ESAIM: PROCEEDINGS, March 2012, Vol. 35, p. 216-221
Fédération Denis Poisson (Orléans-Tours) et E. Trélat (UPMC), Editors

A MATRIX-BASED NUMERICAL METHOD FOR THE SIMULATION OF THE TWO-DIMENSIONAL SINE-GORDON EQUATION*,**

Francisco de la Hoz[1]

Abstract. This paper describes a numerical method for the two-dimensional sine-Gordon equation over a rectangular domain using differentiation matrices, in the theoretical frame of matrix differential equations.

Résumé. Cette courte note décrit une méthode numérique pour l'équation de sine-Gordon bidimensionnelle sur un domaine rectangulaire en utilisant des matrices de différenciation, dans le cadre théorique des équations différentielles matricielles.

Introduction

In this paper, we consider the sine-Gordon equation (SGE):

$$u_{tt}(\bar{x},t) = \triangle u(\bar{x},t) - \sin(u(\bar{x},t)), \qquad u : \mathbb{R}^N \times \mathbb{R}^+ \to \mathbb{R}. \tag{1}$$

This equation is particularly relevant, because it appears in many areas in mathematics, mechanics and theoretical physics. It describes, for instance, the deformation of a nonlinear crystal-lattice, dislocation in solids, properties of ferromagnets, etc. For many of its applications, see for example [19, pg. 1448], [12, pg. 199], or the introduction of [1], together with their references.

As with other equations, SGE has been most intensively studied for the one-dimensional case. Nevertheless, in the last years, there have appeared a pretty large number of papers devoted to studying numerically the two-dimensional SGE defined over a rectangular domain:

$$u_{tt}(x,y,t) = u_{xx} + u_{yy} - \sin(u(x,y,t)), \qquad (x,y) \in [-L_x,+L_x] \times [-L_y,+L_y], \tag{2}$$

and imposing almost exclusively homogeneous Neumann boundary conditions ([3], [4], [21, pg. 134], [11], [17], [18], [10], etc.). Despite using different techniques, the authors do not compare their methods; and their results for the test problems taken from [6] look very similar. On the other hand, in a recently submitted paper [9] (which can be obtained on request), F. de la Hoz and F. Vadillo have developed a pseudo-spectral matrix-based method to solve numerically SGE over an axiparallel rectangular domain in an arbitrary number of spatial dimensions, and with arbitrary time-dependent Neumann boundary conditions. The idea is to discretize the domain at the Chebyshev-Lobatto nodes, approximating the partial derivatives by means of differentiation

* *This work was supported by MEC (Spain), with the project MTM2007-62186, and by the Basque Government, with the project IT-305-07.*

** *I want to thank Fernando Vadillo for his ongoing collaboration on this research area.*

[1] University of the Basque Country, Department of Applied Mathematics, Plaza de La Casilla 3, 48012 Bilbao (Spain); e-mail: francisco.delahoz@ehu.es

matrices, and to use a fourth-order Runge-Kutta method with integrating factor [20, chap. 10] to advance in time, avoiding completely the calculation of matrix exponentials and of tensorization.

In this short paper, we announce the main results of [9] for the two dimensional case, i.e., we develop a matrix-based numerical method for the two-dimensional SGE over a rectangular domain with arbitrary Neumann boundary conditions. The structure of the paper is as follows: In Section 1, we formulate the matrix problem; in Section 2, we solve the corresponding linear problem; in Section 3, we develop a first-order numerical scheme with integrating factor; in Section 4, we perform the numerical tests; and, finally, in Section 5, we draw the main conclusions.

1. Formulation of the matrix problem

Spectral methods have been successfully applied to time-dependent partial differential equations (PDE) and there is an ample literature on this subject (see for instance [14], [2] and [20], as well as the more classic references [15] and [5]). The idea is to approximate the solution $u(x, y, t)$ by a finite sum:

$$u(x,y,t) \approx U(x,y,t) = \sum_{k=0}^{n_x} \sum_{l=0}^{n_y} a_{kl}(t)\ \phi_k\left(\frac{x}{L_x}\right) \phi_l\left(\frac{y}{L_y}\right), \tag{3}$$

where $\phi_m(\cdot) = \cos(m \arccos(\cdot))$ is the Chebyshev polynomial of degree m; $x \in [-L_x, L_x]$ and $y \in [-L_y, L_y]$. This approximation is an equality (i.e., a spectral equality) for a large enough number of addends, i.e., a large enough spectrum, except for errors smaller than the accuracy of the machine.

Chebyshev polynomials appear when dealing with non-periodic problems like (2), while, for periodic problems, trigonometric polynomials are the correct choice. There are different approaches to determine the expansion coefficients $a_{kl}(t)$: we will focus on pseudo-spectral methods, where $a_{kl}(t)$ are required to make the residual equal zero at as many (suitably chosen) spatial points (x_j, y_i) as possible. In this case, the natural choice are the Lobatto-Chebyshev nodes, i.e., $(x_j, y_i) = (L_x \cos(j\pi/n_x), L_y \cos(i\pi/n_y))$, $0 \le j \le n_x$, $0 \le i \le n_y$.

In our matrix-based philosophy, we do not obtain explicitly the coefficients $a_{kl}(t)$. Instead, our evolution variable is the matrix $\mathbf{U}(t) \in \mathcal{M}_{(n_y+1)\times(n_x+1)}$, where $U_{ij}(t) \equiv U(x_j, y_i, t)$, $0 \le j \le n_x$, $0 \le i \le n_y$. Notice that we write (x_j, y_i) rather than (x_i, y_j), to be coherent with MATLAB commands, such as `meshgrid`. To discretize (2), we approximate u_{xx} and u_{yy} by means of the Chebyshev differentiation matrix $\mathbf{D}$ [20, chap. 6] [22] [13]:

$$u_{yy}(x_j, y_i) \approx \left(\frac{1}{L_y^2}\mathbf{D}^2\right) \cdot \mathbf{U}, \qquad\qquad u_{xx}(x_j, y_i) \approx \mathbf{U} \cdot \left(\frac{1}{L_x^2}\mathbf{D}^2\right)^T,$$

where $\mathbf{D}^T$ denotes the transpose of $\mathbf{D}$. Then, (2) becomes

$$\mathbf{U}_{tt}(t) = \mathbf{U}(t) \cdot \left(\frac{1}{L_x^2}\mathbf{D}^2\right)^T + \left(\frac{1}{L_y^2}\mathbf{D}^2\right) \cdot \mathbf{U}(t) - \sin(\mathbf{U}(t)), \tag{4}$$

where the sine is applied pointwise. In the discretization of (2), only the inner elements U_{ij} of the matrix are considered, i.e., those with $1 \le j \le n_x - 1$, $1 \le i \le n_y - 1$. Therefore, we have to recover the border elements U_{0j}, $U_{n_y j}$, U_{i0} and $U_{i n_x}$ in function of the inner elements of $\mathbf{U}$ by means of the boundary conditions corresponding to (2). More precisely, if $\tilde{\mathbf{U}} \in \mathcal{M}_{(n_y-1)\times(n_x-1)}$ denotes the inner points of $\mathbf{U}$, then it is not difficult [7] to find matrices $\mathbf{P}_1 \in \mathcal{M}_{(n_y+1)\times(n_x-1)}$, $\mathbf{P}_2 \in \mathcal{M}_{(n_y-1)\times(n_x+1)}$ and $\mathbf{Q} \in \mathcal{M}_{(n_y+1)\times(n_x+1)}$, such that

$$\mathbf{U}(t) = \mathbf{P}_1 \cdot \tilde{\mathbf{U}}(t) \cdot \mathbf{P}_2 + \mathbf{Q}(t) \tag{5}$$

is a spectral equality. $\mathbf{P}_1$ and $\mathbf{P}_2$ are always time-independent, while $\mathbf{Q}$ is time-independent only if the boundary conditions are time-independent, being zero if we are dealing with homogeneous boundary conditions.

Introducing (5) into (4) and restricting ourselves to the inner points, (4) becomes

$$\mathbf{U}_{tt}(t) = \mathbf{A}\cdot\mathbf{U}(t) + \mathbf{U}(t)\cdot\mathbf{B}^T + \mathbf{C}(t) - \sin(\mathbf{U}(t)), \tag{6}$$

where, in order not to burden the notation, and without lost of generality, we have omitted the tildes of $\tilde{\mathbf{U}}$. $\mathbf{C}$ is time-independent if the boundary conditions are time-independent, and zero if the boundary conditions are homogeneous. $\mathbf{A}$ and $\mathbf{B}$ are actually the second-order Chebyshev differentiating matrices with homogeneous Neumann boundary-conditions and applied to the inner nodes; if $n_x = n_y$, then $L_y^2\mathbf{A} = L_x^2\mathbf{B}$. We will solve numerically (6) and, once calculated its evolution, will recover the border elements of $\mathbf{U}(t)$ from (5).

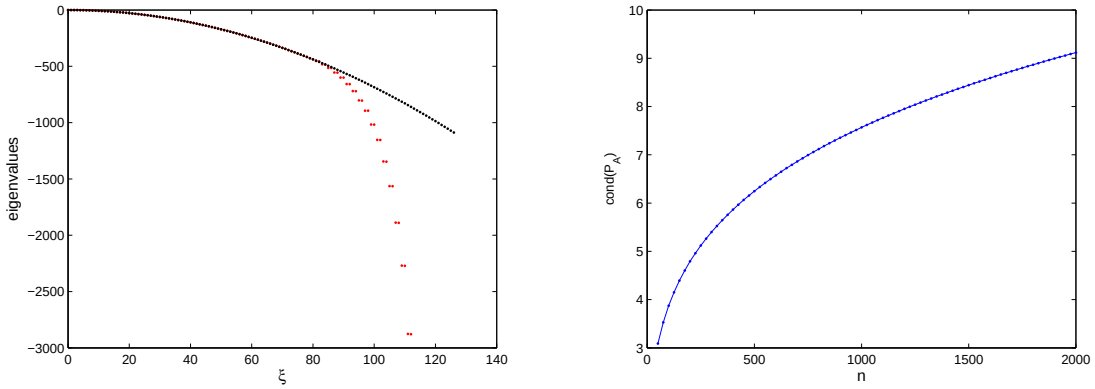

FIGURE 1. Left: Eigenvalues $\lambda_\xi(\mathbf{A})$ of the second-order differentiation matrix $\mathbf{A}$ with homogeneous Neumann boundary conditions, $n_y = 128$, $L_y = 6$ (red), versus the corresponding eigenvalues $\lambda_\xi = -(\xi\pi/2L_y)^2$ of the continuous problem $u_{xx} = \lambda u, u_x(-L_y) = u_x(+L_y) = 0$ (black). Right: Condition number of $\mathbf{P}_A$, as a function of the number of spatial nodes n_y.

At this point, two **crucial** observations should be done. The first one is that all the eigenvalues of $\mathbf{A}$ and $\mathbf{B}$ are negative, except for one, which equals zero except maybe for infinitesimal rounding errors. This fact, which is required for stability, is evident from a numerical point of view from the left hand-side of Figure 1. The second crucial observation is the well-conditionedness of the matrices of eigenvectors $\mathbf{P}_A$ and $\mathbf{P}_B$ that appear in the diagonal decomposition of $\mathbf{A}$ and $\mathbf{B}$, i.e., $\mathbf{D}_A = \mathbf{P}_A^{-1}\cdot\mathbf{A}\cdot\mathbf{P}_A$ and $\mathbf{D}_B = \mathbf{P}_B^{-1}\cdot\mathbf{B}\cdot\mathbf{P}_B$; this is required to avoid direct calculation of exponential matrices, which is a central idea of this paper. In the right-hand side of Figure 1, we have plotted the 2-norm condition number of the eigenvector matrix $\mathbf{P}_A$ of $\mathbf{A}\in\mathcal{M}_{(n_y-1)\times(n_y-1)}$, for $L = 6$, and different n_y, being evident that the condition numbers are small and grow sublinearly.

2. Solving the linear problem

The integrating-factor method is based on the idea that a problem with a linear part plus a nonlinear one can be transformed, so that its linear part is solved exactly (see [20, chap. 10], [8], and their references). Hence, to apply an integrating factor to (6), we need to solve first its linear part

$$\mathbf{U}_{tt} = \mathbf{A}\cdot\mathbf{U} + \mathbf{U}\cdot\mathbf{B}^T, \tag{7}$$

with initial data $\mathbf{U}(0)$, $\mathbf{U}_t(0)$. Denoting $\mathbf{V}(t)\equiv\mathbf{U}_t(t)$, (7) can be solved efficiently by the following theorem [9]:

Theorem 2.1. *Given the time-independent matrices* $\mathbf{A} \in \mathcal{M}_{(n_y-1)\times(n_y-1)}$, *and* $\mathbf{B} \in \mathcal{M}_{(n_x-1)\times(n_x-1)}$, *with diagonal decompositions* $\mathbf{D}_A = \mathbf{P}_A^{-1} \cdot \mathbf{A} \cdot \mathbf{P}_A$ *and* $\mathbf{D}_B = \mathbf{P}_B^{-1} \cdot \mathbf{B} \cdot \mathbf{P}_B$, *the solution of* (7) *is*

$$\begin{aligned}
\mathbf{U}(t) &= \mathbf{P}_A \cdot \left[\left(\mathbf{P}_A^{-1} \cdot \mathbf{U}(0) \cdot (\mathbf{P}_B^{-1})^T\right) \circ \cos\left(t\sqrt{-\Lambda_{AB}}\right)\right] \cdot \mathbf{P}_B^T \\
&\quad + \mathbf{P}_A \cdot \left[\left(\mathbf{P}_A^{-1} \cdot \mathbf{U}_t(0) \cdot (\mathbf{P}_B^{-1})^T\right) \circ \left(\sqrt{-\Lambda_{AB}}\right)^{-1} \circ \sin\left(t\sqrt{-\Lambda_{AB}}\right)\right] \cdot \mathbf{P}_B^T \\
\mathbf{V}(t) &= \mathbf{P}_A \cdot \left[-\left(\mathbf{P}_A^{-1} \cdot \mathbf{U}(0) \cdot (\mathbf{P}_B^{-1})^T\right) \circ \sqrt{-\Lambda_{AB}} \circ \sin\left(t\sqrt{-\Lambda_{AB}}\right)\right] \cdot \mathbf{P}_B^T \\
&\quad + \mathbf{P}_A \cdot \left[\left(\mathbf{P}_A^{-1} \cdot \mathbf{U}_t(0) \cdot (\mathbf{P}_B^{-1})^T\right) \circ \cos\left(t\sqrt{-\Lambda_{AB}}\right)\right] \cdot \mathbf{P}_B^T,
\end{aligned} \tag{8}$$

where all the operations are ***pointwise****;* $\circ$ *denotes the Hadamard or pointwise product of two matrices [16]; and* $\mathbf{\Lambda}_{AB} = [\lambda_{ij}]$ *is the matrix whose elements are* $\lambda_{ij} = \lambda_i(\mathbf{D}_A) + \lambda_j(\mathbf{D}_B)$, *for* $i = 1, \ldots, n_y - 1$, $j = 1, \ldots, n_x - 1$.

Observe that if $\lambda_{ij} = 0$, we take $(\sqrt{-\lambda_{ij}})^{-1} \sin(t\sqrt{-\lambda_{ij}}) \equiv t$, so **zero eigenvalues cause no concern**. Moreover, since we have proven numerically (left-hand side of Figure 1) that $\lambda_{ij} \leq 0$, $\mathbf{U}(t)$ and $\mathbf{V}(t)$ are real and bounded $\forall t > 0$. On the other hand, if there were $\lambda_{ij} > 0$, there would be stability issues as $t \to \infty$.

3. Integrating factor with forward Euler discretization

Let us transform (6) by means of the vec operator [16]:

$$\begin{pmatrix} \mathrm{vec}(\mathbf{U}) \\ \mathrm{vec}(\mathbf{V}) \end{pmatrix}_t = \mathbf{M} \cdot \begin{pmatrix} \mathrm{vec}(\mathbf{U}) \\ \mathrm{vec}(\mathbf{V}) \end{pmatrix} + \begin{pmatrix} 0 \\ \mathrm{vec}(\mathbf{C}) - \sin(\mathrm{vec}(\mathbf{U})) \end{pmatrix}, \tag{9}$$

where $\mathbf{V} = \mathbf{U}_t$, and $\mathbf{M}$ is the block matrix

$$\mathbf{M} = \begin{pmatrix} 0 & \mathbf{I} \\ \mathbf{B} \oplus \mathbf{A} & 0 \end{pmatrix}, \tag{10}$$

where $\oplus$ denotes the Kronecker sum [16], and we have used that $\mathrm{vec}(\mathbf{A} \cdot \mathbf{U}(t) + \mathbf{U}(t) \cdot \mathbf{B}^T) \equiv (\mathbf{B} \oplus \mathbf{A}) \cdot \mathrm{vec}(\mathbf{U})$. In (9), there is a linear part plus a non-linear part; to get rid of the linear part, we multiply at both sides by the integrating factor $\exp(-t\,\mathbf{M})$, where exp denotes the matrix exponential, getting

$$\left(\exp(-t\,\mathbf{M}) \cdot \begin{pmatrix} \mathrm{vec}(\mathbf{U}) \\ \mathrm{vec}(\mathbf{V}) \end{pmatrix} \right)_t = \exp(-t\,\mathbf{M}) \cdot \begin{pmatrix} 0 \\ \mathrm{vec}(\mathbf{C}) - \sin(\mathrm{vec}(\mathbf{U})) \end{pmatrix}. \tag{11}$$

Using a forward Euler discretization in time, it becomes

$$\exp(-t^{n+1}\mathbf{M}) \cdot \begin{pmatrix} \mathrm{vec}(\mathbf{U}^{n+1}) \\ \mathrm{vec}(\mathbf{V}^{n+1}) \end{pmatrix} - \exp(-t^n\mathbf{M}) \cdot \begin{pmatrix} \mathrm{vec}(\mathbf{U}^n) \\ \mathrm{vec}(\mathbf{V}^n) \end{pmatrix} = \Delta t \exp(-t^n\mathbf{M}) \cdot \begin{pmatrix} 0 \\ \mathrm{vec}(\mathbf{C}^n) - \sin(\mathrm{vec}(\mathbf{U}^n)) \end{pmatrix}. \tag{12}$$

Multiplying at both sides by $\exp(t^{n+1}\mathbf{M})$, we get finally

$$\begin{pmatrix} \mathrm{vec}(\mathbf{U}^{n+1}) \\ \mathrm{vec}(\mathbf{V}^{n+1}) \end{pmatrix} = \exp(\Delta t\ \mathbf{M}) \cdot \left[\begin{pmatrix} \mathrm{vec}(\mathbf{U}^n) \\ \mathrm{vec}(\mathbf{V}^n) \end{pmatrix} + \Delta t \begin{pmatrix} 0 \\ \mathrm{vec}(\mathbf{C}^n) - \sin(\mathrm{vec}(\mathbf{U}^n)) \end{pmatrix} \right], \tag{13}$$

which is obviously the solution of (7) at time $t = \Delta t$, with initial data $\mathbf{U}(0) = \mathbf{U}^n$, and $\mathbf{V}(0) = \mathbf{V}^n + \Delta t\,(\mathrm{vec}(\mathbf{C}^n) - \sin(\mathrm{vec}(\mathbf{U}^n)))$; hence, by Theorem 2.1,

$$\begin{aligned}
\mathbf{U}^{n+1} &= \mathbf{P}_A \cdot \Bigg[\left(\mathbf{P}_A^{-1} \cdot \mathbf{U}^n \cdot (\mathbf{P}_B^{-1})^T\right) \circ \cos\left(\Delta t \sqrt{-\Lambda_{AB}}\right) \\
&\quad + \left(\mathbf{P}_A^{-1} \cdot [\mathbf{V}^n + \Delta t\,(\mathbf{C}^n - \sin(\mathbf{U}^n))] \cdot (\mathbf{P}_B^{-1})^T\right) \circ \left(\sqrt{-\Lambda_{AB}}\right)^{-1} \circ \sin\left(\Delta t \sqrt{-\Lambda_{AB}}\right) \Bigg] \cdot \mathbf{P}_B^T, \\
\mathbf{V}^{n+1} &= \mathbf{P}_A \cdot \Bigg[- \left(\mathbf{P}_A^{-1} \cdot [\mathbf{V}^n + \Delta t\,(\mathbf{C}^n - \sin(\mathbf{U}^n))] \cdot (\mathbf{P}_B^{-1})^T\right) \circ \sqrt{-\Lambda_{AB}} \circ \sin\left(\Delta t \sqrt{-\Lambda_{AB}}\right) \\
&\quad + \left(\mathbf{P}_A^{-1} \cdot \mathbf{U}^n \cdot (\mathbf{P}_B^{-1})^T\right) \circ \cos\left(\Delta t \sqrt{-\Lambda_{AB}}\right) \Bigg] \cdot \mathbf{P}_B^T. \qquad (14)
\end{aligned}$$

In brief, we have tensorized (6), applied the integrating factor, discretized the equation in time, and destensorized the resulting scheme. The previous ideas can be easily extended to higher-order schemes, by applying a higher-order discretization to (11). In the following section, for comparison's sake, we have also considered a fourth-order Runge-Kutta discretization of (11).

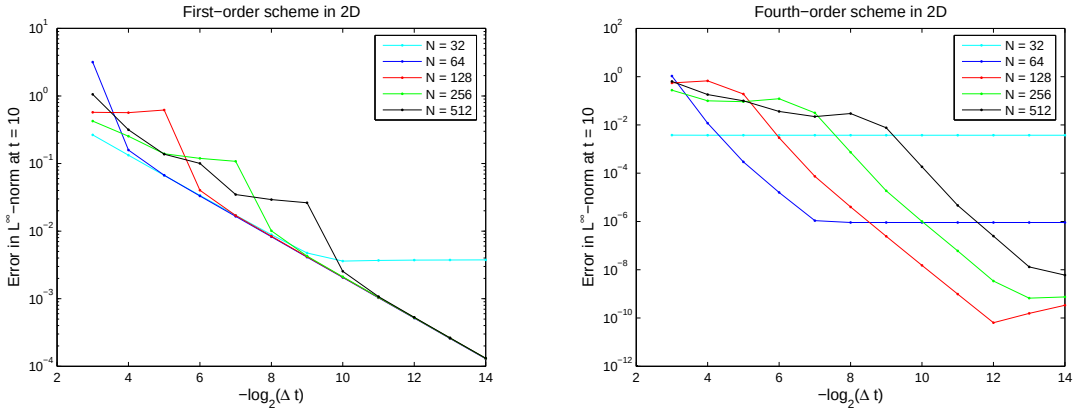

FIGURE 2. Errors in L^∞-norm at $t = 10$, for $L_x = L_y = 6$, and different $n_x = n_y = N$ and Δt, corresponding to the first-order scheme (left), and to the fourth-order scheme (right).

4. Numerical tests

We have considered the theoretical solution of (2), $u(x, y, t) = 4 \arctan(\exp[(\sqrt{2}/2)x + (\sqrt{6}/2)y - t])$, $(x, y) \in [-6, 6]^2$, $\forall t \geq 0$; introducing exactly the initial data $u(x, y, 0)$ and $u_t(x, y, 0)$, and the inhomogeneous Neumann boundary conditions $\partial_x u(\pm 6, y, t)$, and $\partial_y u(x, \pm 6, t)$, $\forall t$. We have executed the first-order scheme described above, as well as a fourth-order Runge-Kutta with integrating factor [9], for $L_x = L_y = 6$ and different $n_x = n_y$. Figure 2 shows the errors obtained in L^∞-norm at $t = 10$, for different Δt; notice that $\max_{(x,y)} u(x, y, 10) \approx 5.48$. The left-hand side and right-hand side correspond respectively to the first and fourth-order schemes; the orders are evident from the slopes of the (approximately) straight lines that appear: while the first-order may need a prohibitive Δt to yield good results, the fourth-order method offers a highly remarkable accuracy.

To obtain the highest possible accuracy, a minimum number of nodes N is required. For example, with $N = 32$ and $N = 64$, the minimum possible errors are only about $3.7581 \cdot 10^{-3}$ and $9.0774 \cdot 10^{-7}$, respectively. On the other hand, the method is very stable, even for big N.

From the right-hand side, it is apparent that if we double N, we have to divide Δt by four to get a similar accuracy. Therefore, we have a $\Delta t = \mathcal{O}(1/N^2)$ restriction if we want to get the maximum accuracy. Although

the results get slightly worse when we increase N, they are still very good, even for large N. For instance, for $N = 512$ and $\Delta t = 2^{-14}$, we have an absolute error of about $6.0143 \cdot 10^{-9}$.

5. Conclusions

We have developed a new numerical matrix-based method with integrating factor to solve efficiently and accurately the two-dimensional SGE (2), avoiding the explicit calculation of matrix exponentials and the use of Kronecker tensor products. To understand why avoiding tensor products is vital, let be $\mathbf{A}, \mathbf{B} \in \mathcal{M}_{N\times N}$, where all the components of both $\mathbf{A}$ and $\mathbf{B}$ are positive (so that there are no cancellations); then it can be shown that $\mathbf{B} \oplus \mathbf{A} \in \mathcal{M}_{N^2\times N^2}$ has exactly $2N^3 - N^2$ non-zero elements, i.e. a sparsity ratio of $\mathcal{O}(1/N)$. Therefore, if $N = 511$, $\mathbf{A}$ and $\mathbf{B}$ have just 261121 elements, versus $\mathbf{B} \oplus \mathbf{A}$, which has 68184176641 elements, of which 266604541 are nonzero! In other words, the problem becomes quickly intractable. Another virtue of the non-tensor approach is that it can be extended to higher dimensions [9], overcoming the curse of dimensionality.

The method can be applied, with little modification, to other types of nonlinear Klein-Gordon equations, with other types of boundary conditions. Furthermore, the techniques of this paper are not necessarily restricted to axiparallel rectangular domains. Indeed, as long as the spatially semi-discretized problem can be written in the form of (6), with eventually another nonlinear term, the general ideas in this paper are applicable.

References

1. E. L. Aero, A. N. Bulygin, and Y. V. Pavlov, *Solutions of the three-dimensional sine-Gordon equation*, Theoretical and Mathematical Physics **158** (2009), 313–319.
2. J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, Dover, 2001.
3. A. G. Bratsos, *The solution of the two-dimensional sine-Gordon equation using the method of lines*, J. Comput. Appl. Math. **206** (2007), 251–277.
4. ———, *An improved numerical scheme for the sine-Gordon equation in 2+1 dimensions*, Int. J. Numer. Meth. Engng **75** (2008), 787–799.
5. C. Canuto, M. Y. Hussain, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics*, Springer, 1988.
6. P. L. Christiansen and P. S. Lomdahl, *Numerical solutions of 2 + 1 dimensional sine-Gordon solitons*, Physica 2D (1981), 482–494.
7. F. de la Hoz and F. Vadillo, *A Simple Method for Two-Dimensional Advection-Diffusion Equations via Operational Matrices*, preprint.
8. ———, *An integrating factor for nonlinear Dirac equations*, Computer Physics Communications **181** (2010), 1195–1203.
9. ———, *Numerical simulation of the N-dimensional sine-Gordon equation via operational matrices*, Computer Physics Communications **183** (2012), 864–879.
10. M. Dehghan and A. Ghesmati, *Numerical simulation of two-dimensional sine-Gordon solitons via a local weak meshless technique based on the radial point interpolation method (RPIM)*, Computer Physics Communications **181** (2010), 772–786.
11. M. Dehghan and A. Shokri, *A numerical method for solution of the two-dimensional sine-Gordon equation using the radial basis functions*, Mathematics and Computers in Simulation **79** (2008), 700–715.
12. P. G. Drazin and R. S. Johnson, *Solitons : an introduction*, Cambridge University Press, 1989.
13. E. M. E. Elbarbary and S. M. El-Sayed, *Higher order pseudospectral differentiation matrices*, Applied Numerical Mathematics **55** (2005), no. 4, 425–438.
14. B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, 1998.
15. D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*, SIAM, 1977.
16. N. J. Higham, *Function of Matrices. Theory and Computation*, SIAM, 2008.
17. D. Mirzaei and M. Dehghan, *Boundary element solution of the two-dimensional sine-Gordon equation using continuous linear elements*, Engineering Analysis with Boundary Elements **33** (2009), 12–24.
18. ———, *Meshless local Petrov-Galerkin (MLPG) approximation to the two dimensional sine-Gordon equation*, J. Comput. Appl. Math. **233** (2010), 2737–2754.
19. A. C. Scott, F. Y. F. Chu, and D. W. McLaughlin, *The Soliton: A New Concept in Applied Science*, Proceedings of the IEEE **61** (1973), no. 10, 1443–1483.
20. L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, 2000.
21. E. H. Twizell, *Computational Methods for Partial Differential Equations*, John Wiley & Sons, 1984.
22. J. A. C. Weideman and S. C. Reddy, *A MATLAB Differentiation Matrix Suite*, ACM Trans. Math. Software **26** (2000), no. 4, 465–519.