JOURNAL OF COMPUTATIONAL PHYSICS **22**, 131–149 (1976)

The Method of Lines Applied to a Simple Hyperbolic Equation*

JOHN GARY

Computer Science Department, University of Colorado, Boulder, Colorado

Received August 4, 1975; revised March 25, 1976

This paper is concerned with the semidiscrete approximation of hyperbolic equations in one-space dimension. A finite-difference approximation is used for the spatial derivative, but the time derivative is retained to yield a system of ordinary-differential equations. We compare fixed time step methods such as the leapfrog and fourth-order Runge–Kutta with variable time step ODE solvers such as Runge–Kutta–Fehlberg and the Adams method of Shampine. We are especially interested in the determination of the optimal time step for the fixed time step methods. A graphical method for this determination is described. We also test two predictor–corrector schemes, one based on the Milne corrector, the other based on an implicit Runge–Kutta formula.

1. INTRODUCTION

We are concerned with the application of a few of the standard methods for ordinary differential equations to the simple hyperbolic equations $u_t + u_x = 0$ or $u_t + uu_x = g(x, t)$, and the shallow water equations given in Section 5. Our objective is to compare the computational efficiency of these schemes. We are mainly interested in the time discretization. We replace the spatial derivative u_x, or $\frac{1}{2}(u^2)_x$ by a fourth-order finite difference using an equally spaced mesh. This yields a system of ordinary differential equations to which we apply the various methods. Thus we are using the well-known method of lines. This method has been used extensively [4, 8, 10, 13, 14] but we are not aware of comparisons with fixed time step methods which use an optimal time step. In some cases we use a fixed ratio $\Delta t/\Delta x$. To make the comparison we must therefore decide on the desired accuracy, then choose Δt and Δx to yield this accuracy at minimal cost [1]. We use a graphical method to find these optimal values. We compare the leapfrog scheme, a fourth-order Runge–Kutta, and two predictor–corrector schemes. The first is

* This work was supported in part by the National Center for Atmospheric Research which is funded by the National Science Foundation. A brief summary of this work is given in the AICA Symposium Proceedings [15]. This summary was also published in the AICA Journal [16].

based on the Milne corrector and the second on an implicit Runge–Kutta corrector [2]. We also use a variable step Runge–Kutta–Fehlberg ordinary-differential equation solver [5] and a variable order, variable step Adams method [6].

Our tests are limited to a few simple problems. Even if the tests were more extensive they could not define an optimal scheme. The solution may contain a shock, boundary layer, discontinuous material constants, or it may be defined on an irregular domain. A steady state solution may be desired, a very crude solution may be adequate, some global statistics such as the spectrum or long term averages may be desired rather than pointwise values. These factors can have a profound effect on the choice of a difference scheme. However, we feel that limited experiments such as this one can provide useful insight.

Our results tend to favor the fixed time step leapfrog or Runge–Kutta schemes. However, the ODE solvers tend to be almost as effective, and this comparison depends on the problem and the desired error. The ODE solvers do not require the user to select the time step. The tests described here are limited to periodic boundary conditions. To use more realistic boundary conditions is certainly of interest, but would create a much more complex test.

2. The Difference Schemes

We first describe the schemes as they apply to ordinary-differential equations. We write the ordinary-differential equation in the form

$$y' = f(y, t)$$

and denote the approximation to $y(t_n) = y(n\Delta t)$ by y_n. Then the leapfrog scheme is

$$y_{n+1} = y_{n-1} + 2\Delta t f(y_n, t_n). \tag{1}$$

The Runge–Kutta version which we use is

$$\begin{aligned}
k_1 &= \Delta t f(y_n, t_n)\\
k_2 &= \Delta t f(y_n + 0.5k_1, t_n + 0.5\Delta t)\\
k_3 &= \Delta t f(y_n + 0.5k_2, t_n + 0.5\Delta t)\\
k_4 &= \Delta t f(y_n + k_3, t_{n+1})\\
y_{n+1} &= y_n + (k_1 + 2k_2 + 2k_3 + k_4)/6.
\end{aligned} \tag{2}$$

The Milne corrector is based on

$$y_{n+1} = y_{n-1} + \Delta t(f(y_{n-1}, t_{n-1}) + 4f(y_n, t_n) + f(y_{n+1}, t_{n+1}))/3. \tag{3}$$

The other corrector is based on the following implicit Runge–Kutta scheme.

$$\begin{aligned} y_{n+1/2} &= (y_n + y_{n+1})/2 + \Delta t(f(y_n, t_n) - f(y_{n+1}, t_{n+1}))/8 \\ y_{n+1} &= y_n + \Delta t(f(y_n, t_n) + 4f(y_{n+1/2}, t_{n+1/2}) + f(y_{n+1}, t_{n+1}))/6. \end{aligned} \tag{4}$$

This scheme, which is implicit in $y_{n+1/2}$ and y_{n+1}, has been applied to diffusion problems by Watanabe and Flood [3].

We use the following Adams predictor which has third-order accuracy.

$$y_{n+1} = y_n + \Delta t(23f(y_n, t_n) - 16f(y_{n-1}, t_{n-1}) + 5f(y_{n-2}, t_{n-2}))/12. \tag{5}$$

These schemes are applied to the following hyperbolic equations

$$u_t + u_x = 0 \tag{6}$$

$$u_t + \tfrac{1}{2}(u^2)_x = g(x, t). \tag{7}$$

Periodic boundary conditions on the interval $0 \leqslant x \leqslant 2\pi$ are used. Two different solutions are used.

$$u(x, t) = \sin(x - t) \tag{8}$$

$$u(x, t) = \exp(0.5\sin(x - t)). \tag{9}$$

In the case of Eq. (7), the function $g(x, t)$ is chosen to yield the desired solution (8) or (9). Therefore, we know the exact solution in all our test cases.

In order to apply the method of lines to Eqs. (6) and (7), the spatial derivatives must be approximated. We use an equally spaced mesh.

$$x_j = j\Delta x = 2\pi j/J \qquad 1 \leqslant j \leqslant J$$

A fourth-order difference approximation is used for u_x and $(u^2)_x$, namely,

$$\delta_x(u) = (u_{j-1} - 8u_{j-1} + 8u_{j+1} - u_{j+2})/(12\Delta x).$$

This defines a system of ordinary differential equations which approximate (6) and (7)

$$(du/dt)j = f_j(\underline{u}, t) \qquad 1 \leqslant j \leqslant J$$

where for (6)

$$f_j = -\delta_x(\underline{u})_j$$

and for (7)

$$f_j = -\tfrac{1}{2}\delta_x(U^2)_j + g(x_j, t).$$

For a fixed ratio $\lambda = \Delta t/\Delta x$, the schemes defined above are subject to a stability restriction. By making the substitution

$$u_j^n = A^n e^{i2\pi m x_j}$$

and imposing the restriction that A^n should be bounded we obtain the stability restriction on the ratio λ. These are

1. Leapfrog: $\lambda \leqslant 0.73$
2. Runge–Kutta: $\lambda \leqslant 2.0$
3. Milne implicit: $\lambda \leqslant 1.26$ (10)
4. Implicit Runge–Kutta: unconditional stability, $\lambda \leqslant \infty$.

(Note that this applies to the corrector, as discussed below.)

5. Adams predictor: $\lambda \leqslant 0.5$.

The condition for the Adams predictor was determined numerically using a rootfinder. The same result is obtained from the stability regions given in the book by Shampine and Gordon [6, p. 136], although the numerical results must be divided by the maximum amplitude of the Fourier transform of the spatial difference.

We also use a version of the Lax Wendroff scheme [12, p. 302]. For the equation

$$u_t + f_x = g$$

this scheme is given by

$$\begin{aligned} u_{j+1/2}^{n+1/2} &= 1/2(u_{j+1}^n + u_j^n) - (\Delta t/2\Delta x)(f_{j+1}^n - f_j^n) + (\Delta t/2)\, g_{j+1/2}^n \\ u_j^{n+1} &= u_j^n - (\Delta t/\Delta x)(f_{j+1/2}^{n+1/2} - f_{j-1/2}^{n+1/2}) + \Delta t\, g_j^{n+1/2}. \end{aligned} \tag{11}$$

The stability condition here is $\lambda \leqslant 1$.

3. The Predictor–Corrector Methods

This scheme was suggested by J. Klemp at NCAR. The Milne corrector iteration takes an approximate value for y_{n+1}, and substitutes this value into the right-hand side of Eq. (3) to obtain a new approximation for y_{n+1}. This is a multipstep scheme and requires the values of both y_n and y_{n-1}. In our experiments with Eqs. (6) and (7) we started the integration for the leapfrog and Milne schemes by using the exact solution for the required previous time levels. By using the Fourier transform of Eq. (10) we can show that this corrector iteration will converge provided $\lambda \leqslant 2.19$. Therefore, the stability requirement (10) is more restrictive for the Milne corrector.

We used three predictors. The first is the Lax–Wendroff of Eq. (11), predicting from u^n to u^{n+1}. The second is the Adams predictor, from u^{n-2}, u^{n-1}, and u^n to u^{n+1}. The third is another version of the Adams predictor which predicts in two steps. The first from u^{n-2}, u^{n-1}, u^n to $u^{n+1+1/2}$ to u^{n+1}. This predictor is based on the following Adams schemes for ordinary differential equations.

$$\begin{aligned} y_{n+1/2} &= y_n + \Delta t(17f(y_n, t_n) - 7f(y_{n-1}, t_{n-1}) + 2f(y_{n-2}, t_{n-2}))/24 \\ y_{n+1} &= y_{n+1/2} + \Delta t(64f(y_{n+1/2}, t_{n+1/2}) - 33f(y_n, t_n) + 5f(y_{n-1}, t_{n-1}))/72. \end{aligned} \tag{12}$$

The Lax-Wendroff is a dissipative scheme, at least for $\lambda < 1$. Therefore it might make a good predictor provided $\lambda < 1$. However it has only second-order accuracy. The Adams predictor (5) has third-order accuracy. However, it is stable only if $\lambda < 0.5$. Since the Milne corrector is stable for $\lambda \leqslant 1.26$, the Adams predictor would be used in an unstable region. Therefore we also tried the Adams predictor in the two step version (12) under the assumption that the effective reduction in Δt would yield a more accurate predictor and avoid amplification of higher wave numbers due to a stronger instability in the predictor. We did not determine the stability condition for this predictor. The two step predictor does require fewer corrector iterations, but seems to be no more efficient than the one step predictor because of the additional predictor iteration.

TABLE I

Comparison of Predictors[a]

Predictor	Error	ϵ	CPU time	N_E
Lax-Wendroff	4.5E − 2	5.0E − 3	202	97
predictor	3.2E − 3	2.0E − 3	194	93
	8.7E − 3	1.0E − 3	251	121
	4.6E − 3	5.0E − 4	280	134
Single step	1.2E − 2	5.0E − 3	193	97
Adams predictor	6.0E − 3	2.0E − 3	222	111
	4.4E − 3	1.0E − 3	248	125
	4.5E − 3	5.0E − 4	292	147
Double step	5.0E − 3	5.0E − 3	185	94
Adams predictor	7.0E − 3	2.0E − 3	220	111
	5.3E − 3	1.0E − 3	246	124
	4.9E − 3	5.0E − 4	283	141

[a] Used with the Milne corrector, $J = 20$, $\lambda = \Delta t u_m/\Delta x = 1.13$, for the nonlinear equation (7) with solution (9). Here ϵ is the convergence tolerance used for the corrector iteration; N_E the number of evaluations of the right side including the predictor; the CPU time is in milliseconds on the NCAR CDC 7600.

In all cases we iterate the corrector until the absolute difference between successive iterates is less than some preassigned ϵ. The comparison of these predictors for the Milne corrector is shown in Table I. These results are for the nonlinear equation

(7) with the solution (9). The error is the maximum relative error over the mesh at $t = 2\pi$. The number of evaluations of the right side of the equation is given. A prediction using Lax-Wendroff is counted as a single evaluation. A double Adams predictor followed by a single Milne corrector requires three evaluations for a single time step. In all cases 20 mesh points ($J = 20$) are used. The total number of time steps is 30 with $\lambda = 1.13$. We use this value of λ because it is 10% below the maximum permitted by stability. Note that $\lambda = u_m \Delta t / \Delta x$ where

$$u_m = \max_j | u(x_j, 0)|.$$

The average number of corrector iterations per time step at $\epsilon = 2.0\text{E} - 3$ is 2.1 for the Lax–Wendroff predictor, 2.7 for single-step Adams, and 1.7 for double-step Adams.

The implicit Runge–Kutta corrector (4) computes the $y_{n+1/2}$ approximation using the predicted y_{n+1}. Then a corrected y_{n+1} is obtained from the second equation of (4). This iteration will converge for the linear equation (6) provided $\lambda \leqslant 1.33$. The implicit corrector is unconditionally stable, but the iteration is not.

These predictors seem to give equivalent results within the variability one might expect from problem to problem. Note that the Lax–Wendroff predictor shows an increase in error going from $\epsilon = 2.0\text{E} - 3$ to $\epsilon = 1.0\text{E} - 3$, which is especially serious. The larger error for the single Adams predictor at $\epsilon = 5.0\text{E} - 3$ is probably not too serious, since ϵ can be somewhat below the desired error although this does increase the computational cost. We use the Lax–Wendroff predictor with the implicit Runge–Kutta scheme since it is compatible with the single step (U^n to U^{n+1}) nature of the Runge–Kutta, unlike the Adams predictor which requires previous time levels. We use the double step Adams predictor with the Milne corrector since it seems to be slightly superior to the other predictors. However, this comparison is likely to be problem dependent.

4. A Graphical Determination of the Optimal Mesh Ratio

We wish to choose values of Δx and Δt which will yield a preassigned error with minimal computational effort. We assume the computational cost is

$$C = K / \Delta t \Delta x^d \tag{13}$$

where d is the dimension of the space ($d = 1, 2,$ or 3) and K is a constant depending on the problem and the difference scheme. We assume the error has the asymptotic representation

$$\epsilon(\Delta t, \Delta x) = O(\Delta t^p) + (\Delta x^q)$$

for a fixed t and a fixed problem. For example, the leapfrog scheme with a fourth-order spatial difference has $q = 4$ and $p = 2$. Our objective is to achieve a preassigned error $\epsilon(\Delta t, \Delta x) = \epsilon$ at a minimal cost. We choose Δt so that the asymptotic spatial and temporal errors are balanced, that is $\Delta t = \alpha \Delta x^{q/p}$. More exactly, we define α for the nonlinear problem (7) by

$$\alpha = \Delta t u_m / \Delta x^{q/p}$$

where the normalizing factor u_m is

$$u_m = \max_j | u(x_j, 0)|.$$

We will further assume that the truncation error has the following form ($u(x, t)$ is the exact solution and u_j^n the solution of the difference scheme on the mesh (x_j, t_n)).

$$u(x_j, t_n) - u_j^n = \Delta x^q \epsilon(x_j, t_n, \alpha) + O(\Delta x^{q+1}). \qquad (14)$$

Note that the function $\epsilon(x, t, \alpha)$ is independent of Δx. It is then possible to show the existence of an optimal value of α which yields a given error ϵ at minimal computational cost [1]. This optimal value is independent of the preassigned error ϵ.

In this paper we determine by computational experiment the error

$$\max_j \tau_j^n = \max_j | u(x_j, t_n) - u_j^n |. \qquad (15)$$

The error is computed for various values α_μ and J_ν ($1 \leqslant \mu \leqslant N$, $1\, \nu \leqslant M$ for integers N and M). The equation is integrated to a fixed value $t = T$ (usually $T = 2\pi$, or one period). The value of Δx is fixed by J_ν, and then the value of Δt is fixed by α_μ.

Since $T/\Delta t$ may not be an integer we have to change Δt on the last time-step. For the single step schemes (Lax–Wendroff or Runge–Kutta), we simply change the value of Δt in the last step. In the case of the leapfrog scheme, which is second order in time, we use quadratic interpolation to obtain a value u^{n-1} with $t_{n+1} = T$, $t_{n+1} - t_n = t_n - t_{n-1}$. The value of u^n is unchanged. For the other multistep schemes (Milne and implicit Runge–Kutta correctors) we use the Runge–Kutta for this last time step. This fails to provide a pure comparison of the difference schemes. However, the contribution to the error due to this last step seemed to be small and we prefer to run at a preassigned α_μ.

Thus we compute $\tau(\alpha_\mu, J_\nu)$ at $t = T$. We choose α_μ and J_ν so that the ratios $\alpha_{\mu+1}/\alpha_\mu$ and $J_{\nu+1}/J_\nu$ are nearly constant. We display the results in the form of a contour plot of $\log_{10}(\tau)$ against $\ln(J_\nu)$ and $\ln(\alpha_\mu)$ on the axes. The values J_ν must be integers and therefore the values $\ln(J_\nu)$ are not equally spaced along the axis. In order to avoid modification of the contour plotting routine, we use cubic spline interpolation to obtain values of $\tau(\alpha_\mu, J_\nu)$ at equally spaced values of $\ln(J_\nu)$. The contour plots are based on these interpolated values. From Eq. (13) for the cost

$$\begin{aligned}\log_{10} C &= \log_{10} K - \log_{10} \Delta t - d \log_{10} \Delta x \\ &= \log_{10} \hat{K} + \log_{10} \alpha + (d + q/p) \log_{10} J.\end{aligned}$$

Here $\hat{K}$ is a new constant which differs from K. Therefore, the contour plot of the cost function is a set of straight lines. These are shown as dashed lines on the plots. The contour interval for the curves of constant error is $\log \sqrt{10} = 0.5$. Thus a spacing of two curves corresponds to a factor of 10 in the error. The contour interval for the logarithmic cost function is 0.25.

5. The Comparison of the Schemes

In fig. 1 we show the curves for the implicit–Runge–Kutta corrector. There is no predictor involved here; this is an implicit scheme. This plot is for the linear equation (6) with solution (8). To avoid coding an implicit scheme we obtain the error from the Fourier transform of the difference equation. This should yield the same result as would be obtained by running the difference scheme. The optimum operating point is obtained where the cost and error curves are tangent. In Fig. 1, this optimum is approximately $\alpha_{\text{opt}} = 2.2$ or 2.3. Note that the error curves are parallel in this case which yields an α_{opt} independent of the error ϵ. The asymptotic form of the error $\Delta x^q \epsilon(x, t, \alpha)$ in Eq. (14) predicts parallel error curves.

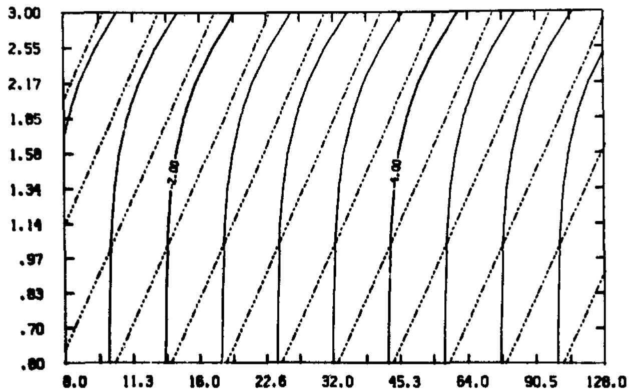

Fig. 1. Curves of constant error (logarithmic) for the implicit Runge–Kutta applied to (6) with solution (8). The horizontal axis is in J and the vertical in α.

In Fig. 2 we show the curves for the leapfrog scheme with fourth-order spatial differences applied to the linear problem (6) with solution (8). Here we observe the cancellation between the spatial and temporal errors which has been noted by Swartz and Wendroff [10]. It is unreasonable to use this problem to compare schemes, since this cancellation cannot be expected on more complex problems.

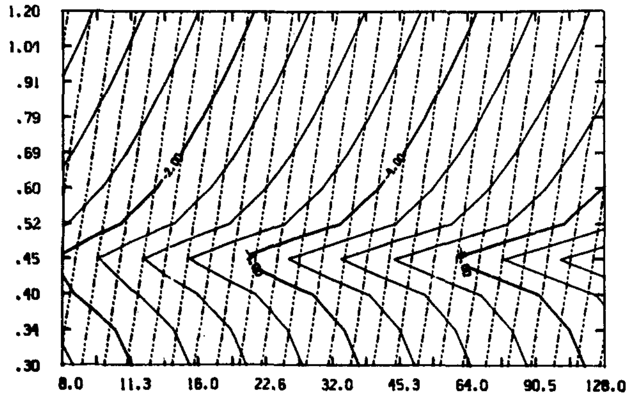

FIG. 2. Curves of constant error for the leapfrog applied to (6) with solution (8).

We do not include the error curves for the Milne and implicit Runge–Kutta correctors since they are not tangent to the cost curves within the range of α for which the corrector iteration (or the corrector) is stable. Thus we should use the maximum value of α permitted by stability. We actually use 90 % of this value, since it is impossible to run at the limit in most nonlinear problems.

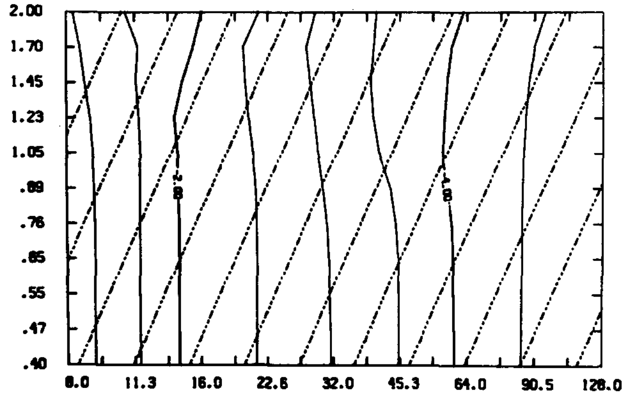

FIG. 3. Curves of constant error for Runge–Kutta applied to (7) with multiwave solution (9).

For the nonlinear problem (7) with the solution (9) the error curves are not parallel. In Fig. 3 the curves for the Runge–Kutta scheme are shown. The curves generally indicate that the maximum value of α should be used, so we set $\alpha = 1.8$. Figure 4 displays the same case as Fig. 3 except the solution is the single wave number of Eq. (8) instead of the multiple wave numbers of Eq. (9). In both cases the Runge–Kutta scheme is applied to the nonlinear equation (7). The curves for the two cases are certainly different, although the optimal values of α are not much different in the two cases. We do not see any way to draw a general conclusion concerning the dependence of the optimal α on the problem.

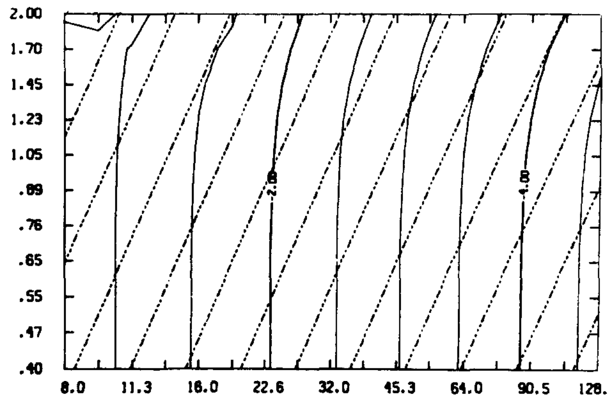

FIG. 4. Curves of constant error for Runge–Kutta applied to equation (7) with single wave solution (8).

The straight dashed lines on these figures are lines of constant computational cost. They are spaced at a logarithmic interval of 0.25, therefore the cost factor between two adjacent lines is $10^{0.25} = 1.8$. Looking at Fig. 3 (Runge–Kutta, Eq. (7) solution (9)) we see that a change in α from 0.89 to approximately 1.4 along the constant error curve at 10^{-3} would reduce the cost by a factor of 1.8 since these points lie on adjacent constant cost lines.

In Tables II and III we compare the schemes. The previous discussion of Fig. 1–4 concerned the choice of an optimal time-step for a fixed time-step scheme. Here we are concerned with a comparison between schemes including those based on an ODE solver. The value of α used in Table II is chosen to give optimal performance from contour plots of cost and error. In these tables J is the number of mesh points in one wavelength. The error is relative error. The ratio α is $\alpha = \Delta t u_m / \Delta x^{q/p}$ as defined earlier. The convergence parameter for the corrector iteration is ϵ and refers to an absolute rather than relative difference between successive iterates. The CPU time is measured in milliseconds on the CDC 7600 at NCAR. The number

TABLE II

Comparison of Schemes for the Linear Equation (6) with Solution (9)[a]

Scheme	J	Error (rel)	α	ϵ	CPU time	N_E
Leapfrog	20	2.1E − 3	0.7	—	103	92
Runge–Kutta	20	7.1E − 3	1.4	—	66	61
Milne	20	2.4E − 3	1.13	5.0E − 4	89	82
Imp. Runge–Kutta	20	7.5E − 3	1.2	5.0E − 4	125	122
Imp. Runge–Kutta	20	4.3E − 3	1.2	1.0E − 4	214	198
Lax–Wendroff	20	1.2E − 2	0.9	—	11	24
Leapfrog	8	4.7E − 2	0.7	—	7	16
Runge–Kutta	9	5.8E − 2	1.4	—	15	29
Lax–Wendroff	8	5.6E − 2	0.9	—	2	10
Leapfrog	15	6.6E − 3	0.7	—	45	53
Runge–Kutta	21	5.9E − 3	1.4	—	71	61
Lax–Wendroff	26	6.9E − 3	0.9	—	18	30

[a] Here $\alpha = \Delta t u_m / \Delta x^{q/p}$; ϵ = convergence tolerance for corrector iteration; N_E = number of evaluations of right side; the CPU time is in milliseconds on the NCAR CDC 7600.

TABLE III

Comparison of Schemes for the Nonlinear Equation (7) with Solution (9)[a]

Scheme	J	Error	α	ϵ	CPU time	N_E
Runge–Kutta	20	3.5E − 3	1.80	—	153	77
Leapfrog	20	5.5E − 3	1.20	—	178	89
Milne	20	5.3E − 3	1.13	1.0E − 3	247	124
Imp. Runge–Kutta	20	3.5E − 3	1.20	1.0E − 3	565	275
Lax–Wendroff	20	5.4E − 2	0.90	—	87	38
Runge–Kutta	9	6.9E − 2	1.80	—	34	37
Leapfrog	11	5.0E − 2	1.20	—	30	28
Milne	10	7.1E − 2	1.13	0.01	63	63
Lax–Wendroff	18	6.3E − 2	0.90	—	71	34
Runge–Kutta	17	6.9E − 3	1.80	—	112	65
Leapfrog	19	6.5E − 3	1.20	—	154	80
Milne	18	6.5E − 3	1.13	0.001	234	127
Lax–Wendroff	100	6.6E − 3	0.90	—	2 1E + 3	185

[a] Notation same as Table II.

of evaluations refers to the right side of the ordinary-differential equation obtained by the method of lines. Table II refers to the linear problem (6) with the multi-wavenumber solution (9). In this problem the leapfrog scheme still exhibits some cancellation between temporal and spatial truncation. We first compare the schemes by running all of them at the same resolution, $J = 20$. The value of α (i.e., Δt) is chosen from the contour plots to yield the optimum approximation. Additional comparisons are obtained by using a value of J which yields (approximately) the relative error 0.06 and 0.006. This corresponds with an error in the phase angle of 1 % and 0.1 %. Tables II and III indicate that the implicit Runge–Kutta corrector is quite inefficient compared with the other schemes. Table III indicates that the Milne may not be as good as the Runge–Kutta or leapfrog although they are close enough so that this conclusion may be problem dependent. Use of the Milne corrector requires that the additional corrector convergence parameter ϵ be determined, which is a disadvantage since the efficiency of the scheme is sensitive to this choice.

The Lax–Wendroff scheme does very well on the linear problem of Table II. But this is due to cancellation of spatial and temporal truncation errors, and therefore would not apply to nonlinear problems as Table III indicates. The leapfrog scheme also exhibits cancellation on this problem along with the Milne corrector. Therefore, this linear problem is a poor test case. We prefer to use Table III.

For the nonlinear problem of Table III, the leapfrog scheme or the Runge–Kutta seem to be the best. The predictor–corrector schemes are not as efficient, and they also require adjustment of the convergence parameter ϵ, which is a disadvantage. These results are likely to be highly problem dependent. Also, we have not considered difficulties due to boundary conditions. However, the leapfrog and Runge–Kutta seem to be superior. As we might expect, the leapfrog, which is only secondorder accurate in time, tends to be best at lower resolution. The results are in agreement with those of Swartz. Although the publication [10] does not include the simple spatial discretization which we used, he does have unpublished results using the same discretization which agree with ours.

There are many excellent programs available for the integration of ordinary-differential equations. These programs set the time step dynamically in order to achieve a preassigned temporal error. We selected a Runge–Kutta–Fehlberg (RKF) routine as given in a paper by Enright *et al.* [5].

We are indebted to Jay Chalmers for coding this program from the listing given by Enright *et al.* The results obtained from this program applied to the nonlinear problem (7) are given in Table IV. We modified the program in order to set the initial time step equal to that determined by the input parameter α. The error tolerance ϵ is given per unit step. This variable step method is apparently less efficient than the fixed time step Runge–Kutta. Its use seems to require about twice the computing for the same error for this particular problem. However, the variable step method is probably more robust and also more convenient provided the choice of ϵ is not troublesome. The results in Table IV show the dependence of the computational efficiency on the choice of ϵ.

TABLE IV

Results using the Runge–Kutta–Fehlberg (RKF) Variable Step Method for the Nonlinear Equation (7) with Solution (9)[a]

J	Error	Initial α	ϵ	CPU time	N_E
20	4.1E − 3	2.0	1.0E − 1	206	103
20	4.1E − 3	2.0	5.0E − 3	207	103
20	5.3E − 3	2.0	1.0E − 3	254	127
20	5.2E − 3	2.0	5.0E − 4	293	145
11	4.5E − 2	2.0	5.0E − 2	68	61
11	4.7E − 2	2.0	1.0E − 2	67	61
11	4.6E − 2	2.0	1.0E − 3	119	107
11	1.1E − 1	4.0	5.0E − 2	64	58
11	5.1E − 2	4.0	1.0E − 2	78	70
11	4.6E − 2	4.0	1.0E − 3	124	112
18	6.6E − 3	3.0	1.0E − 3	228	126

[a] The initial α is used to set both the initial and maximum step size. The ϵ refers to the convergence tolerance used in the RKF routine (absolute error per unit step). The remaining parameters are as in Table II.

An ODE solver due to Shampine was also used. This program is based on a variable order, variable step Adam's integrator. The results are given in Table V. The Adams integrator requires fewer functional evaluations for our test cases, as it does for most test cases [7]. The code we used allowed a maximum of 12th-order accuracy. This is more than we require and cost us 22 working storage locations for each unknown in the ordinary differential equation system. If we had reduced the maximum order to 6, then 14 locations would be required. If this order is reduced too much, then an underestimate of the error tolerance ϵ can result in an expensive computation. The fourth-order Runge–Kutta–Fehlberg that we used requires six locations per unknown.

This routine requires specification of both a relative and an absolute error tolerance. We set the absolute tolerance to ϵU_m where U_m is the maximum norm of the solution. The first line of Table V indicates that the error tolerance ϵ must be sufficiently small to avoid an unreasonably large error. On the other hand, a reduction in ϵ does not seem to increase the computational cost too much, although it will cause the scheme to run at a higher order which requires more storage. The anomalous behavior for $J = 18$ is probably due to an interaction between spatial and temporal truncation error. We printed Δt and the order used by this ODE solver at each step for $J = 18$ over integrations using different values of ϵ. We could see nothing peculiar in the behavior of the integrator. The larger ϵ ($\epsilon = 0.001$) allows the integrator to reach $t_g = 1.55$ in nine time steps, whereas $t_g = 1.17$ after nine steps with $\epsilon = 0.0005$. The error at t_g corresponding to these two ϵ values is 0.014 and 0.0038. The larger ϵ apparently allowed too large a step initially. The larger ϵ produced a smaller Δt later which caused the number of evaluations to increase. The average order used in the Adams method after the first four steps was 5.0 with $\epsilon = 0.001$ and 4.3 with $\epsilon = 0.0005$. In the second case where the average order was 4.3, the order ranged from 3 to 6.

TABLE V

Results using the Variable Step Adams Method of Shampine for the Nonlinear Equation (7) with Solution (9)[a]

J	Error	Initial α	ϵ	CPU time	N_E
20	4.0E − 2	3.0	5.0E − 3	170	78
20	4.9E − 3	3.0	1.0E − 3	195	89
20	5.0E − 3	3.0	5.0E − 4	202	92
11	5.3E − 2	3.0	5.0E − 3	58	47
11	4.9E − 2	3.0	1.0E − 3	82	65
18	9.4E − 3	3.0	1.0E − 3	185	93
18	5.5E − 3	3.0	5.0E − 4	158	79

[a] The initial α is used to set both the initial and maximum step size. The ϵ refers to the convergence tolerance used in the RKF routine (relative error per unit step). The remaining parameters are as in Table II.

We ran one nonlinear test case which did not have a large source term. This is more typical of many problems, however as a consequence we do not have an exact solution for this case. We used a dimensionless form of the following shallow water equations

$$\begin{aligned} u_t + uu_x - gh_x &= 0 \\ h_t + H_0 u_x + (uh)_x &= 0. \end{aligned} \tag{16}$$

Here H_0 is a constant which represents a "mean" height. If the scaling factors $H_1(g/H_0)^{1/2}$ and H_1 are used for u and h, then the equations reduce to the dimensionless form

$$\begin{aligned} u_t + ((\alpha/2)\, u^2 + h)_x &= 0 \\ h_t + (u(1 + \alpha h))_x &= 0 \end{aligned} \tag{17}$$

where α is the ratio $\alpha = H_1/H_0$; that is, α is the measure of the perturbation in h relative to the mean height H_0. We assume periodic boundary conditions on the interval $0 \leqslant x \leqslant 1$, use the value $\alpha = 0.05$, and the initial conditions

$$u(x, 0) = h(x, 0) = \sin 2\pi x.$$

If the value $\alpha = 0.1$ is used, a shock seems to develop before $t = 2\pi$. If $\alpha = 0.05$, the sine curve for $u(x, t)$ is clearly distorted at $t = 2\pi$, but seems to remain continuous. In Table VI the result of solving this system with the Adam's integrator of Shampine is shown. Several values of $h = \Delta x/2 = 1/J$ are used and the solution is checked at three mesh points. The maximum values of u and h for $t = 2\pi$ occur near $x = 0.32 \times 2\pi$.

TABLE VI

	$U(x, h_0) - U(x, h_0)$	$\dfrac{U(x, h_{\nu+1}) - U(x, h_0)}{U(x, h_\nu) - U(x, h_0)}$	$\dfrac{(h_{\nu+1}/h_0)^4 - 1}{(h_\nu/h_0)^4 - 1}$	$\dfrac{E(h_\nu, 2\pi) - E(h_\nu, 0)}{E(h_\nu, 0)}$	ϵ
$x = 0.3 \times 2\pi$					
$h_0 = 1/250$	—	—	—	−1.57E − 4	1.E − 8
$h_1 = 1/140$	6.E − 7	—	—	−1.54E − 4	1.E − 7
$h_2 = 1/80$	5.5E − 6	9.2	10.3	−1.21E − 4	1.E − 6
$h_3 = 1/40$	2.64E − 4	48.	16.2	4.15E − 4	1.E − 5
$h_4 = 1/20$	4.13E − 4	1.6	16.0	8.02E − 3	5.E − 4
$h_5 = 1/10$	1.4E − 1	340.	16.0	6.18E − 2	5.E − 4
$x = 0.4 \times 2\pi$					
h_1	9.4E − 6	—	—		
h_2	9.66E − 5	10.3	10.3		
h_3	2.56E − 3	26.	16.2		
h_4	2.32E − 2	9.1	16.0		
h_5	−1.5E − 1	−64.	16.0		
$x = 0.6 \times 2\pi$					
h_1	8.9E − 6	—	—		
h_2	9.49E − 5	10.7	10.3		
h_3	1.62E − 3	17.1	16.2		
h_4	1.83E − 2	11.3	16.0		
h_5	7.7E − 2	4.	16.0		

[a] Adams ODE integrator used with ϵ = error tolerance.

The system of Eqs. (17) has the following "energy" invariant

$$\int_0^{2\pi} [\alpha(u(x,t)^2 + h(x,t)^2) + (2 + \alpha^2 u(u,t)^2)\, h(x,t)]\, dx = E(t) = E.$$

This integral is approximated by the trapezoid rule. Because of the periodic boundary conditions the trapezoid rule shows the same error as an approximation of the solution by a finite Fourier series taken over the mesh points [11]. Therefore use of the trapezoid rule should introduce less error than the difference approximation. We use the method of lines with fourth-order spatial differences to compute an approximate solution $U(x, h)$ (we drop t since all the results are at $t = 2\pi$). If we assume an asymptotic error expansion and drop the higher-order terms we have

$$(U(x, h_{\nu+1}) - U(x, h_0))/(U(x, h_\nu) - U(x, h_0)) = ((h_{\nu+1}/h_0)^4 - 1)/((h_\nu/h_0)^4 - 1).$$

A comparison of the two sides of this equation is given in Table VI for the indicated mesh points x and the mesh spacing h_ν. Agreement seems to be good only for the last three values of h. The computation is too expensive to permit smaller values of h. The error is much smaller at $x = 0.3 \times 2\pi$ for some reason.

Extrapolation to the limit using

$$U(x, 0) \cong U(x, h) - (U(u, h_1) - U(x, h_0))/((h_1/h_0)^4 - 1)$$

yields an error around 10^{-6} for $h = 1/250$ at $x = 0.4 \times 2\pi$. We have $E(h, 0) = \alpha = 0.05$, and thus we might expect a relative error in $E(h, 2\pi)$ around 2×10^{-5}. The actual error shown in Table VI is much larger and $E(h, 2\pi)$ does not appear to converge to zero with h. We have no explanation for this. The scheme does seem to converge pointwise. It is natural to suspect a programming error. However, we have run the integration of (17) on two independent programs. The program has been checked out on simple problems such as $u_t = u_x + x^3 - 3x^2(1 + t)$ with the solution $u(x, t) = (1 + t)x^3$ for which the solution of the difference equation is exact (the code allows nonperiodic boundary conditions although we have not discussed that here). We checked the trapezoid rule on the integral of $\exp(\sin(2\pi x))$ over one period. We perturbed the calculation to check its sensitivity to rounding error. But we have no explanation for the failure of this energy check.

In Table VII we show a comparison of the three ODE methods for the system (17). The error is estimated by assuming that the solution at $h = 1/140$ is exact. The results for the leapfrog show some error cancellation at $J = 10$. The Adams method requires fewer functional evaluations than the Runge–Kutta, but more CPU time. The right side of the equation is apparently too simple to cover the overhead in the Adams scheme. We made no effort to program this efficiently, however the Adams and Runge–Kutta–Fehlberg codes should suffer equally from any inefficiency in our program. The leapfrog scheme appears to be superior for this problem even if it is run at $\lambda = 0.4$ at $J = 10$ and $\lambda = 0.2$ at $J = 20$.

TABLE VII

Estimated Error for Shallow Water Equations at $t = 1.0$ with $\alpha = 0.05$[a]

	Error at $x = 0.3 \times 2\pi$	Error at $x = 0.4 \times 2\pi$	Energy check	Number of Eval	CPU time
$\epsilon = 1.E-2$	1.50E − 1	−1.95E − 1	1.12E − 1	34	0.55
$\epsilon = 1.E-3$	1.43E − 1	−1.50E − 1	6.30E − 2	56	0.97
$\epsilon = 1.E-4$	1.41E − 1	−1.49E − 1	6.16E − 2	67	1.35
	$J = 10$, Adams method				
$\epsilon = 1.E-2$	4.13E − 2	3.26E − 3	2.16E − 2	61	1.75
$\epsilon = 5.E-3$	1.56E − 2	1.87E − 2	1.27E − 2	69	2.03
$\epsilon = 5.E-4$	−3.00E − 3	2.59E − 2	8.02E − 3	82	2.62
$\epsilon = 5.E-5$	3.1E − 5	2.25E − 2	7.63E − 3	101	3.23
	$J = 20$, Adams method				
$\epsilon = 1.E-2$	1.41E − 1	−1.52E − 1	6.50E − 2	66	0.66
$\epsilon = 1.E-3$	1.40E − 1	−1.49E − 1	6.16E − 2	120	1.27
$\epsilon = 1.E-4$	1.40E − 1	−1.49E − 1	6.14E − 2	204	2.15
	$J = 10$, Runge–Kutta–Fehlberg				
$\epsilon = 1.E-2$	6.54E − 3	2.06E − 2	1.00E − 2	78	1.43
$\epsilon = 5.E-3$	2.83E − 3	2.19E − 2	8.76E − E	96	1.80
$\epsilon = 5.E-4$	4.82E − 4	2.23E − 2	7.71E − 3	174	3.27
$\epsilon = 5.E-5$	4.01E − 4	2.22E − 2	7.64E − 3	306	5.66
	$J = 20$, Runge–Kutta–Fehlberg				
$\lambda = 0.6$	4.39E − 2	1.64E − 1	3.43E − 2	21	0.12
$\lambda = 0.4$	1.42E − 1	−2.10E − 2	5.22E − 2	29	0.17
$\lambda = 0.2$	1.44E − 1	−1.19E − 1	5.96E − 2	54	0.36
$\lambda = 0.1$	1.42E − 1	−1.42E − 1	6.10E − 2	104	0.72
	$J = 10$, Leapfrog				
$\lambda = 0.6$	3.68E − 2	3.69E − 2	−1.25E − 2	38	0.42
$\lambda = 0.4$	−2.91E − 2	4.58E − 2	−6.44E − 4	54	0.63
$\lambda = 0.2$	−7.33E − 3	2.94E − 2	5.71E − 3	104	1.38
$\lambda = 0.1$	−1.53E − 3	2.41E − 2	7.17E − 3	204	2.59
	$J = 20$, Leapfrog				

[a] Here $\lambda = \Delta t/\Delta x$ and ϵ is the convergence tolerance. Estimated error $= U(x, h_0) - U(x, h_1)$, where $h_1 = 1/140$ and $h_0 = 1/10$ or $1/20$ (see Table VI). The CPU time is seconds on a CDC 6400.

Equations (17) were also run with different initial conditions intended to include higher frequencies in the solution. These are

$$u(x, 0) = h(x, 0) = \exp(\sin(2\pi(x - t))).$$

The solution for this case is highly distorted at $t = 1.$, however it appears to be continuous. The results are shown in Table VIII. The leapfrog appears to be superior in this case.

TABLE VIII

Estimated Error for Shallow Water Equations at $t = 1.0$ with $\alpha = 0.05$ and $4(x, 0) = h(x, 0) = \exp(\sin(2\pi(x - t)))$[a]

	Error at $x = 0.4 \times 2\pi$	Error at $x = 0.5 \times 2\pi$	Energy check	Number of Eval	*CPU* time
$\epsilon = 1.E - 2$	0.321	−0.362	6.2E − 4	67	1.9
$\epsilon = 5.E - 3$	0.214	−0.322	5.6E − 4	84	2.5
$\epsilon = 5.E - 4$	0.204	−0.314	3.4E − 4	105	3.2
		$J = 20$, Adams method			
$\epsilon = 1.E - 2$	0.335	−0.315	3.3E − 4	150	2.8
$\epsilon = 5.E - 3$	0.202	−0.314	3.1E − 4	186	3.3
$\epsilon = 5.E - 4$	0.201	−0.313	2.9E − 4	318	5.8
		$J = 20$, Runge–Kutta–Fehlberg			
$\lambda = 0.6$	0.013	−0.059	1.9E − 4	54	0.6
$\lambda = 0.4$	0.126	−0.187	2.1E − 4	71	0.8
$\lambda = 0.2$	0.211	−0.261	2.6E − 4	104	1.2
$\lambda = 0.1$	0.196	−0.301	2.8E − 4	204	2.4
		$J = 20$, Leapfrog			

[a] Estimated error $= u(x, h_0) - u(x, h_2)$ where $h_2 = 1/80$. The CPU time is seconds on a CDC 6400.

These results indicate that the fixed time-step leapfrog scheme may be best if the differential equation is simple. Otherwise the Adams method may be best, since it tends to minimize the number of functional evaluations of the time derivative. We are currently testing a program package for "hyperbolic" type systems in two dimensions coupled with a single elliptic equation. This package will allow a choice between the leapfrog and Runge–Kutta–Fehlberg schemes. We hope to add the Adams code to the package, but this requires some rewriting since we allow the data to be contained in LCM(ECS) on the CDC 7600 (6600). For this package the maximum order should probably be specified by the user (or by default). Also, the fact that we are in two space dimensions may allow the storage requirements to be reduced by keeping some temporary arrays only along a one-dimensional line instead of over the two-dimensional mesh.

REFERENCES

1. J. GARY, *J. Comput. Phys.* **16** (1974), 298–303.
2. J. BUTCHER, *Math. Comp.* **18** (1964), 50–64.
3. D. WATANABE AND J. FLOOD, *Math. Comp.* **28** (1974), 27–32.
4. B. SWARTZ AND B. WENDROFF, *SIAM J. Numer Anal.* **11** (1974), 979–993.
5. W. ENRIGHT, R. BEDET, I. FARKAS, AND T. HULL, Tech. Rep. 68, Dept. Comp. Science, Univ. Toronto, 1974.
6. L. SHAMPINE AND M. GORDON, "Computer Solution of Ordinary Differential Equations," W. H. Freeman, San Francisco, 1975.
7. L. SHAMPINE, H. WATTS, AND S. DAVENPORT, SAND–75–0182, Sandia Laboratories, Albuquerque, NM, 87115.
8. J. OLIGER, *Math. Comp.* **28** (1974), 15–25.
9. IMSL Library 3, IMSL, Houston, Texas, 77036, 1974.
10. B. SWARTZ, *in* Proc. of the Conference on Mathematical Aspects of Finite Elements in Partial Differential Equations, Math. Research Center, Univ. Wisconsin, 1974.
11. G. DAHLQUIST, A. BJORCK, AND N. ANDERSON, "Numerical Methods," Prentice Hall, Englewood Cliffs, N. J., 1974.
12. R. RICHTMYER AND K. MORTON, "Difference Methods for Initial Value Problems," Wiley, New York, 1967.
13. R. SINCOVEC AND N. MADSEN, *Trans. Math. Software* **1** (1975), 232–286.
14. S. MORRIS AND W. SCHIESSER, *in* Proc. AFIPS Fall Joint Comp. Conf., **33**, part 1, pp. 353–357, 1968.
15. J. GARY, *in* Proceedings of AICA Symposium on Computer Methods for Partial Differential Equations, Lehigh Univ., Bethlehem, PA, Juna 17–19, 1975.
16. J. GARY, *Ann. Inter. Calcul. Analog.* **3** (1975), 192–194.