Numerical Heat Transfer, vol. 14, pp. 149–164, 1988

EVALUATION OF THE FILTERED LEAPFROG–TRAPEZOIDAL TIME INTEGRATION METHOD

Patrick J. Roache and David E. Dietrich
Ecodynamics Research Associates, Inc., Albuquerque, New Mexico 87198

An analysis and evaluation are presented for a new method of time integration for fluid dynamic equations proposed by Dietrich. The method, called the filtered leapfrog-trapezoidal (FLT) scheme, is analyzed for the one-dimensional constant-coefficient advection equation and is shown to have some advantages for quasi-steady flows. A modification (FLTW) using a weighted combination of FLT and leapfrog is developed which retains the advantages for steady flows, increases accuracy for time-dependent flows, and involves little coding effort. Merits and applicability are discussed.

INTRODUCTION: THE FLT AND FLTW METHODS

Dietrich and Wormeck [1], in an article published in this journal, developed the filtered leapfrog-trapezoidal (FLT) time differencing method for advection terms and applied it to combustion problems, for which it was successful in preventing instability. This method (and the modification proposed herein) has more recently been applied to oceanographic calculations [2] in the Sandia Ocean Modeling System (SOMS).

Several implementations and interpretations of the FLT method are possible, including a predictor-corrector formulation or filtering, but the simple and straightforward algorithm is as follows. Using the half-time step notation (common to geophysical literature) with superscripts indicating the time level at regular time increments Δt, and with q being any dynamically advanced quantity and Q the spatial operator in the dynamic equation, we have

$$\frac{q^{n+1} - q^n}{\Delta t} = Q(q^{n+1/2}) \tag{1a}$$

$$q^{n+1/2} = \frac{q^n + q^{n+1}}{2} \tag{1b}$$

$$\frac{q^{n+3/2} - q^{n+1/2}}{\Delta t} = Q(q^{n+1}) \tag{1c}$$

For the purposes of the present analysis, we can define the FLT method more economically, considering only the model one-dimensional constant-coefficient advection equation

$$q_t = -uq_x \tag{2}$$

This work was supported in part by Sandia National Laboratories. The authors gratefully acknowledge the technical contributions and support of Dr. M. Marietta.

NOMENCLATURE

FLT	filtered leapfrog-trapezoidal method	t	time
FLTW	weighted filtered leapfrog-trapezoidal method	u	convection speed
G	amplification matrix	V	amplitude function of the Fourier component
I	$\sqrt{(-1)}$	w	weighting factor for FLTW method
k	component wave number	x	space
$L(q)$	operator	Δ	finite increment
O	order	θ	phase angle of the Fourier component
Q	spatial operator in transport equation	Λ	wavelength of the solution
q	any advected quantity	λ	eigenvalues of G

where subscripts t and x (in continuum equations) represent partial derivatives with respect to time and space, and u is the advection velocity, assumed herein to be a linear coefficient, constant in time and space. The familiar leapfrog method (e.g., see Roache [3]) uses centered two-point differences in space and time, producing the following three-time-level equation. Subscripts (in a discrete equation) indicate spatial position in a regular mesh with spacing Δx.

$$\frac{q_i^{n+1} - q_i^{n-1}}{2\Delta t} = -\frac{u(q_{i+1}^n - q_{i-1}^n)}{2\Delta x} \tag{3}$$

In terms of the Courant number c (also called the Courant-Friedrichs-Lewy or CFL number), defined as

$$c = \frac{u\,\Delta t}{\Delta x} \tag{4}$$

the leapfrog method [Eq. (3)] can be written as

$$q_i^{n+1} = q_i^{n-1} - c(q_{i+1}^n - q_{i-1}^n) \tag{5}$$

Adopting a more concise notation, in which omitted superscripts and subscripts are understood to represent n and i, Eq. (5) becomes

$$q^{n+1} = q^{n-1} - c(q_{i+1} - q_{i-1}) \tag{6}$$

Dietrich's FLT method applies a time filter to the above algorithm, smoothing the q values at the $(n-1)$ time level by two-point averaging in time, replacing q^{n-1} by

$$q^{n-1} \simeq \tfrac{1}{2}(q^n + q^{n-2}) \tag{7}$$

The FLT method then gives

$$q^{n+1} = \tfrac{1}{2}(q^n + q^{n-2}) - c(q_{i+1} - q_{i-1}) \tag{8}$$

In this paper we also consider a simple extension of FLT. It involves a user-selected weighted combination of the new FLT method with standard leapfrog differencing. The leapfrog differencing itself would be adequate for many problems, and has the considerable advantage of giving the correct unity damping factor for constant-coefficient problems, except for a well-known difficulty (e.g., see [3]). Leapfrog exhibits a time-splitting instability in which solutions at alternate time steps tend to decouple, the coupling occurring only through (small) viscous terms and boundary conditions. As will be shown below, only a little weighting (say 1/10) of the FLT method with leapfrog is sufficient to prevent the time-splitting instability, while affecting the long-wavelength damping factor and the stability limit only slightly. This modification is easy to implement in multidimensional codes and is simpler than the often used three-point implicit filters. Multiplying Eq. (8) by a weighting factor w and Eq. (6) by $(1 - w)$ and adding the results gives the FLTW method.

$$q^{n+1} = \frac{w}{2}(q + q^{n-2}) + (1 - w)q^{n-1} - c(q_{i+1} - q_{i-1}) \tag{9}$$

The FLTW method reduces to the FLT method for $w = 1$ and to leapfrog for $w = 0$. (There is no physical basis for choosing a value for w; it will be determined by numerical experimentation below.)

OPERATION COUNT EFFICIENCY

The operation count efficiency of the FLTW method will be evaluated by comparison to the standard leapfrog method. The limitations of operation count comparisons were discussed by Roache [4]. Although not precise, they are valuable for comparing different methods. (This is especially so in the present case, in which relative performance does not depend on the computer-dependent ratio of CPU time for multiplies versus adds.)

Thus, we will compare the FLT method of Eq. (8) and the FLTW method of Eq. (9) with the standard leapfrog method of Eq. (6) in terms of fundamental arithmetic operations of multiplies (*) and adds (+) per time step. The assumption is made that good programming practice (or at least a good compiler) will precalculate and store grouped terms such as $(1 - w)$, $(w/2)$, and $(2\Delta x)$. However, practical applications are not made to a model equation with constant u; consequently, we include in the operation count an additional multiplication for each local evaluation of Courant number c from local velocity u. The dimensional extensions of FLT and FLTW are straightforward. The filtering process is not dimensional, so the absolute penalty is not a function of dimensionality; consequently, the relative penalty decreases with the increasing cost of higher dimensionality. The results are summarized in Table 1.

Note that in a practical fluid dynamics code, these relative penalties would be reduced further by the operations for viscous terms, pressure solutions, etc., which

Table 1 Operation Count Comparisons for the Leapfrog, FLT, and FLTW Methods Applied to an Advection Equation Requiring Local Evaluation of the Local Courant Number

Dimension	Method	Operations	Penalty (%)
1D	Leapfrog	2*, 2+	
	FLT	3*, 3+	50
	FLTW	4*, 4+	100
2D	Leapfrog	4*, 4+	
	FLT	5*, 5+	25
	FLTW	6*, 6+	50
3D	Leapfrog	6*, 6+	
	FLT	7*, 7+	17
	FLTW	8*, 8+	33

are not affected by the operation counts for filtering. The net penalty is considered small. For multidimensional codes with significant additional physics (e.g., turbulence, chemistry, radiation) the typical penalty will be a few percent or less.

OBSERVATIONS

Comparing the FLT method [Eq. (8)] with the leapfrog method [Eq. (6)], we see that the leapfrog is a three-time-level method, whereas the FLT method extends over four time levels, $(n - 2)$ to $(n + 1)$, but skips the level $(n - 1)$. Simple programming devices may be used in leapfrog to overwrite level $(n - 1)$ with level $(n + 1)$, so only two storage arrays are required for velocities and advected quantities like energy. Similarly, programming the FLT method can overwrite level $(n - 2)$ with level $(n + 1)$, requiring two storage arrays. However, the four-level FLTW method does not skip the level $(n - 1)$ for $w < 1$ and thus requires three, rather than two, storage arrays for velocities and advected quantities. Note that in fluid dynamics calculations the pressure array (and the vertical velocity in a hydrostatic approximation code) needs to be stored at only one level, so the storage penalty for FLTW compared to FLT or leapfrog is always less than 50%. Also, in the development of modern scientific computers (either supercomputers or microcomputers) storage is relatively cheap compared to computing time, so this is not considered to be a disqualifying disadvantage.

It is obvious from inspection that both leapfrog and FLT would give the same steady-state solutions. (The model problem has only the trivial steady-state solution, but the observation applies equally well to multidimensional, nonlinear, and viscous flows, for which the steady-state solutions may not be trivial.) It is instructive to rewrite the FLTW method in terms of an addition to the leapfrog method; rearrangement of Eq. (9) to fit the form of Eq. (6) gives

$$q^{n+1} = q^{n-1} - c(q_{i+1} - q_{i-1}) + \frac{w}{2}(q^n - 2q^{n-1} + q^{n-2}) \tag{10}$$

This form indicates clearly that the method has no spatial damping compared to the leapfrog method but adds a time-only damping term; in a steady state, the added term = 0 identically. Loosely, the added term approximates some constant times the finite-difference analog of q_{tt}, although it is not centered in time. That is, leapfrog approximates the operator

$$L(q) = q_t + uq_x \tag{11}$$

to second-order accuracy as

$$L(q) = O(\Delta t^2, \Delta x^2) \tag{12}$$

whereas the FLTW method approximates the operator

$$-\left(w\frac{\Delta t}{4}\right)q_{tt} + L(q) = O(\Delta t^2, \Delta x^2) \tag{13}$$

The time-only damping, or time filtering, of the method is, of course, an error in the mathematical sense, but it exists by design and is useful for many problems. First, it avoids the decoupling behavior of the leapfrog method for quasi-steady flows. Second, it damps out errors arising from commonly occurring difficulties with initial conditions. Whereas some physically meaningful initial conditions can give rise to rapid time oscillations in the initial transients, a more common cause is inaccurate and/or physically unrealistic initial conditions. Even if physically realistic initial conditions are accurate and produce continuum solutions with no rapid time oscillations, these initial conditions may be incompatible with the discretization, i.e., the discretization error may induce rapid time oscillations. (For example, if discrete values of an exact continuum steady-state solution are merely injected into a discrete model as initial conditions, they will not ordinarily produce a steady-state solution of the discretized equations, giving rise to transients.) The FLTW method with $w > 0$ artificially damps all (temporal) frequencies, whether they are physically valid or are merely numerical artifacts, but it damps the rapid time variations more heavily; hence the justification of the term "filtering." This was the motivation for the original application to compressible reactive gas flows by Dietrich and Wormeck [1]. In such cases, the original method with large time-only damping ($w = 1$) is preferable. (Our experience with the ability of the method to successfully damp out errors from initial conditions includes coupled fluid dynamics and heat transfer problems in compressible reactive gas flows, incompressible high Re flows in spray nozzles, and ocean circulation models.)

Neither leapfrog nor FLTW is self-starting, since another method must be used to initiate the multiple time levels. [The adequacy of the starting solution depends on the problem under consideration, of course. If an oscillatory long-time solution is of interest, the details of the initial conditions are unimportant. If the early transients are of interest and the initial conditions are accurately known (often not the case in practical problems) any method may be used, with very small time steps if necessary to preserve accuracy, to march the initial solution out to the time level at which the leapfrog or FLTW method may be turned on.] This is common to all multi-time-level methods.

Neither leapfrog nor FLTW moves the short-wavelength (Λ) component of the solution; i.e., the $\Lambda = 2\Delta x$ component is stationary except for (nonphysical) boundary condition effects. (See, e.g., Roache [3].)

The use of the time-smoothing operator in FLTW is reminiscent of the spatial-smoothing operator used in the Lax method (e.g., see [3]). However, the Lax method shows the peculiar and undesirable property of advancing a solution even when $\Delta t = 0$, which is not the case with FLTW. (Some interpretation is required for any multilevel method, such as FLTW or even leapfrog itself, but if consistent initial conditions are used, $\Delta t = 0$ in FLTW gives a stationary solution, as expected.) However, the FLTW method does not maintain a desirable property shared by most conventional methods, in that it does not preserve the exact solution for the special case of unity Courant number. For the case of $c = 1$, the characteristics of the continuum hyperbolic advection equation pass exactly through the node points in the space-time mesh (e.g., see [3]), and most methods give the exact answer,

$$q_i^{n+1} = q_{i-1}^n \tag{14}$$

To obtain the exact solution for multi-time-level methods like leapfrog, it is necessary to start the method with two correct time level values, i.e., with

$$q_{i+1}^n = q_i^{n-1} \tag{15}$$

Use of this value in the leapfrog method, Eq. (5), with $c = 1$ gives the exact solution for the next step, as in Eq. (14). However, starting the FLTW method with exact values as in Eq. (15) and

$$q_{i+2}^n = q_{i+1}^{n-1} = q_i^{n-2} \tag{16}$$

gives

$$q_i^{n+1} = q_{i-1} + \frac{w}{2}(q_i - 2q_{i+1} + q_{i+2}) \tag{17}$$

which indicates that the exact solution is not preserved.

TRUNCATION ERROR ANALYSIS

The (local) truncation error analysis of the FLTW method, Eq. (9), is performed using the standard Taylor series approach. Although the smoothing operation of Eq. (7) may appear to "center" the replacement for q^{n-1}, the method is shown to be only first-order accurate in time, contrary to the claim in Dietrich and Wormeck [1]. (Their iterated trapezoidal method is second order, but the final filtering operation reduces it to first order.)

Expanding Eq. (9) in terms of the increments Δt and Δx gives

$$\frac{q^{n+1} - q^{n-1}}{2\Delta t} = -\frac{u(q_{i+1} - q_{i-1})}{2\Delta x} + \frac{w}{4\Delta t}(q^n - 2q^{n-1} + q^{n-2}) \tag{18}$$

Substituting into Eq. (18) the usual Taylor series expansions, including that over the $-2\Delta t$ time increment,

$$q^{n-2} = q - (2\Delta t)q_t + \tfrac{1}{2}(2\Delta t)^2 q_{tt} + O(\Delta t^3) \tag{19}$$

and rearranging, we obtain

$$q_t = -uq_x + \frac{w\,\Delta t}{4} q_{tt} + O(\Delta t^2, \Delta x^2) \tag{20}$$

For the weighting parameter $w = 0$, the FLTW reverts to the leapfrog method, and the error term above is second order in space and time. For the FLT method with $w = 1$, the leading error term is $(\Delta t/4)q_{tt}$ which is clearly first-order accurate in time. For practical calculations, the *size* of the truncation error for the modified FLTW method may be small, since $w = O(1/10)$ is acceptable, as indicated by tests (see below). However, the formal error is first order, unless $w \propto \Delta t$ as the time step is reduced. This would bring the FLTW method back to the leapfrog method, with its attendant problems of decoupling. Therefore, the FLT and FLTW methods must be interpreted as only first-order accurate in time. [Note that in SOMS the turbulent dissipation terms are also $O(\Delta t)$.]

HEURISTIC STABILITY ANALYSIS

The heuristic stability analysis of Hirt [5] can be applied readily to the FLTW method. For the lowest-order terms, as used here, it is equivalent to the modified-equation analysis of Warming and Hyett [6]. For the simple model equation here, the modified-equation analysis is rigorous. The higher time derivative terms in the expanded equation are evaluated by successive partial differentiation of the continuum equation in Hirt's original method or, more properly, of the modified equation itself in the Warming and Hyett method.

To relate the q_{tt} term in Eq. (20) to spatial variables, we differentiate the model equation (2), first with respect to time and then with respect to x, to obtain

$$q_{tt} = -uq_{xt}, \qquad q_{tx} = -uq_{xx} \tag{21}$$

Substituting (21) into (20), and then using (4), gives

$$q_t = -uq_x + \frac{wu^2\,\Delta t}{4} q_{xx} + O(\Delta x^2, \Delta t^2) \tag{22}$$

$$q_t = -uq_x + \frac{wuc\,\Delta x}{4} q_{xx} + O(\Delta x^2, \Delta t^2) \tag{23}$$

In the interpretation of heuristic stability analysis, the FLTW method represents the original continuum inviscid equation (2) to $O(\Delta x^2, \Delta t)$, but to $O(\Delta x^2, \Delta t^2)$ represents a viscous equation with an effective diffusion coefficient of $wuc\,\Delta x/4$. The positivity of this *numerical* diffusion coefficient indicates at least conditional sta-

bility, the concept being that positive *physical* (i.e., continuum) diffusion would be stable in the continuum equations. Negative numerical diffusion of other numerical methods correctly predicts instability [5].

VON NEUMANN ANALYSIS

The von Neumann analysis of FLTW is somewhat involved because the method involves four time levels, and the complex eigenvalues are obtained from a cubic equation. We utilize the symbolic manipulation code Macsyma to assist in the analysis.

Following the well-established Fourier decomposition in time and space (e.g., see [3]), we write

$$q_i^n = V^n \exp(Ii\theta) \tag{24}$$

where V is the amplitude function, $I = \sqrt{(-1)}$, and θ is the phase angle of the Fourier component, equal to $k\,\Delta x$, where k is the component wave number (wavelength $\Lambda = 2\pi/k$). Substituting Eq. (24) into Eq. (9), dividing through by $\exp(Ii\theta)$, and defining

$$a = -c[\exp(I\theta) - \exp(-I\theta)] = -2Ic\sin(\theta) \tag{25}$$

gives the evolutionary equation for V.

$$\begin{aligned} V^{n+1} &= aV^n + V^{n-1} + \frac{w}{2}(V^n - 2V^{n-1} + V^{n-2}) \\ &= \left(a + \frac{w}{2}\right) V^n + (1 - w)V^{n-1} + \frac{w}{2} V^{n-2} \end{aligned} \tag{26}$$

The matrix recursion relation is obtained by augmenting Eq. (26) with identities and zeros as required, resulting in

$$\begin{bmatrix} V^{n+1} \\ V^n \\ V^{n-1} \end{bmatrix} = G \begin{bmatrix} V^n \\ V^{n-1} \\ V^{n-2} \end{bmatrix} \tag{27}$$

where G is the amplification matrix, given by

$$G = \begin{bmatrix} \left(\dfrac{a + w}{2}\right) & (1 - w) & \dfrac{w}{2} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{28}$$

The eigenvalues λ of Eq. (28) are found from the following determinant equation of the characteristic polynomial equation for (28):

$$\begin{bmatrix} \left(\dfrac{a + w}{2} - \lambda\right) & (1 - w) & \dfrac{w}{2} \\ 1 & -\lambda & 0 \\ 0 & 1 & -\lambda \end{bmatrix} = 0 \tag{29}$$

This expands to the cubic equation for eigenvalues λ as

$$\lambda^3 - \left(a + \frac{w}{2}\right)\lambda^2 - (1 - w)\lambda - \frac{w}{2} = 0 \tag{30}$$

Starting with the amplification matrix equation (28), the symbolic manipulation code Macsyma was used to form the characteristic polynomial equation (29) and expand it to the cubic equation (30). The equation solver in Macsyma was used to solve the cubic equation in closed form. The Fortran capability of Macsyma was then used to convert the cubic solution to a Fortran notation, which was then grouped and simplified by hand calculations to obtain a subroutine. This subroutine was called over a range of phase angles θ from 0 to 360° at 1° increments, and each of the three solutions for $|\lambda|$ was evaluated and plotted.

Scanning for max $|\lambda|$ over θ as a function of Courant number then indicates the stability limit. The numerical evaluation for $w = 1$ showed that max $|\lambda| = 1$ for $c \leq 1/2$, but max $|\lambda| > 1$ for $c > 1/2$, indicating instability. Examples of the distribution of $|\lambda|$ in θ are shown in Fig. 1. Figure 1a shows the three solutions of $|\lambda|$ versus θ for $c = 0.4$; all three solutions give max $|\lambda| \leq 1$, indicating stability. Figure 1b shows the three solutions of $|\lambda|$ versus θ for $c = 0.6$; two of the three solutions give max $|\lambda| > 1$, indicating instability.

The dependence of max $|\lambda|$ on w is linear, as is to be expected, and the analysis indicates a stability limit on the Courant number c of

$$c \leq 1 - \frac{w}{2} \tag{31}$$

For $w = 0$ the present FLTW method reverts to leapfrog and (31) gives the usual restriction of $c \leq 1$. For the FLT method with $w = 1$, (31) gives a limit of $c \leq 1/2$. This limit has been verified experimentally (see below). Note that the time filtering of Asselin [7], although more difficult to apply, gives a somewhat less restrictive limit of $c \leq 0.57$. The phase error behavior is evaluated experimentally in the next section.

EXPERIMENTAL TESTS

In this section the new FLTW method is evaluated and compared to other methods using a common technique. An initial condition of a sine wave is set up in a grid and is advected through a time t by the numerical method. Exact initial conditions are used for all time levels in multi-time-level methods, and periodic boundary conditions are used; thus, interpretation is not clouded by questions regarding starting methods for multi-time-level methods or outflow boundary conditions for methods using centered space derivatives. The exact solution is simply a sine wave translated a distance ut, without change in amplitude or shape. The figures show the solution from the numerical method (symbol □) overplotted on the exact solution (symbol +). Damping errors are indicated by the reduced peak amplitude, and horizontal shifts are the result of accumulated phase errors.

These tests show the results for situations relevant to time-dependent flows. In

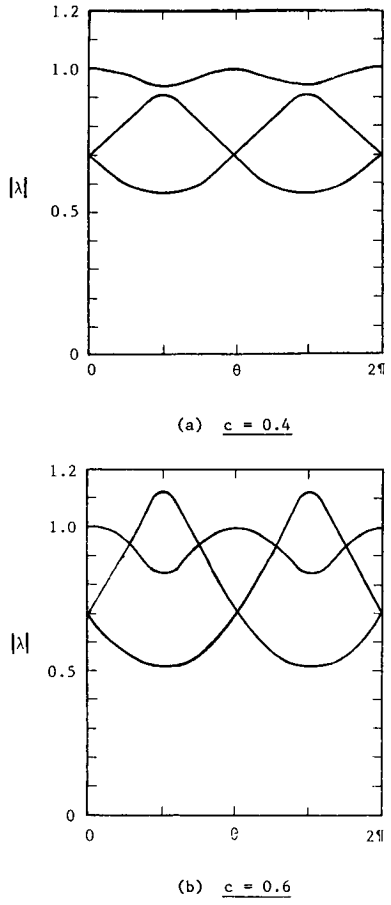


Fig. 1 Three solutions of the cubic stability equation for FLT (FLTW with w = 1).

a true steady flow situation, FLTW gives the same answers as any time differencing method that uses three-point centered space differencing (and is stable and does not decouple). However, as noted, one motivation for the FLTW modification to the leapfrog method is that leapfrog has a tendency to produce decoupled solutions at alternate time steps. This could be prevented by weighting leapfrog with any number of methods that do not themselves decouple: with FLT as in the present FLTW method, with upwind differencing, with Lax-Wendroff methods, etc. (e.g., see [3]). However, the (two-point) upwind method produces a strong artificial viscosity effect in both transient and steady-state results, as is well known. The Lax-Wendroff method is not so easily adapted to turbulence terms; note that viscous terms cannot simply be added to the inviscid Lax-Wendroff formulations but must be derived anew. Also, the mixed space-time formulation of the Lax-Wendroff methods results in the peculiar property that steady-state solutions are a weak function of the time step used, and may be interpreted as displaying a steady-state artificial viscosity. (See Appendix B of Roache [3].) Thus, any comparisons that indicate roughly comparable performance for FLTW on a strongly transient problem may be regarded as a good evaluation of FLTW for quasi-steady problems.

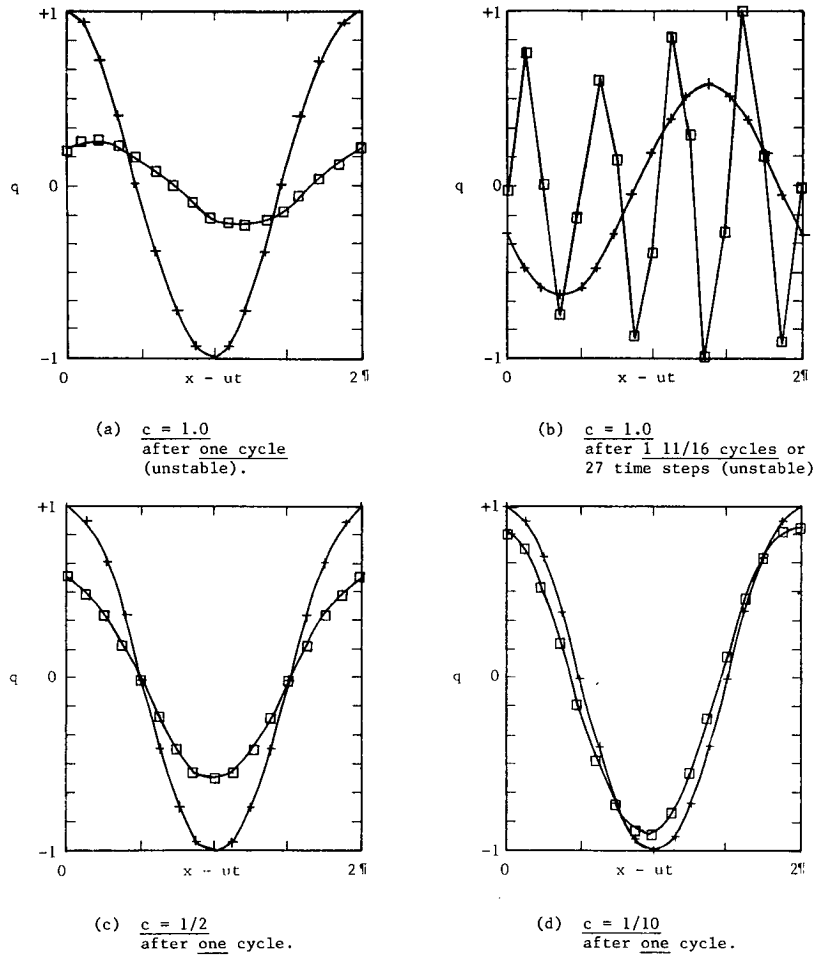


Fig. 2 Exact solution (+) and numerical solution produced by the FLT method (FLTW with $w = 1$).

In the following tests, the mesh consists of 16 cells. The initial amplitude is unity, and the exact solution maintains this amplitude. The modeled times are chosen to correspond to an integer number of cycles in the exact sine wave translation, e.g., one cycle or five cycles. The actual time step is determined by the Courant number c, which is a major factor in the performance of all methods.

Figures 2a and b show results for $c = 1$, a condition for which most methods (including leapfrog, upwind, and Lax-Wendroff) give exact results for the constant-coefficient model problem. From Fig. 2a, the FLT method (FLTW with $w = 1$) is seen to produce severe damping initially, with the amplitude reduced to 0.241 after only one cycle; a noticeable phase error has accumulated as well. However, this damping applies to the long-wavelength components. According to the von Neumann stability analysis, the short wavelengths are unstable at $c > 1$, so we expect to see these amplified eventually. Experimentally, damping continues through 21 time steps,

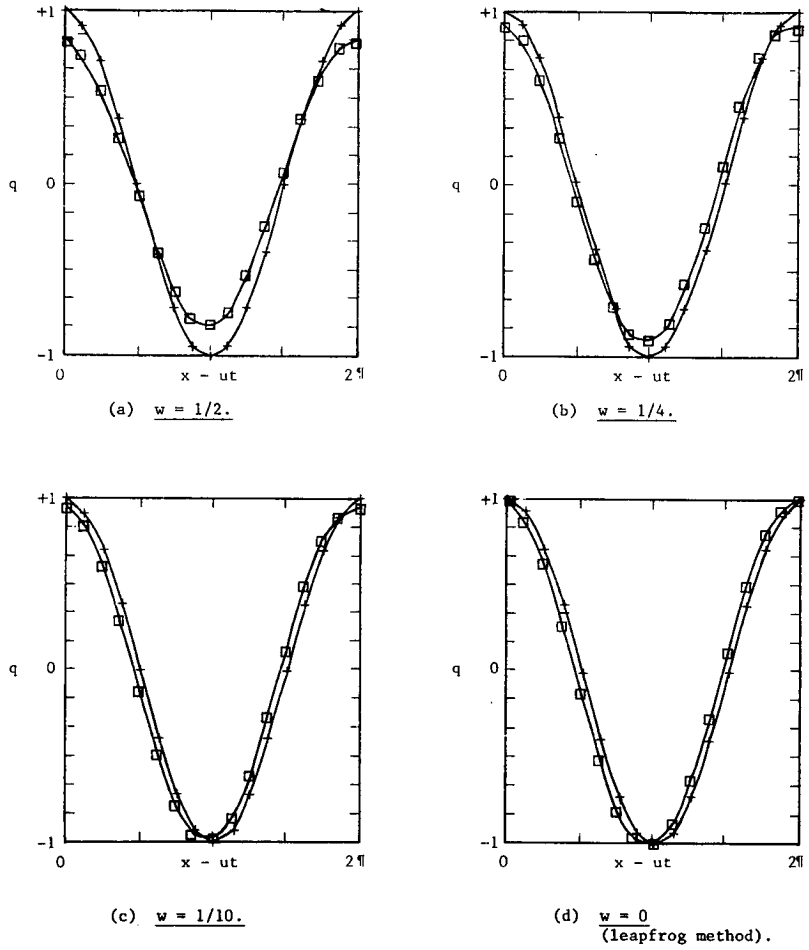
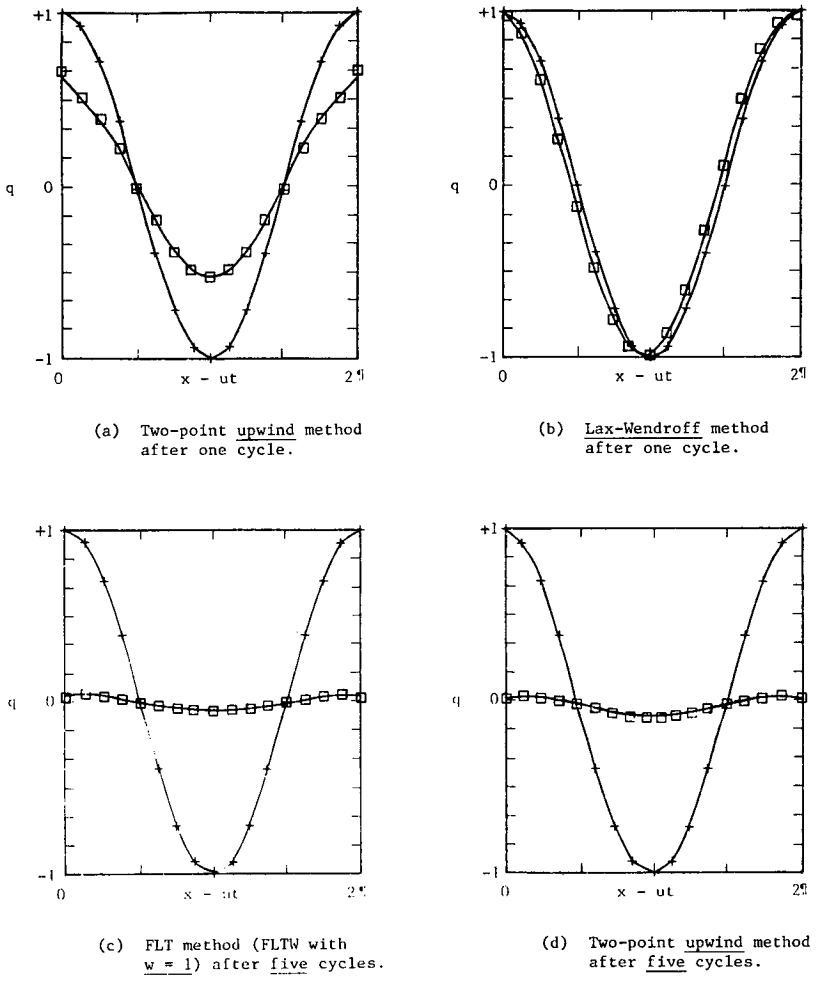


Fig. 3 Exact solution (+) and numerical solution produced for $c = 1/2$ by the FLTW method after one cycle.

with amplitude ratio = 0.17. However, beyond that point the shorter-wavelength instability is manifested, with the amplitude ratio exceeding 1 at time step 27 (or $1\frac{11}{16}$ cycles) as shown in Fig. 2b. By the time equivalent to two cycles, the calculation has completely blown up.

Figures 2c and d show the results of the FLT method (FLTW with $w = 1$) for stable time steps. Figure 2c shows the solution for $c = 1/2$ after one cycle. The amplitude is reduced to 0.548. Comparisons with other methods will be given shortly, but it is clear that this performance is inadequate for transient problems at this Courant number, which is its stability limit. Figure 2d shows the solution for $c = 1/10$ after one cycle. The amplitude is reduced to 0.879, considerably better than the $c = 1/2$ case.

The series of results in Figs. 3 through 5 are for the condition $c = 1/2$. This is the stability limit for the original FLT method (FLTW with $w = 1$) but is below



(a) Two-point upwind method after one cycle.

(b) Lax-Wendroff method after one cycle.

(c) FLT method (FLTW with w = 1) after five cycles.

(d) Two-point upwind method after five cycles.

Fig. 4 Exact solution (+) and numerical solution produced for $c = 1/2$.

the stability limit for $w < 1$ and therefore represents a realistic condition for actual simulations for all the methods compared.

In spite of the high damping of the FLT method (or FLTW with $w = 1$) for $c = 0.5$, it is still useful for steady-state problems, since its high temporal damping reduces the computer CPU time required to attain a steady state and it requires one less array storage of velocity fields than FLTW. (It would also be useful in a "false transient" approach using a local time step to efficiently attain steady-state solutions, but of course it does not cure the problem of possible oscillatory character of these steady solutions at high Re; see, e.g., Roache [3]). However, for strongly transient problems it is clear that the FLT method is not accurate at $c = 0.5$ and that $w < 1$ is advisable. Figures 2c and 3 show the effect of reducing w, with $w = 1$ (FLT) in Fig. 2c and $w = 1/2$, $1/4$, $1/10$, and 0 (leapfrog) in Figs. 3a–d. Comparisons with other methods are given in Fig. 4a, showing the (two-point) upwind method, and in

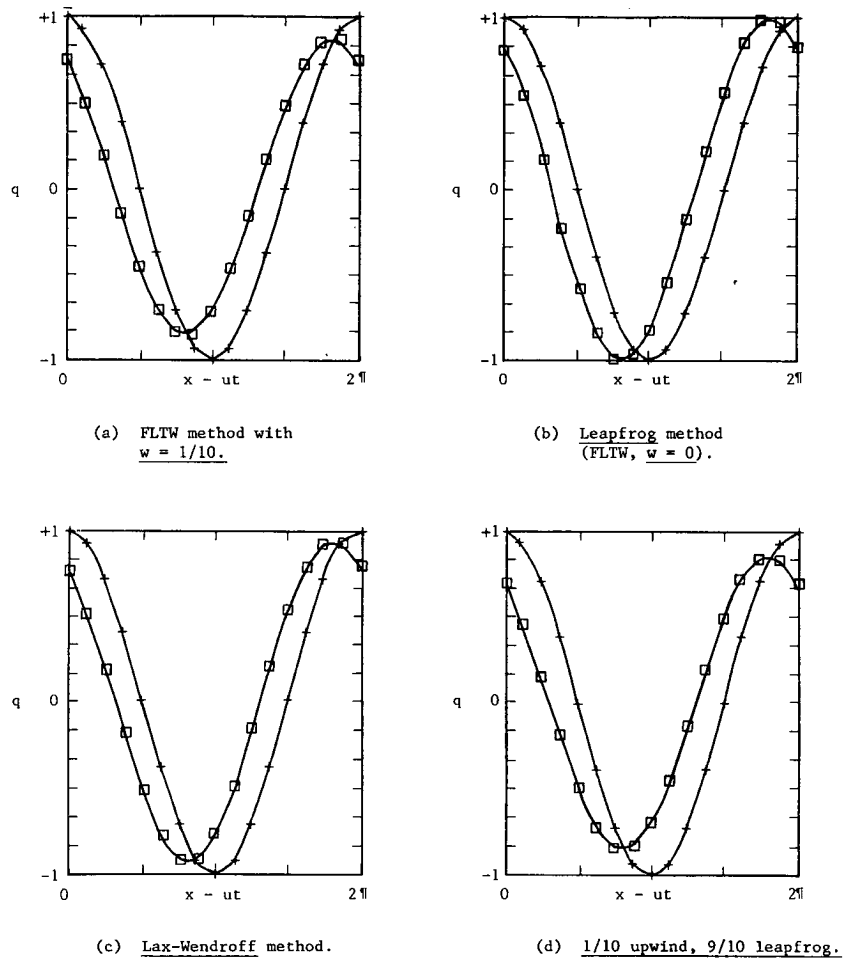


Fig. 5 Exact solution (+) and numerical solution produced for $c = 1/2$ after five cycles.

Fig. 4b, showing the Lax-Wendroff method. The FLTW method with $w = 1/10$ (Fig. 3c), which appears to be sufficient to prevent solution decoupling of the leapfrog method in steady-state problems, performs respectably well; the amplitude after one cycle is reduced to 0.967, certainly better than pure upwinding (0.537, Fig. 4a) and comparable to Lax-Wendroff (0.976, Fig. 4b) and leapfrog (0.994, Fig. 3d). Note that this experimental amplitude decrease for leapfrog is not the same as that which arises from its von Neumann analysis, which predicts zero damping (unity amplication factor). The reason [3] is that this experimental amplitude decrease is the result of both amplitude damping and phase errors. As the phase error of the wave moves the peak, sampling errors miss the resolution of the peak, and the discrete maximum value is decreased. Over many cycles, if the phase error occurs at multiples of the wavelength, the amplitude may return to unity for the leapfrog method.

More severe transient errors are demonstrated by calculating over more cycles. Figure 4c shows the FLT method (FLTW with $w = 1$) for $c = 1/2$ over five cycles.

The wave is virtually obliterated, with amplitude = 0.046. This is comparable to the well-known degradation of the upwind method, Fig. 4d, with amplitude = 0.045. However, the FLTW method with $w = 1/10$ shown in Fig. 5a gives amplitude = 0.840 after five cycles, not as good as the leapfrog method (0.980, Fig. 5b) or the Lax-Wendroff method (0.900, Fig. 5c) but still usable. Similar performance can also be obtained by a weighted combination of other methods with leapfrog. Figure 5d shows the results for a 1/10–9/10 weighting of upwind with leapfrog and gives an amplitude of 0.832 compared to 0.840 for FLTW with $w = 1/10$. However, a steady flow calculation with this scheme will have 1/10 the artificial viscosity effect of the upwind method, whereas the FLTW method will have none.

The FLTW method with $w = 1/10$ or less is seen to perform reasonably well on transient problems compared to leapfrog and Lax-Wendroff methods, and much better than upwind methods, for the representative case of $c = 1/2$. It appears that $w = 1/10$ is more than sufficient to prevent decoupling in quasi-steady flows, and this has been proved by our experience with the SOMS code [2] and with a high-Re gas flow code simulating spray nozzles; this experience also validates the applicability of the linear stability analyses to nonlinear problems. For these quasi-steady flows, the FLTW method has advantages over leapfrog, Lax-Wendroff, and certainly upwind methods.

CONCLUSIONS

The FLTW method applied to the model one-dimensional advection equation has been shown to be $O(\Delta t, \Delta x^2)$ accurate and to be conditionally stable for Courant number $c \leq 1 - w/2$, where $w = 1$ gives the original FLT method [1]. Unlike most methods, it does not preserve the exact solution for $c = 1$ or for its own stability limit. Its artificial damping is purely temporal, possessing no artificial viscosity for steady-state problems. The method was designed to overcome the problem of solution decoupling exhibited by the leapfrog method and to provide temporal filtering of rapid time oscillations arising from physically unrealistic, inaccurate, or incompatible initial conditions. The method involves at most a 33% operation count penalty in 3D compared to leapfrog for the simplest model problem, and only a few percent if significant additional physics is modeled. For strongly transient problems, the FLTW method with weighting parameter $w = 1/10$ or less performs reasonably well compared to the leapfrog and Lax-Wendroff methods, as judged by long-wavelength damping error and accumulated phase error.

REFERENCES

1. D. E. Dietrich and J. J. Wormeck, An Optimized Implicit Scheme for Compressible Reactive Gas Flow, *Numer. Heat Transfer,* vol. 8, pp. 335–348, 1985.
2. D. E. Dietrich, M. G. Marietta, and P. J. Roache, An Ocean Modeling System with Turbulent Boundary Layers and Topography: Numerical Description, *Int. J. Numer. Methods Fluids,* vol. 7, pp. 833–855, 1987.
3. P. J. Roache, *Computational Fluid Dynamics,* Hermosa Publishers, Albuquerque, N.M., 1976.
4. P. J. Roache, Marching Methods for Elliptic Problems. Part I, *Numer. Heat Transfer,* vol. 1, pp. 1–25, 1978.

5. C. W. Hirt, Heuristic Stability Theory for Finite-Difference Equations, *J. Comput. Phys.*, vol. 2, pp. 339–355, 1968.
6. R. F. Warming and B. J. Hyett, The Modified Equation Approach to the Stability and Accuracy Analysis of Finite Difference Methods, *J. Comput. Phys.*, vol. 14, pp. 159–179, 1974.
7. R. Asselin, Frequency Filter for Time Integrations, *Mon. Weather Rev.*, vol. 100, pp. 187–190, 1972.

Received April 22, 1987
Accepted December 2, 1987

Requests for reprints should be sent to Patrick J. Roache.