FOURTH-ORDER TIME STEPPING FOR STIFF PDES*

ALY-KHAN KASSAM† AND LLOYD N. TREFETHEN‡

Abstract. A modification of the ETDRK4 (Exponential Time Differencing fourth-order Runge-Kutta) method for solving stiff nonlinear PDEs is presented that solves the problem of numerical instability in the scheme as proposed by Cox and Matthews and generalizes the method to non-diagonal operators. A comparison is made of the performance of this modified ETD scheme against the competing methods of linearly implicit differencing, integrating factors, time-splitting, and Fornberg and Driscoll's "sliders" for the KdV, Kuramoto-Sivashinsky, Burgers, and Allen-Cahn equations in one space dimension. It is found that the modified ETD scheme is the best by a clear margin. Implementation of the method is illustrated by short MATLAB programs for two of the equations.

Key words. ETD, exponential time-differencing, KdV, Kuramoto-Sivashinsky, Burgers, Allen-Cahn, linearly implicit, split step, integrating factor

AMS subject classifications.

1. Introduction. Many time-dependent partial differential equations (PDEs) combine low-order nonlinear terms with higher-order linear terms. Examples include the Allen-Cahn, Burgers, Cahn-Hilliard, Fisher-KPP, Fitzhugh-Naguno, Gray-Scott, Hodgkin-Huxley, Kuramoto-Sivashinsky, Navier-Stokes, nonlinear Schrödinger, and Swift-Hohenberg equations. To obtain accurate numerical solutions of such problems, it is desirable to use high-order approximations in space and time. Yet because of the difficulties introduced by the combination of nonlinearity and stiffness, most computations heretofore have been limited to second order in time.

Our subject in this paper is fourth order time-differencing. We shall write the PDE in the form

$$u_t = \mathcal{L}u + \mathcal{N}(u,t), \tag{1.1}$$

where $\mathcal{L}$ and $\mathcal{N}$ are linear and nonlinear operators, respectively. Once we discretise the spatial part of the PDE we get a system of ODEs,

$$u_t = \mathbf{L}u + \mathbf{N}(u,t). \tag{1.2}$$

There seem to be five principal competing methods for solving problems of this kind, which we will abbreviate by LI, SS, IF, SL, and ETD.

LI = Linearly Implicit. These are a well studied family of schemes that have an established history in the solution of stiff PDE. Early work looking at some stability issues dates to the beginning of the 1980's [44]. Schemes have been proposed for specific examples, e.g. the KdV equation [10] and the Navier-Stokes equations [9, 22, 23], as well as certain classes of problems, for example reaction-diffusion problems [36] and atmospheric modelling problems [45]. An overview of the stability properties and derivations of linearly-implicit schemes can be found in [2].

Linearly-implicit schemes consist of using an explicit multi-step formula, for example the fourth order Adams-Bashforth formula, to advance the nonlinear part of

*This work was supported by the Engineering and Physical Sciences Research Council (UK) and by MathWorks, Inc.

†Oxford University Computing Laboratory, Wolfson Bldg., Parks Road, Oxford OX1 3QD, UK (akk@comlab.ox.ac.uk)

‡same address (LNT@comlab.ox.ac.uk).

the problem and an implicit scheme, for example the fourth order backward differentiation formula, to advance the linear part. Other kinds of formulations also exist; for developments based around Runge-Kutta rather than Adams-Bashforth formulae, for example, see [37].

SS = Split Step. The idea of split step methods seems to have originated with Bagrinovskii and Godunov in the late 1950's [3] and to have been independently developed by Strang for the construction of finite difference schemes [40] (the simplest of these is often called 'Strang Splitting'). The idea has been widely used in modelling Hamiltonian dynamics, with the Hamiltonian of a system split into its potential and kinetic energy parts. Some early work on this was done by Ruth [35]. Yoshida [46] developed a technique to produce split-step methods of arbitrary even order. McLachlan and Atela [28] studied the accuracy of such schemes and McLachlan [29] made some further comparisons of different symplectic and non-symplectic schemes. Overviews of these methods can be found in Sanz-Serna and Calvo [38] and Boyd [5], and a recent discussion of the relative merits of operator splitting in general can be found in a paper by Schatzman [39].

In essence, with the split step method, we want to write the solution as a composition of linear and nonlinear steps

$$u(t) \approx \exp(c_1 t\mathbf{L}) F(d_1 t\mathbf{N}) \exp(c_2 t\mathbf{L}) F(d_2 t\mathbf{N}) \cdots \exp(c_k t\mathbf{L}) F(d_k t\mathbf{N}) u(0), \tag{1.3}$$

where c_i and d_i are real numbers and represent *fractional* time steps (though we use product notation, the nonlinear substeps are nonlinear). Generating split step methods becomes a process of generating the appropriate sets of real numbers, $\{c_i\}$ and $\{d_i\}$, such that this product matches the exact evolution operator to high order.

IF = Integrating factor. Techniques that multiply both sides of a differential equation by some integrating factor and then make a relevant change of variable are well known in the theory of ODEs (see for example [26]). A similar method has been developed for the study of PDEs. The idea is to make a change of variable that allows us to solve for the linear part exactly, and then use a numerical scheme of our choosing to solve the transformed, nonlinear equation. This technique has been used for PDEs by Milewski and Tabak [31], Fornberg and Driscoll [14], Trefethen [43], Boyd [5] and Cox and Matthews [11]. Starting with our generic discretised PDE, we define

$$v = e^{-\mathbf{L}t} u. \tag{1.4}$$

The term $e^{-\mathbf{L}t}$ is known as the *integrating factor*. In many applications we can work in Fourier space and render $\mathbf{L}$ diagonal, so that scalars rather than matrices are involved. Differentiating (1.4) gives

$$v_t = -e^{-\mathbf{L}t}\mathbf{L}u + e^{-\mathbf{L}t} u_t. \tag{1.5}$$

Now, multiplying (1.2) by the integrating factor gives

$$\underbrace{e^{-\mathbf{L}t} u_t - e^{-\mathbf{L}t}\mathbf{L}u}_{v_t} = e^{-\mathbf{L}t}\mathbf{N}(u), \tag{1.6}$$

that is,

$$v_t = e^{-\mathbf{L}t}\mathbf{N}(e^{\mathbf{L}t} v). \tag{1.7}$$

This has the effect of ameliorating the stiff linear part of the PDE, and we can use a timestepping method of our choice (for example a fourth order Runge-Kutta formula) to advance the transformed equation.

SL = Sliders. In a recent paper [14], Fornberg and Driscoll describe a clever extension of the linearly-implicit concept described above. In addition to splitting the problem into a linear and a nonlinear part, they also split the linear part (after transformation to Fourier space) into three regions: low, medium and high Fourier wavenumbers. The slider method involves using a different numerical scheme in each region. The advantage of this method is that one can combine high order methods for the low wave numbers with high-stability methods for the higher wave numbers. We can summarise one version of this method with the following table:

Low $\|k\|$	Medium $\|k\|$	High $\|k\|$
AB4/AB4	AB4/AM6	AB4/AM2*

Here k is the wavenumber, AB4 denotes the fourth order Adams-Bashforth formula, AM6 denotes the sixth order Adams-Moulton formula, and AM2* denotes a modified second order Adams-Moulton formula specified by

$$u^{n+1} = u^n + \frac{h}{2}\left(\frac{3}{2}\mathbf{L}u^{n+1} + \frac{1}{2}\mathbf{L}u^{n-1}\right), \tag{1.8}$$

where h is the timestep.

ETD = Exponential Time Differencing. This method is the main focus of this paper, and we will describe it in Section 2.

***

One might imagine that extensive comparisons would have been carried out of the behavior of these methods for various PDEs such as those listed in the first paragraph, but this is not so. One reason is that SL and ETD are quite new; but even the other three methods have not been compared as systematically as one might expect.

Our aim in beginning this project was to make such a comparison. However, we soon found that further development of the ETD schemes was first needed. As partly recognized by their originators Cox and Matthews, these methods as originally proposed encounter certain problems associated with eigenvalues equal to or close to zero, espcially when the matrix $\mathbf{L}$ is not diagonal. If these problems are not addressed, ETD schemes prove unsuccessful for some PDE applications.

In Section 2 we propose a modification of the ETD schemes that solves these numerical problems. The key idea is to make use of complex analysis and evaluate certain coefficient matrices or scalars via contour integrals in the complex plane. Other modifications would very possibly also achieve the same purpose, but so far as we know, this is the first fully practical ETD method for general use.

In Section 3 we summarize the results of experimental comparison of the five fourth-order methods listed above for four PDEs: the Burgers, Korteweg-de Vries, Allen-Cahn, and Kuramoto-Sivashinsky equations. We find that the ETD scheme outperforms the others. We believe it is the best method currently in existence for stiff PDEs, at least in one space dimension.

Sections 4 and 5 illustrate the methods in a little more detail for a diagonal example (Kuramoto-Sivashinsky) and a non-diagonal example (Allen-Cahn). They also provide brief MATLAB codes for use by our readers as templates.

2. A modified ETD scheme. Low order ETD schemes arose originally in the field of computational electrodynamics [42]. They have been independently derived at least three times [4, 11, 32], but the most comprehensive treatment, and in particular the fourth order ETDRK4 formula, is in the paper by Cox and Matthews [11], and it is from this paper that we take details of the scheme. Cox and Matthews argue that ETD schemes outperform LI schemes because they treat transient solutions better (where the linear term dominates), and outperform IF schemes because they treat non-transient solutions better (where the nonlinear term dominates).

Algebraically, ETD is similar to the IF method. The difference is that we do not make a complete change of variable. If we proceed as in the IF approach and apply the same integrating factor and then integrate over a *single* time step of length h, we get

$$u_{n+1} = e^{\mathbf{L}h} u_n + e^{\mathbf{L}h} \int_0^h e^{-\mathbf{L}\tau} \mathbf{N}(u(t_n+\tau), t_n+\tau) d\tau. \tag{2.1}$$

This equation is exact, so far, and the various order ETD schemes come from how one approximates the integral. In their paper Cox and Matthews first present a sequence of recurrence formulae that provide higher and higher order approximations of a multistep type. They propose a generating formula

$$u_{n+1} = e^{\mathbf{L}h} u_n + h \sum_{m=0}^{s-1} g_m \sum_{k=0}^{m} (-1)^k \binom{m}{k} F_{n-k}, \tag{2.2}$$

where s is the order of the scheme. The coefficients g_m are given by the recurrence relation

$$\begin{aligned} \mathbf{L}h g_0 &= e^{\mathbf{L}h} - \mathbf{I}, \\ \mathbf{L}h g_{m+1} + \mathbf{I} &= g_m + \frac{1}{2} g_{m-1} + \frac{1}{3} g_{m-2} + \ldots + \frac{g_0}{m+1}, \qquad m \geq 0. \end{aligned} \tag{2.3}$$

Cox and Matthews also derive a set of ETD methods based on Runge-Kutta time-stepping, which they call ETDRK schemes. In this report we consider only the fourth order scheme of this type, known as ETDRK4. According to Cox and Matthews, the derivation of this scheme is not at all obvious and required a symbolic manipulation system. Here are the formulae:

$$\boxed{\begin{aligned} a_n &= e^{\mathbf{L}h/2} u_n + \mathbf{L}^{-1}(e^{\mathbf{L}h/2} - \mathbf{I}) \mathbf{N}(u_n, t_n), \\ b_n &= e^{\mathbf{L}h/2} u_n + \mathbf{L}^{-1}(e^{\mathbf{L}h/2} - \mathbf{I}) \mathbf{N}(a_n, t_n + h/2), \\ c_n &= e^{\mathbf{L}h/2} a_n + \mathbf{L}^{-1}(e^{\mathbf{L}h/2} - \mathbf{I})(2\mathbf{N}(b_n, t_n + h/2) - \mathbf{N}(u_n, t_n)), \\ u_{n+1} &= e^{\mathbf{L}h} u_n + h^{-2}\mathbf{L}^{-3}\{\mathbf{N}(u_n, t_n)[-4 - \mathbf{L}h + e^{\mathbf{L}h}(4 - 3\mathbf{L}h + (\mathbf{L}h)^2)] \\ &+ 2(\mathbf{N}(a_n, t_n + h/2) + \mathbf{N}(b_n, t_n + h/2))[2 + \mathbf{L}h + e^{\mathbf{L}h}(-2 + \mathbf{L}h)] \\ &+ \mathbf{N}(c_n, t_n + h)[-4 - 3\mathbf{L}h - (\mathbf{L}h)^2 + e^{\mathbf{L}h}(4 - \mathbf{L}h)]\}. \end{aligned}}$$

Unfortunately, in this form, ETDRK4 (and indeed any of the ETD schemes of order higher than two) suffers from numerical instability. To understand why this is the case, consider the expression

$$g(z) = \frac{e^z - 1}{z}. \tag{2.4}$$

TABLE 2.1

Computation of $g(z)$ by two different methods. The formula (2.4) is inaccurate for small z, and the order-5 partial sum of the Taylor series is inaccurate for larger z. For $z = 1e - 2$, neither is fully accurate.

z	formula (2.4)	5-term Taylor	exact
1	1.71828182845905	1.71666666666667	1.71828182845905
1e-1	1.05170918075648	1.05170916666667	1.05170918075648
1e-2	1.00501670841679	1.00501670841667	1.00501670841681
1e-3	1.00050016670838	1.00050016670834	1.00050016670834
1e-4	1.00005000166714	1.00005000166671	1.00005000166671
1e-5	1.00000500000696	1.00000500001667	1.00000500001667
1e-6	1.00000049996218	1.00000050000017	1.00000050000017
1e-7	1.00000004943368	1.00000005000000	1.00000005000000
1e-8	0.99999999392253	1.00000000500000	1.00000000500000
1e-9	1.00000008274037	1.00000000050000	1.00000000050000
1e-10	1.00000008274037	1.00000000005000	1.00000000005000
1e-11	1.00000008274037	1.00000000000500	1.00000000000500
1e-12	1.00008890058234	1.00000000000050	1.00000000000050
1e-13	0.99920072216264	1.00000000000005	1.00000000000005

The accurate computation of this function is a well known problem in numerical analysis, and is discussed, for example, in the monograph by Higham [18]. The reason it is not straightforward is that for small z, (2.4) suffers from cancellation error. We illustrate this in Table 2.1 by comparing the true value of $g(z)$ to values computed directly from (2.4) and from five terms of a Taylor expansion. For small z, the direct formula is no good because of cancellation, but the Taylor polynomial is excellent. For large z, the direct formula is fine but the Taylor polynomial is inaccurate. For one value of z in the table, neither method gives full precision.

The connection between (2.4) and the scheme ETDRK4 becomes apparent when we consider the coefficients in square brackets in the update formula for ETDRK4:

$$\begin{aligned}
\alpha &= h^{-2}\mathbf{L}^{-3}[-4 - \mathbf{L}h + e^{\mathbf{L}h}(4 - 3\mathbf{L}h + (\mathbf{L}h)^2)], \\
\beta &= h^{-2}\mathbf{L}^{-3}[2 + \mathbf{L}h + e^{\mathbf{L}h}(-2 + \mathbf{L}h)], \\
\gamma &= h^{-2}\mathbf{L}^{-3}[-4 - 3\mathbf{L}h - (\mathbf{L}h)^2 + e^{\mathbf{L}h}(4 - \mathbf{L}h)].
\end{aligned} \tag{2.5}$$

These three coefficients are higher-order analogues of (2.4). The cancellation errors are even more pronounced in these higher-order variants, and all three suffer disastrous cancellation errors when $\mathbf{L}$ has eigenvalues close to zero. This vulnerability to cancellation errors in the higher-order ETD and ETDRK schemes can render them effectively useless for problems which have small eigenvalues in the discretised linear operator.

Cox and Matthews were aware of this problem, and in their paper they use a cutoff point for small eigenvalues. In particular, as they work mainly with linear operators that are diagonal, they use a Taylor series representation of the coefficients for diagonal elements below the cutoff, much as in Table 2.1. This approach, however, entails some problems. One is that as the table illustrates, one must be careful to ensure that there is no overlap region where neither formula is accurate. Another more serious problem is, how does the method generalize to nondiagonal problems, i.e., matrices rather than scalars? To handle such cases gracefully one would like a single formula that is simultaneously accurate for all values of z.

We have found that this can be achieved by making use of ideas of complex analysis. First let us describe the accuracy problem in general terms. We have a function $f(z)$ to evaluate that is analytic except for a removable singularity at $z = 0$. For values of z close to that singularity, though the formula given for $f(z)$ is mathematically exact, it is numerically inaccurate because of cancellation errors. We seek a uniform procedure to evaluate $f(z)$ accurately for values of z that may or may not lie in this difficult region.

The solution we have found is to evaluate $f(z)$ via an integral over a contour Γ in the complex plane that encloses z and is well separated from 0:

$$f(z) = \frac{1}{2\pi i}\int_\Gamma \frac{f(t)}{t-z}\,dt. \tag{2.6}$$

When z becomes a matrix $\mathbf{L}$ instead of a scalar, the same approach works, with the term $1/(t-z)$ becoming the resolvent matrix $(t\mathbf{I}-\mathbf{L})^{-1}$:

$$f(\mathbf{L}) = \frac{1}{2\pi i}\int_\Gamma f(t)(t\mathbf{I}-\mathbf{L})^{-1}\,dt. \tag{2.7}$$

Contour integrals of analytic functions (scalar or matrix) in the complex plane are easy to evaluate by means of the trapezoid rule, which converges exponentially [12, 17, 43]. In practice we take Γ to be a circle and usually find that 32-64 equally spaced points are sufficient. When $\mathbf{L}$ is real, we can exploit the symmetry and evaluate only in equally spaced points on the upper half of a circle centered on the real axis, then take the real part of the result.

The scalars or eigenvalues of $\mathbf{L}$ that arise in a discretized PDE typically lie in or near the left half of the complex plane and may cover a wide range, which grows with the spatial discretization parameter N. For diffusive problems they are close to the negative real axis (e.g., the Kuramoto-Sivashinsky equation), and for dispersive problems they are close to the imaginary axis (KdV). Suitable contours Γ may accordingly vary from problem to problem. Our experience shows that many different choices work well, so long as one is careful to ensure that the eigenvalues are indeed enclosed by Γ. For some diffusive problems, it might be advantageous to use a parabolic contour extending to $\mathrm{Re}z = -\infty$, taking advantage of exponential decay deep in the left half-plane, but we have not used this approach for the problems treated in this paper.

For diagonal problems, we have the additional flexibility of being able to choose a contour Γ that depends on z, such as a circle centred at z. In this special case, the contour integral reduces simply to a mean of $f(t)$ over Γ, which we approximate to full accuracy by a mean over equally spaced points along Γ (or again just the upper half of Γ, followed by taking the real part).

For the details of exactly how this contour integral approach can be implemented, see the MATLAB codes listed in Sections 4 and 5. There is considerable flexibility about this procedure, and we do not claim that our particular implementations are optimal, merely that they work and are easy to program. For non-diagonal problems, quite a bit of computation is involved—say, 32 matrix inverses—but as this is done just once before the time-stepping begins, the impact on the total computing time is small. It would be greater for some problems in multiple space dimensions, but in serious cases one could ameliorate the problem with a preliminary Schur factorisation to bring the matrix to triangular form.

To demonstrate the effectiveness of our stabilization method, Table 2.2 considers the computation of the coefficient $\gamma(z)$ described above (here $z = \mathbf{L}$ and $h = 1$) by use of the formula (2.5) and by a contour integral method. Here we follow the simplest approach in which the contour is a circle of radius 2 sampled at equally spaced points at angles $\pi/64, 3\pi/64, \ldots, 127\pi/64$. The integral becomes just a mean over these points, and because of the $\pm i$ symmetry, it is enough to compute the mean over the 32 points in the upper half-plane and then take the real part of the result. The table shows accuracy in all digits printed. (In fact for this example, the same is achieved with as few as 12 sample points in the upper half-plane.)

Table 2.2

Computation of $\gamma = \gamma(z)$ from the formula (2.5) (unstable) and from a mean over 32 points on a semicircle in the upper half-plane. The contour integral gives full accuracy for all these values of z.

z	formula (2.5)	contour integral	exact
10	−132.292794768840	−132.292794768840	−132.292794768840
1	0.15484548537714	0.15484548537714	0.15484548537714
1e-1	0.16658049502638	0.16658049502574	0.16658049502574
1e-2	0.16666583046998	0.16666583054959	0.16666583054959
1e-3	0.16666668045673	0.16666665833055	0.16666665833055
1e-4	0.16608936448392	0.16666666658333	0.16666666658333
1e-5	0	0.16666666666583	0.16666666666583
1e-6	−888.178419700125	0.16666666666666	0.16666666666666
1e-7	0	0.16666666666667	0.16666666666667
1e-8	0	0.16666666666667	0.16666666666667
1e-9	0	0.16666666666667	0.16666666666667

Another way to demonstrate the effectiveness of our method is to see it in action. Figures 3.1–3.4 of the next section demonstrate fast fourth-order convergence achieved for this method applied to four PDEs. An additional figure in that section, Figure 3.5, shows what happens to the ETDRK4 method for Burgers' equation if instead of our contour integrals it is implemented directly from the formulas as written.

3. Comparison of five numerical methods. We now present the results of our numerical experiments, adapted, in part, from p27.m and p34.m of [43]. In all of the following graphs, the error is divided by the ∞–norm of the actual solution, and thus is a relative rather than an absolute measure. In a similar vein, the timesteps displayed have all been divided by the overall simulation time scale and are thus scaled from 0 to 1. A relative timestep of 1e-2, for example, implies that the actual timestep was one hundredth of the total simulation time, or that the simulation required 100 steps.

We studied the following four problems. In each case we work with a fixed, spectrally accurate space discretisation and vary the timestep.

Korteweg-de Vries equation with periodic boundary conditions,

$$u_t = -uu_x - u_{xxx}, \qquad x \in [-\pi, \pi], \tag{3.1}$$

$$u(x, t = 0) = 3A^2\mathrm{sech}(\frac{A(x+2)}{2})^2 + 3B^2\mathrm{sech}(\frac{B(x+1)}{2})^2,$$

with $A = 25, B = 16$. The simulation runs to $t = 0.004$. Here and for the next PDE we use a 64-point Fourier spectral discretisation in x.

Burgers equation with periodic boundary conditions,

$$u_t = -(\frac{1}{2}u^2)_x + \epsilon u_{xx}, \qquad x \in [-\pi, \pi], \tag{3.2}$$

$$u(x, t = 0) = \exp(-10\sin^2(x/2)),$$

with $\epsilon = 0.03$ and the simulation running to $t = 3$.

Kuramoto-Sivashinsky equation with periodic boundary conditions,

$$u_t = -uu_x - u_{xx} - u_{xxxx}, \qquad x \in [0, 32\pi], \tag{3.3}$$

$$u(x, t = 0) = \cos(\frac{x}{16})(1 + \sin(\frac{x}{16})),$$

with the simulation running to $t = 65$. In this equation we use a 128-point Fourier spectral discretisation in x (see Section 4).

Allen-Cahn equation with constant Dirichlet boundary conditions,

$$u_t = \epsilon u_{xx} + u - u^3, \qquad x \in [-1, 1], \tag{3.4}$$

$$u(x, t = 0) = .53x + .47\sin(-1.5\pi x), \qquad u(-1, t) = -1, \quad u(1, t) = 1,$$

with $\epsilon = 0.001$ and the simulation running to $t = 3$. To impose the boundary conditions we define $u = w + x$ and work with homogeneous boundary conditions in the w variable; the discretisation is by a 20-point Chebyshev spectral method (see Section 5).

We emphasize that the first three problems, because of the periodic boundary conditions, can be reduced to diagonal form by Fourier transformation, whereas the fourth cannot be reduced this way and thus forces us to work with matrices rather than scalars.

Our results are summarised in Figures 3.1–3.4. The upper half of each figure compares accuracy against step size and thus should be reasonably independent of machine and implementation. The accuracy is measured against results as $h \to 0$ for the given spatial discretisation; there may be small spatial errors that are not our concern here. Ultimately, of course, it is computer time that matters, and this is what is displayed in the lower half of each figure, based on our MATLAB implementations on an 800 MHz Pentium 3 machine. Other implementations and other machines would give somewhat different results.

Before considering the differences among methods revealed in Figures 3.1–3.4, let us first highlight the most general point: our computations show that it is entirely practical to solve these difficult nonlinear partial differential equations to high accuracy by fourth-order time-stepping. Most simulations in the past have been of lower order in time, typically second-order, but we believe that for most purposes, a fourth-order method is superior.

KdV equation

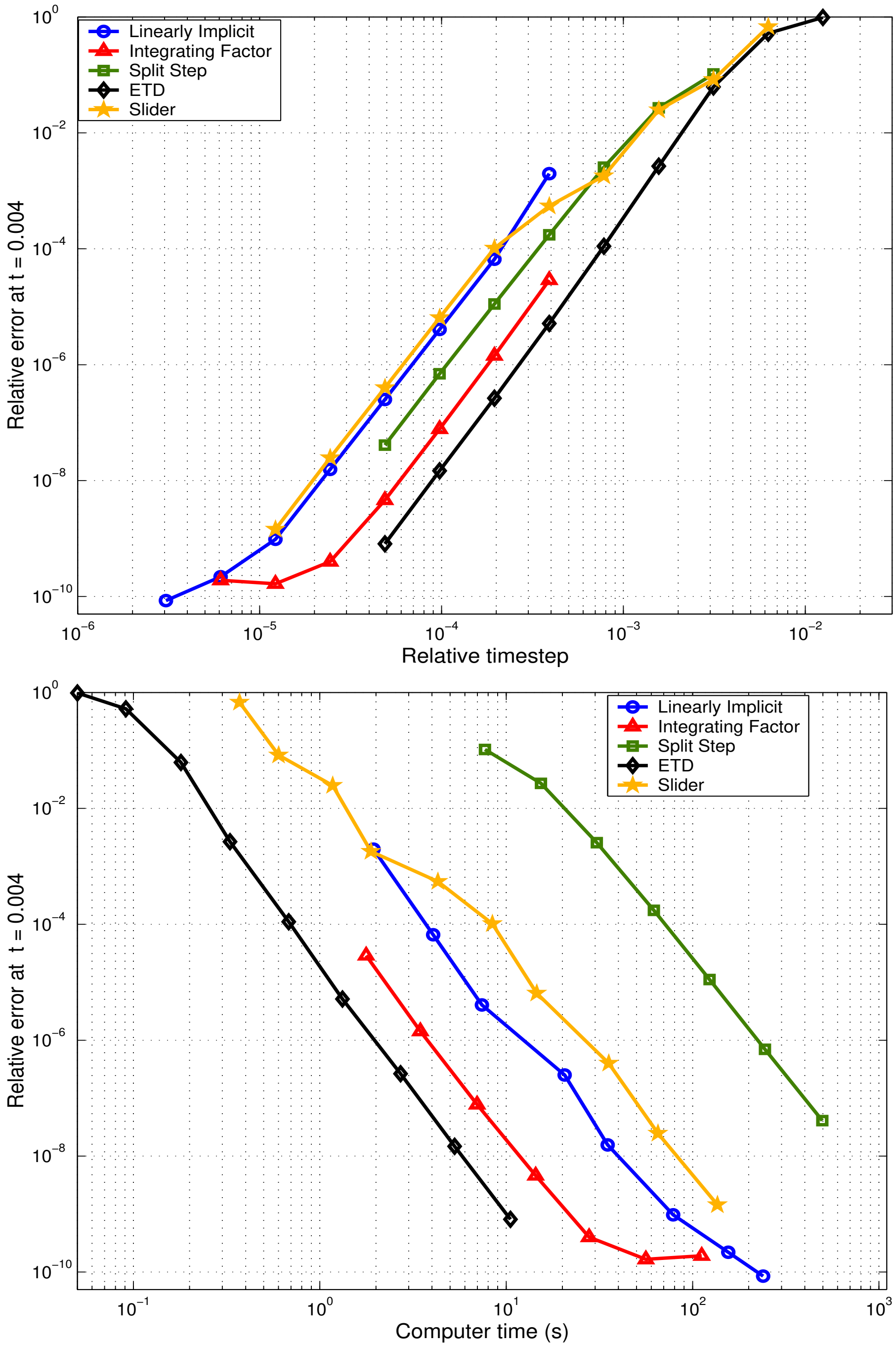

FIG. 3.1. *Accuracy vs. time step and computer time for five schemes for KdV equation. The ETD scheme performs best. Note how the slider scheme "slides" from one curve to another and then to another as the time step, h, is reduced.*

Burgers equation

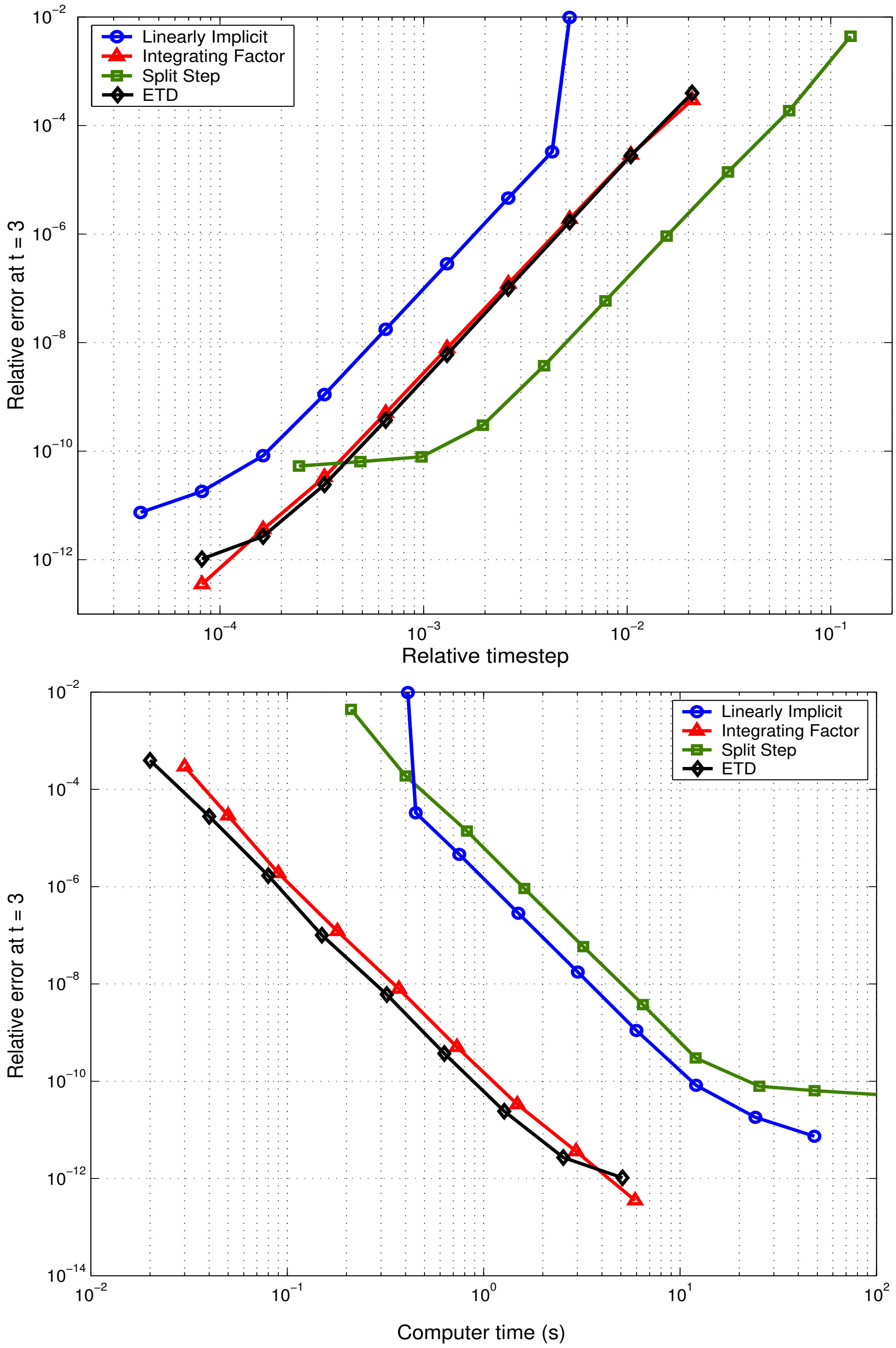

FIG. 3.2. *Accuracy vs. time step and computer time for four schemes for Burgers' equation. No slider scheme is considered here as appropriate formulae have not been published.*

Kuramoto-Sivashinsky equation

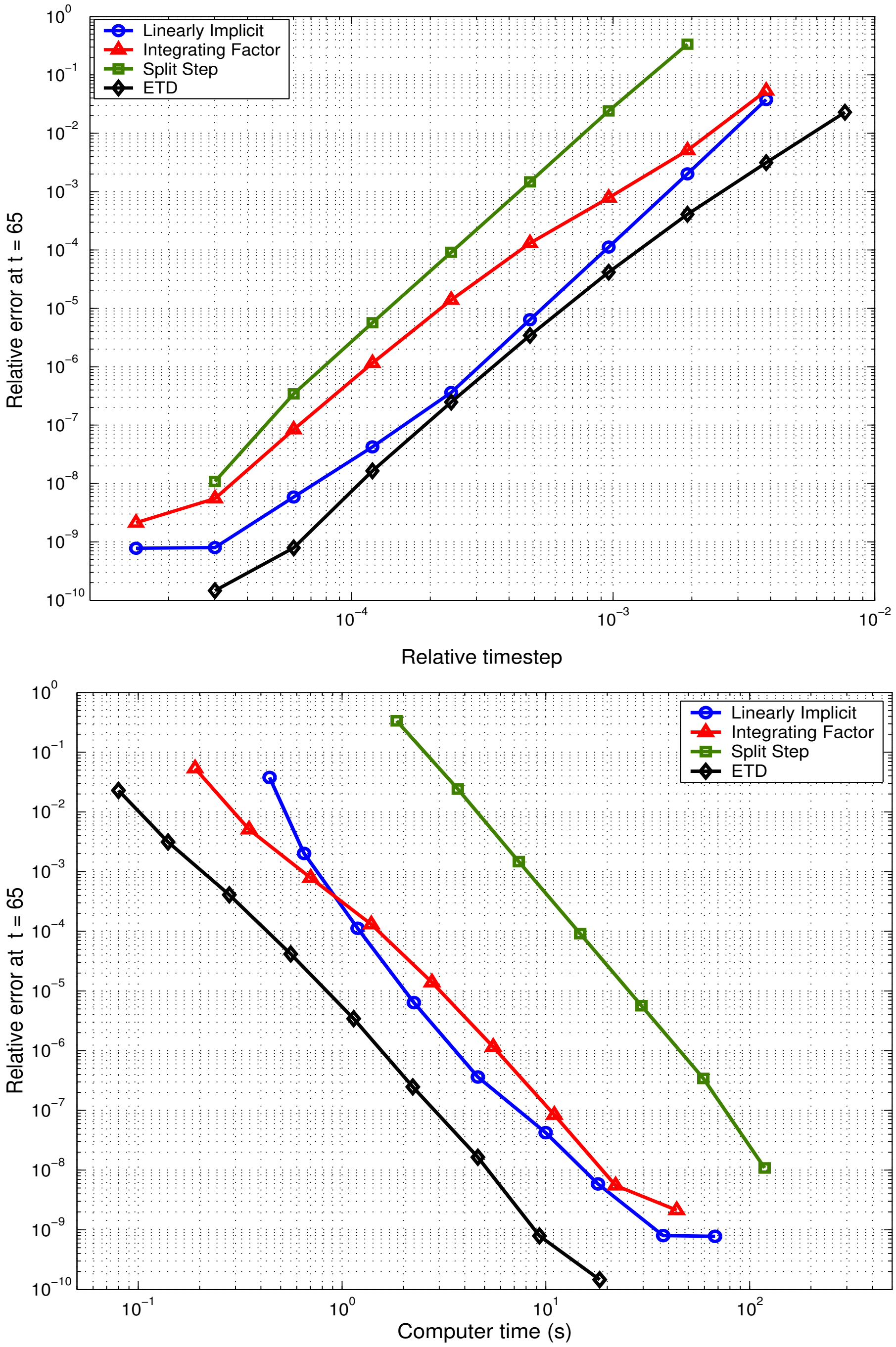

FIG. 3.3. *Results for the Kuramoto-Sivashinsky equation. Our* MATLAB *code is listed in Section* 4.

Allen-Cahn equation

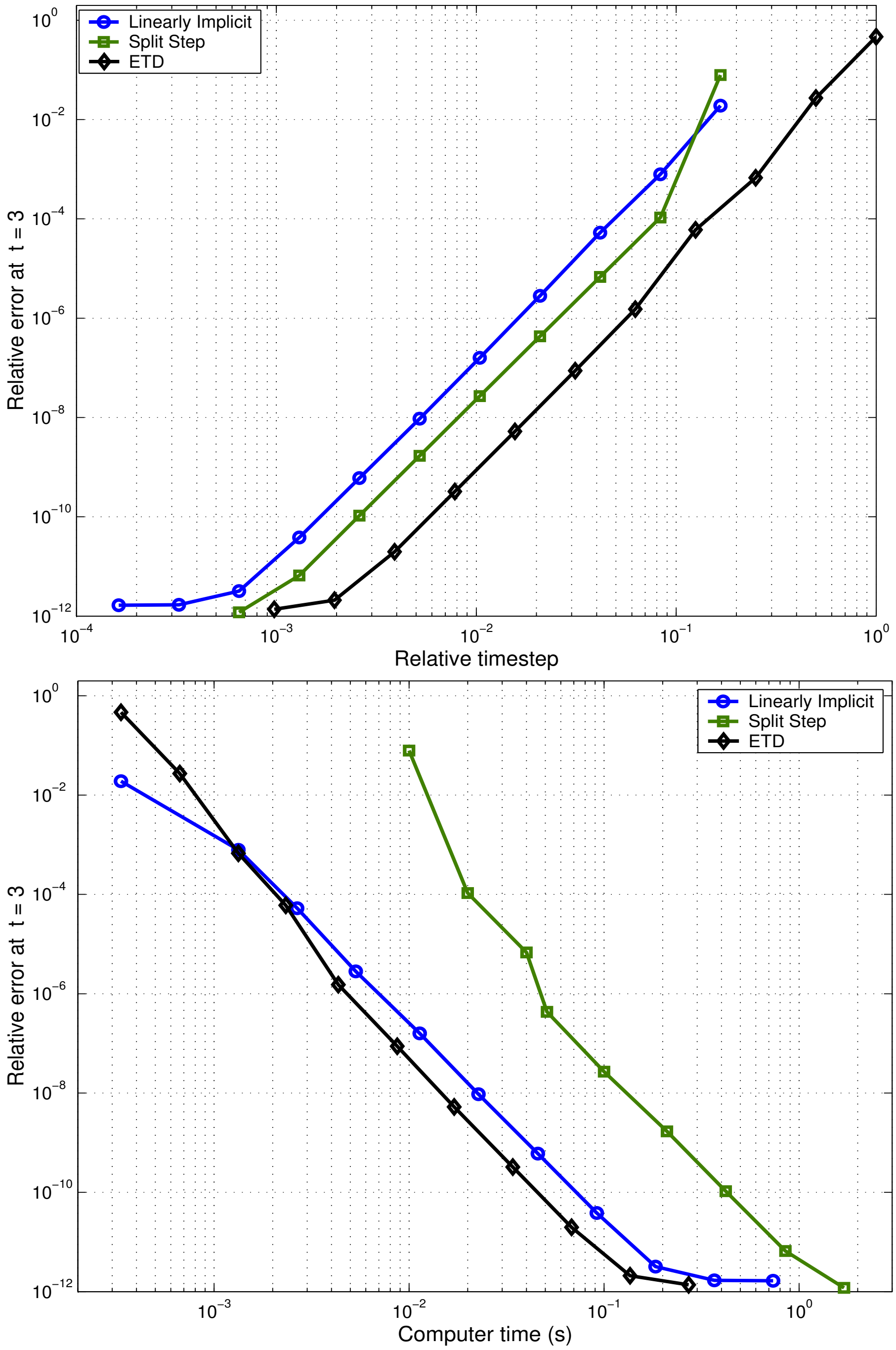

FIG. 3.4. *Results for the Allen-Cahn equation. This problem is non-diagonal and more challenging than the others. Our* MATLAB *code is listed in Section* 5.

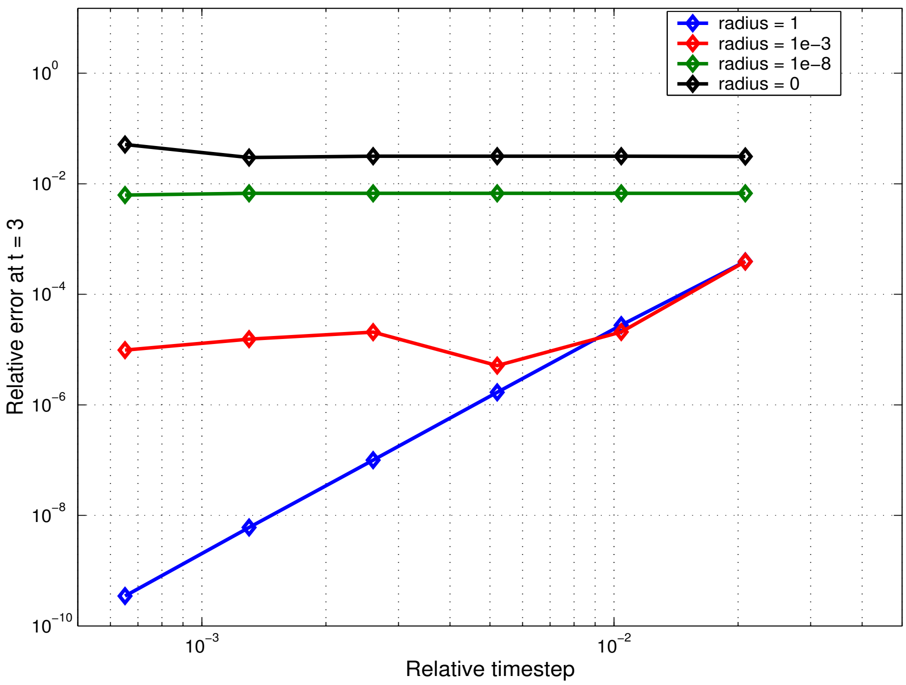

FIG. 3.5. *Loss of stability in the ETD scheme applied to Burgers' equation as the radius of the contour shrinks to zero. The contour of zero radius corresponds to an ETD calculation directly from the defining formula.*

Turning now to the differences between methods revealed in Figures 3.1–3.4, the first thing we note is that the differences are very considerable. The methods differ in efficiency by factors as great as 10 or higher.

A key feature to look for in the upper part of the figures is the relative positioning of the different methods. Schemes that are further to the right for a given accuracy take fewer steps to achieve that accuracy. It is possible that each step is more costly, however, so just because a scheme achieves a good accuracy in few steps does not mean that it is the most efficient. The lower part of the figures give insight into these computation times. The split step method illustrates this point in Figure 3.2: it has small errors for each given time step size, but is inefficient nonetheless. We can make a few comments on each of the methods investigated:

Exponential time differencing is the best in all cases. Overall it appears that for this kind of problem, the ETDRK schemes are the most stable and the most efficient.

The **linearly implicit** scheme used in this study does not perform very well. This is striking, as these are probably the most widely used of all the schemes.

The **split step** method also performed poorly. The main problem with this scheme is the long computation time caused by the large number of Runge-Kutta evaluations at each time-step for the nonlinear term. This becomes a particular problem when we use a scheme which is of higher than second order. For second-order calculations, split step schemes are certainly competitive.

The **slider** method does well in terms of widening the stability region of the standard linearly-implicit schemes. Interestingly, for small timesteps it approaches the fourth order linearly implicit scheme. It is not the quickest method, however. It also suffers from two other problems: it is complicated to extend to multiple dimensions, and, in its current form, it only applies to purely dispersive problems. Further developments are being planned for this scheme, however, including using Runge-Kutta time-stepping.

Finally, the **integrating factor** scheme performs well for problems in diagonal form. However we did not succeed in getting it to work for the non-diagonal Allen-Cahn equation.

An important consideration is, how easily do these schemes generalise to higher spatial dimensions? With the exception of the slider method, they all generalise in a straightforward manner. Formally, almost nothing changes other than the fact that we must work with tensor product matrices. There are a few problems that arise from this fact, though.

The ETD, IF and SS schemes all need to calculate and store a matrix exponential. Even if the original matrix is sparse, this is not an insignificant amount of work, and the matrix exponential will not itself be sparse, which can be a significant amount of storage. It is true that these can be calculated in advance, and for one dimensional-problems the calculation is not very expensive. As the dimension of the problem increases, however, the cost goes up. Our difficulty with the IF method for non-diagonal problems was that numerical instability meant that we were unable to calculate the appropriate exponential at all.

The nature of these matrices depends crucially on whether the problem is periodic. Each periodic dimension corresponds to a diagonalisation; if all dimensions are periodic, we have only scalars to deal with, and if only one dimension is non-periodic we have a collection of small blocks.

Overall it appears that, as a rule, the ETD scheme requires the fewest steps to achieve a given accuracy. It is also the fastest method in computation time.

We conclude this section with an illustration of how crucial our complex contour integrals, or other stabilisation devices, are to the success of ETD schemes. Figure 3.5 shows what happens if instead of a contour of radius 1 we shrink the radius to 10^{-3}, 10^{-8}, or 0. The accuracy is quickly lost. The case of radius 0 corresponds to an ETD scheme implemented directly from the formula without any stabilisation device beyond the use of L'Hôpital's rule to eliminate the removable singularity at $z = 0$.

4. A diagonal example: Kuramoto-Sivashinsky. We now give a little more detail about the Kuramoto-Sivashinsky equation, which dates to the mid–1970's and has been used in the study of a variety of reaction-diffusion systems [27]. In one spatial dimension it can be written as

$$u_t = -uu_x - u_{xx} - u_{xxxx}, \qquad x \in [0, 32\pi]. \tag{4.1}$$

As it contains both second and fourth order derivatives, the K-S equation produces highly complex behaviour. The second order term acts as an energy source and has a destabilising effect, and the nonlinear term transfers energy from low to high wavenumbers where the fourth order term has a stabilising effect. It is also very interesting from a dynamical systems point of view, as it is a PDE that can exhibit chaotic solutions [20, 33].

We use the initial condition

$$u(x, 0) = \cos(x/16)(1 + \sin(x/16)). \tag{4.2}$$

As the equation is periodic, we discretise the spatial part using a Fourier spectral method. Transforming to Fourier space gives,

$$\widehat{u}_t = -\frac{ik}{2}\widehat{u^2} + (k^2 - k^4)\widehat{u}, \tag{4.3}$$

or, in the standard form of (1.2),

$$(\mathbf{L}\hat{u})(k) = (k^2 - k^4)\hat{u}(k), \qquad \mathbf{N}(\hat{u}, t) = \mathbf{N}(\hat{u}) = -\frac{ik}{2}(F((F^{-1}(\hat{u}))^2)), \tag{4.4}$$

where F denotes the discrete Fourier transform.

We solve the problem entirely in Fourier space and use ETDRK4 time stepping scheme to solve to $t = 150$. Figure 4.1 shows the result, which took less than 1 second of computer time on our worksation. Despite the extraodinary sensitivity of the solution at later times to perturbations in the initial data (such perturbations are amplified by as much as 10^8 up to $t = 150$), we are confident that this image is correct to plotting accuracy. It would not have been practical to achieve this with a time-stepping scheme of lower order.

We produced Figure 4.1 with the MATLAB code listed in Figure 4.2.

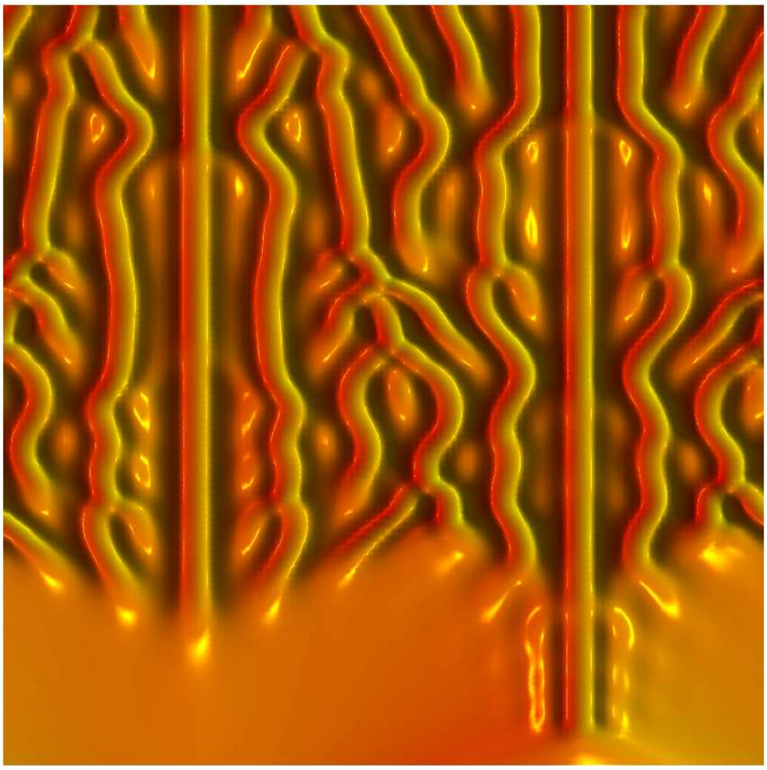

FIG. 4.1. *Time evolution for Kuramoto-Sivashinsky equation. Time runs from* 0 *at the bottom of the figure to* 150 *at the top. This is a PDE example of deterministic chaos* [1].

```
% kursiv.m - solution of Kuramoto-Sivashinsky equation by ETDRK4 scheme
%
%   u_t = -u*u_x - u_xx - u_xxxx, periodic BCs on [0,32*pi]
%   computation is based on v = fft(u), so linear term is diagonal
%   compare p27.m in Trefethen, "Spectral Methods in MATLAB", SIAM 2000
%   AK Kassam and LN Trefethen, July 2002

% Spatial grid and initial condition:
N = 128;
x = 32*pi*(1:N)'/N;
u = cos(x/16).*(1+sin(x/16));
v = fft(u);

% Precompute various ETDRK4 scalar quantities:
h = 1/4;                          % time step
k = [0:N/2-1 0 -N/2+1:-1]'/16;    % wave numbers
L = k.^2 - k.^4;                  % Fourier multipliers
E = exp(h*L); E2 = exp(h*L/2);
M = 16;                           % no. of points for complex means
r = exp(1i*pi*((1:M)-.5)/M);      % roots of unity
LR = h*L(:,ones(M,1)) + r(ones(N,1),:);
Q  = h*real(mean(           (exp(LR/2)-1)./LR              ,2));
f1 = h*real(mean(  (-4-LR+exp(LR).*(4-3*LR+LR.^2))./LR.^3  ,2));
f2 = h*real(mean(    (2+LR+exp(LR).*(-2+LR))./LR.^3    ,2));
f3 = h*real(mean(  (-4-3*LR-LR.^2+exp(LR).*(4-LR))./LR.^3  ,2));

% Main time-stepping loop:
uu = u; tt = 0;
tmax = 150; nmax = round(tmax/h); nplt = floor((tmax/100)/h);
g = -0.5i*k;
for n = 1:nmax
    t = n*h;
    Nv = g.*fft(real(ifft(v)).^2);
    a = E2.*v + Q.*Nv;
    Na = g.*fft(real(ifft(a)).^2);
    b = E2.*v + Q.*Na;
    Nb = g.*fft(real(ifft(b)).^2);
    c = E2.*a + Q.*(2*Nb-Nv);
    Nc = g.*fft(real(ifft(c)).^2);
    v = E.*v + Nv.*f1 + 2*(Na+Nb).*f2 + Nc.*f3;
    if mod(n,nplt)==0
        u = real(ifft(v));
        uu = [uu,u]; tt = [tt,t];
    end
end

% Plot results:
surf(tt,x,uu), shading interp, lighting phong, axis tight
view([-90 90]), colormap(autumn); set(gca,'zlim',[-5 50])
light('color',[1 1 0],'position',[-1,2,2])
material([0.30 0.60 0.60 40.00 1.00]);
```

FIG. 4.2. MATLAB *code to solve the Kuramoto-Sivashinsky equation and produce Figure* 4.1. *Despite the extraordinary sensitivity of this equation to perturbations, this code computes correct results in less than* 1 *second on an* 800 *MHz Pentium.*

5. A non-diagonal example: Allen-Cahn. The Allen-Cahn equation is another well-known equation from the area of reaction-diffusion systems. It has stable equilibria at $u = \pm 1$ and an unstable equilibrium at $u = 0$. One of the interesting features of this equation is the phonomenon of *metastability*. Regions of the solution that are near ± 1 will be flat, and the interface between such areas can remain unchanged over a very long timescale before changing suddenly. The equation can be written as

$$u_t = \epsilon u_{xx} + u - u^3, \qquad x \in [-1, 1], \tag{5.1}$$

and following p34.m of [43], we used the initial condition

$$u(x, 0) = .53x + .47 \sin(-1.5\pi x), \qquad u(-1, t) = -1, \quad u(1, t) = 1. \tag{5.2}$$

We can rewrite this in our standard form (1.2), with

$$\mathbf{L} = \epsilon \mathbf{D}^2, \qquad \mathbf{N}(u, t) = u - u^3, \tag{5.3}$$

where $\mathbf{D}$ is the Chebyshev differentiation matrix [43]. $\mathbf{L}$ is now a full matrix. Again we use ETDRK4 for the time stepping and we solve up to $t = 70$ with $\epsilon = 0.01$. Figure 5.1 shows the result produced by the MATLAB code listed in Figure 5.2, which also runs in less than a second on our workstation. This code calls the function `cheb.m` from [43], available at `http://www.comlab.ox.ac.uk/work/nick.trefethen`. Figure 5.3 provides a listing.

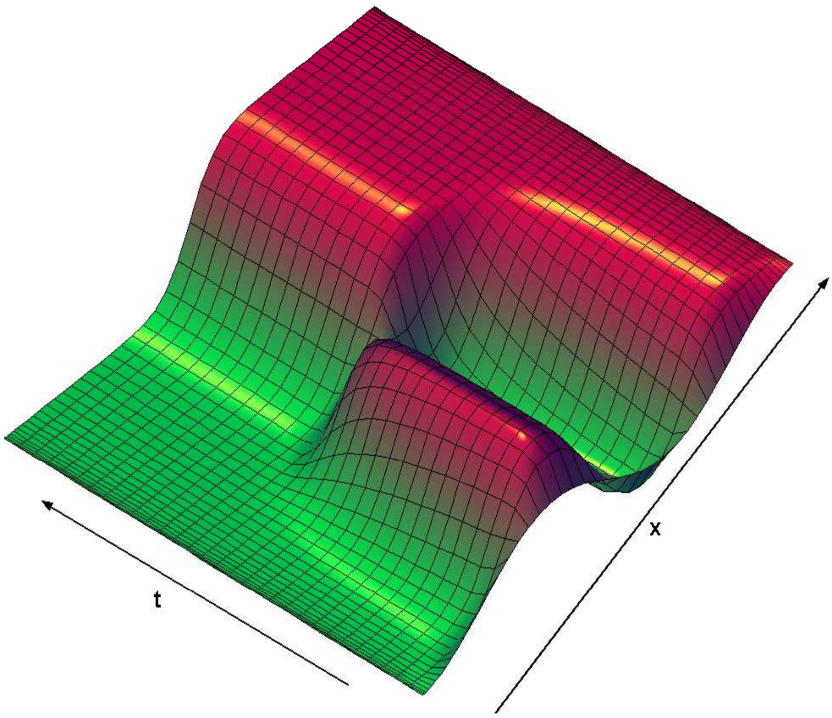

FIG. 5.1. *Time evolution for Allen-Cahn equation. The initial hump is metastable and disappears near $t = 45$. The x axis runs from $x = -1$ to $x = 1$ and the t-axis runs from $t = 0$ to $t = 70$.*

```
% allencahn.m - solution of Allen-Cahn equation by ETDRK4 scheme
%
%   u_t = 0.01*u_xx + u - u^3 on [-1,1], u(-1)=-1, u(1)=1
%   computation is based on Chebyshev points, so linear term is nondiagonal
%   compare p34.m in Trefethen, "Spectral Methods in MATLAB", SIAM 2000
%   AK Kassam and LN Trefethen, July 2002

% Spatial grid and initial condition:
N = 20;
[D,x] = cheb(N); x = x(2:N);          % spectral differentiation matrix
w = .53*x + .47*sin(-1.5*pi*x) - x;  % use w = u-x to make BCs homogeneous
u = [1;w+x;-1];

% Precompute various ETDRK4 matrix quantities:
h = 1/4;                              % time step
M = 32;                               % no. of points for resolvent integral
r = 15*exp(1i*pi*((1:M)-.5)/M);       % points along complex circle
L = D^2; L = .01*L(2:N,2:N);          % 2nd-order differentiation
A = h*L;
E = expm(A); E2 = expm(A/2);
I = eye(N-1); Z = zeros(N-1);
f1 = Z; f2 = Z; f3 = Z; Q = Z;
for j = 1:M
    z = r(j);
    zIA = inv(z*I-A);
    Q  = Q  + h*zIA*(exp(z/2)-1);
    f1 = f1 + h*zIA*(-4-z+exp(z)*(4-3*z+z^2))/z^2;
    f2 = f2 + h*zIA*(2+z+exp(z)*(z-2))/z^2;
    f3 = f3 + h*zIA*(-4-3*z-z^2+exp(z)*(4-z))/z^2;
end
f1 = real(f1/M); f2 = real(f2/M); f3 = real(f3/M); Q = real(Q/M);

% Main time-stepping loop:
uu = u; tt = 0;
tmax = 70; nmax = round(tmax/h); nplt = floor((tmax/70)/h);
for n = 1:nmax
    t = n*h;
    Nu = (w+x) - (w+x).^3;
    a = E2*w  + Q*Nu;
    Na = a + x - (a+x).^3;
    b = E2*w  + Q*Na;
    Nb = b + x - (b+x).^3;
    c = E2*a  + Q*(2*Nb-Nu);
    Nc = c + x - (c+x).^3;
    w = E*w + f1*Nu + 2*f2*(Na+Nb) + f3*Nc;
    if mod(n,nplt)==0
        u = [1;w+x;-1];
        uu = [uu,u]; tt = [tt,t];
    end
end

% Plot results:
surf([1;x;-1],tt,uu'), lighting phong, axis tight
view([-45 60]), colormap(cool), light('col',[1 1 0],'pos',[-10 0 10])
```

FIG. 5.2. MATLAB *code to solve the Allen-Cahn equation and produce Figure* 5.1. *Again, this code takes less than* 1 *second to run on an* 800 *MHz Pentium.*

```
% CHEB  compute D = differentiation matrix, x = Chebyshev grid

  function [D,x] = cheb(N)
  if N==0, D=0; x=1; return, end
  x = cos(pi*(0:N)/N)';
  c = [2; ones(N-1,1); 2].*(-1).^(0:N)';
  X = repmat(x,1,N+1);
  dX = X-X';
  D  = (c*(1./c)')./(dX+(eye(N+1)));      % off-diagonal entries
  D  = D - diag(sum(D'));                 % diagonal entries
```

FIG. 5.3. *cheb.m from [43]. This calculates the Chebyshev differentiation matrix, and is used by the code for the Allen-Cahn equation.*

Acknowledgments. It has been a pleasure to learn about high-order ETD schemes from one of their inventors, Paul Matthews. We are also grateful for useful discussions with Elaine Crooks of Oxford University and Álvaro Meseguer, formerly of Oxford and now at the Universitat Politecnica de Catalunya in Barcelona.

REFERENCES

[1] A. Aceves, H. Adachihara, C. Jones, J. C. Lerman, D. W. McLaughlin, J. V. Moloney and A. C. Newell, Chaos and coherent structures in partial differential equations, *Physica D* **18**, 85–112 (1986).

[2] U. M. Ascher, S. J. Ruuth, and B. T. R. Wetton, Implicit-explicit methods for time-dependent partial differential equations, *SIAM J. Numer. Anal.* **32**, 797–823 (1995).

[3] K. A. Bagrinovskii and S. K. Godunov, Difference schemes for multi-dimensional problems, *Dokl. Akad. Nauk. USSR* **115**, 431–433 (1957).

[4] G. Beylkin, J. M. Keiser, and L. Vozovoi, A new class of time discretization schemes for the solution of nonlinear PDEs, *J. Comp. Phys.* **147**, 362–387 (1998).

[5] J. P. Boyd, *Chebyshev and Fourier Spectral Methods* (Dover, 2001). Online edition at: http://www-personal.engin.umich.edu/∼jpboyd/.

[6] J. M. Burgers, A mathematical model illustrating the theory of turbulence, *Adv. Appl. Mech.* **1**, 171–199 (1948).

[7] G. D. Byrne and A. C. Hindmarsh, Stiff ODE solvers: A review of current and coming attractions, *J. Comp. Phys.* **70** (2), 1–62 (1987).

[8] M. P. Calvo, J. de Frutos, and J. Novo, Linearly implicit Runge-Kutta methods for advection-diffusion equations, *Appl. Numer. Math.* **37**, 535–549 (2001).

[9] C. Canuto, M. Y. Hussaini, A. Quarteroni and T. A. Zang, *Spectral Methods in Fluid Dynamics* (Springer-Verlag, Berlin, 1988).

[10] T. F. Chan and T. Kerkhoven, Fourier methods with extended stability intervals for the Korteweg-de Vries equation, *SIAM J. Numer. Anal.* **22**, 441–454 (1985).

[11] S. M. Cox and P. C. Matthews, Exponential time differencing for stiff systems, *J. Comp. Phys.* **176**, 430–455 (2002).

[12] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*, 2nd ed., (Academic Press, New York, 1984).

[13] B. Fornberg, *A Practical Guide to Pseudospectral Methods* (Cambridge University Press, Cambridge, UK, 1996).

[14] B. Fornberg and T. A. Driscoll, A fast spectral algorithm for nonlinear wave equations with linear dispersion, *J. Comp. Phys.* **155**, 456–467 (1999).

[15] E. Hairer, S. P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I* (Springer-Verlag, Berlin, 1991).

[16] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II* (Springer-Verlag, Berlin, 1996).

[17] P. Henrici, *Applied and Computational Complex Analysis, v. 3*, (Wiley, New York, 1986).

[18] N. J. Higham, *Accuracy and Stability of Numerical Algorithms* (SIAM, Philadelphia, 1996).

[19] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis* (Cambridge University Press, Cambridge, UK, 1991).

[20] J. M. Hyman and B. Nicolaenko, The Kuramoto-Sivashinsky equation: a bridge between PDE's and dynamical systems, *Physica D* **18**, 113–126 (1986).

[21] A. Iserles, *A First Course in the Numerical Analysis of Differential Equations* (Cambridge University Press, Cambridge, UK, 2000).

[22] G. E. Karniadakis, M. Israeli and S. A. Orszag, High order splitting methods for the incompressible Navier-Stokes equations, *J. Comp. Phys* **97**, 414–443 (1991).

[23] J. Kim and P. Moin, Applications of a fractional step method to incompressible Navier-Stokes equations, *J. Comp. Phys.* **59**, 308–323 (1985).

[24] O. M. Knio, H. N. Najim and P. S. Wyckoff, A semi-implicit numerical scheme for reacting flow, *J. Comp. Phys.* **154**, 428–467 (1999).

[25] D. Korteweg and G. de Vries, On the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves, *Phil. Mag.* (Ser. 5) **39** , 422–433 (1895).

[26] M. Krusemeyer, *Differential Equations* (Macmillan College Publishing Company, New York, 1994).

[27] Y. Kuramoto and T. Tsuzuki, Persistent propagation of concentration waves in dissipative media far from thermal equilibrium, *Prog. Theor. Phys.* **55**, 356–369 (1976).

[28] R. I. McLachlan and P. Atela, The accuracy of symplectic integrators, *Nonlinearity* **5**, 541–562 (1992).

[29] R. McLachlan, Symplectic integration of Hamiltonian wave equations, *Numer. Math.* **66**, 465–492 (1994).

[30] W. J. Merryfield and B. Shizgal, Properties of collocation third-derivative operators, *J. Comp. Phys.* **105**, 182–185 (1993).

[31] P. A. Milewski and E. G. Tabak, A pseudo-spectral procedure for the solution of nonlinear wave equations with examples from free-surface flows, *SIAM J. Sci. Comput.* **21**, 1102–1114 (1999).

[32] D. R. Mott, E. S. Oran and B. van Leer, A Quasi-steady state solver for the stiff ordinary differential equations of reaction kinetics, *J. Comp. Phys* **164**, 407–428 (2000).

[33] B. Nicolaenko, B. Scheurer, and T. Temam, Some global properties of the Kuramoto-Sivashinsky equation: nonlinear stability and attractors, *Physica D* **16**, 155–183 (1985).

[34] E. Ott, *Chaos in Dynamical Systems* (Cambridge University Press, Cambridge, 1993).

[35] R. D. Ruth, A canonical integration technique, *IEEE Trans. Nucl. Sci. NS* **30**, 2669–2671 (1983).

[36] S. J. Ruuth, Implicit-explicit methods for reaction-diffusion problems in pattern formation, *J. Math. Biol.* **34** (2), 148–176 (1995).

[37] S. J. Ruuth, Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations, *Appl. Numer. Math.* **25**, 151–167 (1997).

[38] J. M. Sanz-Serna and M. P. Calvo, *Numerical Hamiltonian Problems* (Chapman & Hall, London, 1994).

[39] M. Schatzman, Toward non-commutative numerical analysis: high order integration in time, *J. Sci. Comp.* **17** (1–4), 99–116.

[40] G. Strang, On the construction and comparison of difference schemes, *SIAM J. Numer. Anal.* **5** (3), 506–517.

[41] E. Tadmor, The exponential accuracy of Fourier and Chebyshev differencing methods, *SIAM J. Numer. Anal.* **23** (1), 1–10 (1986).

[42] A. Taflove, *Computational Electrodynamics: The Finite-Difference Time-Domain Method* (Artech House, Boston, 1995).

[43] L. N. Trefethen, *Spectral Methods in* MATLAB (SIAM, Philadelphia 2000).

[44] J. M. Varah, Stability restrictions on second order, three level finite difference schemes for parabolic equations, *SIAM J. Numer. Anal.* **17** (2), 300–309 (1980).

[45] J. G. Verwer, J. G. Blom and W. Hundsdorfer, An implicit-explicit approach for atmospheric transport-chemistry problems, *Appl. Numer. Math* **20**, 191–209 (1996).

[46] H. Yoshida, Construction of higher order symplectic integrators, *Phys. Lett. A* **150**, 262–268 (1990).