JOURNAL OF COMPUTATIONAL PHYSICS 122, 25–29 (1995)

Some Practical Experience with the Time Integration of Dissipative Equations

BOSCO GARCÍA-ARCHILLA*

Departamento de Matemática Aplicada y Computación, Universidad de Valladolid, Valladolid, Spain

Received April 12, 1993; revised December 2, 1994

Different methods for the numerical integration of evolution dissipative partial differential equations are tested with the Kuramoto–Sivashinsky equation. Discretizations in space include Galerkin and nonlinear Galerkin methods. For integration in time three different codes are used, including standard stiff ODE methods. Numerical tests show that standard codes for stiff ODE render a gain of computing time of several orders of magnitude with respect problem-tailored methods. © 1995 Academic Press, Inc.

1. INTRODUCTION

A number of promising ideas have recently been suggested for the numerical integration of dissipative partial differential equations (PDEs). Part of the recent literature on the subject (see, e.g., [3, 4, 14, 18, 23]) has aimed at obtaining theoretically sound, time-saving adaptive procedures to discretize the spatial derivatives. The nonlinear Galerkin method (NLG) is a prominent instance of this. However, little attention has been paid to efficient integration in time. In most cases, time-stepping methods of low accuracy are coupled with sophisticated discretizations in space. The purpose of the present paper is to gain some practical experience on the performance of the new techniques for spatial discretization and that of efficient (but already available) time integrators for ordinary differential equations (ODEs). It turns out that these can improve efficiency by several orders of magnitude with respect to methods currently used.

Many dissipative PDEs [22, 10] can be written as abstract differential equations

$$\frac{du}{dt} + Au + R(u) = 0, \tag{1}$$

in some Hilbert space H. Here, $A : D(A) \subset H \to H$ is a densely defined, unbounded, self-adjoint, and non-negative operator, and $R : D(R) \subset H \to H$ is a nonlinear (possibly time-dependent) operator such that $D(A) \subset D(R)$. The linear operator A involves the higher order spatial derivatives, while R gathers lower order spatial derivatives.

* E-mail: bosco@mac.cie.uva.es.

The discretization of (1) by the standard spectral Galerkin method considers a complete orthonormal set $\{w_0, w_1, ...\}$ in H (for simplicity we assume that the w_i's are the eigenfunctions of A) and projects the equation in $H_m = \text{span}\{w_0, w_1, ..., w_m\}$, that is,

$$\frac{du_m}{dt} + Au_m + P_m R(u_m) = 0, \quad u_m \in H_m, \tag{2}$$

where P_m is the orthogonal projection onto H_m. The relation (2) implies a system of ODEs for the coefficients of u_m in the basis $w_0, ..., w_m$ which can be integrated by a numerical method for ODEs.

The nonlinear Galerkin discretization [18] was introduced in connection with inertial manifolds [8]. The basic idea is to treat the lower and the higher modes in the solution in a different fashion. More precisely, fix $m_1 < m$, and decompose $u_m \in H_m$ as $u_m = p_{m_1} + q_{m_1}$, where $p_{m_1} \in H_{m_1}$ and p_{m_1} and q_{m_1} are orthogonal, that is, $q_{m_1} \in Q_{m_1}H = (P_m - P_{m_1})H$. The component q_{m_1} is usually small and can be integrated less accurately than p_{m_1} [18]. This allows us to set different equations for p_{m_1} and q_{m_1}. A much favoured choice is

$$\frac{dp_{m_1}}{dt} + Ap_{m_1} + P_{m_1}(R(p_{m_1} + q_{m_1}) - R(q_{m_1})) = 0, \quad p_{m_1} \in H_{m_1}, \tag{3}$$

$$\frac{dq_{m_1}}{dt} + Aq_{m_1} + Q_{m_1}R(p_{m_1}) = 0, \quad q_{m_1} \in Q_{m_1}H, \tag{4}$$

although several others have been suggested [3, 18, 19]. The purpose of this decomposition is twofold. On the one hand, (3) is set in a smaller space (hence, it is cheaper to compute) than (2), and, because q_{m_1} is small, it is almost as accurate a discretization of (1) as (2). On the other, q_{m_1}, after a transient phase, evolves more slowly than p_{m_1}, making it unnecessary to update q_{m_1} as frequently as p_{m_1}. A more precise theoretical justification and further details can be found in [6]. Also, some results concerning the limitations of the NLG method can be found in [12].

Numerical experience [4, 14, 23], shows that the numerical

0021-9991/95 $12.00

integration of the nonlinear Galerkin discretization (3)–(4) requires between 40 and 60% less CPU time than the standard Galerkin discretization (2) for the same accuracy. However, it is suggested in this paper that this gain is much smaller than the advantage that can be obtained by integrating the standard Galerkin discretization by efficient ODE methods. In Section 3, an implementation of the nonlinear Galerkin discretization suggested by Jauberteau *et al.* [14] is compared with the standard Galerkin discretization integrated in time with variable step-size backward differentiation formulae (BDF). The errors in this second procedure are several orders of magnitude smaller than in the nonlinear Galerkin algorithm of Jauberteau *et al.*, while the CPU time is 25 times smaller for the BDF algorithm. It remains to ascertain what can be gained by coupling the BDF with the nonlinear Galerkin discretization. This presents several difficulties that are studied in [9].

For numerical tests, the Kuramoto–Sivashinsky (KS) equation

$$u_t + 4\,\Delta^2 u + \theta(\Delta u + \tfrac{1}{2}\,|\nabla u|^2) = 0 \qquad (5)$$

in two spatial dimensions was chosen. This equation has received much attention in recent years as it models phenomena in hydrodynamics, combustion, plasma physics, etc. (see, e.g., [13, 20, 21] and the references cited therein). Also, the rich dynamics the KS equation exhibits make its study interesting from the purely mathematical point of view. In fact, this research started in collaboration with colleagues who were studying this equation. Many and costly numerical experiments had to be carried out. The good performance of the NLG method reported in [14] suggested us to adapt the Jauberteau *et al.* algorithm to the KS equation, a task which required some tuning of the strategy to avoid computations of q_{m_1}. However, the slow rate at which numerical results were obtained made us try other alternatives, which proved to be much more efficient. This surprised us since these new alternatives were standard and straightforward to implement, when compared with the sophistication of the NLG algorithm.

The fact that the KS equation has been shown to possess an inertial manifold [7, 2] makes it well-suited for testing the nonlinear Galerkin discretization. Several authors [5, 15, 24] have already applied the nonlinear Galerkin discretization to the KS equation.

In the rest of the paper, Section 2 is devoted to a description of the methods tested and Section 3, to numerical experiments.

2. THE METHODS TESTED

Equations (2) and (3)–(4) can be rewritten as a system of ODEs of the form

$$\frac{dy}{dt} = F(y) = Ly + NL(y), \quad y \in R^m, \qquad (6)$$

where L and NL are linear and nonlinear operators in R^n, respectively. For example, in Eq. (2), if we denote by $\mathcal{F}$ the mapping from H_m onto R_m giving the coefficients of a function in the basis $w_0, \ldots, w_n$, then $y = \mathcal{F}u_m$, $Ly = -\mathcal{F}A\mathcal{F}^{-1}y$, and $NL(y) = -\mathcal{F}P_mR(\mathcal{F}^{-1}y)$.

Notice that if R in (1) is time-dependent, then, in (6), $F = F(t, y)$. However, as the numerical tests were carried out with the (autonomous) KS equation, for simplicity of exposition, we only treat the autonomous case. The reader will find no difficulty in extending the methods to nonautonomous systems.

An algorithm for the numerical solution of an initial value problem (IVP) of an ODE like (6) produces a series of values y_n starting from an initial condition y_0. Each y_n is meant to be an approximation at time t_n to the solution y of the IVP with initial condition $y(t_0) = y_0$. The time levels t_n can be equally spaced if a fixed step-size is used, or unevenly spaced if variable step-sizes are employed. In this case, the step-sizes are determined by the code as the integration proceeds.

As mentioned in the introduction, two methods for the integration in time are compared. The variable-step size BDF and a fixed step-size method suggested by Jauberteau *et al.* [14]. The second method is used to integrate both the standard Galerkin and the nonlinear Galerkin discretizations, whereas the BDF is only used with the standard Galerkin. Also, to gain further understanding of the performance of the fixed step method by Jauberteau *et al.* [14], another fixed step-size code is tested with the standard Galerkin discretization. In the rest of the section, the three integrators are described.

Variable order variable coefficient BDF (BDF). In the k-step BDF method, the value of y_{n+1} is obtained from previous values by demanding that the Lagrange collocation polynomial $P_n(t)$ based on $y_{n-k+1}, \ldots, y_n, y_{n+1}$ satisfies that $P_n'(t_{n+1}) = F(y_{n+1})$. The formula for the y_n's is

$$G_k(y_{n+1}) = \sum_{j=1}^{k} \left(\prod_{i=1}^{j-1} (t_{n+1} - t_{n+1-i}) \right) y[t_{n+1}, \ldots, t_{n-j+1}] - F(y_{n+1}) = 0. \qquad (7)$$

Here, $y[t_{n+1}, \ldots, t_{n-j+1}]$ denotes the standard divided difference defined by $y[t_n] = y_n$, $y[t_n, \ldots, t_{n-j}] = (y[t_n, \ldots, t_{n-j+1}] - y[t_{n-1}, \ldots, t_{n-j}])/(t_n - t_{n-j})$, $n = 1, 2, \ldots; j = 1, \ldots, n$.

In our code, the resulting system of nonlinear equations is solved by a Newton-like iteration $y_{n+1}^{[\nu]} = y_{n+1}^{[\nu-1]} - J^{-1} G_k(y_{n+1}^{[\nu-1]})$, $\nu = 1, 2, \ldots$, where J is the Jacobian of the linear part of G_k. The initial condition for the iteration is given by $y_{n+1}^{[0]} = Q_n(t_{n+1})$, where Q_n is the Lagrange interpolant of $y_{n-k}, \ldots, y_n$ (if y_{n-k} is not yet available a lower degree interpolant is used).

For monitoring the truncation error, the Milne-type formula [16, p. 108]

$$\mathrm{Est}_k = \frac{t_{n+1} - t_n}{t_n - t_{n-k}} (y_{n+1} - y_{n+1}^{[0]})$$

is used. The value of y_n is rejected if Est_k is larger than a user-supplied tolerance TOL, and accepted otherwise. Being the k-step BDF formula a kth-order method, the value h_{new} for the next step-size is given in terms of the current step-size $h_n = t_{n+1} - t_n$ by the familiar formula

$$h_{\mathrm{new}} = 0.9h_n \min(\mathrm{facmax}, (\mathrm{Est}_k/\mathrm{TOL})^{1/(k+1)})$$

(facmax = 2 in our code). Estimation of the truncation error of $(k+1)$-step and the $(k-1)$-step methods are also calculated, with their corresponding values of the step-size. For the next step, the formula that allows the largest step-size is used. Orders up to six were used. The reader is referred to [11] for more details on the implementation.

The code used in the experiments in this paper includes none of the usual time-saving devices that commercial codes include (constraining the step-sizes between certain bounds, updating Jacobians in the Newton iteration only when this fails, etc.) In fact, the program used was a classroom code modified only to allow for the extra storage required in the computation of the nonlinear term NL.

Linearly exact Runge–Kutta method (LERK). This method is an explicit, three-stage, third-order Runge–Kutta (RK) method in which the linear part is integrated exactly. It has been suggested and tested by Jauberteau *et al.* in [14].

The underlying RK method has the Butcher tableau

$$\begin{array}{c|ccc} 0 & 0 & & \\ 1/3 & 1/3 & & \\ 3/4 & -3/16 & 15/16 & \\ \hline & 1/6 & 3/10 & 8/15 \end{array} \tag{8}$$

With step size h, y_{n+1} is obtained from y_n by the following process:

$$\begin{aligned} H_1 &= he^{(h/3)L}NL(y_n), \\ v_1 &= e^{(h/3)L}v_0 + \tfrac{1}{3}H_1, \\ H_2 &= e^{(5/12)hL}(hNL(v_1) - \tfrac{5}{9}H_1), \\ v_2 &= e^{(5/12)hL}v_1 + \tfrac{15}{16}H_2, \\ H_2 &= e^{(h/4)L}(hNL(v_2) - \tfrac{153}{128}H_2), \\ y_{n+1} &= e^{(h/4)L}v_2 + \tfrac{8}{15}H_3. \end{aligned}$$

The method is based on the following observation. If y is solution of (6), then $w = e^{-Lt}y$ is solution of

$$w' = e^{-Lt}NL(e^{Lt}w).$$

The algorithm above is then obtained by applying the RK method (8) to this system on every step, and then multiplying by $e^{L(t+h)}$ to recover y_{n+1}.

For the time integration of the nonlinear Galerkin method (3)–(4), this procedure is applied alternatively to (3) and (4). The fact that during large time intervals q_{m_1} evolves more slowly than p_{m_1} allows us to keep the value of q_{m_1} in (3) fixed for several consecutive steps. It is updated when it differs from the value of q_{m_1} computed in (4) more than a certain tolerance. In the code used here this tolerance was set to $\|NL(y_n)\|/m^2$.

Linearly implicit second-order Adams formula (LIA). This method treats the linear Ly part in (6) by the second-order implicit Adams formula (trapezoidal rule) and the nonlinear part $NL(y)$ by the explicit second-order (two-step) Adams formula. More precisely, given y_n and y_{n-1}, the value y_{n+1} satisfies

$$y_{n+1} = y_n + h\left(L\frac{y_{n+1} + y_n}{2} + \frac{1}{2}(3NL(y_n) - NL(y_{n-1}))\right),$$

where h is the step-size. Although the method can be implemented with variable step-sizes, it was only used with fixed step-size to make a better comparison with the LERK method. The method is second-order convergent. Its convergence and performance for the KS equation has been studied in [17]. Notice that, in the case of spectral spatial discretizations, the matrix $I - (h/2)L$ of the linear system to be solved at each step is a diagonal matrix.

The additional starting value y_1 was obtained by means of the explicit Euler rule $y_1 = y_0 + h(Ly_0 + NL(y_0))$.

3. NUMERICAL EXPERIMENTS

The three integrators above were tested on the 2π periodic KS equation. As initial condition, $u(x, y, 0) = \cos(x)\cos(y)$ was chosen. The value θ was set to $\theta = 20$. The solution was required to be an even function in both space directions. The basis functions for the Galerkin and nonlinear Galerkin methods were taken to be $\cos(jx)\cos(ky), j, k = 0, 1, 2, \ldots$, the eigenfunctions of the linear operator in the KS equation. The nonlinear term was evaluated by collocation and FFT, using the $\frac{3}{2}$ rule [1, p. 85] to avoid aliasing.

The value of m for both the standard Galerkin and the nonlinear Galerkin was $m = 16^2$ and the value of m_1 for the nonlinear Galerkin method was $m_1 = 8^2$. This values of m and m_1 represent a balance between accuracy and cost. The value of $m = 16^2$ solves the problem for the precision available in the standard Galerkin discretization. A bigger value of m_1 (or m) would make the nonlinear Galerkin discretization more costly than the standard Galerkin, whereas a smaller one would make it too inaccurate. Also, this value of m_1 avoids the need of the $\frac{3}{2}$ rule to prevent aliasing.

Experiments were carried out on a Sun Sparc Station 2. All methods were programmed in FORTRAN. For the FFT the NAG library was used.

The fixed-step methods were run with a range of values of the step-size in which every value is half the previous value. The BDF formulae were run for tolerances $10^{-3}, 10^{-5}, \ldots, 10^{-11}$.

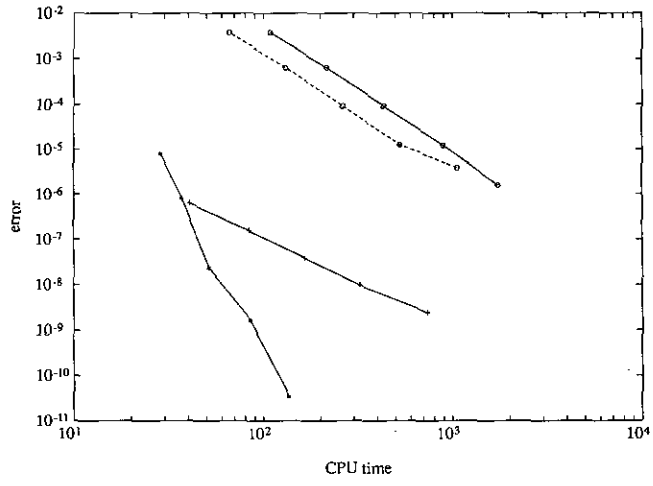

FIG. 1. Efficiency diagram: *, BDF; +, LIA; ∘, LERK; ——, standard Galerkin; --, nonlinear Galerkin.

Figure 1 shows a loglog plot of the CPU time in seconds versus the L_2 norm of the error taken at time $t = 0.25$. The BDF is represented by stars, the LERK by circles, and the LIA method by crosses. The standard Galerkin and the nonlinear Galerkin discretization are represented by continuous and discontinuous lines, respectively.

It can be seen that the most efficient method is the variable step-size variable formula BDF method, and the worst one is the linearly exact Runge–Kutta method. The nonlinear Galerkin discretization saves on the LERK method approximately two-thirds of the CPU time (in agreement with experiments in [14, 23, 4]) but this gain is insignificant compared to the efficiency of the variable step-size BDF. Notice that the smallest errors in the nonlinear Galerkin discretization integrated by the LERK method are comparable to the largest errors in the BDF (with standard Galerkin); but while the BDF takes approximately 40 s to compute the solution, the LERK takes a 1000 s (25 times more).

The results of the LIA method show that the poor performance of the LERK is due not to its order (larger in the latter method) but to its large error constants. Recall that the LERK is third-order convergent, whereas the LIA is only second-order convergent. Measures of the slopes of the plots confirm this fact. The large amount of CPU time used by the LERK is due to the fact that much smaller step-sizes were required in order to obtain errors comparable to those obtained with the other two methods. This can be seen by looking at the size of the truncation errors. The size of these is better understood if we consider simpler systems than (6). Consider then

$$y' = Ly + g(t),$$

with L the same diagonal matrix featuring in (6) and g a smooth forcing term. The LERK then attempts to solve the system

$$w' = e^{-Lt}g(t), \tag{9}$$

which is a quadrature problem. Notice how, by integrating (9), the LERK avoids any stability restriction imposed by the size of the entries of L. However, this is traded off for accuracy. A simple calculation [11, pp. 156–157] shows that the truncation error T_n at t_n (in the y variables) is

$$ce^{L(I-\theta)h}h^4(L^4g(t) + 4L^3g'(t) + 6L^2g''(t) + 4Lg'''(t) + g''''(t)) + O(\|L\|^4h^5),$$

where c is a constant depending only on the coefficients of the method and $\theta = \text{diag}(\theta_1, \ldots, \theta_m)$, with $\theta_j \in (0, 1)$, $j = 1, \ldots, m$. Then, the best that can be said of T_n is that

$$\|T_n\| = O((\|L\|h)^4),$$

whereas for the LIA method, Taylor expansion reveals that the truncation error is $O(\|L\|h^3)$. Loosely speaking, we may say that the error coefficients in the LERK are $\|L\|^3$ times bigger than in the LIA method (recall that in the KS equation, $\|L\| = O(m^4)$). An analysis of the truncation error in the case of (6), although more involved than the one performed above, also reveals that, in general, there is no way of avoiding the presence of L^4 in the expression of T_n, so that, as Fig. 1 shows, one may expect big errors when using the LERK.

We also show, in Fig. 2, measures of the errors at time $t = 2.25$. At this time level, the theoretical solution has passed near a saddle and approaches the attractor (a situation where the nonlinear Galerkin method is supposed to show its capabilities). Notice how the nonlinear Galerkin method reaches the limit of its precision and is not capable of reducing errors below 10^{-5}. Reducing the value of h does not reduce the error, whereas this can still be reduced for the standard Galerkin. That is, while

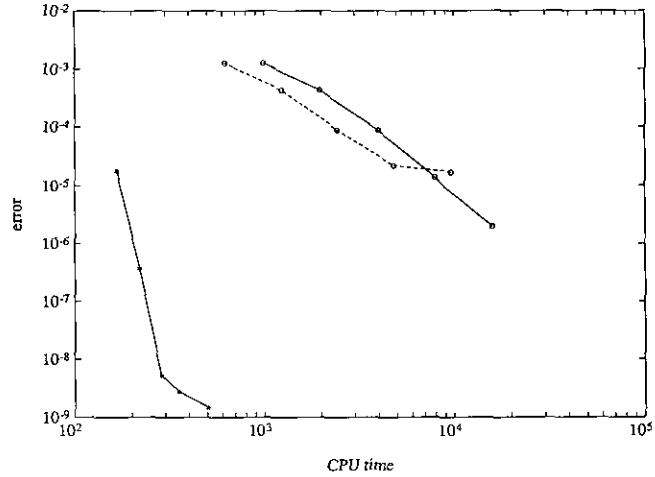

FIG. 2. Efficiency diagram after a saddle point: *, BDF; ∘, LERK: —— standard Galerkin, -- nonlinear Galerkin.

in the standard Galerkin the errors arising from spatial discretization are smaller than those arising from the time integrator (and, hence, reducing h reduces the error) in the NLG method, there is a point where the opposite situation happens.

It must be pointed out that the spatial error in the NLG can be reduced a bit further by better tuning the test to avoid computations of q_{m_1}. This goal was not pursued any further with the LERK in view of the cost of the experiments (3 h each). A different approach with the BDF (which involves the tolerance TOL in the code for the q_{m_1} tests) is followed in [9]. Preliminary experiments show that, in the problem in Fig. 2, the NLG method, cannot produce errors below 10^{-7} (even computing q_{m_1} at every step).

We may conclude then that the computational advantages of the NLG method require further study. Since these are more substantial when the dimension of the problem is large, it seems advisable to use efficient time integrators to avoid reaching premature conclusions.

ACKNOWLEDGMENTS

This research has been partly financed by DGICYT Project PB89-0315. The author thanks Professor Sanz-Serna for his valuable suggestions and careful reading of the manuscript, to Professor Hairer for helpful discussions, and to an anonymous referee whose comments helped to improve the paper.

REFERENCES

1. C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics,* Computational Physics Series, (Springer-Verlag, Berlin, 1988).
2. P. Constantin, C. Foias, B. Nicolenco, and R. Temam, *Integral Manifolds and Inertial Manifolds for Dissipative Partial Differential Equation,* Appl. Math. Sci., Vol. 70 (Springer-Verlag, New York/Berlin, 1989).
3. C. Devulder and M. Marion, *SIAM J. Numer. Anal.* **29,** 462 (1992).
4. T. Dubois, F. Jaberteau, M. Marion, and R. Temam, *Comput. Phys. Commun.* **65,** 100 (1991).
5. C. Foias, M. S. Jolly, I. G. Kevrekidis, G. R. Sell, and E. S. Titi, *Phys. Lett. A* **131,** 433 (1988).
6. C. Foias, O. Manley, and R. Temam, *RAIRO Modél. Math. Anal. Numér.* **22,** 93 (1988).
7. C. Foias, B. Nicolenco, G. R. Sell, and R. Temam, *J. Math. Pures Appl.* **67,** 197 (1988).
8. C. Foias, G. R. Sell, and R. Temam, *J. Differential Equations* **73,** 309 (1988).
9. B. García-Archilla and J. de Frutos, *IMA J. Numer. Anal.* **15,** 221 (1995).
10. J. Hale, *Asymptotic Behavior of Dissipative Systems,* Math. Surveys and Monographs, Vol. 25 (Amer. Math. Soc., Providence, RI, 1988).
11. E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I* (Springer-Verlag, Berlin, 1987).
12. J. G. Heywood and R. Rannacher, *SIAM J. Numer. Anal.* **30,** 1603 (1993).
13. J. M. Hyman and B. Nicolaenko, *Physica D* **18,** 113 (1986).
14. F. Jauberteau, C. Rosier, and R. Temam, "A Nonlinear Galerkin Method for the Navier–Stokes Equations," in *Spectral and High Order Methods for Partial Differential Equations, Proceedings of the ICOSAHOM '89 Conference,* edited by C. Canuto and A. Quarterony (North-Holland, Amsterdam, 1990), p. 245.
15. M. S. Jolly, I. G. Kevrekidis, and E. S. Titi, *Physica D* **44,** 38 (1990).
16. J. D. Lambert, *Numerical Methods for Ordinary Differential Systems. The Initial Value Problem* (Wiley, Chichester, 1991).
17. M. A. López-Marcos, *IMA J. Numer. Anal.* **14,** 233 (1994).
18. M. Marion and R. Temam, *SIAM J. Numer. Anal.* **26,** 1139–1157 (1989).
19. R. Temam, *Math. Comput.* **196,** 477 (1991).
20. B. Nicolaenko, B. Scheurer, and R. Temam, *Commun. Part. Differential Eqs.* **14,** 245 (1989).
21. A. Nicolás-Carrizosa, "A Finite Difference Approach to the Kuramoto–Sivanshinsky Equation," in *Advances in Numerical Partial Differential Equations and Optimization,* edited by S. Gómez, J. P. Hennart, and R. Tapia (SIAM, Philadelphia, 1991), p. 262.
22. R. Temam, *Infinite Dimensional Dynamical Systems in Mechanics and Physics,* Appl. Math. Sci., Vol. 68 (Springer-Verlag, Berlin, 1988).
23. R. Temam, "Dynamical Systems, Turbulence and the Numerical Solution of the Navier–Stokes Equations," in *Proceedings, 11th International Conference on Numerical Methods in Fluid Dynamics (Williamsburg),* Lecture Notes in Physics, edited by D. Dwoyer and R. Voigt (Springer-Verlag, Berlin, 1990).
24. R. D. Russell, D. M. Sloan, and M. D. Trummer, *SIAM J. Sci. Comput.* **14,** 19 (1993).