JOURNAL OF COMPUTATIONAL PHYSICS **134,** 316–331 (1997)
ARTICLE NO. CP975710

Unconventional Schemes for a Class of Ordinary Differential Equations—With Applications to the Korteweg–de Vries Equation[1]

William Kahan* and Ren-Chang Li†

*Computer Science Division and Department of Mathematics, University of California at Berkeley, Berkeley, California 94720; and
†Department of Mathematics, University of Kentucky, Lexington, Kentucky 40506
E-mail: rcli@ms.uky.edu

Received August 26, 1996; revised March 20, 1997

An unconventional numerical method for solving a restrictive and yet often-encountered class of ordinary differential equations is proposed. The method has a crucial, what we call *reflexive*, property and requires solving one linear system per time-step, but is second-order accurate. A systematical and easily implementable scheme is proposed to enhance the computational efficiency of such methods whenever needed. Applications are reported on how the idea can be applied to solve the Korteweg–de Vries Equation discretized in space.

1. INTRODUCTION

Various applications yield systems of ordinary differential equations

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0, \tag{1.1}$$

with a special property: $\mathbf{f}(\mathbf{y})$ is at most quadratic in $\mathbf{y}$,

$$\mathbf{f}(\mathbf{y}) \equiv \mathbf{A}(\mathbf{y}, \mathbf{y}) + B\mathbf{y} + \mathbf{b}, \tag{1.2}$$

where $\mathbf{A}(\cdot, \cdot)$ is a symmetric tensor, B is a matrix with appropriate dimension, and $\mathbf{b}$ is a constant vector. Although a system of this kind looks very restrictive at first sight, it actually appears often in applications—examples including air pollution models [33], many partial differential equations after spatial discretization by finite difference, finite element or pseudospectral methods, like the Korteweg–de Vries equation [14, 31], the Boussinesq equation [31], and potentially many others. Another particularly important example is the *matrix differential Riccati equation* which appears ubiquitously throughout *mathematics, science*, and *engineering*. But we shall study it elsewhere.

In general a system of ordinary differential equations for which $\mathbf{f}(\mathbf{y})$ is a polynomial in $\mathbf{y}$, can be transformed into a big system like (1.1) and (1.2) by introducing a new variables. Unfortunately doing so may end up with an unstable system, even though the original system is stable.

In this paper, we propose efficient numerical methods for solving such a system. Special attention will be given to how the idea can be applied to solve the discretized Korteweg–de Vries (KdV) equations.

2. REFLEXIVE UPDATING FORMULAS

In principle, any one-step method for solving the initial value problem[2] (1.1) yields an *updating formula* $\mathbf{Q}(\theta, \mathbf{g})$ which advances $\mathbf{g} \approx \mathbf{y}(\tau)$ to $\mathbf{Q}(\theta, \mathbf{g}) \approx \mathbf{y}(\tau + \theta)$, where θ is the step-size. An updating formula $\mathbf{Q}(\theta, \mathbf{g})$ is *reflexive* if

$$\mathbf{Q}(-\theta, \mathbf{Q}(\theta, \mathbf{g})) = \mathbf{g}.$$

(It has been called *symmetric*, *reversible*, and *self-adjoint*, too, but as argued by Kahan [12], these terms are already overworked, so we prefer the word *reflexive*.) One example is *the implicit midpoint rule*: $\mathbf{Y} = \mathbf{y} + \theta\mathbf{f}((\mathbf{y} + \mathbf{Y})/2)$.

For the system (1.2), there is a readily available *reflexive* formula obtained by solving a linear system of equations in $\mathbf{Y}$:

$$\frac{\mathbf{Y} - \mathbf{y}}{\theta} = \mathscr{F}(\mathbf{Y}, \mathbf{y}) \equiv \mathbf{A}(\mathbf{Y}, \mathbf{y}) + B\,\frac{\mathbf{Y} + \mathbf{y}}{2} + \mathbf{b}. \tag{2.1}$$

Such a formula is not totally new. In fact, the same idea has been used by many people over years on a variety of special systems like (1.1) and (1.2), e.g., Kahan [11, 12], Li [17], Meyer-Spasche and Düchs [19], Mickens [20], Twizell, Wang, and Price [30], to name a few. What is new here is

[1] This material is based in part upon work supported from August 1995 to January 1997 by a Householder Fellowship in Scientific Computing at Oak Ridge National Laboratory, supported by the Applied Mathematical Sciences Research Program, Office of Energy Research, United States Department of Energy Contract DE-AC05-96OR22464 with Lockheed Martin Energy Research Corp.

[2] This is true regardless of what $\mathbf{f}$ may be.

0021-9991/97 $25.00

our exploitation of its reflexivity which makes it possible to be composed in a simply efficient way to yield higher order methods; see Section 3.

Equation (2.1) admits another formulation which will enable us to discover its link to the implicit midpoint rule (and the trapezoidal rule) and its further generalizations. Denote by $J_{\mathbf{f}}(\mathbf{u})$ the Jacobian matrix of $\mathbf{f}(\mathbf{u})$ evaluated at $\mathbf{u}$, and denote by $J_{\mathbf{A}}(\mathbf{u})$ the Jacobian matrix of $\mathbf{A}(\mathbf{u}, \mathbf{u})$ also evaluated at $\mathbf{u}$. It is easy to see that

$$J_{\mathbf{A}}(\mathbf{u}) + B \equiv J_{\mathbf{f}}(\mathbf{u}), \quad \tfrac{1}{2} J_{\mathbf{A}}(\mathbf{u}) \cdot \mathbf{u} \equiv \mathbf{A}(\mathbf{u}, \mathbf{u}),$$

since $\mathbf{A}(\mathbf{u}, \mathbf{u})$ is a symmetric tensor. Equation (2.1) is equivalent to

$$\frac{\mathbf{Y} - \mathbf{y}}{\theta} = \frac{1}{2} J_{\mathbf{A}}(\mathbf{y}) \cdot \mathbf{Y} + \frac{1}{2} B\mathbf{Y} + \frac{1}{2} B\mathbf{y} + \mathbf{b}.$$

So

$$\begin{aligned}
\frac{\mathbf{Y} - \mathbf{y}}{\theta} &= \frac{1}{2} J_{\mathbf{f}}(\mathbf{y}) \cdot \mathbf{Y} + \frac{1}{2} B\mathbf{y} + \mathbf{b} \\
&= \frac{1}{2} J_{\mathbf{f}}(\mathbf{y}) \cdot (\mathbf{Y} - \mathbf{y}) + \frac{1}{2} J_{\mathbf{f}}(\mathbf{y}) \cdot \mathbf{y} + \frac{1}{2} B\mathbf{y} + \mathbf{b} \\
&= \frac{1}{2} J_{\mathbf{f}}(\mathbf{y}) \cdot (\mathbf{Y} - \mathbf{y}) + \frac{1}{2} J_{\mathbf{A}}(\mathbf{y}) \cdot \mathbf{y} + \frac{1}{2} B\mathbf{y} + \frac{1}{2} B\mathbf{y} + \mathbf{b} \\
&= \frac{1}{2} J_{\mathbf{f}}(\mathbf{y}) \cdot (\mathbf{Y} - \mathbf{y}) + \mathbf{A}(\mathbf{y}, \mathbf{y}) + B\mathbf{y} + \mathbf{b} \\
&= \frac{1}{2} J_{\mathbf{f}}(\mathbf{y}) \cdot (\mathbf{Y} - \mathbf{y}) + \mathbf{f}(\mathbf{y}),
\end{aligned}$$

which means that Eq. (2.1) is equivalent to

$$\left(I - \frac{\theta}{2} J_{\mathbf{f}}(\mathbf{y})\right)(\mathbf{Y} - \mathbf{y}) = \theta \mathbf{f}(\mathbf{y}). \tag{2.2}$$

This is the equation that will link the newly proposed method to the implicit midpoint rule and the trapezoidal rule.

PROPOSITION 2.1. 1. *One interation of Newton's method to solve the implicit midpoint rule* $(\mathbf{Y} - \mathbf{y})/\theta = \mathbf{f}((\mathbf{Y} + \mathbf{y})/2)$ *for* $\mathbf{Y}$, *starting from a first guess* $\mathbf{y}$, *is*

$$\mathbf{Y} \approx \mathbf{y} + \theta \left(I - \frac{\theta}{2} J_{\mathbf{f}}(\mathbf{y})\right)^{-1} \mathbf{f}(\mathbf{y}).$$

In other words, the newly proposed method (2.1) *is just one Newton iteration applied to the implicit midpoint rule.*

2. *One iteration of Newton's method to solve the trapezoidal rule* $(\mathbf{Y} - \mathbf{y})/\theta = (\mathbf{f}(\mathbf{Y}) + \mathbf{f}(\mathbf{y})/2)$ *for* $\mathbf{Y}$, *starting from a first guess* $\mathbf{y}$, *is*

$$\mathbf{Y} \approx \mathbf{y} + \theta \left(I - \frac{\theta}{2} J_{\mathbf{f}}(\mathbf{y})\right)^{-1} \mathbf{f}(\mathbf{y}).$$

In other words, the proposed method (2.1) *is just one Newton iteration applied to the trapezoidal rule.*

Proof. It can be verified easily. ■

The formulation (2.2) makes it possible to be extended to any initial value problems and gives a second-order formula. But when $\mathbf{f}$ is not quadratic, such an extension fails to yield a reflexive formula and thus the efficient constructions of higher order approximations to be discussed in the next section do not apply. Equation (2.2) also suggests that the new method is in the family of so-called *Rosenbrock* method and thus A-stable [16].

Remark. The above treatment to the system (1.1)—(1.2), where $\mathbf{A}$, B, and $\mathbf{b}$ are constants can be easily extended to cover the case where $\mathbf{A}$, B, and $\mathbf{b}$ depend on time t. More generally, it may be generalized to the system

$$\frac{d}{dt}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} = \begin{pmatrix}\mathbf{f}_1(\mathbf{x}, \mathbf{y})\\ \mathbf{f}_2(\mathbf{x}, \mathbf{y})\end{pmatrix},$$

where

- $\mathbf{x}' = \mathbf{f}_1(\mathbf{x}, \mathbf{y})$ with frozen $\mathbf{y}$ can be solved either *exactly* or cheaply by a *reflexive* method.
- $\mathbf{f}_2(\mathbf{x}, \mathbf{y}) = \mathbf{A}(\mathbf{x})(\mathbf{y}, \mathbf{y}) + B(\mathbf{x})\mathbf{y} + \mathbf{b}(\mathbf{x})$.

Then a reflexive method for the whole system can be designed as follows: *given* $\mathbf{x} \approx \mathbf{x}(\tau)$, *and* $\mathbf{y} \approx \mathbf{y}(\tau)$,

1. *Integrate* $\mathbf{x}' = \mathbf{f}_1(\mathbf{x}, \mathbf{y})$ *with frozen* $\mathbf{y}$ *from* $t = \tau$ *to* $t = \tau + \theta/2$ *to get* $\hat{\mathbf{X}} \approx \mathbf{x}(\tau + \theta/2)$;

2. *Integrate* $\mathbf{y}' = \mathbf{f}_2(\hat{\mathbf{X}}, \mathbf{y})$ *from* $t = \tau$ *to* $t = \tau + \theta$ *by the method like* (2.1) *to get* $\mathbf{Y} \approx \mathbf{y}(\tau + \theta)$;

3. *Integrate* $\mathbf{x}' = \mathbf{f}_1(\mathbf{x}, \mathbf{Y})$ *from* $t = \tau + \theta/2$ *to* $t = \tau + \theta$ *to get* $\mathbf{X} \approx \mathbf{x}(\tau + \theta)$.

3. ENHANCE THE EFFICIENCY OF SECOND-ORDER REFLEXIVE UPDATING FORMULAS BY PALINDROMIC COMPOSITIONS

A consistent and reflexive formula has *at least second-order convergence* [9–11, 17] and has other properties which allow efficient constructions of higher order approximations. Assume now $\mathbf{g} \approx \mathbf{y}(\tau)$. By *palindromically composing* the existing reflexive updating formula $\mathbf{Q}(\cdot, \cdot)$ to obtain higher order methods we mean that with appropriately chosen integer m and scalar δ_j's,

$$\mathbf{Q}(\delta_m\theta, \mathbf{Q}(\delta_{m-1}\theta, \mathbf{Q}(..., \mathbf{Q}(\delta_1\theta, \mathbf{g}), ...))) \quad (3.1)$$

approximates $\mathbf{y}(\tau + \theta)$ (much) more accurately than $\mathbf{Q}(\theta, \mathbf{g})$ does, where $\delta_i = \delta_{m-i+1}$ for $i = 1, 2, ..., m$. The reader is referred to Li [17] for a short history of this. Composition schemes (3.1) that work for any reflexive updating formula unify previous work on at least three seemingly unrelated problems:

1. Symplectic integrators for separable Hamiltonian systems, intensively studied by Ruth [23], Forest and Ruth [3], Yoshida [32], McLachlan [18], and many others;

2. Composition schemes based on the implicit midpoint rule due to Sanz-Serna and Abia [25], de Frutos and Sanz-Serna [1];

3. Decompositions of exponential operators, mostly due to Suzuki [27].

Kahan and Li [13] give coefficients δ_j's for orders as high as 10; see http://www.netlib.org/ode/composition.txt. However, what order one should use depends solely on a particular application and generally it is hard to tell. For the discretized KdV equations that we will be solving, we found fourth-order schemes are good enough, partly because errors committed by spatial discretizations make it unnecessary to solve the discretized KdV equations with higher order schemes.

We call (3.1) an *m-stage scheme.* Notation slodrJ stands for an *I*-stage order *J* schemes. (s1odr2 is the formula **Q** itself.) In the sequel, two schemes will be tested on the discretized KdV equations:

1. s3odr4: $m = 3$ and $\delta_1 = \delta_3 = 1/(2 - \sqrt[3]{2})$, $\delta_2 = -\sqrt[3]{2}/(2 - \sqrt[3]{2}) < 0$.

2. s5odr4: $m = 5$ and $\delta_1 = \delta_2 = \delta_4 = \delta_5 = 1/(4 - \sqrt[3]{4})$, $\delta_3 = -\sqrt[3]{4}/(4 - \sqrt[3]{4})$.

Another efficient way to raise the order of reflexive methods is via extrapolation (see Kahan [11] and Hairer, Nørsett, and Wanner [9]).

We now briefly comment on some stability issues associated with the explicitly function-dependent construction (2.1). Notice that (2.1), if applied to linear differential systems, yields the implicit midpoint rule and the trapezoidal rule, based on which a natural linear stability theory may be given for (2.1) and palindromic compositions (3.1). This is done in Li [17]. The conclusion is that, although (2.1) is *A*-stable, palindromic compositions (3.1) are not as long as there are negative δ_j's. As a matter of fact, both linear stability regions for s3odr4 and s5odr4 have a hole[3] in the left half plane. This, of course, coincides with de Frutos and Sanz-Serna's [1] conclusion which is exactly for the implicit midpoint rule. Similarly, we may present a phase error analysis, but it will be similar to de Frutos and Sanz-Serna's analysis as well. We shall omit the details.

[3] The hole for s5odr4 is much smaller and further away to the left from the origin than the one for s3odr4. So s5odr4 could be considered more stable.

4. NUMERICAL TESTS ON THE KORTEWEG–de VRIES EQUATION

We are interested in integrating systems of ordinary differential equations arising from the spatial discretizations of the well-known Korteweg–de Vries (KdV) equation with *smooth* solutions. Two types of spatial discretization will be considered:

1. Finite differences or finite elements. For the purpose of illustration of our idea only, we consider here the space discretization suggested by Sanz-Serna and Christie [26].

2. Pseudospectral methods.

Two kinds of reflexive formulas will be compared to see how well they solve the discretized equations. One family comes from our own methods in Section 2 for quadratic differential equations; the other comes from the implicit midpoint rule explored by de Frutos and Sanz-Serna [1]. They explained briefly why fourth-order explicit Runge–Kutta methods and the popular backward differentiation formulas may be unsuitable for wave problems like the KdV equation. This is also the reason we compare our method here against the implicit midpoint rule studied in [1].

The KdV equation was first proposed in Korteweg and de Vries [14] to describe long waves in water of relatively shallow depth; see also Whitham [31]. It takes the form

$$u_t + 6uu_x + u_{xxx} = 0, \quad (4.1)$$

where $u = u(x, t)$, and subscripts $\cdot_x$ and $\cdot_x$ denote partial derivatives. Later it was discovered that the equation arises in a number of other physical phenomena, e.g., ion-acoustic waves in plasma physics, anharmonic lattices, longitudinal dispersive waves in elastic rods, and pressure waves in liquid gas bubble mixtures.

We shall report both accuracy tests and long time integrations for one soliton solution and collisions of two solitons. The KdV equation (4.1) on the infinite interval $-\infty < x < +\infty$ possesses

One-soliton solutions,

$$u(x, t) = 2k^2 \operatorname{sech}^2(kx - 4k^2t - \eta)$$

for constants k and η; see Taha and Ablowitz [29]. We have chosen a solution with $k = 1$ and $\eta = 0$:

$$u(x, t) = 2 \operatorname{sech}^2(x - 4t) \quad \text{for } -\infty < x < +\infty. \quad (4.2)$$

It has peak amplitude 2 and velocity 4. It was the most difficult one-soliton solution considered in Nouri and Sloan [21] which compared pseudospectral methods for the KdV equation. It also was adopted as a test example in [1].

Two-soliton solutions,

$$u(x,t) = 2(\ln f)_{xx}, \tag{4.3}$$

where $f = 1 + e^{\eta_1} + e^{\eta_2} + ((k_1 - k_2)/(k_1 + k_2))^2\, e^{\eta_1+\eta_2}$, $\eta_i = k_i x - k_i^3 t + \eta_i^{(0)}$ for $i = 1, 2$. We shall test two sets of parameters, as in [29],

$$k_1 = 1, \quad k_2 = \sqrt{2}, \quad \eta_1^{(0)} = 0, \quad \eta_2^{(0)} = 2\sqrt{2}; \tag{4.4}$$

$$k_1 = 1, \quad k_2 = \sqrt{5}, \quad \eta_1^{(0)} = 0, \quad \eta_2^{(0)} = 10.73. \tag{4.5}$$

Since these $u(x, t)$ approaches zero exponentially as $|x|$ increases, for short time integrations practical purposes are served by limiting x to the space interval $\alpha = -20 \le x \le 20 = \beta$ and set $u \equiv 0$ for $x < \alpha$ or $x > \beta$; in such cases we check the accuracies of computed solutions against (4.2) or (4.3) for time interval $0 \le t \le 2$. Notice that such accuracy checking only makes sense for a short period of time due to the limited space interval considered. We also performed long time integration on the limited space interval. It appears our methods enjoy a remarkable long time stability for the KdV equations.

4.1. Finite Element Spatial Discretization

Sanz-Serna and Christie [26] proposed a fourth-order modified Galerkin space discretization: Partition the interval $[\alpha, \beta]$ uniformly by grid points

$$x_j = \alpha + jh \quad \text{for } j = 0, 1, ..., N, \tag{4.6}$$

where $h = (\beta - \alpha)/N$ and N is a positive integer, and let $v_j(t)$ be approximations to $u(x_j, t)$. Then $v_j(t)$ satisfies the system of ordinary differential equations

$$\begin{aligned} &\frac{1}{120}\dot v_{j-2} + \frac{26}{120}\dot v_{j-1} + \frac{66}{120}\dot v_j + \frac{26}{120}\dot v_{j+1} + \frac{1}{120}\dot v_{j+2} \\ &\quad -\frac{1}{8h}v_{j-2}^2 - \frac{10}{8h}v_{j-1}^2 + \frac{10}{8h}v_{j+2}^2 + \frac{1}{8h}v_{j+2}^2 \\ &\quad -\frac{2}{2h^3}v_{j-2} + \frac{2}{2h^3}v_{j-1} - \frac{2}{2h^3}v_{j+1} + \frac{1}{2h^3}v_{j+2} = 0, \end{aligned} \tag{4.7}$$

for $j = 0, 1, ..., N$. Two different boundary treatment were considered:

$v_{-2} \equiv v_{-1} \equiv v_{N+1} \equiv v_{N+2} \equiv 0$. This makes it possible for us to do a brief comparison with [1]; (4.8)

periodic conditions: $v_j = v_{N+j}$ for $j = ..., -2, -1, 0, 1, 2, ...$. This makes long time integration possible. (4.9)

Compactly, this system can be written as

$$M\frac{d\mathbf{v}}{dt} = \mathbf{f}(\mathbf{v}),$$

where M is a $(N + 1) \times (N + 1)$ or $N \times N$ positive definite matrix, depending on which one of (4.8) and (4.9) is used, $\mathbf{v}(t)$ is the $(N + 1)$- or N-dimensional vector-valued function whose jth entry is $v_{j-1}(t)$, and $\mathbf{f}(\mathbf{v})$ is a vector-valued function of $\mathbf{v}$. Since $\mathbf{f}(\mathbf{v})$ turns out to be at most quadratic in $\mathbf{v}$, it has a second-order reflexive updating formula as derived in Section 2 for numerically solving the system (4.7): *Given the approximation* $\mathbf{v}$ *to* $\mathbf{v}(\tau)$, *an approximation* $\mathbf{V}$ *to* $\mathbf{v}(\tau + \theta)$ *can be obtained by solving the linear system*

$$\left(M - \frac{\theta}{2}J(\mathbf{v})\right)(\mathbf{V} - \mathbf{v}) = \theta\mathbf{f}(\mathbf{v}), \tag{4.10}$$

where $J(\mathbf{v})$ *is the Jacobian matrix of* $\mathbf{f}(\cdot)$ *evaluated at* $\mathbf{v}$. Notice that this linear system is easy to solve because its coefficient matrix is pentadiagonal. The method has an advantage over the implicit midpoint rule used in [1] in that there is no system of nonlinear equations to solve at each time step, and no loss of numerical accuracy, as shall be clear soon. (4.10) produces a second-order reflexive updating formula that can be composed or extrapolated to get higher order schemes. There is a limit to the orders worth considering because no reason exists to solve the system (4.7) much more accurately than is compatible with the error of the fourth-order spatial discretization.

4.1.1. *Tests for One-Soliton Solution*

In what follows, we set $h = 0.1$ as in [1] and adopt the boundary values[4] (4.8) in order to compare our results with those reported therein; and thus we have a 401-dimensional system (4.7). With this meshsize, the maximum norm error at the final time $t = 2$ in the solution to (4.7) as an approximation to $u(x, 2)$ in (4.2) has order of magnitude about 10^{-5}.

In Table I, "*Errors*" refer to maximum norm errors at $t = 2$ of the numerical solutions as an approximation to $u(x, 2)$. Such "Errors" may not reflect the distances to the true solution of the system (4.7), especially when "Errors" have order of magnitude about 10^{-5}. We shall return to this later. The numbers related to the implicit midpoint rule are due to de Frutos and Sanz-Serna [1]. The "*Extrapolation*" column contains errors for solutions obtained as

[4] We found that numerical results are of comparable accuracy if solved with the periodic boundary condition (4.9). For this reason, we shall not include numerical results with (4.9) in our accuracy comparisons.

TABLE I

Errors

θ	No. of steps	Mid-point [1]	s3odr4 [1] by mid-point	Scheme (4.10)	s3odr4 by (4.10)	s5odr4 by (4.10)	Extrapolation
5e-2	40		1.6e-2	1.9e-1	1.9e-2	2.1e-3	7.1e-3
2.5e-2	80	3.1e-2	1.1e-3	4.7e-2	1.1e-3	7.2e-5	4.8e-4
1.25e-2	160	7.7e-3	3.4e-5	1.2e-2	6.6e-5		9.4e-5
6.25e-3	320	1.9e-3		2.9e-3			

follows: for each θ, run the second-order method (4.10) with step-size θ first and then run it with step-size $\theta/2$ and finally extrapolate the two solutions to fourth order. de Frutos and Sanz-Serna [1] did not test s5odr4; they might not have been aware of it at that time. We do not include numerical results for the very small step-sizes θ included in [1] for two reasons:

1. At very small step-sizes little difference in cost between solving a nonlinear system and a linear system; They both take one iteration.

2. When step-sizes are sufficiently small, high order explicit schemes might do better.

A primitive implementation of (4.10) factorizes its coefficient matrix every time it is called. Better implementations are conceivable. In any event, even with this primitive implementation, this new method clearly beats schemes based on the implicit midpoint rule that was used in [1], where one matrix factorization was carried out every time the implicit midpoint rule is called.

Table I also shows that s5odr4 is substantially more accurate than s3odr4 at the same step-size, although both are of order 4. Apparently the two extra stages in s5odr4 allow it to take larger steps than s3odr4 for achieving errors of similar magnitude. To get the error below 10^{-4}, s5odr4 calls upon (4.10) 480 times, s5odr4 only 400 times, which takes about 20% less time. To compare the effectiveness of the schemes, we plot their errors versus effort in Fig. 1. Arithmetic operations called *flops* are counted as follows. The coefficient matrices here are pentadiagonal; LU decomposition[5] of a pentadiagonal matrix with dimension n costs about $18n$ flops and solving a linear system after decomposition costs $13n$ flops; each **f** evaluation costs $11n$ and each coefficient matrix evaluation $10n$. With those in mind, together with the information in [1, Table I], *distances* to $u(x, 2)$ in (4.2) versus *numbers of flops* is plotted in Fig. 1, where anything related to *midpoint* is figured out based on information presented in de Frutos and Sanz-Serna [1]. Roughly speaking, for the step-sizes considered, to get about the same accuracy, our method (4.10) is about 1.5 times faster than the implicit midpoint rule; s3odr4 based on (4.10) is roughly 2.3 to 1.5 times faster than s3odr4 based on the implicit midpoint rule. The speed difference diminishes as step-sizes get smaller because the nonlinear systems involved in the implicit midpoint rule require fewer iterations to solve. On the other hand, the extrapolation method may be one of the most efficient ways to go.

As we remarked, comparing numerical solutions of (4.7) against the true solution $u(x, t)$ of the KdV equation may lead us to misinterpret the effectiveness of each scheme because the error introduced by spatial discretization swoops the errors suffered by the higher order schemes when they solve the discretized system (4.7). To overcome this, we have computed an "*exact*" solution to the system (4.7) by using a step-size so small that the "exact" solution comes within at worst about 10^{-7} of the true solution to (4.7). With this, we are able to plot Fig. 2. (Schemes based on the implicit midpoint rule are not included.) Figure 2 shows clearly that the extrapolation method is very competitive for this problem. s5odr4 is less efficient than extrapolation at the beginning and then gets better as step-sizes diminish.

Figure 3 shows favorable linear error growth as functions of t for integration up to $t = 3$.

We also run composition schemes based (4.10) with the periodic boundary condition (4.9) for t as large as 80 without encountering any stability difficulties. It appears the integration can go much longer. The numerical results show a soliton moving to the right periodically in the sense that it disappear at $x = \beta = 20$ but reappear at $x = \alpha = -20$. To save pages, we decide not to say any more than this, but shall report numerical results of long time integra-

[5] A *flop* is defined to be the amount of work of a floating point operation [6, p. 19]. One addition or multiplication of two real numbers is counted as 1 flop; a division is counted as 5 flops (it takes about that long on most commercially significant machines). We consulted [6, pp. 150–151] for flop counts. It turns out the flop counts given there are not accurate enough for our application since matrices here have extremely narrow band width. Our calculation here is based on: LU decomposition for a banded matrix of dimension n and with upper bandwidth q and lower bandwidth p costs about $p(2q + 5)n$ flops (see Algorithm 4.3.1 in [6, pp. 150]); then band forward substitution (column version) costs about $2pn$ flops, and band back substitution (column version) costs about $(2q + 5)n$ flops (see Algorithm 4.3.2 and Algorithm 4.3.3 in [6, pp. 150]).

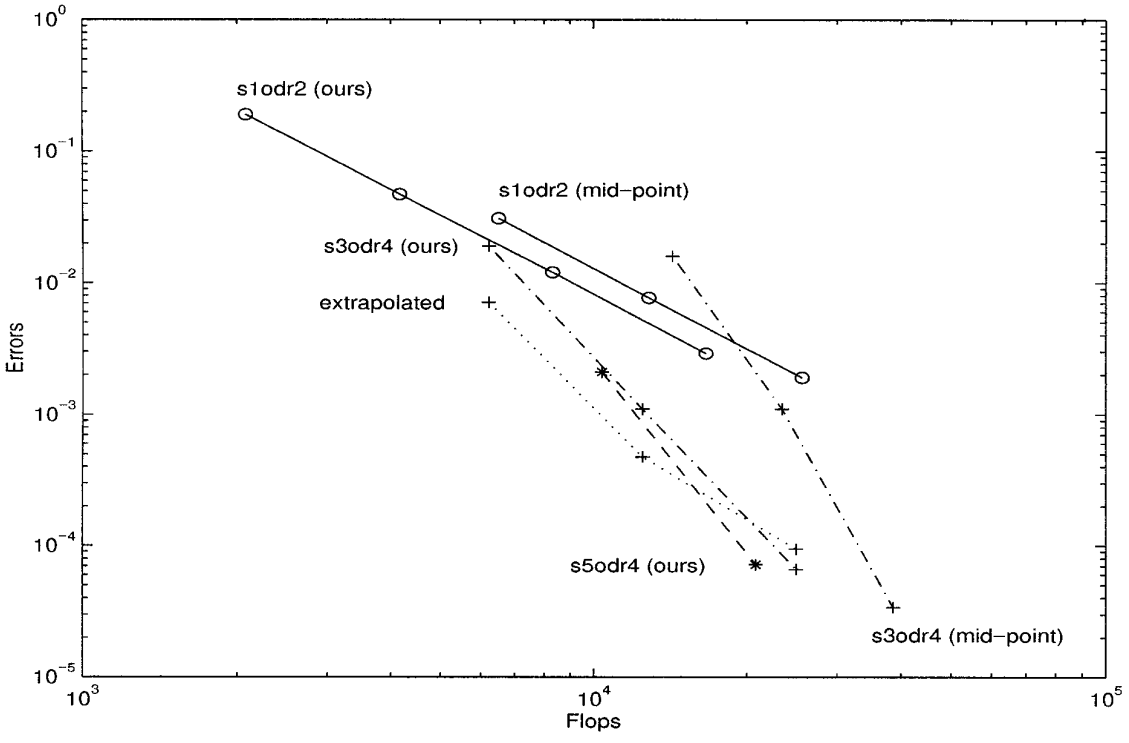

FIG. 1. Distances to $u(x, 2)$ in (4.2) versus costs for solving the KdV equation via finite element spatial discretization.

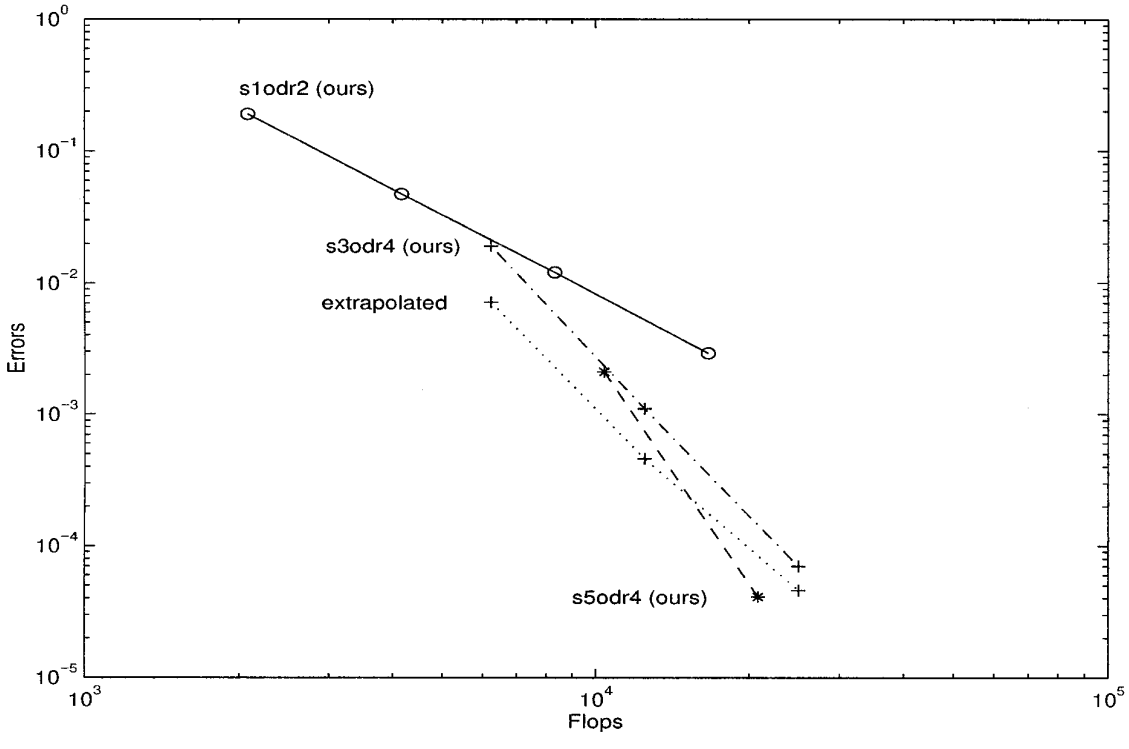

FIG. 2. Distances to the true solution of the discretized system (4.7) versus costs.

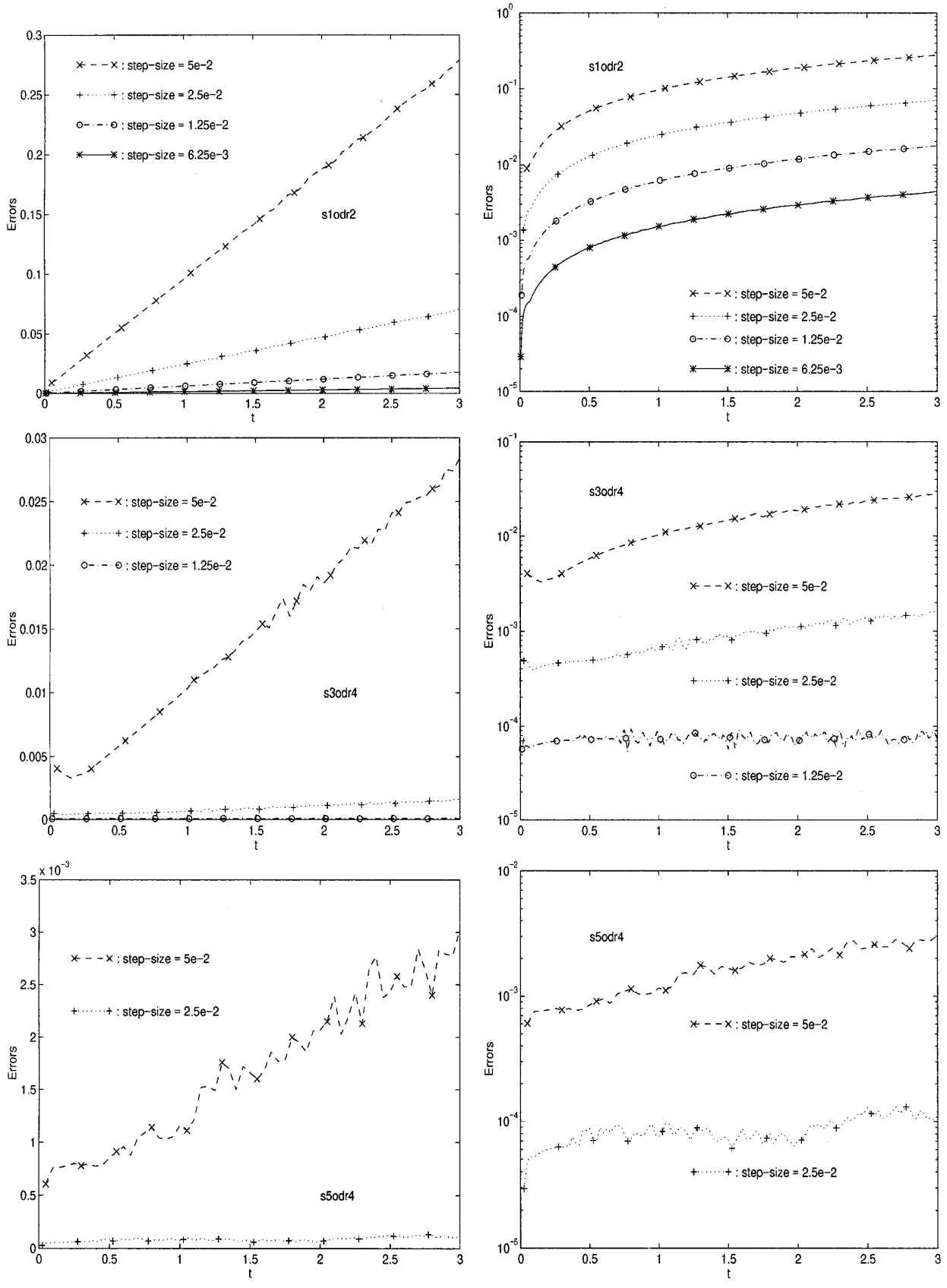

FIG. 3. Temporal changes of errors against one-soliton $u(x, t)$ in (4.2). Compositions are based on (4.10). The errors behave linearly in time t. Each figure on the left and its corresponding one on the right plot the same data but with MATLAB's plot and semilogy, respectively.

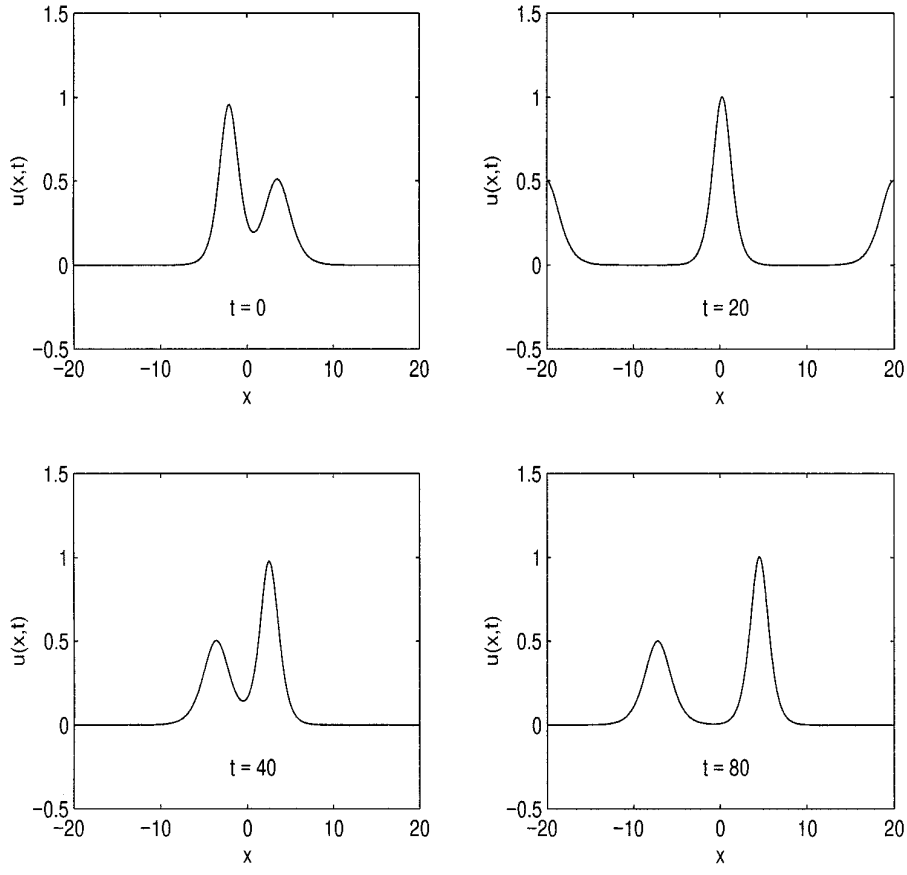

FIG. 4. Long time integration of the spatially discretized KdV equation by finite element method for *collisions of two solitons* with parameters (4.4) (similarly with parameters (4.5)).

tions for *collisions of two solitons* which appear to be more interesting than one-soliton solutions.

4.1.2. *Collisions of Two Solitons*

The accuracy of numerical results against two-soliton (4.3) are always good for t not too big, as expected. So we shall not go into detail in that matter. What we are interested the most is to see if the newly proposed methods run into any stability difficulties for long time integrations. For this purpose, we run schemes `s1odr2`, `s3odr4`, and `s5odr4` based on (4.10) with the periodic boundary condition (4.9) for t up to 80 with $\theta = 0.1$. No stability difficulties have occurred. Figure 4 samples numerical solutions that could be obtained by any one of the three schemes at four different times.

4.2. Discretization by the Pseudospectral Method

The pseudospectral method is an alternative to finite differences and finite elements for certain classes of partial differential equations. Its applicability is restricted in comparison to finite differences or finite elements, but it works much better when it works. Kreiss and Oliger [15] first introduced the pseudospectral method for hyperbolic equations. Early development of its basic theory can be found in Orszag [22], Fornberg [4], Gottlieb and Orszag [7], and more recently Gottlieb and Turkel [8], Tadmore [28], and Fornberg [5].

Let us briefly describe the pseudospectral method. The basic idea is to interpolate a periodic function $g(x)$ by trignometric functions. Here is one way to do it: Suppose $g(x)$ is periodic on the interval $[\alpha, \beta]$, so $g(\alpha) = g(\beta)$. Let N be a positive integer, and let $x_j = \alpha + j(\beta - \alpha)/N$ for $j = 0, 1, ..., N$. Then the discrete Fourier transformation of the sequence of values $g(x_j)$ is given by a sequence

$$\hat{g}(p) = \sum_{j=0}^{N-1} g(x_j) e^{-2\pi i j p/N} \quad \text{for } -N/2 \le p < N/2. \ (i = \sqrt{-1}) \tag{4.11}$$

Accordingly, the inverse discrete Fourier transformation recovers $g(x_j)$:

$$g(x_j) = \frac{1}{N} \sum_{p=-N/2}^{N/2-1} \hat{g}(p) e^{2\pi i p(x_j-\alpha)/\beta-\alpha)} \quad \text{for } j = 0, 1, ..., N. \tag{4.12}$$

For this reason, the trigonometric interpolation $P_N g$ is then given by

$$\begin{aligned} P_N g(x) &= \frac{1}{N} \sum_{p=-N/2}^{N/2-1} \hat{g}(p) e^{2\pi i p(x-\alpha)/(\beta-\alpha)} \\ &= \frac{1}{N} \sum_{j=0}^{N-1} g(x_j) \sum_{p=-N/2}^{N/2-1} e^{2\pi i p[(x-\alpha)/(\beta-\alpha)-j/N]}. \end{aligned} \tag{4.13}$$

It is easy to see that $P_N g(x_j) = g(x_j)$. The derivatives of $g(x)$ can be approximated by the derivatives of $P_N g(x)$:

$$\begin{aligned} (P_N g)^{(\nu)}(x_j) &= \frac{1}{N} \sum_{j=0}^{N-1} g(x_j) \sum_{p_{=-N/2}}^{N/2-1} \left(\frac{2\pi i p}{\beta-\alpha}\right)^{\nu} e^{2\pi i p[(x_j-\alpha)/(\beta-\alpha)-j/N]}, \\ &= \frac{1}{N} \sum_{p=-N/2}^{N/2-1} e^{2\pi i p(x_j-\alpha)/(\beta-\alpha)} \left(\frac{2\pi i p}{\beta-\alpha}\right)^{\nu} \sum_{j=0}^{N-1} g(x_j) e^{-2\pi i j p/N}. \end{aligned} \tag{4.14}$$

Set $\omega = e^{-2\pi i/N}$ and define an $N \times N$ matrix F whose (k, ℓ) entry is $\omega^{(k-N/2-1)*(\ell-1)}$. Denote $\mathbf{g} = (g(x_0), g(x_1), ..., g(x_{N-1}))^{\mathrm{T}}$ and $\hat{\mathbf{g}} = (\hat{g}(-N/2), \hat{g}(-N/2+1), ..., \hat{g}(N/2-1))^{\mathrm{T}}$. Equations (4.11) and (4.12) read as[6]

$$\hat{\mathbf{g}} = F\mathbf{g}, \quad \mathbf{g} = F^{-1}\hat{\mathbf{g}},$$

and the derivative vector $(g^{(\nu)}(x_0), g^{(\nu)}(x_1), ..., g^{(\nu)}(x_{N-1}))^{\mathrm{T}}$ is approximated by (see (4.14))

$$F^{-1}\Lambda^{\nu} F\mathbf{g},$$

where

$$\Lambda = \operatorname{diag}\left(-\frac{2\pi i}{\beta-\alpha}\frac{N}{2}, -\frac{2\pi i}{\beta-\alpha}\left(\frac{N}{2}-1\right), ..., -1, 0, 1, ..., \frac{2\pi i}{\beta-\alpha}\left(\frac{N}{2}-1\right)\right).$$

Let us go back to the KdV equation. For the case we mentioned above, $\alpha = -L$, $\beta = L$, and $L = 20$. Although the functions involved are not really periodic, they can be approximated this way. Again, we work with the spatial grid $\{x_j\}$ as defined by (4.6); but now $v_0(t) \equiv v_N(t)$ and the vector-valued function $\mathbf{v}(t)$ is of length N always. The spatial discretization by the pseudospectral method can be written as

$$\frac{d\mathbf{v}}{dt} + 6F^{-1}\Lambda F\frac{1}{2}\mathbf{v}^2 + F^{-1}\Lambda^3 F\mathbf{v} = 0, \tag{4.15}$$

where $\mathbf{v}^2$ should be interpreted entry-wise. This form again enables us to derive a reflexive method with no systems of nonlinear equations involved. Given the approximation $\mathbf{v}$ to $\mathbf{v}(\tau)$, an approximation $\mathbf{V}$ to $\mathbf{v}(\tau+\theta)$ can be obtained by solving a linear system,

$$\left(I - \frac{\theta}{2}J(\mathbf{v})\right)(\mathbf{V}-\mathbf{v}) = \theta\mathbf{f}(\mathbf{v}), \tag{4.16}$$

where $\mathbf{f}(\mathbf{v}) = -(6F^{-1}\Lambda F\frac{1}{2}\mathbf{v}^2 + F^{-1}\Lambda^3 F\mathbf{v})$, and $J(\mathbf{v})$ is the Jacobian matrix of $\mathbf{f}(\cdot)$ evaluated at $\mathbf{v}$,

$$J(\mathbf{v}) = -6F^{-1}\Lambda\ \operatorname{diag}(\mathbf{v}) - F^{-1}\Lambda^3 F.$$

Unfortunately, it is a full matrix, so is the coefficient matrix in (4.16). Premultiplying the two sides of (4.16) by F yields

$$(F + \theta 3\Lambda\mathbf{F}\operatorname{diag}(\mathbf{v}) + \theta\tfrac{1}{2}\Lambda^3 F)(\mathbf{V}-\mathbf{v}) = -\theta(3\Lambda F\mathbf{v}^2 + \Lambda^3 F\mathbf{v}). \tag{4.17}$$

Applying a direct solver requires work $O(N^3)$ at each time step, which is too expensive; but because of its special form there are iterative methods which solve this linear system cheaply. Two particularly simple-minded iteration methods can be obtained from the rearrangements

$$\begin{aligned} (I + \theta\tfrac{1}{2}\Lambda^3)F(\mathbf{V}-\mathbf{v}) = &-\theta 3\Lambda F\operatorname{diag}(\mathbf{v})F^{-1}F(\mathbf{V}-\mathbf{v}) \\ &- \theta(3\Lambda F\mathbf{v}^2 + \Lambda^3 F\mathbf{v}), \end{aligned} \tag{4.18}$$

and

$$\begin{aligned} &(I + \theta 3\Lambda\eta + \theta\tfrac{1}{2}\Lambda^3)F(\mathbf{V}-\mathbf{v}) \\ &\quad = -\theta 3\Lambda F(\operatorname{diag}(\mathbf{v}) - \eta I) \\ &\qquad F^{-1}F(\mathbf{V}-\mathbf{v}) - \theta(3\Lambda F\mathbf{v}^2 + \Lambda^3 F\mathbf{v}), \end{aligned} \tag{4.19}$$

where η is the average of the entries of $\mathbf{v}$. (Notice that the diagonal entries of $F\operatorname{diag}(\mathbf{v})F^{-1}$ are equal to η). How to solve the linear system (4.17) is of independent interest. Later, we will present an implementation using GMRES (see Saad and Schultz [24]).

De Frutos and Sanz-Serna [1] proposed the implicit midpoint rule to solve the system (4.15), and thus had to solve a system of nonlinear equations at each time step. Unfortunately, the Jacobian matrix associated with the system is full, instead of Newton iteration, de Frutos and Sanz-Serna designed a functional iteration which requires about one pair of FFT/IFFT per iteration.

[6] The matrix–vector product $F\mathbf{g}$ can be realized via a fast Fourier transformation (FFT) and the product $F^{-1}\hat{\mathbf{g}}$ via inverse fast Fourier transformation (IFFT). In the language of MATLAB, they can be realized by fftshift(fft($\mathbf{g}$)) and ifft(fftshift($\hat{\mathbf{g}}$)), respectively.

Our later implementation of GMRES shows our updating formula is cheaper for θ not too small. When θ gets very small GMRES provides little help, because simple iterations based on either rearrangement (4.18) or (4.19) are then good enough for quick convergence.

4.3. Numerical Results with GMRES

Let us briefly review GMRES, which stands for *generalized minimal residual algorithms*, for solving nonsymmetric linear systems. In particular, we are interested in using GMRES(m) with no restart to solve a linear system $A\mathbf{x} = \mathbf{b}$.

ALGORITHM GMRES(m).

1. Choose an initial guess $\mathbf{x}_0$, compute[7] $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\beta = \|\mathbf{r}_0\|$, and $\mathbf{q}_1 = \mathbf{r}_0/\beta$;
2. For $j = 1, 2, ..., m$ do;
   $\hat{\mathbf{q}}_{j+1} = A\mathbf{q}_j$;
   For $i = 1, 2, ..., j$ do:
   $h_{ij} = \mathbf{q}_i^* \hat{\mathbf{q}}_{j+1}$; $\hat{\mathbf{q}}_{j+1} = \hat{\mathbf{q}}_{j+1} - h_{ij}\mathbf{q}_i$;
   enddo;
   $h_{j+1j} = \|\hat{\mathbf{q}}_{j+1}\|$; $\mathbf{q}_{j+1} = \hat{\mathbf{q}}_{j+1}/h_{j+1j}$;
   enddo;
3. Solve for $\mathbf{y}_m$ which minimize $\|\beta\mathbf{e}_1 - H_m\mathbf{y}\|$, where $H_m = (h_{ij})$ is $(m + 1) \times m$;
4. Take $\mathbf{x}_m = \mathbf{x}_0 + Q_m\mathbf{y}_m$ as an approximation solution to the system $A\mathbf{x} = \mathbf{b}$, where $Q_m = (\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_m)$.

GMRES(m) [24] in its most general form has a *restart* mechanism, namely after Step 4 residual $\mathbf{r}_m = \mathbf{b} - A\mathbf{x}_m$ is computed and checked; if a prescribed tolerance is satisfied then stop, else set $\mathbf{x}_0 = \mathbf{x}_m$ and $\mathbf{q}_1 = \mathbf{r}_m/\|\mathbf{r}_m\|$ and go back to Step 2. In our case, this *restart* mechanism will not be considered.

Let us now explain two improvements to GMRES(m) in our particular case. (It may apply to some other cases as well.) Notice that GMRES(m) requires $m + 1$ matrix–vector multiplications; and the last matrix–vector multiplication to get $\hat{\mathbf{q}}_{m+1}$ is not fully used in the sense that only h_{m+1m} is incorporated to get $\mathbf{y}_m$. It follows from Step 4 that

$$A\mathbf{x}_m = A\mathbf{x}_0 + VQ_m\mathbf{y}_m = A\mathbf{x}_0 + Q_{m+1}H_m\mathbf{y}_m.$$

Subtracting $\mathbf{b}$ from these equations gives

$$\mathbf{r}_m = \mathbf{r}_0 = Q_{m+1}H_m\mathbf{y}_m. \tag{4.20}$$

Computing $\mathbf{r}_m$ this way costs about $(m + 1)8m + N8(m + 1) + 2N = (8m + 10)N + 8m(m + 1)$ flops.[8] On the other hand, the cost of computing directly $\mathbf{r}_m = \mathbf{b} - A\mathbf{x}_m$ depends on that of computing $A\mathbf{x}_m$. Before we count the number of flops for doing a matrix–vector multiplication, let us reformulate the system (4.17) into the form that we will actually use in our implementation. Set

$$D_1 = (I + \theta\tfrac{1}{2}\Lambda^3)^{-1}\theta 3\Lambda, \quad D_2 = (I + \theta\tfrac{1}{2}\Lambda^3)^{-1}\theta\Lambda^3.$$

D_1 and D_2 should be computed prior to entering GMRES(m). Then the system takes the form

$$(I + \Omega)\mathbf{x} = \mathbf{b}, \tag{4.21}$$

where $\mathbf{b} = -D_1F\mathbf{v}^2 - D_2F\mathbf{v}$ is precomputed, and $\Omega = D_1F\,\mathrm{diag}(\mathbf{v})F^{-1}$ is kept in this factored form, and $\mathbf{x} = F(\mathbf{V} - \mathbf{v})$ is to be found. Counting as Demmel [2] did, we find FFT in complex arithmetic costs about $5N\log_2 N$ and IFFT costs $2N$ more. So a matrix–vector multiplication in our case costs $2 \times 5N\log_2 N + 2N + 2 \times 6N + 2N = 16N + 10N\log_2 N$ flops; therefore the straightforward way of computing a residual in our case costs $18N + 10N\log_2 N$. The flop ratio

$$\frac{18N + 10N\log_2 N}{(8m + 10)N + 8m(m + 1)}$$

is plotted for $N = 128$ and $N = 256$ with m running from 1 to 13 in Fig. 5. The picture shows that it is worthwhile to use (4.20) for $m \le 9$ when $N = 128$ and for $m \le 10$ when $N = 256$. In our tests, m does satisfy these bounds.

Our second improvement to GMRES is again to utilize the last $\mathbf{q}$-vector $\mathbf{q}_{m+1}$ to improve $\mathbf{x}_m$. The idea is that simple iterations based on either (4.18) or (4.19) will improve a given approximation for reasonable θ; and it turns out for step-sizes we are interested in these simple iterations will reduce residuals by at least about $\frac{1}{5}$ and much more when step-size gets smaller. We observed that (4.19) is a little bit better than (4.18). So what we do is: separate the diagonal and off-diagonal entries of $I + \Omega$ as $I + \Omega = D + B$; it can be seen that $D = I + \eta D_1$, where η is the average of the entries of $\mathbf{v}$; rewrite Eq. (4.21) into $D\mathbf{x} = -B\mathbf{x} + \mathbf{b}$; define new improved approximation $\mathbf{x}_{\text{new}}$ by $\mathbf{x}_{\text{new}} = \mathbf{x}_m + (I + \eta D_1)^{-1}\mathbf{r}_m$ since

$$D\mathbf{x}_{\text{new}} = -B\mathbf{x}_m + b \Rightarrow D(\mathbf{x}_{\text{new}} - \mathbf{x}_m) = \mathbf{b} - (D + B)\mathbf{x}_m = \mathbf{r}_m.$$

Lastly, we point out our initial guess $\mathbf{x}_0$ to the system (4.21) is gotten either by quadratic interpolations or by the leap-frog-like method,[9] depending on which is more

[7] $\|\mathbf{x}\|$ here is the Euclidean length of vector $\mathbf{x}$. The superscript * denotes complex conjugate transpose.

[8] One multiplication of two complex numbers takes 6 flops, and addition/subtraction 2 flops.

[9] The leap-frog method for the system of ordinary differential equations $\mathbf{y}' = \mathbf{f}(\mathbf{y})$ is

$$\mathbf{y}_{n+1} - \mathbf{y}_{n-1} = 2\theta\mathbf{f}(\mathbf{y}_n), \tag{4.22}$$

where $\mathbf{y}_n \approx \mathbf{y}(t_0 + n\theta)$, the true solution at time $t_n = t_0 + n\theta$. In the case when step-size varies, i.e., $t_{n+1} - t_n$ depends on n, one can construct the following second-order scheme,

$$\gamma\mathbf{y}_{n+1} - (\gamma - 1/\gamma)\mathbf{y}_n - (1/\gamma)\mathbf{y}_{n-1} = (t_{n+1} - t_{n-1})\mathbf{f}(\mathbf{y}_n), \tag{4.23}$$

where $\gamma = (t_n - t_{n-1})/(t_{n+1} - t_n)$. In the constant step-size case, $\gamma = 1$ and thus (4.23) degenerates to (4.22).

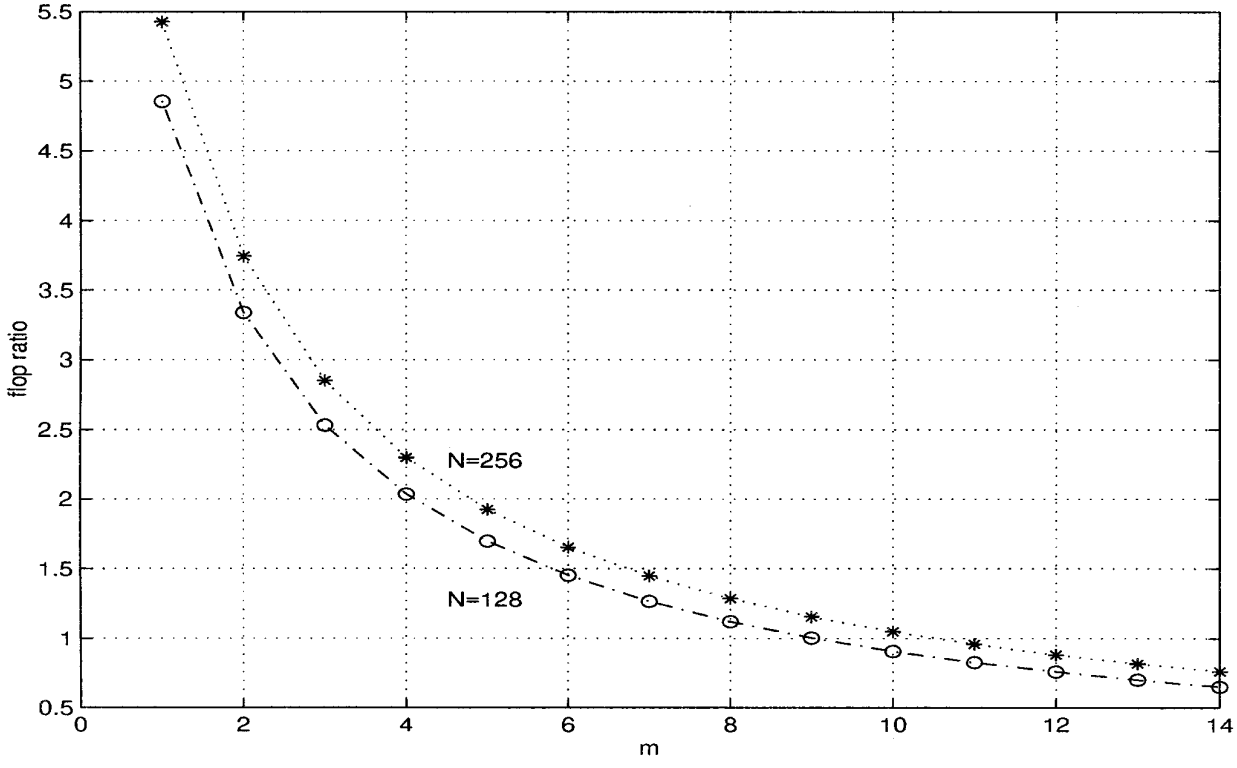

FIG. 5. The ratio $[18N + 10N \log_2 N]/(8m + 10)N + 8m(m + 1)]$ as m varies.

convenient to invoke at various points in the program. Both guesses provide approximations with errors of order $O(\theta^3)$. (It is conceivable that with quadratic interpolations both $\mathbf{x}_0$ and $F^{-1}\mathbf{x}_0$ could be available without doing any FFT/IFFT and thus the first residual vector could be computed using one IFFT in addition to some $O(N)$ flops.)

Roughly speaking, using GMRES(m) to solve the system (4.21) costs about $m + 2$ pairs of FFT/IFFT operations. As we just commented, if quadratic interpolations were always used for initial guesses, cost may be reduced to $m + 1.5$ pairs of FFT/IFFT operations. For the moment, we count costs as $m + 2$ pairs of FFT/IFFT operations for using GMRES(m) to solve the system (4.21).

4.3.1. *Tests for One-Soliton Solution*

Tables II and III list our numerical results. The residual columns refer to the maximums among all 2-norms of residuals for $\mathbf{x}_{\text{new}}$ for all linear systems involved in a particular scheme. The values m are for GMRES(m).

Figure 6 plots distances to $u(x, 2)$ in (4.2) versus costs in the numbers of pairs of FFT/IFFT operations for the case $N = 128$. Information regarding the implicit midpoint rule is due to [1, Table II]. Schemes based on the newly proposed method are more efficient than schemes based on the implicit midpoint rule at larger step-sizes and gradually the speed difference diminishes as step-sizes decrease. Our second-order scheme starts by almost twice as fast as

TABLE II

Errors for $N = 128$

		Scheme (4.16)			s3odr4 by (4.16)			s5odr4 by (4.16)		
θ	No. of steps	Error	m	Residual	Error	m	Residual	Error	m	Residual
1.6e-2	125	2e-2	1	4e-4	2e-4	4	8e-6	7e-6	3	2e-7
8.0e-3	250	5e-3	1	3e-5	8e-6	3	2e-6	2e-6	2	1e-7
4.0e-3	500	1e-3	0	4e-5	3e-6	2	4e-7			
2.0e-3	1000	3e-4	0	3e-6						
1.0e3	2000	8e-5	0	3e-7						

TABLE III

Errors for $N = 256$

		Scheme (4.16)			s3odr4 by (4.16)			s5odr4 by (4.16)		
θ	No. of steps	Error	m	Residual	Error	m	Residual	Error	m	Residual
1.6e-2	125	2e-2	1	9e-04	2e-04	4	2e-05	9e-06	3	4e-07
8.0e-3	250	5e-3	1	5e-05	1e-05	4	3e-07	4e-07	3	2e-08
4.0e-3	500	1e-3	1	3e-06	9e-07	3	5e-08	2e-08	2	9e-09
2.0e-3	1000	3e-4	1	1e-07	8e-08	3	1e-09	73-10	2	2e-10
1.0e-3	2000	8e-5	0	1e-07	7e-09	2	7e-10	6e-10	1	2e-10
5.0e-4	4000	2e-5	0	8e-09	4e-10	2	2e-11			
2.5e-4	8000	5e-6	0	5e-10	3e-10	1	2e-11			

the implicit midpoint rule and then goes at about the same speed as step-sizes get smaller. Our s3odr4 is 1.5 to 1.2 times faster than s3odr4 based on the implicit midpoint rule. Figure 7 plots distances to $u(x, 2)$ in (4.2) versus costs in the numbers of pairs of FFT/IFFT operations for the case $N = 256$. Information regarding the implicit midpoint rule is due to [1, Table III]. Again, schemes based on the newly proposed method are more efficient than schemes based on the implicit midpoint rule at larger step-sizes. Our schemes become less favorable choices at smaller step-sizes. This is no surprise and due entirely to the fact that we still use GMRES, which becomes less efficient than simple functional iterations as step-sizes get much smaller. For $\theta =$ 2.0e − 3 cand $\theta =$ 1.0e − 3, our second-order scheme is about 1.2 times faster than the implicit midpoint rule. Our s3odr4 is from 2 to 1.2 times faster than s3odr4 based on the implicit midpoint rule for the first three θ's in Table III. Our s5odr4 seems to be the only favorable choice for $\theta =$ 1.6e − 2. For other θ's, our current implementation with GMRES does not as well for reasons adduced above.

Figure 8 shows temporal changes of errors for integration up to $t = 3$. We see that errors grow very slowly. We also see working with the limited space interval $[-20, 20]$

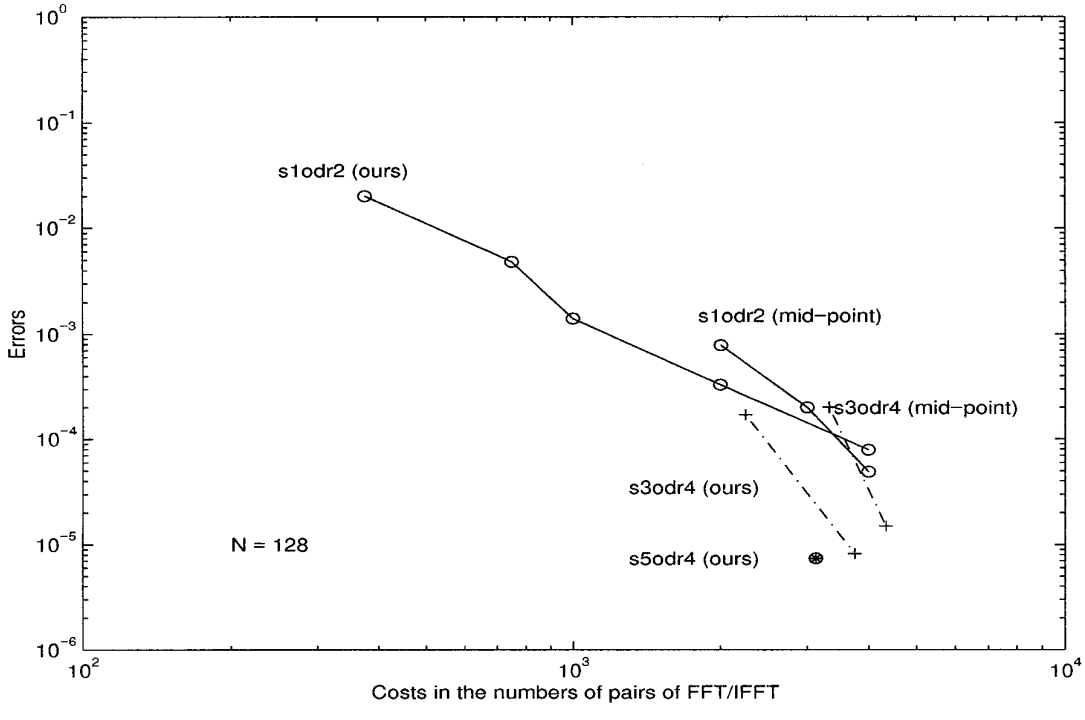

FIG. 6. Distances to $u(x, 2)$ in (4.2) versus costs (in the numbers of pairs of FFT/IFFT).

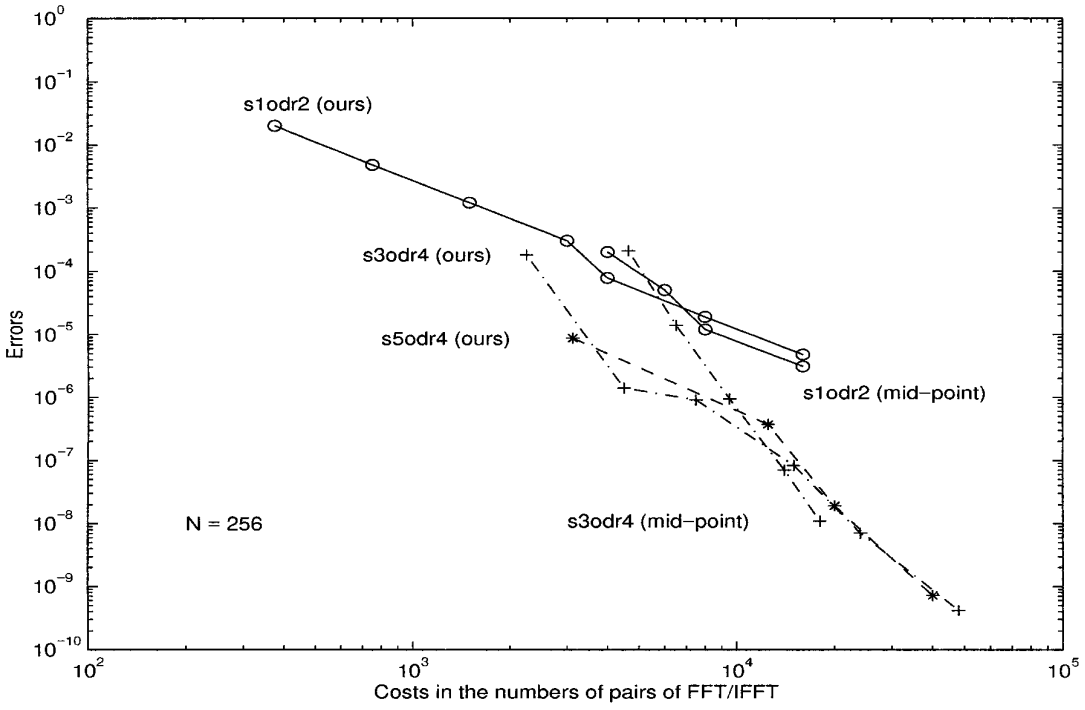

FIG. 7. Distances to $u(x, 2)$ in (4.2) versus costs (in the numbers of pairs of FFT/IFFT).

produces poor approximation for large t to the true one-soliton (4.2) which moves at a constant speed to the right towards infinity. Because of the limited space interval and the periodic boundary condition, numerically we actually see a soliton moving to the right periodically.

It worth noting that the numerical tests in [1] stopped prematurally in their iteratively solving nonlinear equations from the implicit midpoint rule for the case $N = 256$. Such premature stops hurt the numerical accuracy when time step-sizes were small. In fact, for $\theta = 1.0e-3$, s3odr4 in [1] should compute a solution at $t = 2$ with maximum norm error about $O(10^{-10})$, had their iteratively solving nonlinear equations been properly stopped, but the error reported in [1] was $O(10^{-8})$.

4.3.2. *Collisions of Two Solitons*

Long time integrations for *collisions of two solitons* were also conducted. Our methods turn out to work pretty well without running into any stability difficulties. Figure 9 samples numerical solutions that could be obtained by any one of s1odr2, s3odr4, or s5odr4 at four different times.

5. CONCLUSIONS

Through solving the discretized KdV equations, we have presented an unconventional method for solving a special and yet often encountered kind of differential equations. The method requires no nonlinear equations to solve, is of second-order accuracy and, most importantly, *reflexive*. A systematical scheme is proposed to enhance the computational efficiency of such methods. Numerical experiments show that the method is suitable for smooth solutions and significantly faster than the implicit midpoint rule advocated by de Frutos and Sanz-Serna [1]. When high accuracy is required, the enhanced schemes s3odr4 and s5odr4 shall be used. It appears even though both s3odr4 and s5odr4 are of order 4 accuracy and s5odr4 takes two more stages than s3odr4, in terms of computational efficiency s5odr4 may do better. Higher order palindromic composition schemes are not considered here to integrate spatially discretized KdV equations because no reason exists to solve the discretized systems far more accurately than is compatible with the error committed by spatial discretization. Also we looked into a comparable way—*extrapolation*—to increase the order of the method. Our new methods appear to have no difficulties in long time integration for the spatially discretized KdV equations with periodic boundary conditions.

ACKNOWLEDGMENT

The authors thank referees' constructive comments for improving the presentation of their numerical tests on the KdV equation.

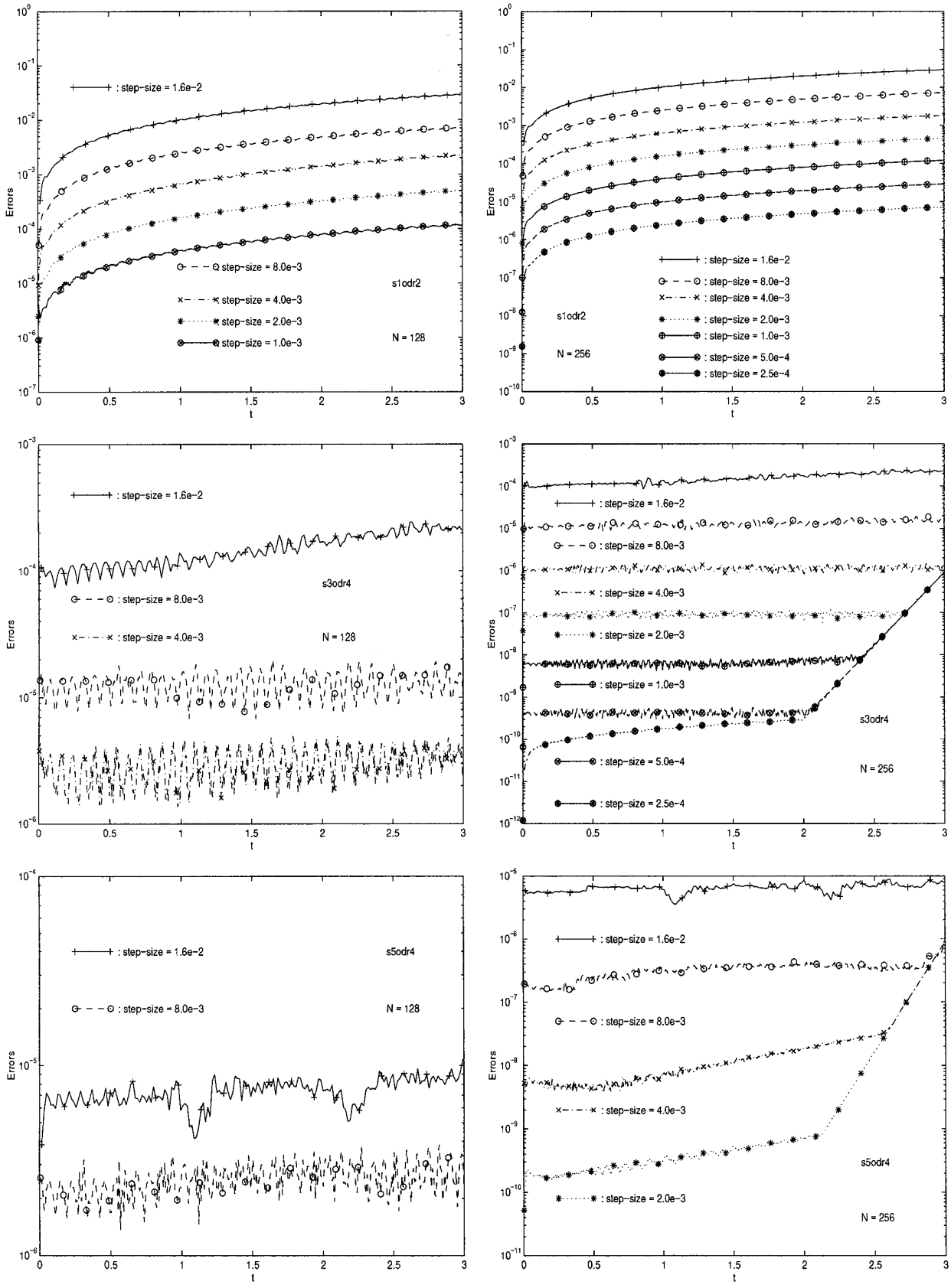

FIG. 8. Temporal changes of errors against one-soliton $u(x, t)$ in (4.2). Compositions are based on (4.16). The errors behave favorably for t not too big, as expected, as t increases. The suddenly rapid error growth for s3odr4 and s5odr4 when $N = 256$ and when t is slightly over 2 is due to the limited space interval $[-20, 20]$ we used; while the exact one-soliton always moves to the right towards infinity.

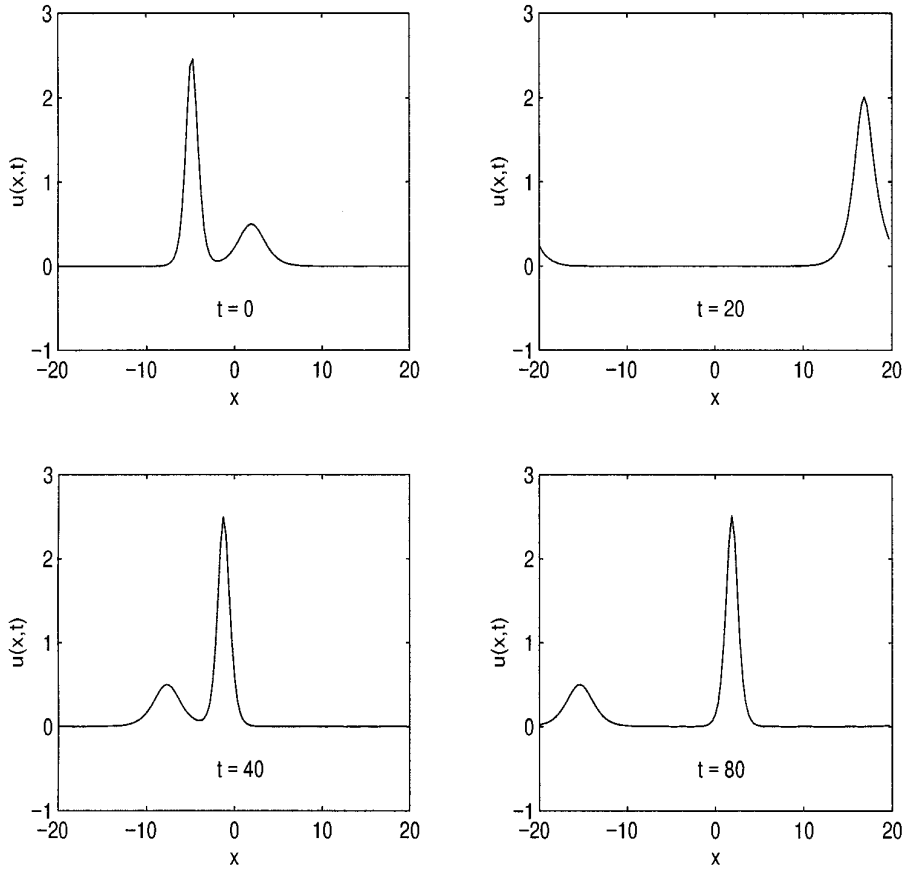

FIG. 9. Long time integration of the spatially discretized KdV equation by pseudospectral method for *collisions of two solitons* with parameters (4.5), and $N = 128$, $\theta = 0.02$ (similarly with parameters (4.4)).

REFERENCES

1. J. de Frutos and J. M. Sanz-Serna, An easily implementable fourth-order method for the time integration of wave problems, *J. Comput. Phys.* **103**(1), 160 (1992).
2. J. Demmel, *Numerical Linear Algebra*, Department of Mathematics (Univ. of California, Berkeley, CA, 1993).
3. E. Forest and R. D. Ruth, Fourth-order symplectic integrator, *Physica D* **43,** 105 (1990).
4. B. Fornberg, On a Fourier method for the integration of hyperbolic equations, *SIAM J. Numer. Anal.* **12,** 509 (1975).
5. B. Fornberg, The pseudospectral method: Comparisons with finite differences for the elastic wave equation, *Geophysics* **52,** 483 (1987).
6. G. H. Golub and C. F. Van Loan, *Matrix Computations,* (Johns Hopkins Univ. Press, Baltimore, MD, 1989).
7. D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications* (Soc. Indus. Appl. Math., Philadelphia, 1977).
8. D. Gottlieb and E. Turkel, *Spectral Methods for Time-Dependent Partial Differential Equations*, Report NASA CR-172241, Institute for Computer Applications in Science and Engineering, NSAS Langley Research Center, Hampton, VA, 1983.
9. E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I*, 2nd ed. (Springer-Verlag, New York, 1992).
10. E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II* (Springer-Verlag, New York, 1991).
11. W. Kahan, Relaxation methods for solving systems of ordinary differential equations, manuscript, CS Division, Department of EECS, University of California at Berkeley, October 1977.
12. W. Kahan, Unconventional numerical methods for trajectory calculations, Lectures notes, CS Division, Department of EECS, University of California at Berkeley, October 1993.
13. W. Kahan and R.-C. Li, Composition constants for raising the orders of unconventional schemes for ordinary differential equations, *Math. Comput.* [to appear]
14. D. J. Korteweg and G. deVries, On the change of form of long waves advancing in a rectangular channel, and on a new type of long stationary waves, *Phil. Mag.* **39,** 422 (1895).
15. H.-O. Kreiss and J. Oliger, Comparison of accurate methods for the integration of hyperbolic equations, *Tellus* **24,** 199 (1972).
16. J. D. Lambert, *Numerical Methods for Ordinary Differential Systems* (Wiley, New York, 1991).
17. R.-C. Li, *Raising the Orders of Unconventional Schemes for Ordinary Differential Equations*, Ph.D. thesis, Department of Mathematics, University of California at Berkeley, CA, 1995.
18. R. I. McLachlan, On the numerical integration of ordinary differential equations by symmetric composition methods, *SIAM J. Sci. Comput.* **16,** 151 (1995).

19. R. Meyer-Spasche and D. Düchs, *A General Method for Obtaining Unconventional and Nonstandard Difference Schemes*, Technical Report IPP 6/3434, Max-Planck-Institut für Plasmaphysik, Munich, 1995.
20. R. E. Mickens, *Nonstandard Finite Difference Models of Differential Equations* (World Scientific, Singapore, 1994).
21. F. Z. Nouri and D. M. Sloan, A comparison of fourier pseudospectral methods for the solution of the Korteweg–de Vries equations, *J. Comput. Phys.* **83,** 324 (1989).
22. S. A. Orszag, Comparison of pseudospectral and spectral approximation, *Stud. Appl. Math.* **51,** 253 (1972).
23. R. D. Ruth, A canonical integration technique, *IEEE Trans. Nucl. Sci.* **NS-30**(4), (1983).
24. Y. Saad and M. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Statis. Comput.* **7,** 856 (1986).
25. J. M. Sanz-Serna and L. Abia, Order conditions for canonical Runge–Kutta schemes, *SIAM J. Numer. Anal.* **28**(4), 1081 (1991).
26. J. M. Sanz-Serna and I. Christie, Petrov–Galerkin methods for nonlinear dispersive waves, *J. Comput. Phys.* **39,** 94 (1981).
27. M. Suzuki, General theory of higher-order decomposition of exponential operators and symplectic integrators, *Phys. Lett. A* **165,** 387 (1992).
28. E. Tadmor, The exponential accuracy of Fourier and Chebyshev differencing methods, *SIAM J. Numer. Anal.* **23,** 1 (1986).
29. T. R. Taha and M. J. Ablowitz, Analytical and numerical aspects of certain nonlinear evolution equations. III. numerical, Korteweg-de Vries equation, *J. Comput. Phys.* **55,** 231 (1984).
30. E. H. Twizell, Y. Wang, and W. G. Price, Chaos-free numerical solutions of reaction-diffusion equations, *Proc. R. Soc. Lond. A.* **430,** 541 (1990).
31. G. B. Whitham, *Linear and Nonlinear Waves* (Wiley, New York, 1974).
32. H. Yoshida, Construction of higher order symplectic integrators, *Phys. Lett. A* **150**(5, 6, 7), 262 (1990).
33. Z. Zlatev and J. Waśniewski, Running air pollution models on the connection machine, *Math. Comput. Modeling* **20**(6), 1 (1993).