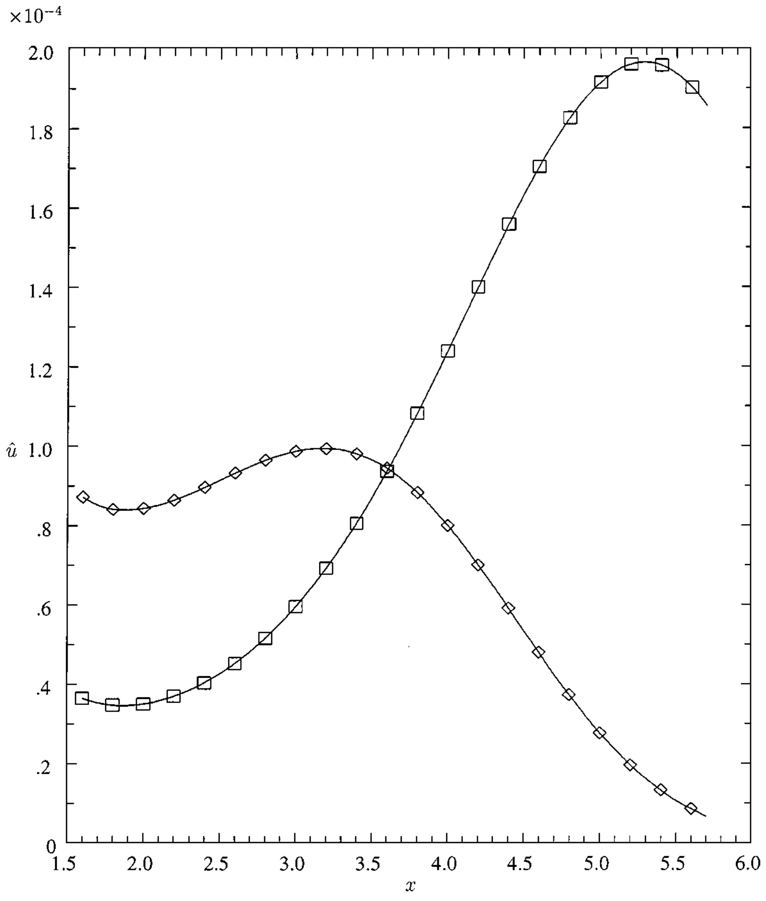

FIG. 11. u-amplitudes of 2-D (□) and 3-D (◇) TS-waves.

TABLE V
Results of Different TS-Wave Calculations

Case	$u_{\max,2-D}$ $(\times 10^{-4})$	p_{2-D}	$u_{\max,3-D}$ $(\times 10^{-4})$	p_{3-D}	CPU $(\times 10^{-6}$ s)
REF	1.96657		0.99517		49
X1	2.08851	3.3	1.01854	3.8	39
X2	2.47288		1.11131		37
T1	1.96037	2.4	0.99351	2.4	49
T2	1.93995		0.98805		49
extrap	1.9756		0.9950		
SI	2.49878		1.13006		37
SC	2.49995		1.13026		26
SM	2.29489		1.09104		26
SS	1.83418		0.99344		26

Note. CPU time in μs/gridpoint/timestep.

x-derivatives of the vorticity, which reduces the overall accuracy by one order. The results of test cases T1 and T2 show that the semi-implicit three-stage Runge–Kutta/Crank–Nicolson method is indeed second-order accurate in Δt. The convergence tests for Δx and Δt were performed with central differences for the nonlinear terms, without splitting (i.e., $w_c = 1$). These tests show that lower resolution in t results in an underprediction of the disturbance growth, while lower resoltution in x tends to overpredict it.

The effect of splitting was investigated in test cases SC, SM, and SS. Since the effects of the splitting are most pronounced for poorly resolved waves, we have selected a resolution of approximately 7 points per wavelength for these three cases. One would not use such a poor resolution for a practical TS-wave calculation. However, in a large-eddy simulation of the later stages of transition and early turbulence, large-scale structures might well contain considerable energy at such short wavelengths. Thus, this test is indicative of the improvements that can be expected from the split differences for such calculations.

In these three test cases, the explicit four-stage Runge–Kutta method was used for time integration, with second-order accurate integration of the y diffusion terms. Since this method is less stable than the combined Runge–Kutta/Crank–Nicolson method, the timestep had to be reduced relative to the reference case. Thus, for comparison, the calculation SI was performed with the combined Runge–Kutta/Crank–Nicolson and with the reduced timestep. We note that the amplitudes in case SI, with reduced Δt, are increased relative to the case X2. This agrees with the previous observation that lower resolution in t results in an underprediction of the amplitude; this effect is apparently more pronounced when the x-resolution is low. Table V shows that the change due to the smaller timestep is an order of magnitude smaller than the change due to the different splitting factors w_c. The effect of the splitting is indeed profound. In both cases SM and SS, the amplitude error is reduced by about 50%, for both the 2-D wave and the 3-D wave. Note also that the error changes sign as we go from $w_c = 0.5$ (SM) to $w_c = 0$ (SS); thus, careful tuning of the splitting might lead to even further accuracy improvements.

The last column in Table V lists the CPU time (in μs/gridpoint/timestep) used for the individual calculations. These times were obtained with the code running on a single 90-MHz R8000 processor on an SGI Power challenge. A comparison between the cases SI and SC shows that the semi-implicit Runge–Kutta/Crank–Nicolson method RKCN requires about 50% more CPU time per timestep than the explicit four-stage Runge–Kutta method RK4-2. This is the penalty for the extra computational effort of the iteration. However, for the computational grid used in this study, RKCN is much more stable than RK4-2. The stability limit for RKCN is about 80 timesteps per period of the TS-waves, while RK4-2 needs about 240 timesteps per period. Thus, if we use the maximum timestep allowed by numerical stability, RKCN needs about half the CPU time of RK4-2, a considerable savings. Also, note that the results with 80 timesteps per period (case T2) are within 2% of the result estimated by Richardson extrapolation. Such accuracy is probably more than sufficient in most cases.

Finally, we investigated the effect of different damping functions and buffer domain lengths l_B relative to the TS-wavelength λ_{TS}. Figure 12 shows the u-amplitude $\hat{u}$ of the 2-D TS-wave for seven different buffer domain parameters: The reference case was computed with a very long buffer domain ($l_B \approx 4\lambda_{TS}$), using the exponential weighting function (50). In cases E15, E25, and E35 the exponential weighting function was used with a buffer domain

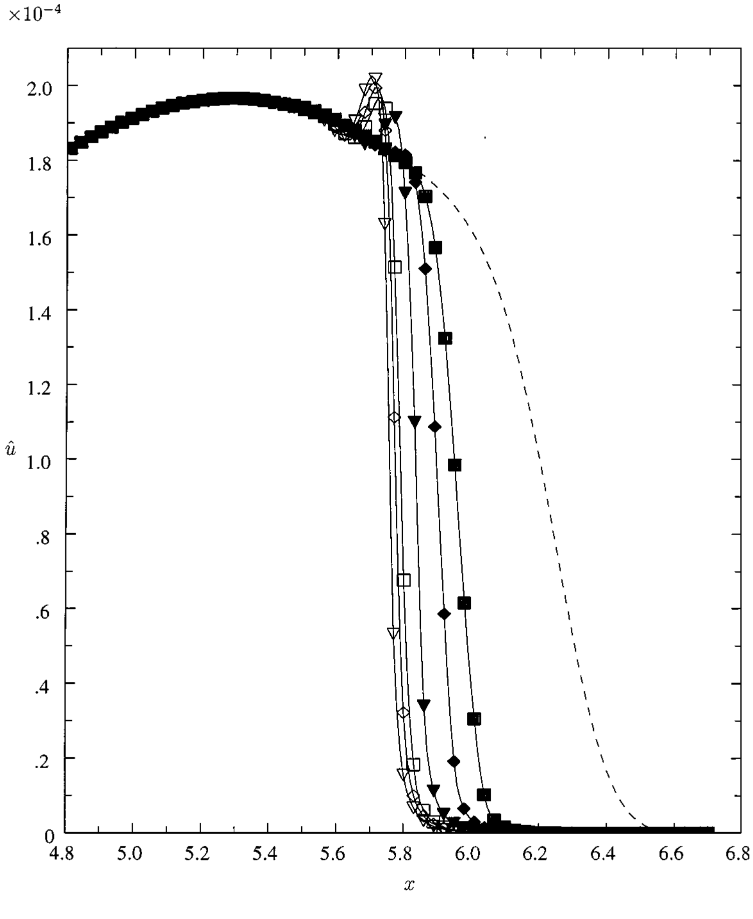

FIG. 12. Effect of different buffer domain lengths l_B and damping functions $c(\xi)$ on the amplitude of the 2-D TS-wave. Damping begins at $x_B = 5.7$. The curves are reference case with $l_B = 0.95$ (---); polynomial damping with $l_B = 0.35$ (□), $l_B = 0.25$ (◇), and $l_B = 0.15$ (▽); and exponential damping with $l_B = 0.35$ (■), $l_B = 0.25$ (◆), and $l_B = 0.15$ (▼).

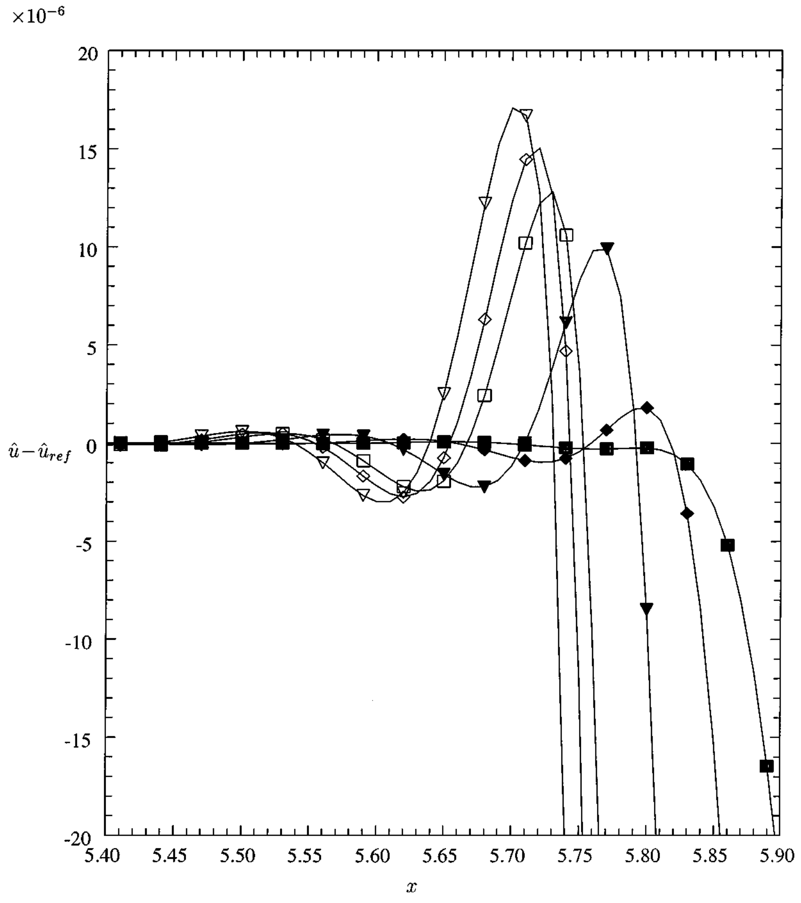

FIG. 13. Difference between calculations with short buffer domains and reference case. The curves are polynomial damping with $l_B = 0.35$ (□), $l_B = 0.25$ (◇), and $l_B = 0.15$ (▽); and exponential damping with $l_B = 0.35$ (■), $l_B = 0.25$ (◆) and $l_B = 0.15$ (▼).

length of 0.15, 0.25, and 0.35, respectively. In cases P15, P25, and P35 the polynomial damping function (47) was used, again with a buffer domain length of 0.15, 0.25, and 0.35, respectively. The difference $\hat{u} - \hat{u}_{\text{ref}}$ between the different E and P cases and the reference case is plotted in Fig. 13. These curves clearly show the dramatic improvement due to the exponential damping function. Even with a buffer length that is substantially smaller than the TS-wavelength (case E15), upstream effects on the amplitude are no larger than 1% of the maximum amplitude, and they decay rapidly for $x < x_B$. On the other hand, the polynomial damping function causes distortions that extend up to one wavelength upstream of the buffer domain. Thus, the exponential damping allows for a reduction in the length of the buffer domain of up to two wavelengths. For turbulence simulations that include only a few wavelengths in streamwise direction, this can amount to a substantial reduction in the computational effort.

5. CONCLUSIONS

We have presented a new numerical method for solving the incompressible, unsteady Navier–Stokes equations in vorticity–velocity formulation. The method is highly suited for simulations of transition and turbulence in wall-bounded shear flows. It combines several new numerical techniques that were discussed in detail. The discretization of the convective terms with split-compact differences and the use of nonequidistant compact differences in the wall-normal direction considerably improved the overall accuracy of the numerical scheme. A new exponential damping function leads to a more efficient implementation of the buffer domain technique to prevent reflections of waves at the outflow boundary. A new iteration scheme for the calculation of the wall vorticity allowed for a semi-implicit time integration of the wall-normal diffusion terms. This resulted in substantially increased numerical stability of the scheme.

It should be emphasized that these new techniques are not restricted to the Navier–Stokes equations in vorticity–transport form. The split-compact differences and the improved buffer domain technique are well suited for wave propagation problems in many areas of mathematical physics. The nonequidistant compact differences provide substantial improvement over conventional high-order finite differences for problems with boundary layer characteristics that require highly stretched grids.

The numerical code has been used in direct numerical simulations of laminar flow control and transition in boundary layers [18, 20] and wall jets [30]. With the addition of an eddy viscosity to model the subgrid scale Reynolds stresses, it has been used to carry out large-eddy simulations of boundary layers and wall jets [2]. It has recently been extended to non-Cartesian coordinates for flows over curved surfaces [31].

ACKNOWLEDGMENTS

This work was supported by the Office of Naval Research under contract N00014-91-J-1787 and by NASA–Langley under contract NAG 1-1422.

REFERENCES

1. D. A. Anderson, J. C. Tannehill, and R. H. Pletcher, *Computational Fluid Mechanics and Heat Transfer* (McGraw–Hill, New York, 1984).
2. C. R. Bachman and H. F. Fasel, Large eddy simulation for a transitional flat plate boundary layer, in *48th Annual APS/DFD Meeting, Irvine, CA*, 1995.
3. F. P. Bertolotti, Th. Herbert, and P. R. Spalart, Linear and nonlinear stability of the Blasius boundary layer, *J. Fluid Mech*. **242**, 441 (1992).
4. F. Bertolotti, Receptivity of cross-flow and Görtler vortices to distributed forcing, Presented at the *IUTAM Symposium on Laminar-Turbulent Transition, Sendai, Japan*, 1999.
5. A. J. Chorin, A numerical method for solving incompressible viscous flow problems, *J. Comput. Phys*. **2**, 12 (1967).
6. A. J. Chorin, Numerical solutions of the Navier–Stokes equations, *Math. Comput*. **22**, 745 (1968).
7. W. E and J.-G. Liu, Vorticity boundary condition and related issues for finite difference schemes, *J. Comput. Phys*. **124**, 368 (1996).
8. W. E and J.-G. Liu, Essentially compact schemes for unsteady viscous incompressible flows, *J. Comput. Phys*. **126**, 122 (1996).
9. W. E and J.-G. Liu, Finite difference methods for 3d viscous incompressible flows in the vorticity–vector potential formulation on nonstaggered grids, *J. Comput. Phys*. **138**, 57 (1997).
10. H. Fasel, U. Rist, and U. Konzelmann, Numerical investigation of the three dimensional development in boundary layer transition, *AIAA Paper* 87-1203, 1987.
11. K.-Y. Fung, R. S. O. Man, and S. Davis, Implicit high-order compact algorithm for computational acoustics, *AIAA J*. **34**, 2029 (1996).

12. D. S. Henningson, S. Berlin, and A. Lundbladh, Spatial simulations of bypass transition in boundary layers, presented at the *IUTAM Symposium on Laminar-Turbulent Transition, Sendai, Japan*, 1994.

13. G. E. Karniadakis, M. Israeli, and S. A. Orszag, High-order splitting methods for the incompressible Navier–Stokes equations, *J. Comput. Phys*. **97**, 414 (1991).

14. L. Kleiser and U. Schumann, in *Notes on Numerical Fluid Mechanics*, edited by E. H. Hirschel (Vieweg, Braunschweig, 1980), p. 165.

15. M. Kloker, U. Konzelmann, and H. Fasel, Outflow boundary conditions for spatial Navier–Stokes simulations of transitional boundary layers, *AIAA J*. **31**, 620 (1993).

16. S. K. Lele, Compact finite difference schemes with spectral-like resolution, *J. Comput. Phys*. **103**, 16 (1992).

17. R. K. Madabushi, S. Balachandar, and S. P. Vanka, A divergence-free Chebyshev collocation procedure for incompressible flows with two non-periodic directions, *J. Comput. Phys*. **105**, 199 (1993).

18. H. Meitz and H. Fasel, Navier–Stokes simulations of the effect of suction holes on a flat plate boundary layer, in *AGARD Conference Proceedings, AGARD-CP-551. Sendai, Japan*, 1994.

19. H. L. Meitz, *Numerical Investigation of Suction in a Transitional Flat-Plate Boundary Layer*, Ph.D. thesis, University of Arizona, 1996.

20. H. Meitz and H. Fasel, Numerical investigation of the interaction of the Klebanoff-mode with Tollmien–Schlichting waves, *J. Fluid Mech.*, in press.

21. R. Mittal and S. Balachandar, Direct numerical simulation of flow past elliptic cylinders, *J. Comput. Phys*. **124**, 351 (1996).

22. M. V. Morkovin, Bypass-transition research: Issues and philosophy, in *Instabilities and Turbulence in Engineering Flows*, edited by D. E. Asphis, T. B. Gatski, and R. Hirsh (Kluwer, Dordrecht/Norwell, MA, 1993), p. 3.

23. S. A. Orszag, Numerical simulation of incompressible flows within simple boundaries: Galerkin (spectral) representations, *Stud. Appl. Math*. **50**, 293 (1971).

24. J. B. Perot, An analysis of the fractional step method, *J. Comput. Phys*. **103**, 51 (1993).

25. S. E. Rogers, Comparison of implicit schemes for the incompressible Navier–Stokes equations, *AIAA J*. **33**, 2066 (1995).

26. C. L. Streett and M. G. Macaraeg, Spectral multidomain for large-scale fluid dynamic simulations, *Int. J. Appl. Numer. Math*. **6**, 123 (1989).

27. P. N. Swarztrauber, The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle, *SIAM Rev*. **19**, 490 (1977).

28. P. N. Swarztrauber, Vectorizing the FFTs, in *Parallel Computations*, edited by G. Rodrigue (Academic Press, New York, 1982), p. 51.

29. P. Tamamidis, G. Zhang, and D. N. Assanis, Comparison of pressure-based and artificial compressibility methods for solving 3D steady incompressible viscous flows, *J. Comput. Phys*. **124**, 1 (1995).

30. S. Wernz and H. F. Fasel, Numerical Investigation of Unsteady Phenomena in Wall Jets, *AIAA Paper* 96-0079, 1996.

31. H. L. Zhang and H. F. Fasel, Spatial direct numerical simulation of Görtler vortices, in *Thirteenth Arizona Fluid Mechanics Conference*, 1997.