Fixed Point Theory, 10(2009), No. 1, 173-183
http://www.math.ubbcluj.ro/~nodeacj/sfptcj.html

A HERMITE SPECTRAL METHOD FOR SOLITONS

DAMIAN TRIF AND TITUS PETRILA

Department of Applied Mathematics
Babeş-Bolyai University of Cluj-Napoca
Str. M. Kogalniceanu nr. 1
400084 Cluj-Napoca, Romania
E-mail: dtrif@math.ubbcluj.ro

Abstract. We use the analytical soliton solutions of the Korteweg-de Vries (KdV) equation to test a new spectral numerical method for partial differential evolution equations with unbounded spatial domain. The proposed spatial discretization uses Hermite functions in the spectral space while the temporal discretization is performed by a symmetric exponential integrator coupled with fixed point iterations. The algorithm could be used to numerically describe the soliton behaviour, such as small-amplitude long waves on the free surface of water.

Key Words and Phrases: Free surface, solitons, Hermite spectral method, exponential integrators, KdV equation.
2000 Mathematics Subject Classification: 65N35, 35Q53, 47H10.

1. Introduction

A *soliton* is a self-reinforcing solitary wave that maintains its shape while it travels at constant speed. Following Drazin and Johnson [4] the main properties of solitons are

a) they are of permanent form,

b) they are localized within a region,

c) they can interact with other solitons, and emerge unchanged from the collision, except for a phase shift.

John Scott Russell was the first who described such a solitary wave, observed in the Union Canal in Scotland, in 1834. This phenomenon can be expressed

by the Korteweg-de Vries (1895) equation

$$u_t + 6u\,u_x + u_{xxx} = 0, \;\; x \in \mathbb{R}, \mathrm{t} > 0 \tag{1}$$

but solitons arise as the solutions of many other nonlinear dispersive partial differential equations describing physical systems.

In 1965 Norman Zabusky and Martin Kruskal firstly demonstrated soliton behaviour in a computational research using a finite difference approach for the KdV equation and in 1967, Gardner, Greene, Kruskal and Miura discovered an inverse scattering transform enabling analytical solution of the KdV equation of the form

$$u = \frac{c}{2\cosh^2\left(\frac{\sqrt{c}}{2}\left(x - ct - f\right)\right)}.$$

This technique was extended to the solutions of many related soliton-generating equations.

Nowadays the solitons are used to describe the complex dynamical behaviour of wave systems in hydrodynamics, optical fibers, plasmas, shock waves, tornados, the Great Red Spot of Jupiter, etc.

The KdV equation is well suited as a test object in applying numerical methods to non-linear PDEs with unbounded spatial domain, since we have analytical solutions and we can appreciate the quality of the numerical approximation.

The next section presents the spatial discretization for an unbounded spatial domain and proposes a technique using Hermite functions in the spectral space. The third section deals with the temporal discretization using symmetric exponential integrators coupled with fixed point iterations, while the last section contains numerical examples and conclusions.

2. Spatial discretization

The main numerical methods used to describe the soliton propagation are the finite difference method, the Fourier method and the spectral methods. The main difficulty comes from the boundlessness of the spatial domain.

The finite differences method needs a bounded spatial domain. If we truncate the unbounded domain and impose artificial homogeneous boundary conditions, we must stop the calculations if the nonzero part of the solution overtakes that boundary, in order to avoid the reflections. Finding transparent

boundary conditions, which do not modify the solution is an active field of research.

The Fourier method uses periodic functions. We must again truncate the domain and impose periodic boundary conditions. In fact, we solve a modified problem, not the given problem.

We can map the unbounded domain to a bounded one, but the transformed equation is usually very complicated. A better idea is to use new families of orthogonal functions which are images of classical Jacobi polynomials for example, under suitable mappings, see Shen and Wang [7] for more details.

For problems on the whole real line, the natural choice is, however, to use spectral methods based on Hermite polynomials/functions, because of their close connection to the physics. Spectral methods have been used for solving problems on unbounded domains for over thirty years and one of the early applications was the computation of interacting solitons by Fornberg and Whithman in 1978. However, Weideman and Reddy [8] did not find in 2000 any references to the use of the Laguerre spectral collocation method for the Schrödinger equation on $[0, \infty)$ or to the use of the Hermite spectral method for the sine-Gordon equation on $(-\infty, \infty)$. They suspected that their package *MATLAB Differentiation Matrix Suite* contains the first application of the Hermite spectral method to simulate solitons.

The modern applications of the Hermite spectral method are in fact pseudospectral methods, i.e. they work with the physical values of the unknown functions on a suitable grid (in the physical space), not with the coefficients of the Fourier-Hermite expansions of that functions (in the spectral space). If we apply a spectral method to a nonlinear equation, the best strategy is to evaluate the derivatives in the spectral space and the nonlinear part in the physical space. There exist fast transforms between physical space and the spectral space for the Fourier or Chebyshev spectral methods but not for Hermite spectral method. Consequently, the use of the physical space only, even for the evaluation of the derivatives in the Hermite pseudospectral method with a *full* pseudospectral differentiation matrix, seems to be more efficient.

The computational power of the modern computers however changes the hierarchy of the numerical methods from the efficiency point of view. The lack of a fast transform between the spectral space and the physical space is

no longer an impediment to use the Hermite spectral method in the spectral space, with a *two-diagonal* spectral differentiation matrix.

If we have to solve numerically a nonlinear evolution equation on the whole real line, like KdV equation (1), we perform at first a spatial semidiscretization using the Hermite spectral method and we obtain a differential system for the expansion coefficients. We shortly present this spectral method.

The solution is represented as a truncated series

$$u_N(x) = \sum_{n=0}^{N-1} c_n \psi_n(x) \tag{2}$$

where

$$\psi_n(x) = \frac{1}{\sqrt{2^n n! \sqrt{\pi}}} H_n(x), \tag{3}$$

H_n are the usual Hermite polynomials and N is a truncation parameter. In the matrix form,

$$u_N(\mathbf{x}) = T^T(\mathbf{x})\ \mathbf{c} \tag{4}$$

where $\mathbf{c}$ is the column of the coefficients and T is the matrix of the values of $\psi_n(\mathbf{x})$ on the grid points $\mathbf{x}$, (the upper script T transposes the matrix). The grid points $\mathbf{x}$, again a column vector, are the Gauss-Hermite quadrature nodes, i.e.

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx \approx \sum_{k=1}^{N} f(x_k) w_k$$

where w_k are the weights of the formula. For MATLAB calculations, see the functions `[x,w]=pd(N)`and `T=x2t(N,x)` from [6].

The coefficients c_k are given by the formula

$$c_k = \int_{-\infty}^{\infty} e^{-x^2} u_N(x) \psi_k(x) dx, \quad k = 0, ..., N-1$$

or, in the matrix form,

$$\mathbf{c} = T(\mathbf{x})\,(diag(w)\, u_N(\mathbf{x})). \tag{5}$$

The relations (4) and (5) give the transformations between the representation of $u_N(x)$ in the physical space $(u_N(x_1), ..., u_N(x_N))$ and the representation in the spectral space $(c_0, ..., c_{N-1})$ and involve only matrix times vector multiplications.

We use now the recurrence relations for the Hermite polynomials

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x),\ H_n'(x) = 2nH_{n-1}(x)$$

to calculate the matrices X (multiplication by x in the spectral space) and D (differentiation matrix in the spectral space) so that

$$\mathbf{x} \cdot u_N(\mathbf{x}) = T^T(X\mathbf{c}), \quad \frac{du_N(\mathbf{x})}{dx} = T^T(D\mathbf{c}).$$

The functions `D=deriv(N)` and `X=mult(N)` from `[6]` give the two-diagonal matrices D and X.

Taking into account the behaviour of our solutions at $\pm\infty$, we must use as a basis the complete orthonormal sequence of Hermite-Weber functions

$$\phi_n(x) = \frac{1}{\sqrt{2^n n!\sqrt{\pi}}} e^{-x^2/2} H_n(x).$$

The multiplication matrix X is the same but the differentiation matrix D must be replaced by $D - X$, denoted again by D. We refer to [7] for details about the convergence of the Fourier-Hermite series and to [5], [6] for details about this kind of spatial discretization and for MATLAB implementation.

3. Temporal discretization

Let us consider, as an example, the problem

$$\begin{gathered} \frac{\partial u}{\partial t} = -\frac{\partial^3 u}{\partial x^3} - u\frac{\partial u}{\partial x}, \quad x \in (-\infty, \infty), t > 0 \\ u(\pm\infty, t) = 0, \quad t > 0 \\ u(x, 0) = u_0(x), \quad x \in (-\infty, \infty) \end{gathered} \tag{6}$$

The numerical solution is represented by $\mathbf{c}(t) = (c_0(t), ..., c_{N-1}(t))^T$, the coefficients of the truncated expansion

$$u_N(\mathbf{x}) = \sum_{n=0}^{N-1} c_n(t)\phi_n(\mathbf{x}) \tag{7}$$

where $\mathbf{x}$ are the Gauss-Hermite nodes. From (6) we obtain the differential system

$$\mathbf{c}'(t) = -D^3\mathbf{c}(t) - T\,diag(w)\,\left[(T^T\mathbf{c}) \cdot \left(T^T D\mathbf{c}\right)\right], \tag{8}$$

$$\mathbf{c}(0) = T\,diag(w)\,u_0(\mathbf{x}). \tag{9}$$

where $(v_1, ..., v_N)^T \cdot (w_1, ..., w_N)^T = (v_1 w_1, ..., v_N w_N)^T$. The equation (6) has the general form

$$\frac{\partial u}{\partial t} = \mathbf{L}u + \mathbf{N}[u] \tag{10}$$

where $\mathbf{L}$ is the linear part and $\mathbf{N}$ is the nonlinear part. The differential system for the coefficients has the general form

$$\mathbf{c}'(t) = \mathcal{L}\mathbf{c}(t) + \mathcal{N}[\mathbf{c}(t)] \tag{11}$$

and it is a stiff system, with a *sparse* matrix $\mathcal{L}$. We consider only matrices $\mathcal{L}$ with large and imaginary eigenvalues (the KdV equation case) or with large and negative eigenvalues.

In order to avoid the small time step imposed by explicit integrators, we will use an exponential time differencing integrator for (11), that is an exact method when $\mathcal{N} = constant$.

If we multiply (11) by the exponential matrix $e^{-\mathcal{L}t}$ as an integrating factor, we obtain

$$\frac{d}{dt}\left(e^{-\mathcal{L}t}\mathbf{c}(t)\right) = e^{-\mathcal{L}t}\mathcal{N}[\mathbf{c}(t)]$$

or, by integrating between time levels t_n and t_{n+1} and denoting $\mathbf{c}_n = \mathbf{c}(t_n)$,

$$e^{-\mathcal{L}t_{n+1}}\mathbf{c}_{n+1} - e^{-\mathcal{L}t_n}\mathbf{c}_n = \int_{t_n}^{t_{n+1}} e^{-\mathcal{L}\tau}\mathcal{N}[\mathbf{c}(\tau)]\, d\tau.$$

Finally we obtain

$$\mathbf{c}_{n+1} = e^{\mathcal{L}h}\mathbf{c}_n + \int_0^h e^{-\mathcal{L}(\tau-h)}\mathcal{N}[\mathbf{c}(t_n+\tau)]\, d\tau,$$

where $h = t_{n+1} - t_n$ is the time step.

Now we approximate $\mathcal{N}[\mathbf{c}(t_n+\tau)]$ by a polynomial in τ and do the integral exactly. For simplicity, assuming $\mathcal{N}[\mathbf{c}(t_n+\tau)] = constant = \mathcal{N}[\mathbf{c}(t_n)] = \mathcal{N}_n$ we obtain

$$\mathbf{c}_{n+1} = e^{\mathcal{L}h}\mathbf{c}_n + \left(e^{\mathcal{L}h} - I\right)\mathcal{L}^{-1}\mathcal{N}_n$$

where I is the $N \times N$ identity matrix. Denoting by $\Phi_1(z) = (e^z - 1)/z$, we obtain the explicit exponential time differencing Euler method,

$$\mathbf{c}_{n+1} = e^{\mathcal{L}h}\mathbf{c}_n + h\Phi_1(\mathcal{L}h)\mathcal{N}_n.$$

Assuming $\mathcal{N}[\mathbf{c}(t_n+\tau)] = constant = \mathcal{N}[\mathbf{c}(t_n+h)] = \mathcal{N}_{n+1}$ we obtain the implicit exponential time differencing Euler method,

$$\mathbf{c}_{n+1} = e^{\mathcal{L}h}\mathbf{c}_n + h\Phi_1(\mathcal{L}h)\mathcal{N}[\mathbf{c}_{n+1}].$$

These methods can be combined into a symmetric exponential integrator

$$\begin{aligned} \mathbf{c} =& e^{\mathcal{L}\frac{h}{2}}\mathbf{c}_n + \frac{h}{2}\Phi_1(\mathcal{L}\frac{h}{2})\mathcal{N}[\mathbf{c}], \\ \mathbf{c}_{n+1} =& e^{\mathcal{L}\frac{h}{2}}\mathbf{c} + \frac{h}{2}\Phi_1(\mathcal{L}\frac{h}{2})\mathcal{N}[\mathbf{c}], \\ & n = 0, 1, ..., \quad \mathbf{c}_0 = \mathbf{c}(0) \end{aligned} \tag{12}$$

and we refer to [3] for the properties of the symmetric exponential integrators and to [1] for higher order exponential integrators and MATLAB implementation.

In order to solve the first (implicit) relation with respect to $\mathbf{c}$

$$\mathbf{c} = \mathbf{f}(\mathbf{c}) := e^{\mathcal{L}\frac{h}{2}}\mathbf{c}_n + \frac{h}{2}\Phi_1(\mathcal{L}\frac{h}{2})\mathcal{N}[\mathbf{c}]$$

we can calculate the fixed point iterations

$$\mathbf{c}^{[k+1]} = e^{\mathcal{L}\frac{h}{2}}\mathbf{c}_n + \frac{h}{2}\Phi_1(\mathcal{L}\frac{h}{2})\mathcal{N}\left[\mathbf{c}^{[k]}\right], \quad \mathbf{c}^{[0]} = \mathbf{c}_n.$$

The iteration error $err^{[k]} = \mathbf{c}^{[k]} - \mathbf{c}$ verifies

$$err^{[k+1]} = \frac{h}{2}\Phi_1(\mathcal{L}\frac{h}{2})\left(\mathcal{N}\left[\mathbf{c}^{[k]}\right] - \mathcal{N}[\mathbf{c}]\right) \approx \frac{h}{2}\Phi_1(\mathcal{L}\frac{h}{2})\mathcal{N}'[\mathbf{c}]\, err^{[k]}.$$

For the KdV equation we have

$$\mathcal{N}[\mathbf{c}] = -T\, diag(w)\, \left[(T^T\mathbf{c})\cdot\left(T'D\mathbf{c}\right)\right]$$

so that

$$\begin{aligned} \mathcal{N}'[\mathbf{c}]\, err^{[k]} &= -T\, diag(w)\left[T^T\mathbf{c}\cdot T^T D\, err^{[k]} + T^T D\mathbf{c}\cdot T^T err^{[k]}\right] = \\ &= -T\, diag(w)\, \left[diag(T^T\mathbf{c})\, T^T D + diag(T^T D\mathbf{c})T^T\right] err^{[k]}. \end{aligned}$$

For a fixed spatial discretization and a bounded solution $\mathbf{c}(t)$, $\mathcal{N}'[\mathbf{c}]$ is bounded, the norm of $\Phi_1(\mathcal{L}\frac{h}{2})$ is bounded by 1 and then, for a sufficiently small time step h, $\mathbf{c}^{[k]}$ converges toward $\mathbf{c}$ and the relations (12), (7) give the approximate solution at the time level t_{n+1}.

4. Numerical experiments

The above algorithm was tested on the KdV equation with a concrete initial condition

$$\begin{aligned} &\frac{\partial u}{\partial t}+\frac{\partial^3 u}{\partial x^3}+6u\frac{\partial u}{\partial x}=0, \quad x\in(-\infty,\infty), t>0 \\ &u(\pm\infty,t)=0, \quad u(x,0)=2\mathrm{sech}^2(x) \end{aligned} \tag{13}$$

with the exact solution $u(x,t)=2\mathrm{sech}^2(x-4t)$. We choose the time interval $[0,4]$ with the time step $h=0.001$ and $N=256$. At each time level, 4 fixed point iterations are necessary to obtain $\left\|\mathbf{c}^{[4]}-\mathbf{c}^{[3]}\right\|\leq 10^{-7}$. Finally, the global error between the exact and numerical solution is less than 3×10^{-4}.

The KdV equation has as time invariants

$$I_1=\int_{-\infty}^{\infty}u(x,t)dx,\ I_2=\int_{-\infty}^{\infty}u^2(x,t)dx,\quad I_3=\int_{-\infty}^{\infty}(u_x^2-u^3)dx.$$

For this example we obtain

$$\begin{aligned} &I_1\in[3.999984842240054, 4.000000878927037] \\ &I_2\in[5.333333308695872, 5.333333333326634] \\ &I_3\in[-4.266700574817008, -4.266665604040662] \end{aligned}$$

which confirm the qualities of the algorithm. We remark that the algorithm is stable even for $h=0.01$ but in this case the global error is about 0.05 and higher order exponential integrators must be used.

As a second test we take a nonlinear superposition of two solitons again for the KdV equation (13). We use the exact solution [2]

$$u(x,t)=-\frac{2\left(b_1-b_2\right)\left[b_1\mathrm{sech}^2\left(\sqrt{\frac{b_1}{2}}\left(x-2tb_1\right)\right)+b_2\mathrm{csch}^2\left(\sqrt{\frac{b_2}{2}}\left(x-2tb_2\right)\right)\right]}{\left[-\sqrt{2b_2}\coth\left(\sqrt{\frac{b_2}{2}}\left(x-2tb_2\right)\right)+\sqrt{2b_1}\tanh\left(\sqrt{\frac{b_1}{2}}\left(x-2tb_1\right)\right)\right]^2}$$

for $b_1=0.5$ and $b_2=1$. For the numerical calculation we take the initial condition $u(x,0)$ from this exact solution.

We choose the time interval $[-4,4]$ with the time step $h=0.01$ and $N=256$. At each time level 6 fixed point iterations are necessary to obtain $\left\|\mathbf{c}^{[6]}-\mathbf{c}^{[5]}\right\|\leq 10^{-7}$. Finally, the global error between the exact and numerical solution is less than 8×10^{-4} and

$$I_1 \in [4.828410267676169, 4.828462604082842].$$

Figure 1 shows a plot of the numerical solution, figure 2 contains the first 100 coefficients of the solution at $t = 4$ and figure 3 shows the error at $t = 4$.

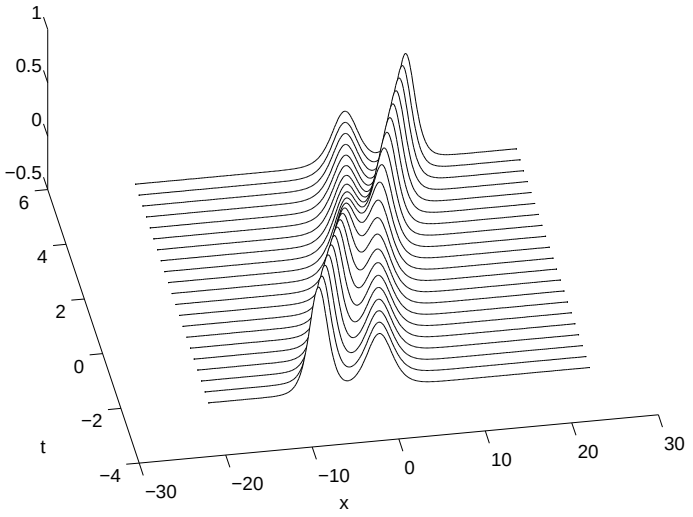

FIGURE 1. Numerical solution for nonlinear superposition of two solitons

The pseudospectral algorithm using DMS [8] does not work for $N = 256$ (the differentiation matrix is "singular, close to singular or badly scaled"). If $N = 180$ the computing time for one time-step is 0.0030 sec for our spectral Hermite method while DMS needs 0.0034 sec, i.e. 13% more computing time.

As a final conclusion, we appreciate that the presented algorithm is a useful algorithm for partial differential evolution equations on unbounded domains with soliton type solutions, taking into account the fact that the methods for calculating analytical solutions are difficult to apply for arbitrary initial conditions. Our algorithm acts directly on the equation on the whole real line, without imposing any unnatural boundary conditions. The combination between the spectral Hermite method (formulated in spectral space) and the

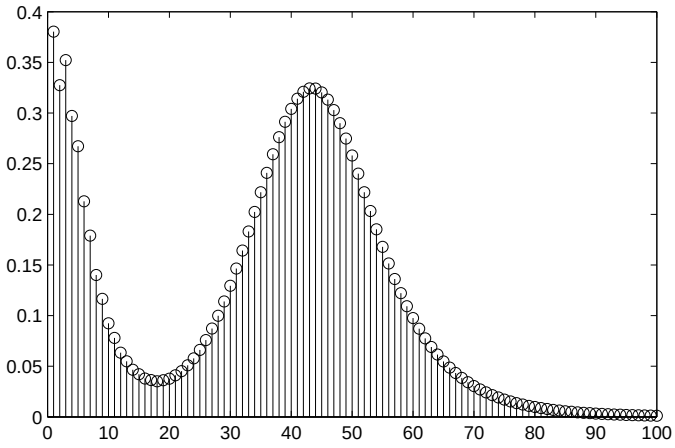

FIGURE 2. The first 100 coefficients of the solution at $t = 4$

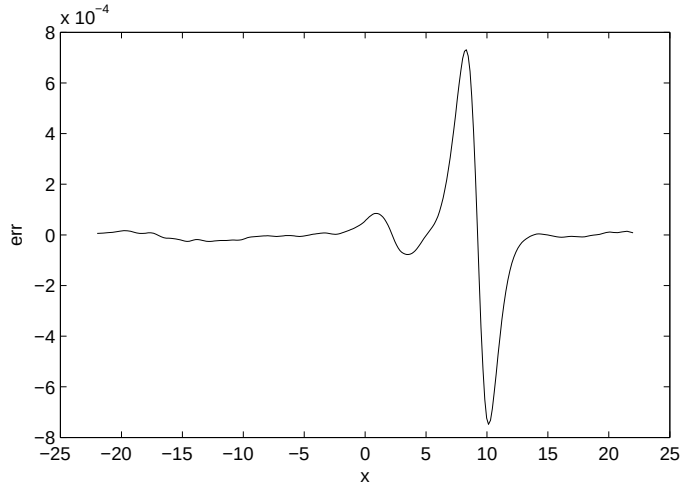

FIGURE 3. The error at $t = 4$

exponential integrators (for the resulting differential system) give an accurate, fast and stable algorithm with a reasonable time step and spatial grid.

Acknowledgement. This work was financially supported through the project CNCSIS 1348/2006.

References

[1] ***, *Exponential Integrators*, http://www.math.ntnu.no/num/expint/

[2] K. Brauer, *The Korteweg-de Vries Equation: History, exact Solutions, and graphycal Representation*, http://www.usf.uos.de/~kbrauer/solitons.html

[3] E. Celledoni, D. Cohen and B. Owren, *Symmetric Exponential Integrators with an Application to the Cubic Schrödinger Equation*, Found Comput Math, **8**(2008), 303-317.

[4] P.G. Drazin and R.S. Johnson, *Solitons: an introduction*, Cambridge University Press, 1989.

[5] D. Trif, *Matlab Package for the Schrödinger Equation*, J. Math. Chem., **43**(3)(2008), 1163-1176.

[6] D. Trif, *HermiteEig*, http://www.mathworks.com/matlabcentral/fileexchange/11522

[7] J. Shen and L.-L. Wang, *Some Recent Advances on Spectral Methods for Unbounded Domains*, Comm. Comp. Phys. **5**, No. 2-4 (2009), 195-241.

[8] J. A. C. Weideman and S. C. Reddy, *A MATLAB Differentiation Matrix Suite*, ACM TOMS, **26** (2000), 465-519,
http://www.mathworks.com/matlabcentral/fileexchange/29

Received: September 29, 2008; Accepted: January 12, 2009.