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

FRANCISCO DE LA HOZ¹

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):

\[ u_{tt}(\bar{x}, t) = \Delta u(\bar{x}, t) - \sin(u(\bar{x}, t)), \quad u : \mathbb{R}^N \times \mathbb{R}^+ \to \mathbb{R}. \]  

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:

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

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 axisparallel 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:

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

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., \( (x_j, y_i) = (L_x \cos(j \pi/n_x), L_y \cos(i \pi/n_y)) \), \( 0 \leq j \leq n_x, 0 \leq i \leq n_y \).

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

\[
\begin{align*}
&u_{yy}(x_j, y_i) \approx \left( \frac{1}{L_y^2} D^2 \right) \cdot U, \\
&u_{xx}(x_j, y_i) \approx U \cdot \left( \frac{1}{L_x^2} D^2 \right)^T,
\end{align*}
\]

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

\[
U_{tt}(t) = U(t) \cdot \left( \frac{1}{L_x^2} D^2 \right)^T + \left( \frac{1}{L_y^2} D^2 \right) \cdot U(t) - \sin(U(t)),
\]

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 \( 1 \leq j \leq n_x - 1, 1 \leq i \leq n_y - 1 \). Therefore, we have to recover the border elements \( U_{0j}, U_{nj}, U_{i0} \) and \( U_{in} \) in function of the inner elements of \( U \) by means of the boundary conditions corresponding to (2). More precisely, if \( \hat{U} \in \mathcal{M}_{(n_x-1) \times (n_y-1)} \) denotes the inner points of \( U \), then it is not difficult [7] to find matrices \( P_1 \in \mathcal{M}_{(n_x+1) \times (n_y-1)} \), \( P_2 \in \mathcal{M}_{(n_x-1) \times (n_y+1)} \) and \( Q \in \mathcal{M}_{(n_x+1) \times (n_y+1)} \), such that

\[
U(t) = P_1 \cdot \hat{U}(t) \cdot P_2 + Q(t)
\]

is a spectral equality. \( P_1 \) and \( P_2 \) are always time-independent, while \( Q \) 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

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

where, in order not to burden the notation, and without lost of generality, we have omitted the tildes of $\mathbf{U}$. $\mathbf{C}$ 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).

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

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

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]:

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1}
\caption{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 $n_y$.}
\end{figure}
Given the time-independent matrices \( A \in \mathcal{M}_{(n_y-1) \times (n_y-1)} \), and \( B \in \mathcal{M}_{(n_x-1) \times (n_x-1)} \), with diagonal decompositions \( D_A = P_A^{-1} \cdot A \cdot P_A \) and \( D_B = P_B^{-1} \cdot B \cdot P_B \), the solution of (7) is

\[
U(t) = P_A \cdot \left[ (P_A^{-1} \cdot U(0) \cdot (P_B^{-1})^T) \circ \cos \left( t \sqrt{-\Lambda_{AB}} \right) \right] \cdot P_B^T \\
+ P_A \cdot \left[ (P_A^{-1} \cdot U(t) \cdot (P_B^{-1})^T) \circ \left( \sqrt{-\Lambda_{AB}} \right)^{-1} \circ \sin \left( t \sqrt{-\Lambda_{AB}} \right) \right] \cdot P_B^T \\
V(t) = P_A \cdot \left[ - (P_A^{-1} \cdot U(0) \cdot (P_B^{-1})^T) \circ \sqrt{-\Lambda_{AB}} \circ \sin \left( t \sqrt{-\Lambda_{AB}} \right) \right] \cdot P_B^T \\
+ P_A \cdot \left[ (P_A^{-1} \cdot U(t) \cdot (P_B^{-1})^T) \circ \cos \left( t \sqrt{-\Lambda_{AB}} \right) \right] \cdot P_B^T,
\]

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

Observe that if \( \lambda_{ij} = 0 \), we take \( \sqrt{-\lambda_{ij}} = 0 \), so zero eigenvalues cause no concern. Moreover, since we have proven numerically (left-hand side of Figure 1) that \( \lambda_{ij} \leq 0 \), \( U(t) \) and \( V(t) \) 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]:

\[
\begin{pmatrix}
\text{vec}(U) \\
\text{vec}(V)
\end{pmatrix}_t = M \cdot \begin{pmatrix}
\text{vec}(U) \\
\text{vec}(V)
\end{pmatrix} + \begin{pmatrix}
0 \\
\text{vec}(C) - \sin(\text{vec}(U))
\end{pmatrix},
\]

where \( V = U_t \), and \( M \) is the block matrix

\[
M = \begin{pmatrix}
0 & A \\
B \oplus A & 0
\end{pmatrix},
\]

where \( \oplus \) denotes the Kronecker sum [16], and we have used that \( \text{vec}(A \cdot U(t) + U(t) \cdot B^T) \equiv (B \oplus A) \cdot \text{vec}(U) \). 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 \( \exp(-tM) \), where \( \exp \) denotes the matrix exponential, getting

\[
\begin{pmatrix}
\exp(-tM) \cdot \begin{pmatrix}
\text{vec}(U) \\
\text{vec}(V)
\end{pmatrix}
\end{pmatrix}_t = \exp(-tM) \cdot \begin{pmatrix}
0 \\
\text{vec}(C) - \sin(\text{vec}(U))
\end{pmatrix}.
\]

Using a forward Euler discretization in time, it becomes

\[
\exp(-t^{n+1}M) \cdot \begin{pmatrix}
\text{vec}(U^{n+1}) \\
\text{vec}(V^{n+1})
\end{pmatrix} - \exp(-t^nM) \cdot \begin{pmatrix}
\text{vec}(U^n) \\
\text{vec}(V^n)
\end{pmatrix} = \Delta t \exp(-t^nM) \cdot \begin{pmatrix}
0 \\
\text{vec}(C^n) - \sin(\text{vec}(U^n))
\end{pmatrix}.
\]

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

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

where all the operations are pointwise; \( \circ \) denotes the Hadamard or pointwise product of two matrices [16]; and \( \Lambda_{AB} = \lambda_{ij} \) is the matrix whose elements are \( \lambda_{ij} = \lambda_i(D_A) + \lambda_j(D_B) \), for \( i = 1, \ldots, n_y - 1, j = 1, \ldots, n_x - 1 \).
which is obviously the solution of (7) at time \( t = \Delta t \), with initial data \( U(0) = U^n \), and \( V(0) = V^n + \Delta t (\text{vec}(C^n) - \sin(\text{vec}(U^n))) \); hence, by Theorem 2.1,

\[
U^{n+1} = P_A \cdot \left( P_A^{-1} \cdot U^n \cdot (P_B^{-1})^T \right) \circ \cos \left( \Delta t \sqrt{-\Lambda_{AB}} \right)
+ \left( P_A^{-1} \cdot [V^n + \Delta t (C^n - \sin(U^n))] \cdot (P_B^{-1})^T \right) \circ \left( \sqrt{-\Lambda_{AB}} \right)^{-1} \circ \sin \left( \Delta t \sqrt{-\Lambda_{AB}} \right) \cdot P_B^T,
\]

\[
V^{n+1} = P_A \cdot \left[ -\left( P_A^{-1} \cdot [V^n + \Delta t (C^n - \sin(U^n))] \cdot (P_B^{-1})^T \right) \circ \sqrt{-\Lambda_{AB}} \circ \sin \left( \Delta t \sqrt{-\Lambda_{AB}} \right) \right] \cdot P_B^T.
\]

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 Runge-Kutta discretization to (11). In the following section, for comparison’s sake, we have also considered a fourth-order Runge-Kutta discretization of (11).

![First-order scheme in 2D](image1)

![Fourth-order scheme in 2D](image2)

**Figure 2.** Errors in \( L^\infty \)-norm at \( t = 10 \), for \( L_x = L_y = 6 \), and different \( n_x = n_y = N \) and \( \Delta t \), 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 \( L^\infty \)-norm at \( t = 10 \), for different \( \Delta t \); 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 \( \Delta t \) 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 \( \Delta t \) 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 \( O(1/N) \). 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