SIAM J. SCI. COMPUT.
Vol. 26, No. 4, pp. 1214–1233

FOURTH-ORDER TIME-STEPPING FOR STIFF PDEs*

ALY-KHAN KASSAM† AND LLOYD N. TREFETHEN†

Abstract. A modification of the 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 nondiagonal operators. A comparison is made of the performance of this modified exponential time-differencing (ETD) scheme against the competing methods of implicit-explicit 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. Implementation of the method is illustrated by short MATLAB programs for two of the equations. It is found that for these applications with fixed time steps, the modified ETD scheme is the best.

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

AMS subject classifications. Primary, 65M70; Secondary, 65L05, 65M20

DOI. 10.1137/S1064827502410633

1. Introduction.

Many time-dependent 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 (KS), 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 discretize 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 IMEX, SS, IF, SL, and ETD. Of course these are not the only schemes that are being used. Noteworthy schemes that we ignore are the exponential Runge–Kutta schemes [26] and deferred correction [37] or semi-implicit deferred correction [6, 45].

IMEX = Implicit-explicit. These are a well-studied family of schemes that have an established history in the solution of stiff PDEs. Early work looking at some stability issues dates to the beginning of the 1980s [61]. Schemes have been proposed for specific examples, e.g., the KdV equation [13] and the Navier–Stokes equations

*Received by the editors July 8, 2002; accepted for publication (in revised form) December 8, 2003; published electronically March 11, 2005. This work was supported by the Engineering and Physical Sciences Research Council (UK) and by MathWorks, Inc.
http://www.siam.org/journals/sisc/26-4/41063.html
†Oxford University Computing Laboratory, Wolfson Bldg., Parks Road, Oxford OX1 3QD, UK (LNT@comlab.ox.ac.uk).

[11, 32, 34], as well as certain classes of problems, for example, reaction-diffusion problems [51] and atmospheric modelling problems [62]. An overview of the stability properties and derivations of implicit-explicit schemes can be found in [2].

Implicit-explicit schemes consist of using an explicit multistep formula, for example, the second order Adams–Bashforth formula, to advance the nonlinear part of the problem and an implicit scheme, for example, the second order Adams–Moulton formula, to advance the linear part. Other kinds of formulations also exist; for developments based on Runge–Kutta rather than Adams–Bashforth formulae, for example, again see work by Ascher, Ruuth, and Spiteri [3], as well as very recent work by Calvo, Frutos, and Novo [10] and Kennedy and Carpenter [33]. In this report, we use a scheme known either as AB4BD4 (in [14]) or SBDF4 (in [2]), which consists of combining a fourth-order Adams–Bashforth and a fourth-order backward differentiation scheme. The formula for this scheme is

$$u_{n+1} = (25 - 12h\mathbf{L})^{-1}(48u_n - 36u_{n-1} + 16u_{n-2} - 3u_{n-3} + 48h\mathbf{N}_n - 72\mathbf{N}_{n-1} + 48\mathbf{N}_{n-2} - 12\mathbf{N}_{n-3}) \quad \text{(AB4BD4)}. \tag{1.3}$$

SS = Split step. The idea of split step methods seems to have originated with Bagrinovskii and Godunov in the late 1950s [4] and to have been independently developed by Strang for the construction of finite difference schemes [57] (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 [50]. Yoshida [63] developed a technique to produce split step methods of arbitrary even order. McLachlan and Atela [41] studied the accuracy of such schemes, and McLachlan [42] made some further comparisons of different symplectic and nonsymplectic schemes. Overviews of these methods can be found in Sanz-Serna and Calvo [53] and Boyd [7], and a recent discussion of the relative merits of operator splitting in general can be found in a paper by Schatzman [54].

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

$$u(t) \approx \exp(c_1 t\mathbf{L})F(d_1 t\mathbf{N})\exp(c_2 t\mathbf{L})F(d_2 t\mathbf{N})\cdots\exp(c_k t\mathbf{L})F(d_k t\mathbf{N})u(0), \tag{1.4}$$

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. The time-stepping for such a scheme can be either a multistep or a Runge–Kutta formula. We use a fourth-order Runge–Kutta formula for the time-stepping in this experiment.

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, [38]). 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 [44], Maday, Patera, and Rønquist [40], Smith and Waleffe [55, 56], Fornberg and Driscoll [20], Trefethen [60], Boyd [7], and Cox and Matthews [14]. Starting with our generic discretized PDE, we define

$$v = e^{-\mathbf{L}t}u. \tag{1.5}$$

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 (1.5) gives

$$v_t = -e^{-\mathbf{L}t}\mathbf{L}u + e^{-\mathbf{L}t}u_t. \tag{1.6}$$

Now, multiplying (1.2) by the integrating factor gives

$$\underbrace{e^{-\mathbf{L}t}u_t - e^{-\mathbf{L}t}\mathbf{L}u}_{v_t} = e^{-\mathbf{L}t}\mathbf{N}(u), \tag{1.7}$$

that is,

$$v_t = e^{-\mathbf{L}t}\mathbf{N}(e^{\mathbf{L}t}v). \tag{1.8}$$

This has the effect of ameliorating the stiff linear part of the PDE, and we can use a time-stepping method of our choice (for example, a fourth-order Runge–Kutta formula) to advance the transformed equation. In practice, one doesn't use the equation as it is written in (1.8), but rather replaces actual time, t, with the time step, Δt, and incrementally updates the formula from one time step to the next. This greatly improves the stability.

In both the split step method and the integrating factor method, we use a fourth-order Runge–Kutta method for the time-stepping. The fourth-order Runge–Kutta algorithm that we used to perform the time integration for this method was

$$\begin{aligned} a &= hf(v_n, t_n), \\ b &= hf(v_n + a/2, t_n + h/2), \\ c &= hf(v_n + b/2, t_n + h/2), \\ d &= hf(v_n + c, t_n + h), \\ v_{n+1} &= v_n + \frac{1}{6}(a + 2b + 2c + d) \quad \text{(Fourth-order RK)}, \end{aligned} \tag{1.9}$$

where h is the time step and f is the nonlinear functional on the right-hand side of (1.8). For the split step method, we simply replace f in (1.9) with F from (1.4).

SL = Sliders. In a recent paper [20], Fornberg and Driscoll describe a clever extension of the implicit-explicit 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 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 summarize 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

$$u^{n+1} = u^n + \frac{h}{2}\left(\frac{3}{2}\mathbf{L}u^{n+1} + \frac{1}{2}\mathbf{L}u^{n-1}\right), \tag{1.10}$$

where h is the time step.

Unfortunately, this scheme is stable only for purely dispersive equations. In order to generalize the concept, Driscoll has developed a very similar idea using Runge–Kutta time-stepping [17]. Again, the idea is to make use of different schemes for "fast" and "slow" modes. In this case, he uses the fourth-order Runge–Kutta formula to deal with the slow, nonlinear modes, and an implicit-explicit third-order Runge–Kutta method to advance the "fast" linear modes. This is the method that we explore in this paper.

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, especially 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, KdV, Allen–Cahn, and KS 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. In making such a bold statement, however, we should add the caveat that we are considering only fixed time steps. Our ETD methods do not extend cheaply to variable time-stepping; an IMEX scheme, for example, is a more natural candidate for such problems.

Sections 4 and 5 illustrate the methods in a little more detail for a diagonal example (KS) and a nondiagonal 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 [59]. They have been independently derived several times [5, 12, 14, 21, 46, 48]—indeed Iserles has pointed out to us that in the ODE context, related ideas go back as far as Filon in 1928 [18, 30]—but the most comprehensive treatment, and in particular the exponential time-differencing fourth-order Runge–Kutta (ETDRK4) formula, is in the paper by Cox and Matthews [14], and it is from this paper that we take details of the scheme. Cox and Matthews argue that ETD schemes outperform IMEX schemes because they treat transient solutions better (where the linear term dominates), and outperform IF schemes because they treat nontransient 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, 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

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

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

$$\begin{aligned} \mathbf{L}hg_0 &= e^{\mathbf{L}h} - \mathbf{I}, \\ \mathbf{L}hg_{m+1} + \mathbf{I} &= g_m + \frac{1}{2}g_{m-1} + \frac{1}{3}g_{m-2} + \cdots + \frac{g_0}{m+1}, \quad m \geq 0. \end{aligned} \tag{2.3}$$

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 requires a symbolic manipulation system. The Cox and Matthews ETDRK4 formulae are:

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

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

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

The accurate computation of this function is a well-known problem in numerical analysis and is discussed, for example, in the monograph by Higham [25], as well as the paper by Friesner et al. [21]. 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

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 $z = 1\mathrm{e}-2$, neither is fully accurate. All computations are done in IEEE double precision artithmetic.

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

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 discretized 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 $(t\mathbf{I} - \mathbf{L})^{-1}$:

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

Here Γ can be any contour that encloses the eigenvalues of l.

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 [15, 16, 24, 60]. In practice we take Γ to be a circle and usually find that 32 or 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 KS 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 centered 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 nondiagonal problems, quite a bit of computation is involved—say, 32 matrix inverses—but as this is done just once before the time-stepping begins (assuming that the time steps are of a fixed size), the impact on the total computing time is small. It would be greater for some problems in multiple space dimensions, but in some cases one could ameliorate the problem with a preliminary Schur factorization to bring the matrix to triangular form.

Contour integrals and Taylor series are not the only solutions that have been proposed for this problem. Both Beylkin [5] and Friesner et al. [21], for example, use a method that is based on scaling and squaring. That method is also effective, but the contour integral method appeals to us because of its greater generality for dealing with arbitrary functions.

To demonstrate the effectiveness of our stabilization method, Table 2.2 considers the computation of the coefficient $\gamma(z)$ of (2.5) (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

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. Again, all computations are done in double precision.

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

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.)

Another way to demonstrate the effectiveness of our method is to see it in action. Figures 1–4 of the next section demonstrate fast fourth-order convergence achieved in application to four PDEs. An additional figure in that section, Figure 5, shows what happens to the ETDRK4 method if instead of using 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 [60]. We estimate the true solution by using a "very small" time step (half the smallest time step shown in the graphs). The error is then calculated as the ∞-norm of the difference between the solution at larger time steps and the "exact" solution at the small time step divided by the maximum value of the solution. Thus the error plotted in the graphs is a relative one. In a similar vein, the time steps displayed have all been divided by the overall simulation time scale and are thus scaled from 0 to 1. A relative time step of 1e-2, for example, implies that the actual time step 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 discretization and vary the time step.

KdV equation with periodic boundary conditions,

$$\begin{aligned} u_t &= -uu_x - u_{xxx}, \quad x \in [-\pi, \pi], \\ u(x, t=0) &= 3A^2 \operatorname{sech}\left(\frac{A(x+2)}{2}\right)^2 + 3B^2 \operatorname{sech}\left(\frac{B(x+1)}{2}\right)^2, \end{aligned} \tag{3.1}$$

with $A = 25$, $B = 16$. The simulation runs to $t = 0.001$. Here and for the next two equations we use a 512-point Fourier spectral discretization in x.

Burgers equation with periodic boundary conditions,

$$\begin{aligned} u_t &= -\left(\frac{1}{2}u^2\right)_x + \epsilon u_{xx}, \quad x \in [-\pi, \pi], \\ u(x, t=0) &= \exp(-10\sin^2(x/2)), \end{aligned} \tag{3.2}$$

with $\epsilon = 0.03$ and the simulation running to $t = 1$.

KS equation with periodic boundary conditions,

$$\begin{aligned} u_t &= -uu_x - u_{xx} - u_{xxxx}, \quad x \in [0, 32\pi], \\ u(x, t=0) &= \cos\left(\frac{x}{16}\right)\left(1 + \sin\left(\frac{x}{16}\right)\right), \end{aligned} \tag{3.3}$$

with the simulation running to $t = 30$.

Allen–Cahn equation with constant Dirichlet boundary conditions,

$$\begin{aligned} u_t &= \epsilon u_{xx} + u - u^3, \quad x \in [-1, 1], \\ u(x, t=0) &= .53x + .47\sin(-1.5\pi x), \quad u(-1, t) = -1, \quad u(1, t) = 1, \end{aligned} \tag{3.4}$$

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 spatial discretization is by an 80-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 summarized in Figures 1–4. The first plot in each figure compares accuracy against step size and thus should be reasonably independent of machine and implementation. Ultimately, of course, it is computer time that matters, and this is what is displayed in the second plot in 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 1–4, let us first highlight the most general point: our computations show that it is entirely practical to solve these difficult nonlinear PDEs 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.

Turning now to the differences between methods revealed in Figures 1–4, 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. We were not able to make every method work in every case. If a method does not appear on a graph it means that it did not succeed, seemingly for reasons of nonlinear instability (which perhaps might have been tackled by dealiasing in our spectral discretizations).

A key feature to look for in the first plot in each figure 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 second plot in the figures gives insight into these computation times. We can make a few comments on each of the methods investigated.

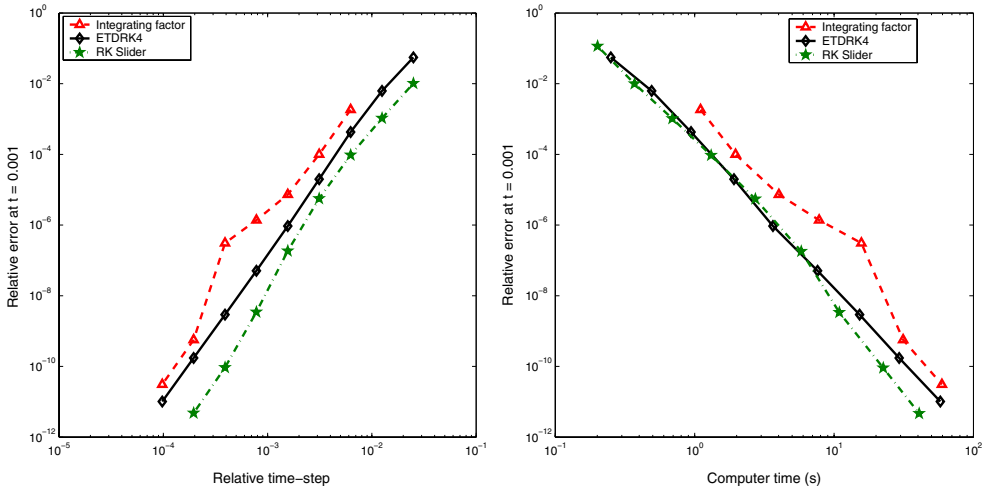

FIG. 1. *Accuracy versus time step and computer time for three schemes for the KdV equation.*

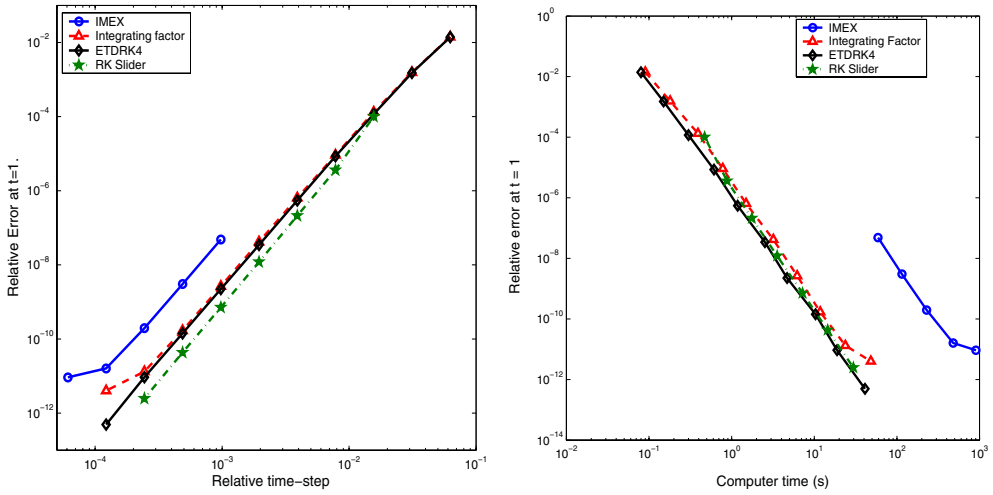

FIG. 2. *Accuracy versus time step and computer time for four schemes for the Burgers equation.*

Exponential time-differencing is very good in every case. It works equally well for diagonal and nondiagonal problems, it is fast and accurate, and it can take large time steps.

The *implicit-explicit* scheme used in this study does not perform well. This is striking, as this is probably the most widely used of all the schemes. This fares particularly poorly for the dispersive equations, KdV, and Burgers equation, and for the KdV, we could not get the IMEX scheme to work at all at the spatial resolution that we used.

The *split step* method also performed poorly. It was unstable for all of the experiments that we performed with 512 points. The main problem with this scheme, even if it is stable, 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

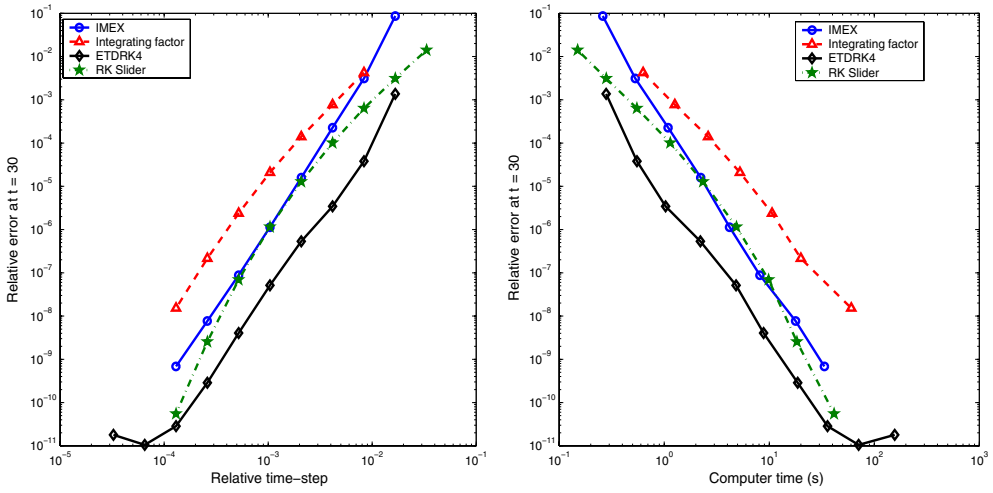

FIG. 3. *Results for the KS equation. Our* MATLAB *code is listed in section* 4.

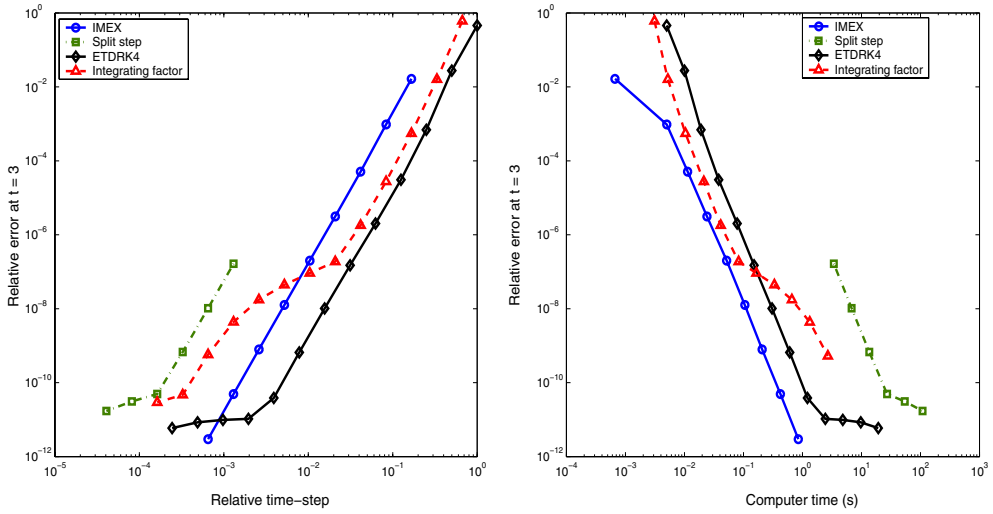

FIG. 4. *Results for the Allen–Cahn equation. This problem is nondiagonal and more challenging than the others. Our* MATLAB *code is listed in section* 5.

problem with schemes of higher than second order. For second-order calculations, split step schemes are certainly competitive.

The *slider* method does well in all of the diagonal cases. It is fast, accurate, and very stable. This is remarkable when we compare its performance with that of the IMEX schemes from which it was derived. The only problem with this scheme is the difficulty in generalizing it to nondiagonal cases.

Finally, the *integrating factor* scheme performs well for the Burgers equation. It doesn't do well for the KS equation, though, coming off worst of all, and we couldn't get it to work for the KdV equation or the Allen–Cahn equation with the spatial discretization that we used. This is also a little surprising, considering how widely used this scheme is.

One important consideration is, how easily do these schemes generalize to several space dimensions? With the exception of the slider method, they all generalize 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 the nondiagonal problem 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 diagonalization; if all dimensions are periodic, we have only scalars to deal with, and if only one dimension is nonperiodic, we have a collection of small blocks.

Another generalization that one might consider would be how easily these schemes could be adapted to take advantage of variable time-stepping. The IMEX and slider methods should be relatively easy to adapt, while those methods that use a matrix exponential would present some difficulties.

Overall then, it appears that the ETD scheme requires the fewest steps to achieve a given accuracy. It is also the fastest method in computation time, has excellent stability properties, and is the most general. It might be noted that the numerical experiments we have presented, since they resolve the spatial part of the problem to very high accuracy, are somewhat stiffer than if the spatial errors had been permitted on the same scale as the temporal errors, raising the question of whether ETD would also be the best for less fully resolved problems. Experiments with coarser resolutions indicate that yes, the advantages of ETD are compelling there, too.

We conclude this section with an illustration of how crucial our complex contour integrals, or other stabilization devices, are to the success of ETD schemes. Figure 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 stabilization 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 KS equation, which dates to the mid-1970s and has been used in the study of a variety of reaction-diffusion systems [39]. Our one-dimensional problem can be written as

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

As it contains both second- and fourth-order derivatives, the KS equation produces complex behavior. The second-order term acts as an energy source and has a destabilizing effect, and the nonlinear term transfers energy from low to high wavenumbers where the fourth-order term has a stabilizing effect. The KS equation is also very interesting from a dynamical systems point of view, as it is a PDE that can exhibit chaotic solutions [28, 47].

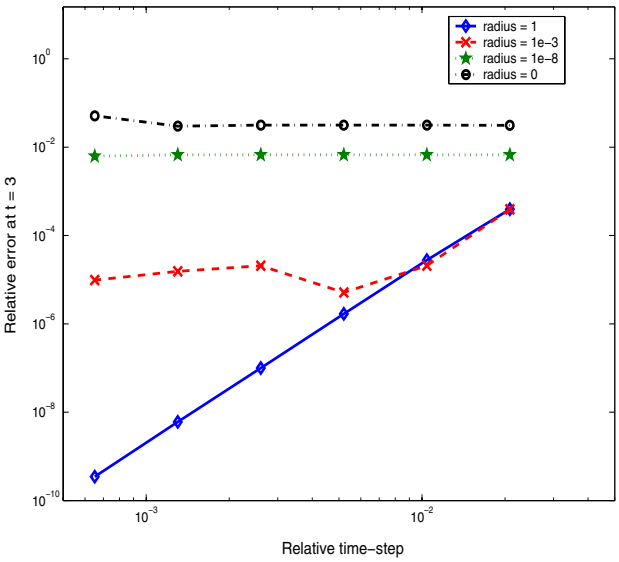

FIG. 5. *Loss of stability in the ETD scheme applied to the Burgers equation as the radius of the contour shrinks to zero. The contour of zero radius corresponds to an ETD calculation directly from the defining formula.*

We use the initial condition

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

As the equation is periodic, we discretize 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 to solve to $t = 150$. Figure 6 shows the result, which took less than 1 second of computer time on our workstation. Despite the extraordinary 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 6 with the MATLAB code listed in Figure 7.

5. A nondiagonal example: Allen–Cahn. The Allen–Cahn equation is another well-known equation from the area of reaction-diffusion systems:

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

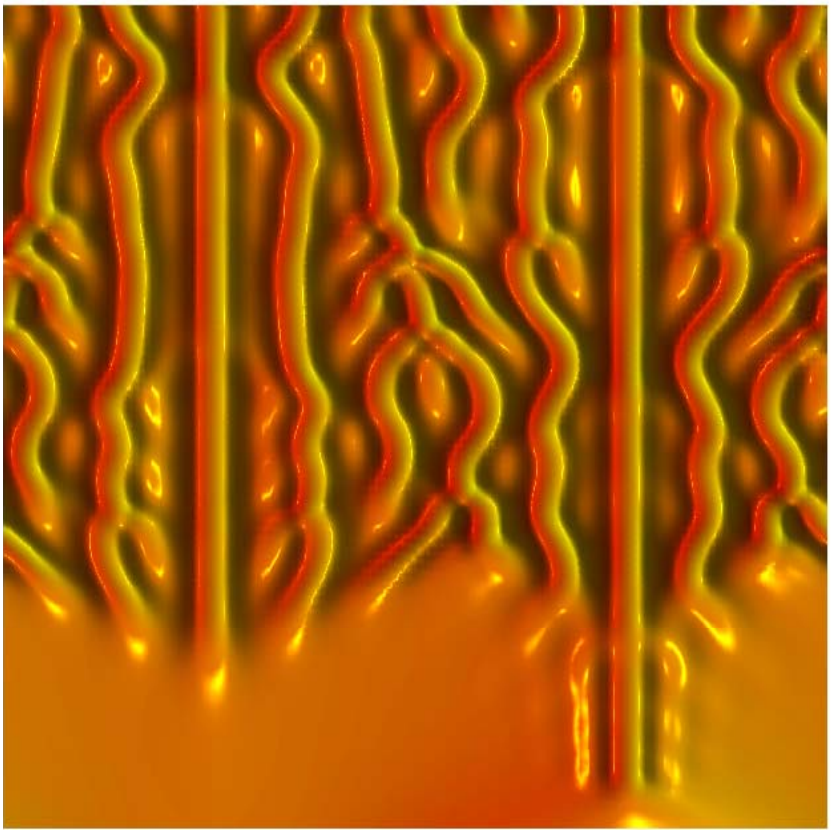

FIG. 6. *Time evolution for the KS equation. Time runs from* 0 *at the bottom of the figure to* 150 *at the top. This is a PDE example of deterministic chaos* [1].

and following p34.m of [60], we used the initial conditions

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

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 phenomenon 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 time scale before changing suddenly.

We can write a discretization of this equation 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 [60]. $\mathbf{L}$ 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 8 shows the result produced by the MATLAB code listed in Figure 9, which also runs in less than a second on our workstation. This code calls the function cheb.m from [60], available at http://www.comlab.ox.ac.uk/work/nick.trefethen.

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 Uri Ascher, Gino Biondini, Elaine Crooks, Arieh Iserles, Álvaro Meseguer, and Brynwulf Owren.

```
% 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. 7. MATLAB *code to solve the KS equation and produce Figure* 6. *Despite the extraordinary sensitivity of this equation to perturbations, this code computes correct results in less than* 1 *second on an* 800 MHz *Pentium machine.*

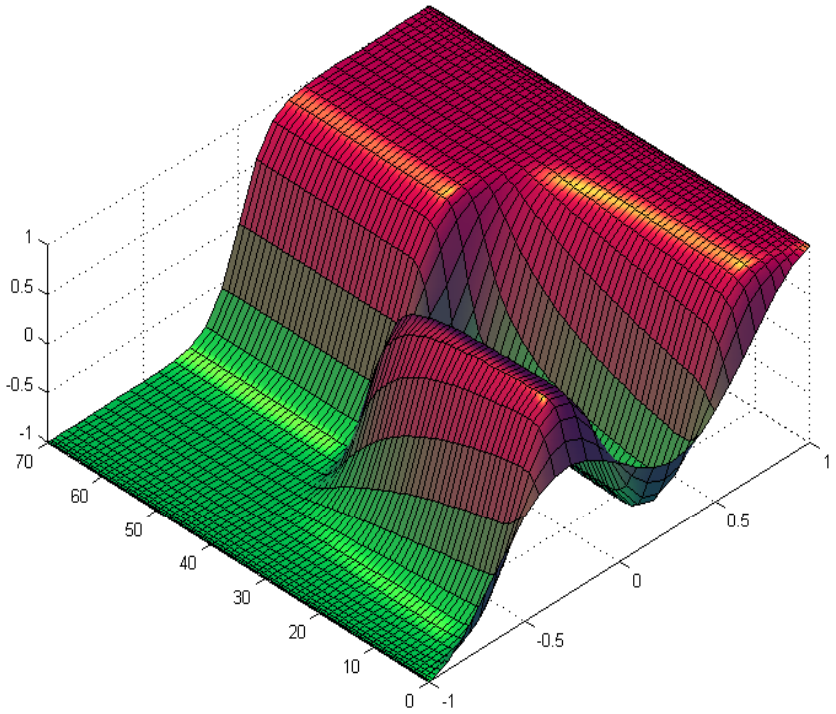

FIG. 8. *Time evolution for the Allen–Cahn equation. The x axis runs from $x = -1$ to $x = 1$, and the t-axis runs from $t = 0$ to $t = 70$. The initial hump is metastable and disappears near $t = 45$.*

```
% 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. 9. MATLAB *code to solve the Allen–Cahn equation and produce Figure* 8. *Again, this code takes less than* 1 *second to run on an* 800 MHz *Pentium machine.*

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*, Phys. D, 18 (1986), pp. 85–112.

[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 (1995), pp. 797–823.

[3] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri, *Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations*, Appl. Numer. Math., 25 (1997), pp. 151–167.

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

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

[6] A. Bourlioux, A. T. Layton, and M. L. Minion, *High-order multi-implicit spectral deferred correction methods for problems of reactive flows*, J. Comput. Phys., 189 (2003), pp. 651–675.

[7] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, Dover, Mineola, NY, 2001; also available online at http://www-personal.engin.umich.edu/∼jpboyd/.

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

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

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

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

[12] J. Certaine, *The solution of ordinary differential equations with large time constants*, in Mathematical Methods for Digital Computers, A. Ralston and H. S. Wilf, eds., Wiley, New York, 1960, pp. 128–132.

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

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

[15] P. J. Davis, *On the numerical integration of periodic analytic functions*, in On Numerical Approximation, E. R. Langer, ed., University of Wisconsin Press, Madison, WI, 1959, pp. 45–49.

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

[17] T. A. Driscoll, *A composite Runge–Kutta method for the spectral solution of semilinear PDEs*, J. Comput. Phys., 182 (2002), pp. 357–367.

[18] L. N. G. Filon, *On a quadrature formula for trigonometric integrals*, Proc. Roy. Soc. Edinburgh Sect. A, 49 (1928–1929), pp. 38–47.

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

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

[21] R. A. Friesner, L. S. Tuckerman, B. C. Dornblaser, and T. V. Russo, *A method for exponential propagation of large stiff nonlinear differential equations*, J. Sci. Comput., 4 (1989), pp. 327–354.

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

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

[24] P. Henrici, *Applied and Computational Complex Analysis*, Vol. 3, Wiley, New York, 1986.

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

[26] M. Hochbruck, C. Lubich, and H. Selhofer, *Exponential integrators for large systems of differential equations*, SIAM J. Sci. Comput., 19 (1998), pp. 1552–1574.

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

[28] J. M. Hyman and B. Nicolaenko, *The Kuramoto-Sivashinsky equation: A bridge between PDEs and dynamical systems*, Phys. D, 18 (1986), pp. 113–126.

[29] A. Iserles, *A First Course in the Numerical Analysis of Differential Equations*, Cambridge University Press, Cambridge, UK, 2000.
[30] A. Iserles, *On the numerical quadrature of highly oscillating integrals* I*: Fourier transforms*, IMA J. Numer. Anal., 24 (2004), pp. 365–391.
[31] J. C. Jiminez, R. Biscay, C. Mora, and L. M. Rodriguez, *Dynamic properties of the local linearization method for initial-value problems*, Appl. Math. Comput., 126 (2002), pp. 63–81.
[32] G. E. Karniadakis, M. Israeli, and S. A. Orszag, *High order splitting methods for the incompressible Navier-Stokes equations*, J. Comput. Phys., 97 (1991), pp. 414–443.
[33] C. A. Kennedy and M. H. Carpenter, *Additive Runge–Kutta schemes for convection-diffusion-reaction equations*, Appl. Numer. Math., 44 (2003), pp. 139–181.
[34] J. Kim and P. Moin, *Applications of a fractional step method to incompressible Navier-Stokes equations*, J. Comput. Phys., 59 (1985), pp. 308–323.
[35] O. M. Knio, H. N. Najim, and P. S. Wyckoff, *A semi-implicit numerical scheme for reacting flow*, J. Comput. Phys., 154 (1999), pp. 428–467.
[36] 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*, Philos. Mag. Ser. 5, 39 (1895), pp. 422–433.
[37] W. Kress and B. Gustafsson, *Deferred correction methods for initial value boundary problems*, J. Sci. Comput., 17 (2002), pp. 241–251.
[38] M. Krusemeyer, *Differential Equations*, Macmillan College Publishing, New York, 1994.
[39] Y. Kuramoto and T. Tsuzuki, *Persistent propagation of concentration waves in dissipative media far from thermal equilibrium*, Prog. Theoret. Phys., 55 (1976), pp. 356–369.
[40] Y. Maday, A. T. Patera, and E. M. Rønquist, *An operator-integration-factor splitting method for time-dependent problems: Application to incompressible fluid flow*, J. Sci. Comput., 5 (1990), pp. 263–292.
[41] R. I. McLachlan and P. Atela, *The accuracy of symplectic integrators*, Nonlinearity, 5 (1992), pp. 541–562.
[42] R. McLachlan, *Symplectic integration of Hamiltonian wave equations*, Numer. Math., 66 (1994), pp. 465–492.
[43] W. J. Merryfield and B. Shizgal, *Properties of collocation third-derivative operators*, J. Comput. Phys., 105 (1993), pp. 182–185.
[44] P. A. Milewski and E. G. Tabak, *A pseudospectral procedure for the solution of nonlinear wave equations with examples from free-surface flows*, SIAM J. Sci. Comput., 21 (1999), pp. 1102–1114.
[45] M. L. Minion, *Semi-implicit spectral deferred correction methods for ordinary differential equations*, Commun. Math. Sci., 1 (2003), pp. 471–500.
[46] 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. Comput. Phys., 164 (2000), pp. 407–428.
[47] B. Nicolaenko, B. Scheurer, and T. Temam, *Some global properties of the Kuramoto-Sivashinsky equation: Nonlinear stability and attractors*, Phys. D, 16 (1985), pp. 155–183.
[48] S. P. Nørsett, *An A-stable modification of the Adams–Bashforth methods*, in Conference on Numerical Solution of Differential Equations (Dundee, 1969), Lecture Notes in Math. 109, Springer-Verlag, Berlin, 1969, pp. 214–219.
[49] E. Ott, *Chaos in Dynamical Systems*, Cambridge University Press, Cambridge, UK, 1993.
[50] R. D. Ruth, *A canonical integration technique*, IEEE Trans. Nuclear Science, NS-30 (1983), pp. 2669–2671.
[51] S. J. Ruuth, *Implicit-explicit methods for reaction-diffusion problems in pattern formation*, J. Math. Biol., 34 (2) (1995), pp. 148–176.
[52] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228.
[53] J. M. Sanz-Serna and M. P. Calvo, *Numerical Hamiltonian Problems*, Chapman and Hall, London, 1994.
[54] M. Schatzman, *Toward non-commutative numerical analysis: High order integration in time*, J. Sci. Comput., 17 (2002), pp. 99–116.
[55] L. M. Smith and F. Waleffe, *Transfer of energy to two-dimensional large scales in forced, rotating three-dimensional turbulence*, Phys. Fluids, 11 (1999), pp. 1608–1622.
[56] L. M. Smith and F. Waleffe, *Generation of slow large scales in forced rotating stratified turbulence*, J. Fluid Mech., 451 (2002), pp. 145–169.
[57] G. Strang, *On the construction and comparison of difference schemes*, SIAM J. Numer. Anal., 5 (1968), pp. 506–517.
[58] E. Tadmor, *The exponential accuracy of Fourier and Chebyshev differencing methods*, SIAM J. Numer. Anal., 23 (1986), pp. 1–10.

[59] A. Taflove, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, Artech House, Boston, 1995.
[60] L. N. Trefethen, *Spectral Methods in MATLAB*, Software Environ. Tools 10, SIAM, Philadelphia, 2000.
[61] J. M. Varah, *Stability restrictions on second order, three level finite difference schemes for parabolic equations*, SIAM J. Numer. Anal., 17 (1980), pp. 300–309.
[62] J. G. Verwer, J. G. Blom, and W. Hundsdorfer, *An implicit-explicit approach for atmospheric transport-chemistry problems*, Appl. Numer. Math., 20 (1996), pp. 191–209.
[63] H. Yoshida, *Construction of higher order symplectic integrators*, Phys. Lett. A, 150 (1990), pp. 262–268.