Parallel Split-Step Fourier Methods for the CMKdV Equation

T. Taha and R. Liu
Univeristy of Georgia
Athens, GA 30602
thiab@cs.uga.edu
USA

Abstract

The class of complex modified Korteweg-de Vries (CMKdV) equations has many applications. One form of the CMKdV equation has been used to create models for the nonlinear evolution of plasma waves , for the propagation of transverse waves in a molecular chain, and for a generalized elastic solid. Another form of the CMKdV equation has been used for the traveling-wave and for a double homoclinic orbit.

In this paper we introduce sequential and parallel split-step Fourier methods for numerical simulations of the above equation. These methods are implemented on the Origin 2000 multiprocessor computer. Our numerical experiments have shown that the finite difference and the inverse scattering methods give accurate results and considerable speedup.

Introduction

This paper is concerned with numerical and parallel algorithms for numerical simulation of the complex modified Korteweg-de Vries (CMKdV) equation. The first part of this paper presents split-step Fourier methods for this equation. Three different numerical schemes are studied. We first present those schemes. Then implementations of these schemes for the numerical simulation of the CMKdV equation are carried out and numerical results are reported. In the second part of this paper, we develop parallel algorithms for these three schemes and implement them on the Origin 2000 multiprocessor computer.

Split-step Fourier method has been employed for solving a wide range of nonlinear wave equations. The basic idea of this method is to decompose the original problem into subproblems and then to approximate the solution of the original problem by solving the subproblems in a given sequential order. Various versions of this method have been developed for the nonlinear Schrödinger (NLS), Korteweg-de Vries (KdV) and modified Korteweg-de Vries (MKdV) equations.

Muslu and Erbay have extended the split-step method to the numerical solution of one form of the CMKdV equation. In their work the original equation is decomposed into a linear sub-equation and a nonlinear sub-equation. The linear sub-equation is solved by using the discrete time Fourier transform, and the nonlinear sub-equation is evaluated by using Runge-Kutta scheme in combination with the Fourier transform. Based on approximating the exponential operator in the original equation in different ways, three split-step schemes have been presented, namely; the first-order, second-order and fourth-order methods.

In this paper we study various split-step based algorithms for the CMKdV equation. Three schemes are considered. We consider the same equation as the one considered by Taha, which has a different form from the equation considered by Muslu-Erbay.

The first scheme of Muslu and Erbay is modified slightly to fit the CMKdV equation under consideration.

Then two additional schemes are presented based on different approximations of the nonlinear sub-equation; one using a finite difference approximation, another using a local Inverse Scattering Transform (IST) approximation. Both sequential and parallel versions of these schemes are developed and implemented. For the sequential algorithms, stability, convergence rate and computation time are compared among the different schemes. For the parallel algorithms, the focus is on testing the speedup.

Since the discrete Fourier transform plays a key part in these schemes, we will use the available and efficient FFTW library in their implementation. The FFTW is a free software that provides subroutines for both sequential and parallel discrete Fourier transform, and developed at MIT by Matteo Frigo and Steven G. Johnson.

Split-step Fourier Algorithms for the CMKdV Equation:

Introduction

In this paper we consider the following CMKdV equation:

$$q_t + 6\,|q|^2\, q_x + q_{xxx} = 0, \;\; x \in [a,b], \;\; 0 \le t \le T \tag{1}$$

where $q = q(x,t)$ is a complex valued function, and $|\cdot|$ denotes the modulus.

Taha presented three schemes based on finite difference approximation. In this work we study the split-step based algorithms for this equation and will compare the results with that presented by Taha. In addition a comparison among the algorithms is discussed.

Split-Step Fourier Methods In this section we present three split-step Fourier schemes for the CMKdV equation (1).

Let the periodic boundary interval be $[a, b]$. For convenience, we first normalize this interval to $[0, 2\pi]$ using the transformation $X = \frac{2\pi(x-a)}{(b-a)}$. As a result, the original CMKdV equation (1) becomes

$$q_t + \overline{\alpha}\,|q|^2 q_X + \overline{\beta} q_{XXX} = 0, \quad X \in [0, 2\pi] \qquad (2)$$

where

$$\overline{\alpha} = \frac{12\pi}{b-a}, \quad \overline{\beta} = \frac{(2\pi)^3}{(b-a)^3}$$

The linear sub-equation is given by

$$q_t + \overline{\beta} q_{XXX} = 0 \qquad (3)$$

and the nonlinear sub-equation is given by

$$q_t + \overline{\alpha}\,|q|^2 q_X = 0 \qquad (4)$$

As mentioned earlier, the basic idea in the split-step Fourier scheme is to approximate the exact solution of equation (2) by solving the linear sub-equation (3) and the nonlinear sub-equation (4) in a given sequential order. For

the solution of the linear sub-equation (3), discrete time Fourier transform is used. For the nonlinear sub-equation (4), different numerical approximation schemes can be used. This leads to different split-step algorithms for the CMKdV equation under consideration.

For completeness, we briefly review the discrete Fourier transform, which can be found in most textbooks on the subject of numerical analysis.

Divide the interval $[0, 2\pi]$ into N equal subintervals with grid spacing $\Delta X = 2\pi/N$. The spatial grid points are then given by $X_j = 2\pi j/N$, $j = 0, \ldots, N$. Let $q_j(t)$ denote the approximate solution to $q(X_j, t)$. The discrete Fourier transform of the sequence $\{q_j\}_{j=0}^{N-1}$ is defined as

$$\widehat{q}_k = \mathcal{F}_k(q_j) = \frac{1}{N} \sum_{j=0}^{N-1} q_j \exp(-ikX_j), \quad -\frac{N}{2} \leq k \leq \frac{N}{2} - 1 \tag{5}$$

The inverse transform of (5) is given by

$$q_j = \mathcal{F}_j^{-1}(\widehat{q}_k) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \widehat{q}_k \exp(ikX_j), \;\; 0 \le j \le N-1 \tag{6}$$

Here $\mathcal{F}$ denotes the discrete Fourier transform and $\mathcal{F}^{-1}$ its inverse transform. In some situations, when N is chosen to be $N = 2^p$ for some integer p, these transforms can be implemented efficiently via a fast Fourier transform (FFT) algorithm. See Briggs,Hart and Sweet- for a discussion on the FFT scheme.

With the Fourier transforms, the derivatives of q with respect to the spatial variable X can be easily evaluated. For example, q_X at (X_j, t) is given by $\mathcal{F}_j^{-1}(ik\mathcal{F}_k(q_j))$ and q_{XXX} at (X_j, t) is given by $\mathcal{F}_j^{-1}(-ik^3\mathcal{F}_k(q_j))$. These formulas will be used in the algorithms presented in the following subsections.

A sequential implementation of the discrete Fourier transform (DFT) can be easily done based on (5) and (6). However, to devise a parallel implementation is far complicated. In this work we will use FFTW, a C subroutine library for computing the DFT that has been used for many applications.

Split-Step Schemes presented by Muslu and Erbay:

Muslu and Erbay considered a different CMKdV equation, which differs in the nonlinear term, as follows:

$$q_t + \alpha(|q|^2 q)_x + q_{xxx} = 0 \qquad (7)$$

With minor modifications, we present the three split-step schemes, initially developed by Muslu-Erbay for equation (1).

To illustrate the general idea of this approach, we consider a general evolution equation in the form

$$q_t = (\mathcal{L} + \mathcal{N})q, \;\; q(x,0) = q(x) \qquad (8)$$

where $\mathcal{L}$ and $\mathcal{N}$ are linear and nonlinear operators, respectively. For equation (1), $\mathcal{L} = -\dfrac{\partial^3}{\partial x^3}$ and $\mathcal{N} = -6\,|q|^2 \dfrac{\partial}{\partial x}$. If $\mathcal{L}$ and $\mathcal{N}$ are assumed

to be t independent, a formal solution of (8) is given by

$$q(x, t + \Delta t) = \exp\left(\Delta t(\mathcal{L} + \mathcal{N})\right) q(x, t) \qquad (9)$$

where Δt is the time step in temporal discretization. Suppose that the linear subproblem

$$q_t = \mathcal{L}q,\ q(x, 0) = q(x) \qquad (10)$$

and the nonlinear subproblem

$$q_t = \mathcal{N}q,\ q(x, 0) = q(x) \qquad (11)$$

have known exact solutions

$$q(x, t + \Delta t) = \exp(\Delta t\mathcal{L})q(x, t) \qquad (12)$$

and

$$q(x, t + \Delta t) = \exp(\Delta t\mathcal{N})q(x, t) \qquad (13)$$

respectively. The solution of equation (8) can be approximated by using the solutions of the linear and nonlinear subproblems in a given sequential order. Such an order is determined

by the way of replacing the exponential operator $\exp(\Delta t(\mathcal{L}+\mathcal{N}))$ in (9) by an appropriate combination of products of the expontial operators $\exp(\Delta t\mathcal{L})$ and $\exp(\Delta t\mathcal{N})$. According to the Baker-Campbell-Hausdorf (BCM) formula and the higher order operator approximation in Yoshida, the first-order approximation of the exponential operator $\exp(\Delta t(\mathcal{L}+\mathcal{N}))$ is given by

$$\phi_1(\Delta t) = \exp(\Delta t\mathcal{L})\exp(\Delta t\mathcal{N}) \tag{14}$$

the second-order approximation is given by

$$\phi_2(\Delta t) = \exp(\frac{1}{2}\Delta t\mathcal{N})\exp(\Delta t\mathcal{L})\exp(\frac{1}{2}\Delta t\mathcal{N}) \tag{15}$$

and the fourth-order approximation is given by

$$\phi_4(\Delta t) = \phi_2(\lambda\Delta t)\phi_2((1-2\lambda)\Delta t)\phi_2(\lambda\Delta t) \tag{16}$$

where $\lambda = (2+2^{1/3}+2^{-1/3})/3$. Note that the fourth-order approximation involves the product of nine exponential operators.

The linear equation (3) is solved by means of the discrete Fourier transform and the advancements in time are performed according to

$$q_j^{m+1} = \mathcal{F}_j^{-1}\left(\exp(i\overline{\beta}k^3\Delta t)\mathcal{F}_k(q_j^m)\right) \qquad (17)$$

where q_j^m denotes the approximation to $q(X_j, m\Delta t)$. The spatial discretization of the nonlinear equation (4) by a Fourier pseudo-spectral method can be written as

$$(q_j)_t + \overline{\alpha}\left|q_j\right|^2 \mathcal{F}_j^{-1}\left(ik\mathcal{F}_k(q_j)\right) = 0 \qquad (18)$$

For the time integration of this equation, a Runge-Kutta method is used.

1. First-order scheme:

Given the data q_j^m at time $t = m\Delta t$:

1. Solve the nonlinear equation (18) where the time step is set to Δt. This solution becomes the initial data for the next step.

2. Solve the linear equation (17) by using the Fourier transform.

At the end of this process, we get the new data q_j^{m+1}.

A split-step scheme based on finite difference approximation of the nonlinear sub-equation

Taha and Ablowitz have developed a split-step Fourier method for the KdV equation, using the idea originally presented by F. Tappert. The linear sub-equation is still solved by the discrete Fourier transform (17). For the nonlinear sub-equation, an improved finite difference approximation formula was proposed, which has better truncation error. Inspired by this idea, we modify this formula and derive the finite difference approximation for the CMKdV nonlinear sub-equation (4) as follows:

$$q_n^{m+1} = q_n^m - \frac{\overline{\alpha}}{48}\frac{\Delta t}{\Delta X}\Big\{ \left|q_n^{m+1}\right|^2 (8q_{n+1}^{m+1} - 8q_{n-1}^{m+1}$$

$$\text{-q}_{n+2}^{m+1} + q_{n-2}^{m+1})$$

$$+ |q_n^m|^2 (8q_{n+1}^m - 8q_{n-1}^m - q_{n+2}^m + q_{n-2}^m)\Big\}$$

(19)

Thus our second split-step algorithm, which we may call the finite difference approximation scheme, is listed below.

2. Finite difference approximation scheme:

Given the data q_j^m at time $t = m\Delta t$:

1. Solve the nonlinear equation (4) by using (19) . This solution becomes the initial data for the next step.

2. Solve the linear equation (3) by using the Fourier transform (17).

At the end of this process, we get the new data q_j^{m+1}.

Equation (19) cannot be solved directly since the unknown quantity q_n^{m+1} appears in both sides of the equal symbol. As mentioned by Taha and Ablowitz, an iterative approximation procedure can be implemented to numerically compute the value of q_n^{m+1}.

A split-step scheme based on local IST approximation of the nonlinear sub-equation:

Taha discusses three numerical schemes for the CMKdV equation (1): a standard finite difference scheme, an integrable Inverse Scattering Transform (IST) scheme, and a local IST scheme. As he pointed out by Taha, the major difference among the three schemes is in the discretization of the nonlinear term in the CMKdV equation. The numerical simulations reported by Taha have shown that the standard scheme is subject to instability and the numerical solution becomes unbounded in finite time. The IST schemes do not suffer from any instabilities. In the case of coarse discretization in either space or in time or in both, these schemes produce approximate solutions for the CMKdV equation (1) with high accuracy (see Table 1 by Taha). However, all of these schemes are implemented by solving

a banded circulant Toeplitz systems of equations. This can be very expensive in terms of computation time if Gaussian elimination is used recursively to solve these systems of equations.

One advantage of using Fourier transform is that it is fast. Thus, we can use the Fourier transform (17) for the linear sub-equation (3) and use a local IST scheme for the nonlinear sub-equation (4). This leads to our third split-step method which we call the local IST scheme. The discretization formula for the nonlinear sub-equation can be easily obtained from the approximation by Taha.

$$
\begin{aligned}
q_n^{m+1} = q_n^m - \frac{\overline{\alpha}}{12}\frac{\Delta t}{\Delta X}\Big(& q_{n+2}^{m+1}\left(\left|q_{n+1}^m\right|^2 + |q_n^m|^2\right) \\
& - q_{n-2}^m\left(\left|q_n^{m+1}\right|^2 + \left|q_{n-1}^{m+1}\right|^2\right) \\
& + \frac{q_{n+1}^{m+1}}{2}\left((q^*)_n^m q_{n+1}^m + (q^*)_n^{m+1} q_{n+1}^{m+1} + 2q_n^m (q^*)_{n-1}^m\right) \\
& - \frac{q_{n-1}^m}{2}\left(q_{n-1}^{m+1}(q^*)_n^{m+1} + q_{n-1}^m (q^*)_n^m + 2q_n^{m+1}(q^*)_{n+1}^{m+1}\right) \\
& + \frac{q_n^m}{2}\left((q^*)_n^{m+1} q_{n+1}^{m+1} + (q^*)_n^m q_{n+1}^m\right) \\
& - \frac{q_n^{m+1}}{2}\left((q^*)_n^{m+1} q_{n-1}^{m+1} + (q^*)_n^m q_{n-1}^m\right) \\
& - 3\left(|q_n^m|^2 q_{n+1}^{m+1} - \left|q_n^{m+1}\right|^2 q_{n-1}^m\right)\Big)
\end{aligned}
\tag{20}
$$

where $*$ denotes the complex conjugate. This algorithm is listed below.

3. Local IST scheme:

Given the data q_j^m at time $t = m\Delta t$:

1. Solve the nonlinear equation (4) by using the local IST approximation (20). This solution becomes the initial data for the next step.

2. Solve the linear equation (3) by using the Fourier transform (17).

At the end of this process, we get the new data q_j^{m+1}.

Numerical Experiments

In this section we carry out numerical simulations to compare the performance among the three schemes. We consider the traveling-wave solution of (1) , i.e., the analytical solution of (1) is given by

$$q(x,t) = a \exp\left(i(kx - \omega t)\right) \qquad (21)$$

where $\omega = 6|a|^2 k - k^3$ and a is the complex amplitude. For initial conditions, (21) is used at $t = 0$ and $a = \frac{1}{2}, k = 1$. Periodic boundary conditions on the interval $[0, 2\pi]$ are imposed.

For convenience, we first give each scheme a short name that will be used frequently throughout this section.

1st: the first order split-step Fourier method.

FD: the finite difference approximation scheme.

IST: the local IST scheme.

The implementations of these schemes are done using C programming language on a Sun workstation. For the discrete Fourier transform, whenever it is applicable, e.g. $N = 2^p$ for some integer p, a fast Fourier transform (FFT) algorithm is used to further decrease the computation time.

Two norms, namely, L_∞ and L_2 are used to measure the accuracy of the approximate solutions. These norms are defined as follows.

$$L_\infty = \max_n(||\tilde{q}_n| - |q_n||), \quad L_2 = \sqrt{\sum_n (|\tilde{q}_n| - |q_n|)^2}$$

where $\tilde{q}_n$ is the numerical solution and q_n is the exact solution at point $(n\Delta x, T)$ for all n, where T is the terminating time.

Stability vs. instability Our first numerical simulation aims to investigate the stability of these schemes as the final time T increases. In order to compare our results with the results presented by Taha, we use the same data for both spatial discretization ($N = 40$) and the time step size ($\Delta t = 0.0125$).

Table 2.1 gives the L_∞ and L_2 error norms obtained from this experiment. Fig. 2.1 – 2.3 show the approximate solutions achieved by using these three numerical schemes.

From this experiment, we found that for the coarse discretization the performances of the first order scheme is very poor compared with the results given by Taha, Table 1. The numerical solution based on this scheme blows up at $t = 13.5875$.

On the other hand we found out that the finite difference and the local IST split-step algorithms produce very good performances which are in agreement of what has been reported by Taha. Since our implementations of these schemes do not involve solving a system of equations, the computation time is reduced.

Muslu and Erbay use very small values for the time step size Δt in their numerical examples (e.g., they use $\Delta t = 0.025(\Delta x)^3$ in one simulation and use $\Delta t = 0.3099 \times 10^{-4}$ in another experiment). Our current results show that for large value of Δt, the first method do not work well. In the following simulation, we'll

study the performance of these algorithms as Δt gets smaller, i.e., the convergence rate in time for these methods.

Table 2.1. L_∞ and L_2 norms for various values of T ($\Delta t = 0.0125, N = 40$)

T	Norm	1st	FD	IST
5.0	L_∞	0.043014	0.000001	0.000000
	L_2	0.272044	0.000004	0.000003
10.0	L_∞	0.114950	0.000001	0.000001
	L_2	0.727008	0.000007	0.000005
20.0	L_∞	blow up at	0.000002	0.000002
	L_2	$t = 13.5875$	0.000015	0.000010
40.0	L_∞		0.020831	0.001253
	L_2		0.091500	0.005597

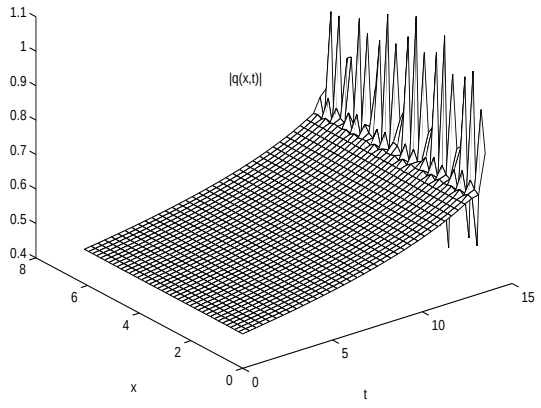

Fig. 2.1: The solution using the 1st scheme ($N = 40$ and $0 \leq t < 13.5875$)

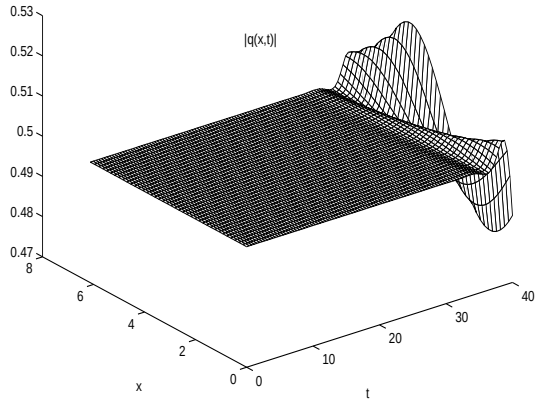

Fig. 2.2. The solution using the FD scheme ($N = 40$ and $0 \leq t \leq 40$)

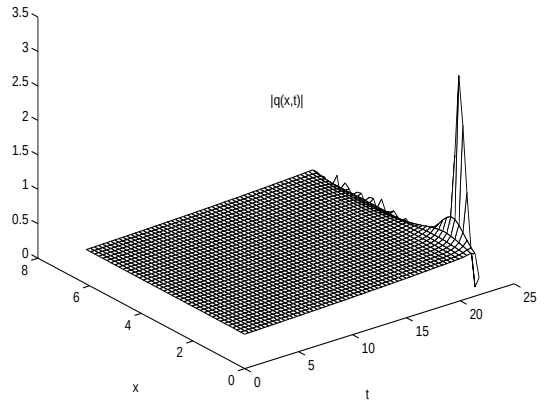

Fig. 2.2. The solution using the 2nd scheme ($N = 40$ and $0 \leq t < 21.775$)

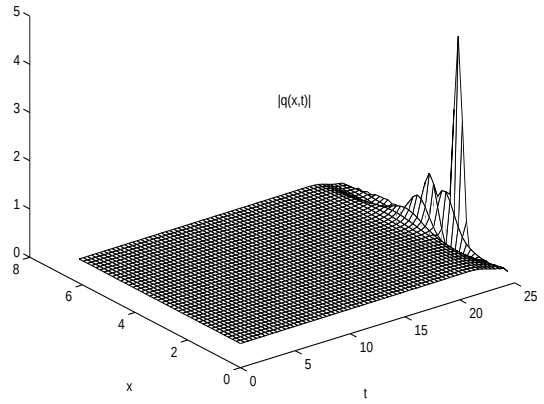

Fig. 2.3. The solution using the 4th scheme ($N = 40$ and $0 \leq t < 24.3875$)

Convergence in time To test the convergence rate of these split-step schemes in time, we perform a number of numerical experiments

for various values of time step Δt and a fixed value of N and of the terminating time T. We choose $T = 10.0$ and $N = 64$ (therefore, a FFT algorithm can be used to speed up the computation further). Table 2.2 gives the results.

From this table, we found:

1. Both the L_∞ errors and the L_2 errors are convergent for all the three schemes as Δt decreases.

2. The error for the first scheme decay approximately linearly in time step Δt. But for the other two schemes, namely FD and IST, the convergence rates seem to be higher than a linear order. Again, the errors of these two schemes are much less than the errors of the first three schemes. IST scheme

produces better approximations than FD. But the errors are of the same order.

Table 2.2. Comparison of the convergence rates in time (N= 40 and T = 10.0)

Δt	Norm	1st	FD	IST
0.005	L_∞	0.032918	7.41e-08	6.45e-08
	L_2	0.263172	5.93e-07	5.16e-07
0.001	L_∞	0.005789	5.93e-10	5.16e-10
	L_2	0.046315	4.75e-09	4.13e-09
0.0005	L_∞	0.002853	7.41e-11	6.44e-11
	L_2	0.022822	5.91e-10	5.14e-10
0.0001	L_∞	0.000564	1.14e-12	6.19e-13
	L_2	0.004513	4.20e-12	2.28e-12

Comparisons of computational times To compare the various schemes in terms of computational efficiency, we fix Δt, N and T and measure the computing times, the L_∞ errors and the L_2 errors. In our simulations, we choose $\Delta t = 0.0002$, $N = 64$, and $T = 10.0$. The results are given in Table 2.2. Note that both

the real CPU time and the normalized time are shown in this table.

Table 2.3. Comparison of computing times
($\Delta t = 0.0002, N = 64, T = 10.0$)

Scheme	L_∞	L_2	Time(sec)	Time
1st	1.13e-03	9.05e-03	123s	1
FD	4.03e-12	3.06e-11	96s	0.78
IST	3.56e-12	2.58e-11	99s	0.80

We notice from this table that the FD and IST schemes perform surprisingly well with very high accuracy and less time expense. Comparing with the time used by the first order scheme (it is normalized to be 1), the times used by FD and IST schemes are even less than 1 (0.78 for FD and 0.80 for IST, respectively). However, the errors of both schemes are reduced by a order of 10^8. This suggests that the last two schemes are much better than the first three algorithms.

Parallel Algorithms for the Split-step Fourier Schemes Introduction In this paper we will devise, implement and test the parallel algorithms of the three numerical schemes for the CMKdV equation which are presented earlier. Recall that the basic idea behind these schemes is to approximate the solution of the CMKdV equation by using the approximations of the linear and nonlinear sub-equations in a pre-specified order. Thus, to develop the parallel version of a split-step numerical algorithm, it is necessary to first develop the parallel algorithms for the approximate solutions of the linear and nonlinear sub-problems.

Parallel Implementations of the Linear and Nonlinear Sub-equations Since the discrete Fourier transform, finite difference and IST approximations, and the Runge-Kutta method are used in the numerical schemes for the linear and/or nonlinear sub-problems of these split-step schemes, we present the parallel implementations of these methods in this section.

Discrete Fourier Transform (DFT)

Suppose that $A(l)$, $l = 0, \ldots, N-1$ is a given array. The discrete Fourier transform is then defined by

$$\mathcal{F}(k) = \sum_{l=0}^{N-1} A(l) \exp(-\frac{2\pi i}{N} lk), \quad k = 0, \ldots, N-1 \tag{1}$$

the FFTW is used in the implementation of the DFT. The FFTW is developed at MIT by Matteo Frigo and Steven G. Johnson, it has become the FFT library of choice for most applications. Its performance, tested by benchmarks on a variety of platforms, is typically superior to that of other publicly available FFT software. FFTW received the 1999 J. H. Wilkinson Prize for Numerical Software, which is awarded every four years to the software that "best addresses all phases of the preparation of high quality numerical software."

FFTW provides parallel routines for distributed-memory (and shared-memory) machines supporting MPI (Message Passing Interface, an introduction will be presented in the next section). The usage of these routines is straightforward and simple. One may refer to the online document of FFTW for details.

Finite Difference and Local IST Approximations

In this subsection we consider the parallel implementations for the finite difference approximation (19) and the local IST approximation (20). It is easy to see that these two schemes can be parallelized in a very similar manner. For this reason we only present the details for the first scheme.

In view of (19), the computation of the n-indexed values depends not only on the previous n-indexed values, but also on the adjacent values, namely, the $(n+1)$-, $(n+2)$-, $(n-1)$-, and $(n-2)$-indexed values. That implies that if we split the computation of the N values evenly among the p processors, data communication must be carried out between adjacent processors.

Let the p processors be labeled by $0, 1, \ldots, p-1$. Processor i will be working on the $M_1 = N/p$

entries q_n^{m+1}, $i * M_1 \leq n < (i+1)M_1$. For easy notation, we denote these M_1 values by $\bar{q}_0, \bar{q}_1, \ldots, \bar{q}_{M_1-2}, \bar{q}_{M_1-1}$. The following is the parallel algorithm for (19):

Procedure 1: Parallel Finite Difference Approximation:

Assume currently each processor i holds its own M_1 data. For $i = 0, 1, \ldots, p-1$:

1. Send $\bar{q}_0, \bar{q}_1$ to its left neighbor (if $i = 0$, send to processor $p-1$).

2. Send $\bar{q}_{M_1-2}, \bar{q}_{M_1-1}$ to its right neighbor (if $i = p-1$, send to processor 0).

3. Update new values by (19).

Parallel Implementations of the three Split-step Schemes Before we present the parallel algorithms, we first give a brief introduction of the Message Passing Interface (MPI), which will be used in implementing the parallel algorithms on the Origin 2000 multiprocessor computer.

Message Passing Interface (MPI)

Message Passing Interface (MPI) is a tool for parallel computing that has become an ad hoc standard. It was designed for high performance on both massively parallel machines and on workstation clusters. MPI is widely available, with both freely available and vendor-supplied implementations. A number of MPI home pages are available. MPI was developed by a broadly based committee of vendors, implementors, and users.

In distributed parallel programming using MPI, different processors work on totally independent data and explicitly use send and receive

commands provided by MPI to exchange data between processors.

For further details on the usage of more MPI commands, one may refer to the MPI on-line document.

Parallel Split-step Fourier Algorithms

Using the parallel procedures we have developed earlier, we now build the parallel implementation algorithms of the three split-step Fourier schemes for the CMKdV equation. We consider the distributed-memory approach for these implementations, using the MPI tool.

Our implementations employ the Master-Slave(s) structure. This means one processor called the master takes the special role of controlling the whole routine of the execution of the program, and the rest processors called slaves are only responsible for their own computation and for sending and receiving data. This way, the master processor will have a heavier load than a slave does.

The implementation algorithms for the three split-step schemes are presented in the following. In these algorithms T_n denotes the number of iterations. Note that the following algorithms use FFTW for the implementation of DFT.

Algorithm 1: parallel first-order scheme (refer to (14), (18) and (17))

The master distributes the initial values q_j^0, $0 \leq j < N$ among the p processors.

For $m = 0$ To $T_n - 1$, Do

step 1: Use Procedure 1 to compute the forward Fourier transform $\mathcal{F}_k(q_j^m)$.

step 2: The master collects the results $\mathcal{F}_k(q_j^m)$, compute the product $ik\mathcal{F}_k(q_j^m)$.

step 3: Use Procedure 1 to compute the inverse transform $\mathcal{F}_k^{-1}(ik\mathcal{F}_k(q_j^m))$.

step 4: The master collects the results $\mathcal{F}_k^{-1}(ik\mathcal{F}_k(q_j^m))$

step 5: Use Procedure 2 for the time integration.

step 6: The master collects the solution of the nonlinear sub-equation (18).

step 7: Use Procedure 1 to compute the forward Fourier transform $\mathcal{F}_k(q_j^m)$.

step 8: The master collects the results $\mathcal{F}_k(q_j^m)$, compute $\exp(i\overline{\beta}k^3\Delta t)\mathcal{F}_k(q_j^m)$.

step 9: Use Procedure 1 to compute the inverse transform.

step 10: The master collects the solution of the linear sub-equation (17). This yields the new values q_j^{m+1}.

In Algorithm 1, the step $1-5$ is used to solve the nonlinear sub-equation, and the step $7-9$ is used for the linear sub-equation.

Algorithm 2: parallel finite difference approximation scheme (refer to (19) and (17))

The master distributes the initial values q_j^0, $0 \leq j < N$ among the p processors.

For $m = 0$ To $T_n - 1$, Do

step 1: Use Procedure 2 to solve the nonlinear sub-equation (4).

step 2: The master collects the solution of the nonlinear sub-equation.

step 3: Use Procedure 1 to compute the forward Fourier transform $\mathcal{F}_k(q_j^m)$.

step 4: The master collects the results $\mathcal{F}_k(q_j^m)$, compute $\exp(i\overline{\beta}k^3\Delta t)\mathcal{F}_k(q_j^m)$.

step 5: Use Procedure 1 to compute the inverse transform.

step 6: The master collects the solution of the linear sub-equation (17). This yields the new values q_j^{m+1}.

Algorithm 3: parallel local IST scheme (refer to (20) and (17))

The master distributes the initial values q_j^0, $0 \leq j < N$ among the p processors.

For $m = 0$ To $T_n - 1$, Do

step 1: Use a modified form of Procedure 2 to solve the nonlinear sub-equation (4).

step 2: The master collects the solution of the nonlinear sub-equation.

step 3: Use Procedure 1 to compute the forward Fourier transform $\mathcal{F}_k(q_j^m)$.

step 4: The master collects the results $\mathcal{F}_k(q_j^m)$, compute $\exp(i\overline{\beta}k^3\Delta t)\mathcal{F}_k(q_j^m)$.

step 5: Use Procedure 1 to compute the inverse transform.

step 6: The master collects the solution of the linear sub-equation (17). This yields the new values q_j^{m+1}.

Numerical Results We have implemented these three parallel algorithms on the Origin 2000 multiprocessor computer, using C programming language and the MPI commands for data communication. Our objective is to test the timing and to compute the speedup.

Experiment 1: In this experiment we test and compare the speedup of these algorithms using FFTW. p Table 3.1(a) - 3.4(e) give the CPU time and speedup for the three methods.

Table 3.1(a). CPU time and speedup for the parallel 1st scheme using FFTW

p	$N = 2048, T_n = 4000$	$N = 4096$	$N = 8192$
	$T_n = 4000$	$T_n = 2000$	$T_n = 1000$
2	19.3 s	20.2 s	18.9 s
4	14.8 s	14.8 s	12.2 s
Sp	1.30	1.36	1.55

Table 3.2(b). CPU time and speedup for the parallel FD scheme using FFTW

p	$N = 2048$	$N = 4096$	$N = 8192$
	$T_n = 4000$	$T_n = 2000$	$T_n = 1000$
2	12.7 s	12.2 s	11.9 s
4	8.9 s	8.3 s	7.1 s
Sp	1.43	1.47	1.68

Table 3.3(c). CPU time and speedup for the parallel IST scheme using FFTW

p	$N = 2048$	$N = 4096$	$N = 8192$
	$T_n = 4000$	$T_n = 2000$	$T_n = 1000$
2	12.5 s	14.1 s	12.5 s
4	9.0 s	9.4 s	7.4 s
Sp	1.39	1.50	1.69

From the experiment we conclude that (1) the speedup is an increasing function of the dimension N; (2) FD and IST yield a higher speedup than the first order schemes.

Conclusions

In this paper we have studied various split-step Fourier transform methods for the numerical solution of the complex modified Korteweg-de Vries (CMKdV) equation. We have dealt with both sequential and parallel implementations of three different split-step Fourier schemes.

Our numerical simulations have demonstrated that the split-step Fourier method is an effective and efficient algorithm for the CMKdV equation.

Based on the studies presented in this paper, we can draw the following conclusions:

1. All of the three schemes presented in this paper give feasible solutions for the CMKdV equation.

2. The finite difference (FD) and the local IST approximation schemes out-perform in general the first order scheme. This point has been demonstrated by comparing the stability, convergence rate, computation time and the speedup.

3. Our parallel versions of these schemes have been developed based on the parallel algorithms of the discrete Fourier transform

and the finite difference approximation. By implementing these algorithms on the Origin 2000 multiprocessor computer, speedup is achieved. The larger the spatial dimension N is, the higher the speedup can be achieved.

References

W. L. Briggs, L. B. Hart, R. A. Sweet and A. O'Gallagher, Multiprocessor FFT methods, *SIAM J. Sci. Stat. Comput.* , **8** (1987), 27-42.

Richard L. Burden and J. Douglas Faires, *Numerical Analysis,* 4th edition, PWS-Kent, Boston (1989).

H. A. Erbay, Nonlinear transverse waves in a generalized elastic solid and the complex modified Korteweg-de Vries equation, *Physica Scripta*, **58** (1998), 9-14.

S. Erbay and E. S. Suhubi, Nonlinear wave propagation in micropolar media II. Special cases, solitary waves and Painlevé analysis, *Int. J. Engng. Sci.* , **27** (1989), 915-919.

FFTW homepage, *http://www.fftw.org*

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

O. B. Gorbacheva and L. A. Ostrovsky, Nonlinear vector waves in a mechanical model of a molecular chain, *Physica*, **D 8** (1983), 223-228.

B. M. Herbst, M. J. Ablowitz and E. Ryan, Numerical homoclinic instabilities and the complex modified Korteweg-de Vries equation, *Comput. Phys. Commun.*, **65** (1991), 137-142.

C. F. F. Karney, A. Sen and F. Y. F. Chu, Nonlinear evolution of lower hybrid waves, *Phys. Fluids*, **22** (1979), 940-952.

MPI homepage, *http://www-unix.mcs.anl.gov/mpi/*

G. M. Muslu and H. A. Erbay, A split-step Fourier method for the complex modified Korteweg-de Vries equation, *Preprint,* (2000).

Khaled Rasheed, Brian Davison, Effect of global parallelism on the behavior of a steady state genetic algorithm for design optimization, Congress on Evolutionary Computation (CEC'99), 1999.

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

Scott Zoldi, Victor Ruban, Alexandre Zenchuk, and Sergey Burtsev, "Parallel implementation of the split-step Fourier method for solving

nonlinear Schrödinger systems," *SIAM news,* January 8, 1999.

T. R. Taha, Numerical simulations of the complex modified Korteweg-de Vries equation, *Mathematics and Computers in Simulation,* **37** (1994), 461-467.

T. R. Taha and M. J. Ablowitz, Analytical and numerical aspects of certain nonlinear evolution equations. II. Numerical, nonlinear Schrödinger equation, *J. Comput. Phys.*, **55** (1984), 203-230.

T. R. Taha and M. J. Ablowitz, Analytical and numerical aspects of certain nonlinear evolution equations. IV. Numerical, Korteweg-de Vries equation, *J. Comput. Phys.*, **55** (1984), 231-253.

T. R. Taha and M. J. Ablowitz, Analytical and numerical aspects of certain nonlinear evolution equations. IV. Numerical, modified Korteweg-de Vries equation, *J. Comput. Phys.*, **77** (1988), 540-548.

F. Tappert, Numerical solutions of the Korteweg-de Vries equation and its generalizations by the split-step Fourier method, *Lect. Appl. Math. Amer. Math. Soc.* , **15** (1974), 215-216.

J. A. C. Weideman and B. M. Herbst, Split-step methods for the solution of the nonlinear Schrödinger equation, *SIAM J. Numer. Anal.* , **23** (1986), 485-507.

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