A numerical study of two dimensional hyperbolic telegraph equation by modified B-spline differential quadrature method

R.C. Mittal, Rachna Bhatia

Department of Mathematics, IIT Roorkee, Roorkee 247667, Uttarakhand, India

ARTICLE INFO

Keywords:
Differential quadrature method
Hyperbolic telegraph equation
Modified B-spline basis functions
Thomas algorithm

ABSTRACT

The present paper uses a relatively new approach and methodology to solve second order two dimensional hyperbolic telegraph equation numerically. We use modified cubic B-spline basis functions based differential quadrature method for space discretization that reduces the problem into an amenable system of ordinary differential equations. The resulting system of ODEs in time subsequently have been solved by SSP-RK43 scheme. Stability of the scheme is studied using matrix stability analysis and found to be stable. The efficacy of proposed approach has been confirmed with seven numerical experiments, where comparison is made with some earlier work. It is clear that the results obtained are acceptable and are in good agreement with earlier studies. However, we obtain these results in much less CPU time. The method is very simple, efficient and produces very accurate numerical results in considerably smaller number of nodes and hence saves computational effort.

1. Introduction

In this paper, we consider the following second-order linear two-space dimensional hyperbolic telegraph equation

\[ u_{tt}(x, y, t) + 2axu_t(x, y, t) + \beta^2u(x, y, t) = u_{xx}(x, y, t) + u_{yy}(x, y, t) + f(x, y, t), \quad (x, y, t) \in \mathbb{R} \times (0, T], \]  

where \( \mathbb{R} \) is the region \([0, 1] \times [0, 0]\) in \(\mathbb{R}^2\) and \(0, T\) is the time interval. \(a, \beta\) are the constants. For \(a > 0, \beta = 0\), Eq. (1.1) represents a damped wave equation and for \(a > 0, \beta > 0\), it is called telegraph equation.

The initial conditions are given by,

\[
\begin{align*}
    u(x, y, 0) &= u_0(x, y), \\
    u_t(x, y, 0) &= v_0(x, y),
\end{align*}
\]  

(1.2)

The Dirichlet boundary conditions are

\[
\begin{align*}
    u(0, y, t) &= f_1(y, t), \\
    u(1, y, t) &= f_2(y, t), \\
    u(x, 0, t) &= f_3(x, t), \\
    u(x, 1, t) &= f_4(x, t),
\end{align*}
\]  

(1.3)

* Corresponding author.
E-mail addresses: rcmmfma@iitr.ernet.in (R.C. Mittal), rachnabhsingh@yahoo.com (R. Bhatia).

http://dx.doi.org/10.1016/j.amc.2014.07.060

© 2014 Elsevier Inc. All rights reserved.
The hyperbolic partial differential equations have significant role in formulating fundamental equations in atomic physics [25] and are also very useful in understanding various phenomena in applied sciences like engineering industry aerospace as well as in chemistry and biology too. On one hand vibrations of structures (e.g. buildings, machines and beams) can be easily analyzed and studied and on the other hand it is more convenient than ordinary diffusion equation in modeling reaction diffusion for such branches of sciences [7].

On delving through the literatures, we found that much effort has been taken for the numerical solution of 1D and 2D hyperbolic telegraph equations. Various numerical schemes were developed for one dimensional telegraph equation such as Taylor matrix method [5], dual reciprocity boundary integral method [7], unconditionally stable finite difference scheme [15], modified B-spline collocation method [26], Chebyshev tau method [32], interpolating scaling function method [25] etc. Numerical solution of linear hyperbolic telegraph equation in three space dimension has been proposed by Mohanty et al. [29] by constructing an unconditionally stable alternating direction implicit scheme. In [27], Mohanty propounded a unconditionally stable implicit difference scheme for one, two and three space dimension telegraphic equations. Dehghan et al. [13], used He’s variational iteration method for solving linear, variable coefficient, fractional derivative and multi space telegraph equations.

In the recent past much emphasis have been given in the literature for numerical solution of two dimensional hyperbolic telegraph equation (1.1). Bülbü and Sezer [4], proposed Taylor matrix method that converts the telegraph equation into the matrix equation. Dehghan and Ghesmati [6] have explored two meshless methods namely meshless local weak-strong (MLWS) and meshless local Petrov–Galerkin (MLPG) method for Eqs. (1.1)–(1.4). Also in [9], Eq. (1.1) is solved using higher order implicit collocation method. Numerical solution of 2D telegraph equation with variable coefficients has been tackled by Dehghan and Shorki [12]. Ding and Zhang [14] have discussed compact finite difference scheme which is of fourth order in both space and time. Mohanty and Jain [28] derived an unconditionally stable alternating direction implicit scheme for Eq. (1.1). A combination of boundary knot method (BKM) and analog equation method (AEM) for Eq. (1.1) has been proposed by Dehghan and Salehi [11]. A differential quadrature method, which approximates the solution of the problem on a finite dimensional space by using polynomials as the basis of the space is applied to the two dimensional telegraph equation with both Dirichlet and Neumann boundary conditions by Jiwari et al. [16].

Differential quadrature method (DQM) is a numerical discretization technique for the approximation of derivatives and has been successfully applied to solve various problem in biosciences, fluid dynamics, chemical engineering, etc [33]. DQM was introduced for the first time by Bellman et al. [3], in the framework to approximate the solution of differential equations. It follows that the partial derivatives of a function with respect to coordinate variable can be approximated by weighted linear combination of functional values at all grid points in the whole computational domain. The key of the DQM is to determine the weighting coefficient in the weighting sum by using some test functions. Various test functions have been used in the literature to calculate these weighting coefficients such as Legendre polynomials and spline functions by Bellman et al. [2,3] respectively. To improve Bellman’s approach, Quan and Chang [31,30], have proposed an explicit formulation using Lagrange interpolation polynomial as the test functions. Shu and Wu presented an implicit approach using radial basis functions as the test functions. After that Shu [33] proposed a general approach to determine the weighting coefficients for first order derivatives and higher order derivatives. A number of differential quadrature methods have been developed in the literature to approximate the solution of differential equations such as quartic B-spline differential quadrature method [19], sinc differential quadrature method [18], modified cubic B-spline differential quadrature method [1], local radial basis functions based differential quadrature method [10], polynomial based differential quadrature method [17], harmonic differential quadrature method [37] etc.

In this paper we present a modified cubic B-spline differential quadrature method (MCB-DQM) to solve two dimensional hyperbolic telegraph equation with Dirichlet and Neumann boundary conditions. B-splines are extensively used in the literature to develop various numerical methods, for detail see [1,8,20–24,26]. We use modified cubic B-spline basis functions [26], to compute the weighting coefficients. First the telegraph equation is converted into system of partial differential equations and then equations are discretized spatially by MCB-DQM. The obtained systems of ODEs in time are solved using SSP-RK43 [36] scheme and consequently the approximate solution is computed.

The outline of the paper is as follows. In Section 2, modified cubic B-spline differential quadrature method is introduced. In Section 3, numerical scheme using MCB-DQM for telegraph equation is presented. In Section 4, we discuss the stability of scheme. In Section 5, computational results for some test problems are illustrated and compared with some previous results and finally conclusions are included in Section 6.

2. Modified cubic B-spline differential quadrature method

To represent the mathematical formulation of two dimensional (DQM), first the region \( a < x < b \), \( c < y < d \) is discretized by taking \( N \) and \( M \) grid points in \( x \) and \( y \) direction respectively, such that \( h_x = x_{i+1} - x_i \) and \( h_y = y_{j+1} - y_j \). Then the \( n \)th order
partial derivatives of the function \( u(x, y, t) \) with respect to \( x \) at a point \((x_i, y_j)\), at any line \( y = y_j \) that is parallel to the \( x \) axis can be approximated as follows:

\[
\begin{align*}
\{ u^{(n)}_i(x_i, y_j, t) = \sum_{k=1}^{N} a_{ik}^{(n)} u(x_k, y_j, t), \quad i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, M. \tag{2.1} \end{align*}
\]

The \( m \)th order partial derivatives of the function \( u(x, y, t) \) with respect to \( y \) at a point \((x_i, y_j)\) at any line \( x = x_i \) that is parallel to the \( y \) axis can be approximated as follows:

\[
\begin{align*}
\{ u^{(m)}_j(x_i, y_j, t) = \sum_{k=1}^{M} b_{jk}^{(m)} u(x_i, y_k, t), \quad i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, M, \tag{2.2} \end{align*}
\]

where \( a_{ij}^{(n)} \) and \( b_{ij}^{(m)} \) represent the weighting coefficients related to \( u^{(n)}_i(x_i, y_j, t) \) and \( u^{(m)}_j(x_i, y_j, t) \) respectively, at the point \((x_i, y_j)\).

2.1. Cubic B-spline basis function

The cubic B-spline \( B_i(x) \) at the knots are given by:

\[
B_i(x) = \begin{cases} 
(x - x_i)^3, \quad &x \in [x_i, x_{i-1}), \\
(x - x_i)^3 - 4(x - x_i)^2 x, \quad &x \in [x_{i-1}, x_i), \\
(x_i - x)^3 - 4(x_i - x)^2 x, \quad &x \in [x_i, x_{i+1}), \\
(x_i - x)^3, \quad &x \in [x_{i+1}, x_{i-1}), \\
0, \quad &\text{otherwise}. 
\end{cases} \tag{2.3}
\]

Each cubic B-spline covers four elements so that an element is covered by four cubic B-splines and the set of functions \( \{B_0, B_1, \ldots, B_{N-1}, B_N\} \), forms a basis for the function define over the region \( R \), with the obvious adjustment of the boundary base functions to avoid undefined knots.

\( B_i(x) \) and its derivatives using (2.3) are tabulated in Table 1. Since \( B_i(x) = 0 \) outside the interval \( [x_{i-2}, x_{i+2}) \), so there is no need to tabulate \( B_i(x) \) for other values of \( x \).

The modified cubic B-spline basis functions are in number matches with the selected point in the given domain are denoted by \( \hat{B}_i(x) \), \( i = 1, 2, \ldots, N \) and form the basis over the region \( R \). These basis functions are obtained using cubic B-spline basis function (2.3), such that resulting matrix system of equations is diagonally dominant:

\[
\begin{align*}
\hat{B}_1(x) = B_1(x) + 2B_0(x), & \quad \text{for } i = 1, \\
\hat{B}_2(x) = B_2(x) - B_0(x), & \quad \text{for } i = 2, \\
\hat{B}_i(x) = B_i(x), & \quad \text{for } i = 3, \ldots, N-2, \\
\hat{B}_{N-1}(x) = B_{N-1}(x) - B_{N+1}(x), & \quad \text{for } i = N-1, \\
\hat{B}_N(x) = B_N(x) + 2B_{N+1}(x), & \quad \text{for } i = N. \tag{2.4} 
\end{align*}
\]

2.2. Calculation of weighting coefficients of first order derivatives

To find the weighting coefficients \( a_{ij}^{(1)} \), we use modified B-splines \( \hat{B}_i(x) \), \( i = 1, 2, \ldots, N \), in the Eq. (2.1), as \( y \) axis is fixed there.

\[
\hat{B}_i(x_l) = \sum_{k=1}^{N} a_{ik}^{(1)} \hat{B}_i(x_k), \quad i = 1, 2, \ldots, N; \quad l = 1, 2, \ldots, N. \tag{2.5}
\]

### Table 1
Values of cubic B-spline and its derivatives at different knots.

<table>
<thead>
<tr>
<th>( x )</th>
<th>( x_{i-2} )</th>
<th>( x_{i-1} )</th>
<th>( x_{i} )</th>
<th>( x_{i+1} )</th>
<th>( x_{i+2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B_i(x) )</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( \hat{B}_i(x) )</td>
<td>0</td>
<td>( \frac{1}{h} )</td>
<td>0</td>
<td>( -\frac{1}{h} )</td>
<td>0</td>
</tr>
<tr>
<td>( \hat{B}'(x) )</td>
<td>0</td>
<td>( \frac{6}{h^2} )</td>
<td>( -\frac{12}{h^2} )</td>
<td>( \frac{6}{h^2} )</td>
<td>0</td>
</tr>
</tbody>
</table>
At the first knot \( x = x_1 \), we have
\[
\tilde{B}_l(x_1) = \sum_{k=1}^{N} a^{(1)}_{ik} \tilde{B}_l(x_k), \quad l = 1, 2, \ldots, N, \tag{2.6}
\]
which produces a tridiagonal system of equations and can be written as,
\[
\begin{bmatrix}
6 & 1 & 0 & 1 & 1 & 1 \\
0 & 4 & 1 & 1 & 1 & 1 \\
1 & 4 & 1 & 1 & 1 & 1 \\
1 & 4 & 0 & 1 & 1 & 1 \\
1 & 4 & 0 & 1 & 1 & 1 \\
1 & 6 & 0 & 1 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
a^{(1)}_{11} \\
a^{(1)}_{12} \\
a^{(1)}_{13} \\
a^{(1)}_{1N-2} \\
a^{(1)}_{1N-1} \\
a^{(1)}_{1N} \\
\end{bmatrix}
= \begin{bmatrix}
-\frac{2}{h} \\
\frac{6}{h} \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\tag{2.7}
\]

The above tridiagonal system of equations have been solved using Thomas algorithm to get the weighting coefficients \( a^{(1)}_{11}, a^{(1)}_{12}, \ldots, a^{(1)}_{1N} \).

Similarly at the second knot \( x = x_2 \), we have
\[
\tilde{B}_l(x_2) = \sum_{k=1}^{N} a^{(1)}_{2k} \tilde{B}_l(x_k), \quad l = 1, 2, \ldots, N, \tag{2.8}
\]
which again gives a tridiagonal system of equations as:
\[
\begin{bmatrix}
6 & 1 & 0 & 1 & 1 & 1 \\
0 & 4 & 1 & 1 & 1 & 1 \\
1 & 4 & 1 & 1 & 1 & 1 \\
1 & 4 & 0 & 1 & 1 & 1 \\
1 & 4 & 0 & 1 & 1 & 1 \\
1 & 6 & 0 & 1 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
a^{(1)}_{21} \\
a^{(1)}_{22} \\
a^{(1)}_{23} \\
a^{(1)}_{2N-2} \\
a^{(1)}_{2N-1} \\
a^{(1)}_{2N} \\
\end{bmatrix}
= \begin{bmatrix}
-\frac{1}{h} \\
\frac{3}{h} \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\tag{2.9}
\]

Using Thomas algorithm to above we get the weighting coefficient \( a^{(1)}_{21}, a^{(1)}_{22}, \ldots, a^{(1)}_{2N} \). In the similar way we can find the weighting coefficients corresponding to the knots \( x = x_3, x_4, \ldots, x_{N-1} \). At the last knot \( x = x_N \), we have
\[
\tilde{B}_l(x_N) = \sum_{k=1}^{N} a^{(1)}_{Nk} \tilde{B}_l(x_k), \quad l = 1, 2, \ldots, N, \tag{2.10}
\]
which gives,
\[
\begin{bmatrix}
6 & 1 & 0 & 1 & 1 & 1 \\
0 & 4 & 1 & 1 & 1 & 1 \\
1 & 4 & 1 & 1 & 1 & 1 \\
1 & 4 & 0 & 1 & 1 & 1 \\
1 & 4 & 0 & 1 & 1 & 1 \\
1 & 6 & 0 & 1 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
a^{(1)}_{N1} \\
a^{(1)}_{N2} \\
a^{(1)}_{N3} \\
a^{(1)}_{N4} \\
a^{(1)}_{N5} \\
a^{(1)}_{N6} \\
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
-\frac{6}{h} \\
\frac{6}{h} \\
\end{bmatrix}
\tag{2.11}
\]

On solving the above tridiagonal system we get \( a^{(1)}_{11}, a^{(1)}_{22}, \ldots, a^{(1)}_{NN} \).

In a similar way using modified B-spline basis functions \( \tilde{B}_l(y) \), \( l = 1, 2, \ldots, M, \) in (2.2), we can compute \( b^{(1)}_i \), for \( i, j = 1, 2, \ldots, M \), related to \( u^{(1)}_y(x, y, t) \). Shu’s [33] gave a recurrence formula to determine the weighting coefficients of second or higher order derivatives. So once the weighting coefficients related to first order partial derivatives have been computed, we use following formulae to get the weighting coefficient related to \( r \)th order derivatives.
\[ a^{(r)}_{ij} = r \left[ a^{(1)}_{ij} u^{(r-1)}_{ij} - \frac{a^{(r-1)}_{ij}}{(x_i-x_j)} \right], \quad i \neq j \]

\[ a^{(r)}_{ii} = - \sum_{j=1, j \neq i}^{N} a^{(r)}_{ij}, \quad i = j; \quad i, j = 1, 2, \ldots, N, \]

\[ b^{(r)}_{ij} = r \left[ b^{(1)}_{ij} u^{(r-1)}_{ij} - \frac{b^{(r-1)}_{ij}}{(y_j-y_i)} \right], \quad i \neq j \]

\[ b^{(r)}_{ii} = - \sum_{j=1, j \neq i}^{M} b^{(r)}_{ij}, \quad i = j; \quad i, j = 1, 2, \ldots, M, \]

where \( r \) denotes the order of the derivative, \( a^{(r)}_{ij} \) and \( a^{(r-1)}_{ij} \) are the weighting coefficients of \( r \text{th} \) and \((r-1)\text{th}\) order partial derivative of \( u(x_i, y_j, t) \) with respect to \( x \) at the point \( (x_i, y_j) \) and \( b^{(r)}_{ij} \) and \( b^{(r-1)}_{ij} \) denote the weighting coefficients of \( r \text{th} \) and \((r-1)\text{th}\) order partial derivative of \( u(x_i, y_j, t) \) with respect to \( y \). We take \( r = 2 \), to get the value of weighting coefficients \( a^{(2)}_{ij} \) and \( b^{(2)}_{ij} \).

3. Numerical solution of two dimensional hyperbolic telegraph equation

The telegraph equation (1.1) is converted into the coupled system using the following transformation

\[ u_i = v_i. \]

Then transformed form of Eq. (1.1) is

\[
\begin{align*}
\frac{d u_i}{d t} &= v_i(x_i, y_j, t), \\
\frac{d v_i}{d t} &= -2xu_i(x_i, y_j, t) - \beta^2 u(x_i, y_j, t) + u_{xx}(x_i, y_j, t) + u_{yy}(x_i, y_j, t) + f(x_i, y_j, t).
\end{align*}
\] (3.1)

Now the first and second order spatial derivatives of \( u \) are discretized by modified B-spline differential quadrature method, which reduces the system of Eq. (3.1) into the following system of first order ordinary differential equations,

\[
\begin{align*}
\frac{d u_i}{d t} &= v_i(x_i, y_j, t), \\
\frac{d v_i}{d t} &= -2xv_i(x_i, y_j, t) - \beta^2 u(x_i, y_j, t) + \sum_{k=1}^{N} a_{ik}^{(2)} u(x_k, y_j, t) + \sum_{k=1}^{M} b_{jk}^{(2)} u(x_i, y_k, t) + f(x_i, y_j, t),
\end{align*}
\] (3.2)

where \( (x_i, y_j, t) \in R \times (0, T); \quad i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, M \)

with the initial conditions,

\[
\begin{align*}
u(x_i, y_j, 0) &= u_0(x_i, y_j), \\
v(x_i, y_j, 0) &= v_0(x_i, y_j).
\end{align*}
\] (3.3)

3.1. Treatment of boundary conditions

The Dirichlet boundary conditions are directly used on the boundary and do not need any further simplification but when the boundary conditions are of Neumann type or mixed, then they are discretized using the modified B-spline differential quadrature method and further simplified to get the solutions at the boundary points.

Dirichlet boundary conditions (1.3) gives:

\[
\begin{align*}
u_{1j} &= f_1(y_j, t), \quad u_{Nj} = f_2(y_j, t), \\
u_1 &= f_3(x_1, t), \quad u_M = f_4(x_M, t), \quad (x_i, y_j, t) \in \partial R \times (0, T).
\end{align*}
\] (3.4)

Neumann boundary conditions (1.4), at \( x = 0 \) and \( x = 1 \) are approximated as:

\[
\begin{align*}
\sum_{k=1}^{N} a_{ik}^{(1)} u(x_k, y_j, t) &= g_1(y_j, t), \\
\sum_{k=1}^{N} a_{nk}^{(1)} u(x_k, y_j, t) &= g_2(y_j, t), \quad j = 1, 2, \ldots, M.
\end{align*}
\]
Fig. 1. Eigen values of matrix $B$ using different grid points.
Rewriting the above system as:

$$a_{11}^{(1)} u_{1j} + a_{1N}^{(1)} u_{Nj} = g_1 - \sum_{k=2}^{N-1} a_{1k}^{(1)} u_{kj},$$

$$a_{N1}^{(1)} u_{1j} + a_{NN}^{(1)} u_{Nj} = g_2 - \sum_{k=2}^{N-1} a_{Nk}^{(1)} u_{kj}, \quad j = 1, 2, \ldots, M.$$ 

On solving the above system, we get approximate value of $u$ at the boundary points $x = 0$ and $x = 1$ as:

$$u_{1j} = \frac{A_{-s_1}}{(a_{11}^{(1)}) - (a_{1N}^{(1)})^2} \tag{3.5},$$

$$u_{Nj} = \frac{B_{-s_2}}{(a_{N1}^{(1)}) - (a_{NN}^{(1)})^2} \quad j = 1, 2, \ldots, M,$$

where,

$$A = (g_1 a_{11}^{(1)} - g_2 a_{1N}^{(1)})^2, \quad s_1 = \sum_{k=2}^{N-1} (a_{11}^{(1)} a_{1k}^{(1)} - a_{1N}^{(1)} a_{Nk}^{(1)}) u_{kj},$$

$$B = (g_2 a_{11}^{(1)} - g_1 a_{1N}^{(1)})^2, \quad s_2 = \sum_{k=2}^{N-1} (a_{11}^{(1)} a_{Nk}^{(1)} - a_{1N}^{(1)} a_{1k}^{(1)}) u_{kj}.$$

In a similar way, from the Neumann boundary conditions (1.4) at $y = 0$ and $y = 1$, we have

$$\sum_{k=1}^{N} b_{1k}^{(1)} u(x, y, t) = g_3(x, t),$$

$$\sum_{k=1}^{N} b_{Mk}^{(1)} u(x, y, t) = g_4(x, t), \quad i = 1, 2, \ldots, N.$$

Solution of above system will give $u_{i1}$ and $u_{iM}$ as:

$$u_{i1} = \frac{C - s_3}{(b_{11}^{(1)})^2 - (b_{1M}^{(1)})^2},$$

$$u_{iM} = \frac{D - s_4}{(b_{M1}^{(1)})^2 - (b_{M1}^{(1)})^2}, \quad i = 1, 2, \ldots, N,$$

where,

$$C = (g_3 b_{11}^{(1)} - g_4 b_{1M}^{(1)})^2, \quad s_3 = \sum_{k=2}^{M-1} (b_{1M}^{(1)} b_{1k}^{(1)} - b_{11}^{(1)} b_{Mk}^{(1)}) u_{ik},$$

$$D = (g_4 b_{11}^{(1)} - g_3 b_{1M}^{(1)})^2, \quad s_4 = \sum_{k=2}^{M-1} (b_{M1}^{(1)} b_{1k}^{(1)} - b_{M1}^{(1)} b_{M1}^{(1)}) u_{ik}.$$
To find the solution at a particular time level, first we compute the initial vector \( \mathbf{u}_0 \) and \( \mathbf{v}_0 \), using the initial conditions (3.3). Then SSP-RK43 scheme has been employed to solve the system of first order ordinary differential equation (3.2) with the appropriate boundary conditions and hence the approximate solution \( \mathbf{u}_k \) and \( \mathbf{v}_k \) at a time level \( t = t_k \) is computed.
4. Stability analysis

We consider the system (3.2) and rewrite it in compact form as:

\[
\frac{dW}{dt} = AW + F
\]

or

\[
\frac{d}{dt} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} O & I \\ B & -2\lambda I \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} O_1 \\ F_i(t) \end{bmatrix},
\]

where,

- \(O\) is null matrix.
- \(I\) is identity matrix of order \((N-2)(M-2)\).
- \(B\) is a square matrix of weighting coefficients of order \((N-2)(M-2)\).
- \(W = [u, v]^T\) is solution vector at the interior grid points, given by.
- \(W = [u_{22}, u_{21}, \ldots, u_{2M-1}, u_{13}, \ldots, u_{1M-1}, \ldots, u_{N-1,1}, \ldots, u_{N,1}, \ldots, u_{N,2}, \ldots, u_{N,2M-1}, \ldots, u_{N-1,M-1}, \ldots, u_{N-1,1}, \ldots, u_{N-1,2}, \ldots, u_{N-1,2M-1}]^T\).
- \(F = [O_1, F_i(t)]^T\) is a vector containing nonhomogeneous part and boundary conditions, where \(O_1\) is null and \(F_i(t)\) is column vector whose entries are given by.

\[F_i(t) = -(a_{ij}^1 u_{ij} + a_{ij}^2 u_{ij}) - (b_{ij}^1 u_{ij} + b_{ij}^2 u_{ij}) + f(x_i, y_j, t), \quad i = 2 : N - 1; \quad j = 2 : M - 1.\]

The stability of system (4.1) is very important since it is related to the stability of numerical scheme for solving it. If the system of ordinary differential equation (4.1) is unstable then stable numerical scheme for temporal discretization may not generate converged solution. The stability of (4.1) is depend on the eigen values of coefficient matrix \(A\), since its exact solution can be directly found using the eigen values. If all the eigen values of \(A\) are having negative real part the system (4.1) will be stable.

Let \(\lambda_k\) be any eigenvalue of \(A\) and \(X_1\) and \(X_2\) be two component (each of order \((N-2)(M-2)\)), of eigenvector corresponding to eigenvalue \(\lambda_k\). We have

\[
\begin{bmatrix} O & I \\ B & -2\lambda I \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \lambda_k \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}.
\]

### Table 4

Error norms and CPU time of Example 2 with \(z = 10, \beta = 5, \Delta t = 0.01\). \(h_x = h_y = 1\).

<table>
<thead>
<tr>
<th>(t)</th>
<th>Proposed method (L_2)</th>
<th>(L_\infty)</th>
<th>Relative-error</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.5</td>
<td>8.3931E-4</td>
<td>3.3019E-3</td>
<td>2.8902E-3</td>
<td>.13</td>
</tr>
<tr>
<td>1</td>
<td>6.0254E-4</td>
<td>2.0597E-3</td>
<td>3.4208E-3</td>
<td>.16</td>
</tr>
<tr>
<td>2</td>
<td>2.4167E-4</td>
<td>7.6531E-4</td>
<td>3.7297E-3</td>
<td>.19</td>
</tr>
<tr>
<td>3</td>
<td>8.9534E-5</td>
<td>2.7920E-4</td>
<td>3.7937E-3</td>
<td>.24</td>
</tr>
<tr>
<td>5</td>
<td>1.2168E-5</td>
<td>3.7800E-5</td>
<td>3.8097E-3</td>
<td>.34</td>
</tr>
</tbody>
</table>

### Table 5

Error norms and CPU time of Example 2 with \(\Delta t = 0.001\). \(h_x = h_y = 0.05\). at different time levels.

| \(z = 10, \beta = 5\) | Proposed method \(L_2\) | \(L_\infty\) | Relative-error | CPU time (s) | Jiwari et al. [16] Relative-error | CPU time (s) |
|---|---|---|---|---|---|---|---|
| 0.5 | 1.0690E-4 | 2.4738E-4 | 1.1088E-4 | .47 | 1.1185E-4 | 6 |
| 1 | 1.5293E-5 | 3.3082E-4 | 1.3266E-4 | 1.1 | 1.8051E-4 | 12 |
| 2 | 4.6488E-5 | 1.1380E-5 | 3.1954E-4 | 1.2 | 4.7289E-4 | 25 |
| 3 | 2.1994E-5 | 4.5377E-5 | 1.3024E-4 | 2.8 | 1.2656E-4 | 37 |
| 5 | 2.7151E-6 | 5.4141E-6 | 1.4439E-4 | 4.3 | 9.2770E-4 | 62 |

<table>
<thead>
<tr>
<th>(z = 10, \beta = 0)</th>
<th>Proposed method (L_2)</th>
<th>(L_\infty)</th>
<th>Relative-error</th>
<th>CPU time (s)</th>
<th>Jiwari et al. [16] Relative-error</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>9.2959E-5</td>
<td>4.2348E-4</td>
<td>3.4675E-4</td>
<td>.52</td>
<td>1.1198E-4</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>6.3652E-5</td>
<td>2.5838E-4</td>
<td>3.9146E-4</td>
<td>.98</td>
<td>1.8633E-4</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>2.5540E-5</td>
<td>9.5843E-5</td>
<td>4.2793E-4</td>
<td>1.8</td>
<td>5.1797E-4</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>9.0234E-6</td>
<td>3.5340E-5</td>
<td>4.5140E-4</td>
<td>2.2</td>
<td>1.4412E-4</td>
<td>37</td>
</tr>
<tr>
<td>5</td>
<td>1.5116E-6</td>
<td>4.8043E-6</td>
<td>5.0758E-4</td>
<td>4.5</td>
<td>1.0883E-4</td>
<td>62</td>
</tr>
</tbody>
</table>
from (4.2), we have

\[ IX_2 = \lambda X_1, \]

\[ BX_1 - 2xX_2 = \lambda X_2. \]
From above system of equations we get,

\[ BX_t = (\lambda_A^2 + 2\lambda_A)X_1 \]  

(4.2)

\[ \Rightarrow \lambda_A(\lambda_A + 2\lambda) \] is an eigenvalue of \( B \).

The Matrix \( B \) is given by:

\[ B = -f^2 I + B_1 + B_2, \]  

(4.3)

where \( B_1 \) and \( B_2 \) are the matrices of weighting coefficients \( a_{ij}^{(2)} \) and \( b_{ij}^{(2)} \), given by

\[
B_1 = \begin{bmatrix}
  a_{22}^{(2)}I_{M-2} & a_{23}^{(2)}I_{M-2} & \cdots & a_{2N-1}^{(2)}I_{M-2} \\
  a_{32}^{(2)}I_{M-2} & a_{33}^{(2)}I_{M-2} & \cdots & a_{3N-1}^{(2)}I_{M-2} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{N-1,2}^{(2)}I_{M-2} & a_{N-1,3}^{(2)}I_{M-2} & \cdots & a_{N-1,N-1}^{(2)}I_{M-2}
\end{bmatrix}
\]

\[
B_2 \quad \text{block diagonal matrix given by}
\]

\[
B_2 = \begin{bmatrix}
  B_{ij} & 0 & \cdots & 0 \\
  0 & B_{ij} & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & B_{ij}
\end{bmatrix}
\]

where,

\[
B_{ij} = \begin{bmatrix}
  b_{22} & b_{23} & \cdots & b_{2,M-1} \\
  b_{32} & b_{33} & \cdots & b_{2,M-1} \\
  \vdots & \vdots & \ddots & \vdots \\
  b_{M-1,2} & b_{M-1,3} & \cdots & b_{M-1,M-1}
\end{bmatrix}_{(M-2)\times(M-2)}
\]

We have computed the eigen values of the matrix \( B_1 \) and \( B_2 \) by taking different grid points. Fig. 1 represents the eigen values for \( 11 \times 11, 21 \times 21, 31 \times 31, 41 \times 41 \) grid points in space and we found that all the eigen values of \( B \) are real and negative, for different values on \( N \) and \( M \). Hence from (4.3), it is clearly seen that all the eigen values of \( B \) are real and negative.

From (4.2) it is clear that.

\[ \lambda_A(2\lambda + \lambda_A) \] will be negative and real.

Let \( \lambda_A = x + iy \), where \( x \) and \( y \) are real numbers.

We have, \((x + iy)(x + iy + 2\lambda)\) is real and negative.

### Table 6

Error norms and CPU time of Example 3 with \( \Delta t = .01 \) and \( h_x = h_y = 1. 

<table>
<thead>
<tr>
<th>( t )</th>
<th>( L_2 )</th>
<th>( L_{\infty} )</th>
<th>Relative-error</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.7047E–4</td>
<td>1.9232E–3</td>
<td>4.9447E–3</td>
<td>.05</td>
</tr>
<tr>
<td>2</td>
<td>3.7546E–4</td>
<td>1.0072E–3</td>
<td>4.3014E–3</td>
<td>.10</td>
</tr>
<tr>
<td>3</td>
<td>5.1383E–4</td>
<td>1.6753E–3</td>
<td>1.9008E–2</td>
<td>.17</td>
</tr>
<tr>
<td>5</td>
<td>6.8021E–4</td>
<td>2.6738E–3</td>
<td>4.8567E–2</td>
<td>.22</td>
</tr>
</tbody>
</table>

### Table 7

Error norms and CPU time of Example 3 with \( \Delta t = .001 \) and \( h_x = h_y = .025. 

<table>
<thead>
<tr>
<th>( t )</th>
<th>Proposed method ( L_2 )</th>
<th>( L_{\infty} )</th>
<th>Relative-error</th>
<th>CPU time (s)</th>
<th>Dehghan and Ghesmati [6] Rel.-error (MLWS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.1370E–4</td>
<td>1.9735E–4</td>
<td>5.7932E–4</td>
<td>1.1</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>1.1044E–4</td>
<td>3.4253E–4</td>
<td>1.2658E–4</td>
<td>1.9</td>
<td>4.731E–4</td>
</tr>
<tr>
<td>3</td>
<td>1.4336E–4</td>
<td>4.5378E–5</td>
<td>3.2997E–4</td>
<td>3.3</td>
<td>–</td>
</tr>
<tr>
<td>5</td>
<td>1.6269E–4</td>
<td>5.8801E–4</td>
<td>1.1501E–4</td>
<td>5.5</td>
<td>–</td>
</tr>
<tr>
<td>7</td>
<td>1.5768E–4</td>
<td>6.7607E–4</td>
<td>2.6368E–3</td>
<td>7.3</td>
<td>–</td>
</tr>
<tr>
<td>10</td>
<td>1.4145E–4</td>
<td>7.9495E–4</td>
<td>6.3276E–2</td>
<td>11.0</td>
<td>–</td>
</tr>
</tbody>
</table>
i.e.,

\[
\begin{align*}
  x(x + 2x) - y^2 &< 0, \\
  y(x + x) &= 0.
\end{align*}
\]

**Fig. 4.** Plots of numerical and exact solution at different time levels for Example 3.
From above set of equations, we get the solutions as:

1. $x + x = 0$ and $y$ is arbitrary real number.
   $\Rightarrow x$ is negative real number, since $x$ is real and positive.

2. If $y = 0$
   $\Rightarrow x(2x < 0$,
   $\Rightarrow (x + x)^2 < x^2$,
   $\Rightarrow x$ is negative and real, since $x$ is real and positive.
   $\Rightarrow$ Real part of eigen values of $A$ is negative.

We conclude that the real part of eigen values of $A$ is always negative, i.e the system (1.1) is stable.

5. Numerical results

In this section Eqs. (1.1)–(1.4) is solved numerically for different values of $\alpha$ and $\beta$ at different time levels. The effectiveness of the approach is demonstrated by performing experiments on the problems available in the literature. To assess the performance of the method we compute $L_2$, $L_\infty$ norms and relative error.

Example 1. We consider (1.1) in the region $0 \leq x, y \leq 1$ with $\alpha = 1, \beta = 1, f(x, y, t) = 2(\cos t - \sin t) \sin x \sin y$ and with the following initial conditions

\[
\begin{align*}
\{ u(x, y, 0) &= \sin x \sin y, \\
u_t(x, y, 0) &= 0.
\end{align*}
\]

The Dirichlet boundary condition are given by,

\[
\begin{align*}
u(0, y, t) &= 0, &0 \leq y \leq 1, x = 0, \\
u(1, y, t) &= \cos t \sin(1) \sin y, &0 \leq y \leq 1, x = 1, \\
u(x, 0, t) &= 0, &0 \leq x \leq 1, y = 0, \\
u(x, 1, t) &= \cos t \sin x \sin(1), &0 \leq x \leq 1, y = 1.
\end{align*}
\]

The exact solution [6] of this example is,

\[u(x, y, t) = \cos t \sin x \sin y\]

In Table 2 different error norms are shown with $\Delta t = .01$ and $h_x = h_y = .1$. Next we take $\Delta t = .001$ and $h_x = h_y = .05$ and the $L_2$, $L_\infty$ and relative errors at different time levels are shown in Table 3. We also report the CPU time. A comparison of exact and numerical solutions at time $t = 1, 2, 3$, are depicted in Fig. 2 with $\Delta t = .001$ and $h_x = h_y = .05$, which shows that numerical solutions are in excellent agreement with the exact solutions.

Example 2. In this example we consider the telegraph equation (1.1) in the region $0 \leq x, y \leq 1$ with $f(x, y, t) = (-2x + \beta^2 - 1)e^{-t} \sinh x \sinh y$ and the following initial conditions

\[
\begin{align*}
\{ u(x, y, 0) &= \sin x \sin y, \\
u_t(x, y, 0) &= 0.
\end{align*}
\]

<table>
<thead>
<tr>
<th>$t$</th>
<th>Proposed method</th>
<th>Jiwari et al. [16]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$L_2$</td>
<td>$L_\infty$</td>
</tr>
<tr>
<td>$x = 10, \beta = 5$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>1.0696E-4</td>
<td>3.7559E-4</td>
</tr>
<tr>
<td>1</td>
<td>1.7174E-4</td>
<td>5.6395E-4</td>
</tr>
<tr>
<td>2</td>
<td>1.6468E-4</td>
<td>5.1298E-4</td>
</tr>
<tr>
<td>3</td>
<td>8.9858E-6</td>
<td>1.9564E-5</td>
</tr>
<tr>
<td>5</td>
<td>1.7737E-4</td>
<td>5.3627E-4</td>
</tr>
<tr>
<td>7</td>
<td>1.4200E-4</td>
<td>4.7231E-4</td>
</tr>
<tr>
<td>10</td>
<td>1.2241E-4</td>
<td>4.1222E-4</td>
</tr>
</tbody>
</table>

| $x = 50, \beta = 5$ |       |       |       |       |       |       |
| 0.5 | 9.8800E-5 | 3.6962E-5 | 1.2735E-5 | .57 | 2.8724E-5 | 7 |
| 1   | 1.6766E-4 | 5.6874E-4 | 3.5104E-5 | .94 | 7.0064E-5 | 13 |
| 2   | 1.7109E-4 | 5.2572E-4 | 6.4608E-5 | 1.4 | 6.7759E-5 | 27 |
| 3   | 1.7406E-5 | 4.3459E-5 | 1.9886E-6 | 2.5 | 1.3856E-5 | 39 |
| 5   | 1.8420E-4 | 5.6940E-4 | 7.3460E-5 | 4.1 | 1.2840E-4 | 69 |
| 7   | 1.3768E-4 | 4.7587E-4 | 2.0647E-5 | 6.0 | - | - |
| 10  | 1.1691E-4 | 4.1396E-4 | 1.5772E-5 | 8.8 | 4.4202E-5 | 139 |
\begin{align*}
&\begin{cases}
  u(x, y, 0) = \sinh x \sinh y, \\
  u_t(x, y, 0) = -\sinh x \sinh y.
\end{cases}
\end{align*}

Dirichlet boundary conditions are:

\begin{align*}
\begin{cases}
  u(0, y, t) = C_0 \
  u(1, y, t) = 0 \
  u(x, 0, t) = C_0 \
  u(x, 1, t) = 0
\end{cases}
\end{align*}

\textbf{Fig. 5.} Plots of numerical and exact solution at different time levels for \textit{Example 4}. 
\[
\begin{aligned}
&u(0, y, t) = 0, \quad 0 \leq y \leq 1, \quad x = 0, \\
&u(1, y, t) = e^{-t} \sinh(1) \sinh y, \quad 0 \leq y \leq 1, \quad x = 1, \\
&u(x, 0, t) = 0, \quad 0 \leq x \leq 1, \quad y = 0, \\
&u(x, 1, t) = e^{-t} \sin x \sinh(1), \quad 0 \leq x \leq 1, \quad y = 1.
\end{aligned}
\]

The exact solution is given by [16],
\[u(x, y, t) = e^{-t} \sin x \sinh y.\]

We take \(x = 10, \beta = 5\) and \(x = 10, \beta = 0\). In Table 4, results are depicted at different time levels with \(\Delta t = .01\) and \(h_x = h_y = .1\). Next we take time step size \(\Delta t = .001\) and \(h_x = h_y = .05\) and compute \(L_2, L_\infty\), relative errors and CPU time at different time levels. In Table 5 we compare our results with those of Jiwari et al. [16] and observe that our results are comparable in terms of relative error. However CPU time taken in our computation is very less. We can conclude that our scheme produces good accuracy using less computational effort. Fig. 3 compares the surface plots of numerical and exact solutions at \(t = 1, 2, 3\), with \(\Delta t = .001\) and \(h_x = h_y = .05\).

**Example 3.** We consider the following two dimensional hyperbolic telegraph equation in the region \(0 \leq x, y \leq 1\),
\[
u_{tt} + 4\pi u_{tt} + 2\pi^2 u = u_{xx} + u_{yy} + 2\pi \sin \pi(x + y) e^{-(x+y)^2} + (x + y - 2\pi^2) \sin(\pi x) \sin(\pi y) e^{-(x+y)^2}.
\]
The initial and Dirichlet boundary conditions are given by
\[
\begin{aligned}
&u(x, y, 0) = \sin \pi x \sin \pi y, \\
u_t(x, y, 0) = -(x + y) \sin \pi x \sin \pi y, \\
&u(0, y, t) = 0, \quad 0 \leq y \leq 1, \quad x = 0, \\
&u(1, y, t) = 0, \quad 0 \leq y \leq 1, \quad x = 1, \\
&u(x, 0, t) = 0, \quad 0 \leq x \leq 1, \quad y = 0, \\
&u(x, 1, t) = 0, \quad 0 \leq x \leq 1, \quad y = 1.
\end{aligned}
\]
The exact solution is given in [6] by,
\[u(x, y, t) = e^{-(x+y)} \sin \pi x \sin \pi y.\]

In Table 6 results are shown with \(\Delta t = .01\) and \(h_x = h_y = .1\). Next we take \(\Delta t = .001\) and \(h_x = h_y = .025\). Table 7 shows the accuracy of computed solutions in terms of \(L_2, L_\infty\) norms and relative errors. We have also reported CPU time used in our computation. A comparison is shown with the results of Dehghan and Ghesmati [6] in terms of relative error \(t = 2\). The surface plots of numerical and exact solutions at different time level are depicted in Fig. 4.

### Table 9
<p>| Error norms and CPU time of Example 5 with (\Delta t = .01) and (h_x = h_y = .1). |
|---|---|---|---|---|</p>
<table>
<thead>
<tr>
<th>(t)</th>
<th>(L_2)</th>
<th>(L_\infty)</th>
<th>Relative-error</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.4441E–2</td>
<td>2.9996E–2</td>
<td>1.0829E–3</td>
<td>.03</td>
</tr>
<tr>
<td>2</td>
<td>1.3898E–3</td>
<td>3.9711E–3</td>
<td>2.8333E–4</td>
<td>.05</td>
</tr>
<tr>
<td>3</td>
<td>1.3018E–3</td>
<td>2.2178E–3</td>
<td>7.2867E–4</td>
<td>.08</td>
</tr>
<tr>
<td>5</td>
<td>1.1112E–4</td>
<td>2.0618E–4</td>
<td>4.5956E–4</td>
<td>.11</td>
</tr>
<tr>
<td>7</td>
<td>1.3695E–5</td>
<td>3.0052E–5</td>
<td>4.1851E–4</td>
<td>.14</td>
</tr>
<tr>
<td>10</td>
<td>1.4408E–6</td>
<td>2.5334E–6</td>
<td>8.8440E–4</td>
<td>.19</td>
</tr>
</tbody>
</table>

### Table 10
| Error norms and CPU time of Example 5 at different time with \(\Delta t = .001\) and \(h_x = h_y = .05\). |
|---|---|---|---|---|
| \(t\) | Proposed method | Dehghan and Ghesmati [6] |
|---|---|---|---|---|
| \(L_2\) | \(L_\infty\) | Relative-error | CPU time (s) | Relative-error (MLWS) | CPU time (s) | Relative-error (MLPG) | CPU-time |
| 0.5 | 3.4808E–3 | 9.5129E–3 | 8.4225E–5 | .5 | 8.014E–5 | 7.4 | 2.032E–5 | 16.6 |
| 2 | 2.8518E–4 | 1.0361E–3 | 3.0957E–5 | 1.3 | 9.791E–5 | 22.9 | 3.004E–4 | 42.3 |
| 3 | 3.1028E–4 | 5.7859E–4 | 9.1555E–5 | 1.9 | 7.029E–4 | 31.0 | 5.201E–4 | 53.7 |
| 4 | 9.0898E–5 | 2.7645E–4 | 7.2908E–5 | 2.3 | 1.703E–3 | 40.2 | 7.065E–4 | 67.4 |
| 5 | 2.4493E–5 | 6.7234E–5 | 5.3354E–5 | 3.3 | – | – | – | – |
| 7 | 2.3278E–6 | 8.2203E–6 | 4.0840E–5 | 3.9 | – | – | – | – |
| 10 | 3.6505E–6 | 8.5897E–6 | 1.1812E–5 | 5.2 | – | – | – | – |
Example 4. In this example we consider the telegraph equation (1.1) in the region $0 \leq x, y \leq 1$ with
$$f(x,y,t) = (\beta \cos t - 2x \sin t + \beta^2 \cos t) \sinh x \sinh y$$
and with the following initial conditions.
The Dirichlet boundary conditions (1.3) are given by,

\[
\begin{align*}
    u(0, y, t) &= 0, & 0 \leq y \leq 1, & x = 0, \\
    u(1, y, t) &= \cos t \sinh 1 \sinh y, & 0 \leq y \leq 1, & x = 1, \\
    u(x, 0, t) &= 0, & 0 \leq x \leq 1, & y = 0, \\
    u(x, 1, t) &= \cos t \sinh x \sinh(1), & 0 \leq x \leq 1, & y = 1.
\end{align*}
\]

The exact solution [16] is given by,

\[
u(x,y,t) = \cos t \sinh x \sinh y.
\]

We consider the telegraph equation (1.1) in the region \(0 \leq x, y \leq 1\) with \(\alpha = 1\), \(\beta = 1\), \(f(x,y,t) = -2e^{x+y-t}\) and with the following initial conditions

\[
\begin{align*}
    u(x,y,0) &= e^{x+y}, \\
    u_t(x,y,0) &= -e^{x+y}.
\end{align*}
\]

We consider the following mixed boundary conditions,

\[
\begin{align*}
    u(0, y, t) &= e^{y-t}, & 0 \leq y \leq 1, & x = 0, \\
    u(1, y, t) &= e^{1+y-t}, & 0 \leq y \leq 1, & x = 1, \\
    \frac{\partial u}{\partial y}(x, 0, t) &= e^{-t}, & 0 \leq x \leq 1, & y = 0, \\
    u(x, 1, t) &= e^{1-x-t}, & 0 \leq x \leq 1, & y = 1.
\end{align*}
\]

The exact solution [6] is

\[
u(x,y,t) = e^{x+y-t}.
\]

### Table 11

<table>
<thead>
<tr>
<th>(t)</th>
<th>(L_2)</th>
<th>(L_{\infty})</th>
<th>Relative-error</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.6144E–3</td>
<td>3.6006E–3</td>
<td>8.7768E–4</td>
<td>.07</td>
</tr>
<tr>
<td>2</td>
<td>2.6345E–3</td>
<td>5.7068E–3</td>
<td>3.8933E–3</td>
<td>.09</td>
</tr>
<tr>
<td>3</td>
<td>5.3845E–4</td>
<td>1.2479E–3</td>
<td>2.1847E–3</td>
<td>.11</td>
</tr>
<tr>
<td>5</td>
<td>1.2418E–4</td>
<td>2.1003E–4</td>
<td>3.7232E–3</td>
<td>.15</td>
</tr>
<tr>
<td>10</td>
<td>7.5592E–6</td>
<td>1.4083E–6</td>
<td>3.3635E–3</td>
<td>.20</td>
</tr>
</tbody>
</table>

### Table 12

<table>
<thead>
<tr>
<th>(t)</th>
<th>Proposed method</th>
<th>(L_2)</th>
<th>(L_{\infty})</th>
<th>Relative-error</th>
<th>CPU time (s)</th>
<th>Dehghan and Ghesmati [6]</th>
<th>Jiwari et al. [16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>3.5833E–4</td>
<td>9.5129E–4</td>
<td>8.4225E–5</td>
<td>.3</td>
<td>7.040E–5</td>
<td>3.701E–5</td>
<td>1.0078E–4</td>
</tr>
<tr>
<td>2</td>
<td>2.8518E–5</td>
<td>1.0361E–4</td>
<td>3.0957E–5</td>
<td>1.3</td>
<td>4.820E–4</td>
<td>1.216E–4</td>
<td>1.1962E–4</td>
</tr>
<tr>
<td>5</td>
<td>2.4495E–6</td>
<td>6.7234E–5</td>
<td>5.3354E–4</td>
<td>2.9</td>
<td>–</td>
<td>–</td>
<td>1.3277E–5</td>
</tr>
<tr>
<td>7</td>
<td>2.5376E–7</td>
<td>8.2203E–7</td>
<td>4.0940E–5</td>
<td>4.1</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Fig. 7. Plots of numerical and exact solution at different time levels for Example 6.
norms and relative errors are shown in Table 9 with $\Delta t = .01$ and $h_x = h_y = .1$. In Table 10, we show different error norms with $\Delta t = .001$ and $h_x = h_y = .05$, up to time $t = 10$. We also compare our results with Dehghan and Ghesmati [6], in terms of relative error. We see that our results are better than that given in [6] and also CPU time taken is much less. A comparison of numerical and exact solution at $t = 1, 2, 4$, is shown in Fig. 6 with $\Delta t = .001$ and $h_x = h_y = .05$.

**Example 6.** Now we consider the telegraph equation (1.1) with $\alpha = 1$, $\beta = 1$, $f(x,y,t) = 2\pi^2 e^{-t} \sin \pi x \sin \pi y$ and the following initial conditions

$$
\begin{align*}
  u(x,y,0) &= \sin \pi x \sin \pi y, \\
  u_t(x,y,0) &= -\sin \pi x \sin \pi y.
\end{align*}
$$

We use the following mixed boundary conditions,

$$
\begin{align*}
  \frac{\partial u}{\partial n}(0,y,t) &= \pi e^{-t} \sin \pi y, & 0 \leq y \leq 1, & x = 0, \\
  u(1,y,t) &= 0, & 0 \leq y \leq 1, & x = 1, \\
  u(x,0,t) &= 0, & 0 \leq x \leq 1, & y = 0, \\
  \frac{\partial u}{\partial n}(x,1,t) &= -\pi e^{-t} \sin \pi x, & 0 \leq x \leq 1, & y = 1.
\end{align*}
$$

The exact solution [6] is

$$
u(x,y,t) = e^{-t} \sin \pi x \sin \pi y.
$$

Table 11 shows the $L_2$, $L_\infty$ and relative error up to time $t = 10$ with $\Delta t = .01$ and $h_x = h_y = .1$. We also report CPU time. Next we take $\Delta t = .001$ and $h_x = h_y = .05$. A comparison is shown in Table 12 with Dehghan and Ghesmati [6] and Jiwari et al. [16] in term of relative errors. We see that our scheme produces much better results in comparison with [6,16].

**Example 7.** In this example consider (1.1) for $\alpha = 1$, $\beta = 1$, with the following initial conditions

$$
\begin{align*}
  u(x,y,0) &= \log(1 + x + y), \\
  u_t(x,y,0) &= \frac{1}{1 + x + y}.
\end{align*}
$$

and with the following mixed boundary conditions

$$
\begin{align*}
  u(0,y,t) &= \log(1 + y + t), & 0 \leq y \leq 1, & x = 0, \\
  \frac{\partial u}{\partial n}(1,y,t) &= \frac{1}{1 + y + t}, & 0 \leq y \leq 1, & x = 1, \\
  \frac{\partial u}{\partial n}(x,0,t) &= \frac{1}{1 + x + t}, & 0 \leq x \leq 1, & y = 0, \\
  u(x,1,t) &= \log(2 + x + t), & 0 \leq x \leq 1, & y = 1.
\end{align*}
$$

The exact solution is given by [6],

$$
u(x,y,t) = \log(1 + x + y + t)
$$

and $f(x,y,t)$ can be extracted from the exact solution.

We take $\Delta t = .001$ and $21 \times 21$ mesh grid points in the region $0 \leq x, y \leq 1$. Computed results for this examples have been shown in Table 13 in terms of $L_2$, $L_\infty$ and relative errors. The results are compared with Dehghan and Ghesmati [6] and found better. The CPU time is also very less. The surface plots of numerical and exact solutions at $t = 1, 3, 5$, are depicted in Fig. 8, which also confirms the good accuracy of the scheme.

<table>
<thead>
<tr>
<th>$t$</th>
<th>Proposed method</th>
<th>Dehghan and Ghesmati [6]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$L_2$</td>
<td>$L_\infty$</td>
</tr>
<tr>
<td></td>
<td>Relative-error</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>1.2959E−3</td>
<td>1.1088E−3</td>
</tr>
<tr>
<td>1</td>
<td>1.255E−3</td>
<td>1.3266E−3</td>
</tr>
<tr>
<td>2</td>
<td>1.219E−3</td>
<td>3.1954E−4</td>
</tr>
<tr>
<td>3</td>
<td>1.219E−3</td>
<td>1.3024E−4</td>
</tr>
<tr>
<td>4</td>
<td>1.219E−3</td>
<td>1.4435E−5</td>
</tr>
<tr>
<td>5</td>
<td>1.219E−3</td>
<td>8.4225E−5</td>
</tr>
<tr>
<td>10</td>
<td>1.219E−3</td>
<td>8.4225E−5</td>
</tr>
</tbody>
</table>

Table 13

Error norms and CPU time of Example 7 at different time.
6. Conclusion

In this paper a differential quadrature method based on modified cubic B-spline basis functions has been developed to solve two dimensional hyperbolic telegraph equation. The telegraph equation is first converted into a system of partial differential equations. The numerical results have been compared with the exact solutions. The numerical results show that the proposed method is efficient and accurate.

Fig. 8. Plots of numerical and exact solution at different time levels for Example 7.
numerical errors. The method is tested on seven test problems. In order to show the efficiency and accuracy of approach SSP-RK43 scheme to solve this system. The SSP-RK43 scheme needs low storage space that causes less accumulation of differential equations and then using MCB-DQM we get a system of first order ordinary differential equations. Finally we use mixed boundary conditions.

Acknowledgments

The authors would like to thank the anonymous referees for their time, efforts and extensive comments on the revision of the manuscript.

References


