A differential quadrature algorithm to solve the two dimensional linear hyperbolic telegraph equation with Dirichlet and Neumann boundary conditions

Ram Jiwari a,*, Sapna Pandit b, R.C. Mittal c

a School of Mathematics & Computer Applications, Thapar University, Patiala, India
b Department of Mathematics, MNNIT Allahabad, India
c Department of Mathematics, Indian Institute of Technology Roorkee, Roorkee, India

A R T I C L E  I N F O

Keywords:
Two dimensional telegraph equation
Differential quadrature method
Gauss–Lobatto–Chebyshev grid points
Runge–Kutta method

A B S T R A C T

In this article, we proposed a numerical technique based on polynomial differential quadrature method (PDQM) to find the numerical solutions of two dimensional hyperbolic telegraph equation with Dirichlet and Neumann boundary condition. The PDQM reduced the problem into a system of second order linear differential equation. Then, the obtained system is changed into a system of ordinary differential equations and lastly, RK4 method is used to solve the obtained system. The accuracy of the proposed method is demonstrated by several test examples. The numerical results are found to be in good agreement with the exact solutions and the numerical solutions exist in literature. The technique is easy to apply for multidimensional problems.

© 2012 Elsevier Inc. All rights reserved.

1. Introduction

We consider the second order two dimensional hyperbolic telegraph equation

\[ \frac{\partial^2 u}{\partial t^2} + 2 \alpha \frac{\partial u}{\partial t} + \beta^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f(x, y, t), \quad (x, y, t) \in [0, 1] \times [0, 1] \times [t > 0] \]  

(1)

with initial conditions

\[ u(x, y, 0) = g_1(x, y), \quad (x, y) \in [0, 1] \times [0, 1], \]
\[ u_t(x, y, 0) = g_2(x, y), \quad (x, y) \in [0, 1] \times [0, 1]. \]  

(2)

The Dirichlet boundary conditions are given by

\[ u(0, y, t) = h_1(y, t), \quad (y, t) \in [0, 1] \times [t > 0], \quad u(1, y, t) = h_2(y, t), \quad (y, t) \in [0, 1] \times [t > 0], \]
\[ u(x, 0, t) = h_3(x, t), \quad (x, t) \in [0, 1] \times [t > 0], \quad u(x, 1, t) = h_4(x, t), \quad (x, t) \in [0, 1] \times [t > 0] \]  

(3)

and Neumann boundary conditions are given by

\[ u_x(0, y, t) = f_1(y, t), \quad (y, t) \in [0, 1] \times [t > 0], \quad u_x(1, y, t) = f_2(y, t), \quad (y, t) \in [0, 1] \times [t > 0], \]
\[ u_y(x, 0, t) = f_3(x, t), \quad (x, t) \in [0, 1] \times [t > 0], \quad u_y(x, 1, t) = f_4(x, t), \quad (x, t) \in [0, 1] \times [t > 0]. \]  

(4)

* Corresponding author.
E-mail address: ram1.maths@gmail.com (R. Jiwari).

0096-3003/– see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2012.01.006
Both the electric voltage and the current in a double conductor, satisfy the telegraph equation, where \( x, y \) are the distances and \( t \) is time. When \( x > 0, \beta = 0 \), Eq. (1) represents a damped wave equation and for \( x > 0, \beta > 0 \), is called telegraph equation.

Recently, it is found that telegraph equation is more suitable than ordinary diffusion equation in modeling reaction diffusion for such branches of sciences. The hyperbolic partial differential equations model the vibrations of structures (e.g., buildings, beams, and machines) and they are the basis for fundamental equations of atomic physics. The telegraph equation is important for modeling several relevant problems such as signal analysis [1], wave propagation [2], random walk theory [3], etc.

In recent years, much attention has been given in the literature to the development of numerical schemes for the second order one dimensional and two dimensional hyperbolic telegraph equations. In [4–10, 21, 22], the authors considered one dimensional telegraph equation to find the numerical solutions of the equation. Mohanty and Jain [11] proposed a three-level implicit unconditionally stable alternating direction implicit method for two dimensional telegraph equation, whose accuracy is second order for both time and space components. Also, Mohanti et al. [12] used an unconditionally stable ADI method for three dimensional hyperbolic telegraph equation (1). Dehghan and Mehebbi [13] proposed a high order implicit collocation method for solving the two dimensional hyperbolic equation (1). Dehghan and Shokri [14] solved the one and two dimensional hyperbolic equations using collocation points and used the thin plate-spline radial basis functions to approximate the solution. In [15], the author proposed a new unconditionally stable difference schemes for the solution of multi-dimensional telegraphic equations with Dirichlet boundary conditions. In [23, 24], Dehghan and Mohebbi proposed some numerical techniques for two dimensional wave equation and schrodinger equation. The latest work is done by Dehghan and Ghesmati [16] by making a combination of meshless local weak and strong (MLWS) forms to solve the two dimensional hyperbolic telegraph equation.

In this article, we proposed a numerical scheme based on polynomial differential quadrature method to find the numerical solutions of two dimensional telegraph equation with Dirichlet and Neumann boundary conditions. The PDQM reduced the problem into a system of second order linear differential equation. Then, the obtained system is changed into coupled differential equations and lastly, RK4 method is used to solve the coupled system. The accuracy and efficiency of the proposed method are demonstrated by several test examples. The main advantage of the present scheme is that it works for big time and gives good accuracy as comparison to the numerical techniques presented in [11–16]. Also, the present technique is easy to apply for multidimensional problems.

2. Polynomial differential quadrature method

Differential quadrature method is a numerical method for solving differential equations. It was firstly, introduced by Bellman et al. [17]. During the past few decades, the idea of using differential quadrature methods for numerical solution of partial differential equations (PDEs) has received much attention throughout the scientific community. Mittal and his associates [25–27] used differential quadrature method for solving some two dimension nonlinear equations.

This method approximates the derivatives of a function using the weighted sum of the functional values at certain discrete points. First the domain \( D \) is discretized by taking \( N \) points along \( x \) direction and \( M \) points along \( y \) direction. According to the two dimensional PDQM, the first order partial derivatives of a dependent function \( u(x,y,t) \) can be approximated as follows [20]

\[
\begin{align*}
  u_x^{(1)}(x_i, y_j, t) & \approx \sum_{k=1}^{N} a_{ik}^{(1)} u(x_k, y_j, t), \\
  u_y^{(1)}(x_i, y_j, t) & \approx \sum_{k=1}^{N} b_{jk}^{(1)} u(x_i, y_k, t), \quad \text{for } i = 1, 2, \ldots, N; \ j = 1, 2, \ldots, M,
\end{align*}
\]

where \( u_x^{(1)}(x_i, y_j, t) \) and \( u_y^{(1)}(x_i, y_j, t) \) are the first order partial derivatives of \( u(x,y,t) \) with respect to \( x \) and \( y \) at grid points \((x_i,y_j)\) and \( a_{ik}^{(1)} \) and \( b_{jk}^{(1)} \) are the corresponding weighting coefficients.

Bellman et al. [17] proposed two approaches to compute the weighting coefficients. To improve Bellman’s approaches in computing the weighting coefficients, many attempts have been made by researchers. One of the most useful approaches is the one introduced by Quan and Chang [18, 19]. After that, Shu [20] proposed the most general approach for finding the weighting coefficients. The following base functions are used to obtain weighting coefficients

\[
\begin{align*}
  r_k(x) & = \frac{M(x)}{(x-x_k)M''(x_k)}, \quad k = 1, 2, \ldots, N, \\
  s_k(x) & = \frac{M(y)}{(y-y_k)M''(y_k)}, \quad k = 1, 2, \ldots, M,
\end{align*}
\]
where
\[ M(x) = (x - x_1)(x - x_2) \ldots (x - x_N), \]
\[ M(1)(x_i) = \prod_{k=1, k \neq i}^N (x_i - x_k). \]  
(9)

The weighting coefficients of the first order partial derivatives are
\[ a_y^{(1)} = \frac{M(1)(x_i)}{(x_i - x_j)M(1)(x_j)}, \text{ for } i \neq j, \]  
(10)
\[ a_u^{(1)} = -\sum_{j=1}^N a_y^{(1)}, \text{ for } i, j = 1, 2, \ldots, N, \]  
(11)
\[ b_y^{(1)} = \frac{M(1)(y_i)}{(y_i - y_j)M(1)(y_j)}, \text{ for } i \neq j, \]  
(12)
\[ b_u^{(1)} = -\sum_{j=1}^N b_y^{(1)}, \text{ for } i, j = 1, 2, \ldots, M. \]  
(13)

Similarly, for the second and higher order derivatives, the recurrence relation of the weighting coefficients are obtained as follows
\[ a_y^{(n)} = n \left[a_y^{(n-1)}a_u^{(n-1)} - a_y^{(n-1)}\right], \text{ for } i \neq j, \]  
(14)
\[ a_u^{(n)} = -\sum_{j=1}^N a_y^{(n)}, \text{ for } i = j, n = 2, 3, \ldots, N - 1; i, j = 1, 2, \ldots, N, \]  
(15)
\[ b_y^{(n)} = m \left[b_y^{(n-1)}b_u^{(n-1)} - b_y^{(n-1)}\right], \text{ for } i \neq j, \]  
(16)
\[ b_u^{(n)} = -\sum_{j=1}^M b_y^{(n)}, \text{ for } i = j, m = 2, 3, \ldots, M - 1; i, j = 1, 2, \ldots, M, \]  
(17)
where \( a_y^{(n)} \) and \( b_y^{(n)} \) are the weighting coefficients related to \( u_y^{(n)}(x_i, y_j, t) \) and \( u_y^{(n)}(x_i, y_j, t) \) at \( (x_i, y_j) \) and \( a_y^{(1)} \) and \( b_y^{(1)} \) are the weighting coefficients of the first order partial derivatives of \( u(x, y, t) \) with respect to \( x \) and \( y \).

3. Numerical scheme for two dimensional telegraph equation

The space derivatives in the two-dimensional linear hyperbolic equation (1) are approximated by the polynomial differential quadrature method. Eq. (1) changed into the following form
\[ \frac{d^2 u_{ij}}{dt^2} + 2x \frac{d u_{ij}}{dt} + \beta^2 u_{ij} = \sum_{k=1}^N a_{ijk}^2 u_{kj} + \sum_{k=1}^M b_{ijk}^2 u_{i+k} + g_{i,j}(x_i, y_j, t) \in [0, 1] \times [0, 1] \times [t > 0], \quad i = 1, 2, \ldots, N, \]
\[ j = 1, 2, \ldots, M \]  
(18)
with initial conditions
\[ u(x_i, y_j, 0) = g_1(x_i, y_j), \]  
(19)
\[ \frac{du}{dt}(x_i, y_j, 0) = g_2(x_i, y_j), \]  
(20)
where \( u_{ij} = u(x_i, y_j, t) \) and \( g_1, g_2 = g(x_i, y_j, t) \).

The system (18) is system of second order differential equations. Now let
\[ \frac{du}{dt}(x_i, y_j, t) = z(x_i, y_j, t), \quad \text{then } \frac{dz}{dt}(x_i, y_j, t) = \frac{dz}{dt}(x_i, y_j, t), \]  
(21)
Using the assumptions of Eq. (21) into Eq. (18), Eq. (18) changed as follows

\[
\begin{align*}
\frac{du}{dt}(x_i, y_j, t) &= z(x_i, y_j, t), \\
\frac{dz}{dt}(x_i, y_j, t) + 2\alpha z(x_i, y_j, t) + \beta^2 u(x_i, y_j, t) &= \sum_{j=1}^{N} a_{1k}^{(1)} u(x_k, y_j, t) + \sum_{j=1}^{M} b_{ik}^{(1)} u(x_i, y_k, t) + g(x_i, y_j, t), \quad (x_i, y_j, t) \in [0, 1] \times [0, 1] \times [t > 0], \quad i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, M
\end{align*}
\]

with initial conditions

\[
\begin{align*}
u(x_i, y_j, 0) &= g_1(x_i, y_j), \\
\frac{du}{dt}(x_i, y_j, 0) &= g_2(x_i, y_j).
\end{align*}
\]

3.1. Implementation of Dirichlet boundary conditions

The Dirichlet boundary conditions are directly applied in the PDQM and gives numerical solutions on boundary in the following way

\[
\begin{align*}
u_{1j} &= h_1(y_j, t), \quad (y_j, t) \in [0, 1] \times [t > 0], \quad u_{Nj} = h_2(y_j, t), \quad (y_j, t) \in [0, 1] \times [t > 0], \quad j = 1, 2, \ldots, M, \\
u_{1i} &= h_3(x_i, t), \quad (x_i, t) \in [0, 1] \times [t > 0], \quad u_{Mi} = h_4(x_i, t), \quad (x_i, t) \in [0, 1] \times [t > 0], \quad i = 1, 2, \ldots, N.
\end{align*}
\]

3.2. Implementation of Neumann boundary conditions

The Neumann boundary conditions (4) at \(x = 0\) and \(x = 1\) can be approximated as

\[
\begin{align*}
\sum_{k=1}^{N} a_{1k}^{(1)} u_{kj} &= f_1(0, y, t), \\
\sum_{k=1}^{N} a_{Nk}^{(1)} u_{kj} &= f_2(1, y, t), \quad j = 1, 2, \ldots, M.
\end{align*}
\]

Eqs. (28) and (29) can be written as follows

\[
\begin{align*}
a_{11}^{(1)} u_{1j} + a_{1N}^{(1)} u_{Nj} + a_{1k}^{(1)} u_{kj} &= f_1 - \sum_{k=2}^{N-1} a_{1k}^{(1)} u_{kj}, \\
a_{N1}^{(1)} u_{1j} + a_{Nk}^{(1)} u_{Nj} + a_{Nk}^{(1)} u_{kj} &= f_2 - \sum_{k=2}^{N-1} a_{Nk}^{(1)} u_{kj}.
\end{align*}
\]

Solving Eqs. (28) and (29) for \(u_{1j}\) and \(u_{Nj}\), we get

\[
\begin{align*}
u_{1j} &= \frac{a_{1k}^{(1)} (f_2 - S_2) - a_{Nk}^{(1)} (f_1 - S_1)}{(a_{1N}^{(1)} a_{N1}^{(1)} - a_{1k}^{(1)} a_{Nk}^{(1)})}, \quad j = 1, 2, \ldots, M, \\
u_{Nj} &= \frac{a_{Nk}^{(1)} (f_1 - S_1) - a_{Nk}^{(1)} (f_2 - S_2)}{(a_{1N}^{(1)} a_{N1}^{(1)} - a_{1k}^{(1)} a_{Nk}^{(1)})}, \quad j = 1, 2, \ldots, M,
\end{align*}
\]

where \(S_1 = \sum_{k=2}^{N-1} a_{1k}^{(1)} u_{kj}\), \(S_2 = \sum_{k=2}^{N-2} a_{Nk}^{(1)} u\), \(f_1 = f_1(0, y, t)\) and \(f_2 = f_2(1, y, t)\).

Similarly, the Neumann boundary conditions at (4) \(y = 0\) and \(y = 1\) can be approximated as

\[
\begin{align*}
\sum_{k=1}^{M} b_{1k}^{(1)} u_{k} &= f_3(x, 0, t), \\
\sum_{k=1}^{M} b_{Mk}^{(1)} u_{k} &= f_4(x, 1, t), \quad i = 1, 2, \ldots, N.
\end{align*}
\]
Solving Eqs. (34) and (35) for $u_{i,1}$ and $u_{i,M}$, we get

$$u_{i,1} = \frac{1}{b_{i,1}(f_4 - S_4) - b_{i,M}(f_3 - S_3)} \left( b_{i,1}(b_{i,M} - b_{1,1})b_{1,1}^M \right), \quad j = 1, 2, \ldots, M,$$  

$$u_{i,M} = \frac{1}{b_{i,1}(f_3 - S_3) - b_{i,M}(f_4 - S_4)} \left( b_{i,1}(b_{i,M} - b_{1,1})b_{1,1}^M \right), \quad j = 1, 2, \ldots, M,$$  

where $S_3 = \sum_{i=2}^{M-1} b_{i,1}u_{i,1}$, $S_4 = \sum_{i=2}^{M-1} b_{i,1}u_{i,1}$, $f_3 = f_1(x_0, t)$ and $f_4 = f_4(x_1, t)$.

The system of ordinary differential equations (22), (23) with initial conditions (24), (25) and boundary conditions discussed in Sections 3.1 and 3.2 is solved by standard RK4 method.

4. Selection of grid points and stability

The stability of the DQM depends on the eigen-values of differential quadrature discretization matrices. These eigen-values in turn very much depend on the distribution of grid points. It has been shown by Shu [17] in his book that the uniform grid points distribution does not give stable solutions. So, in the whole numerical experiments, we have used the Chebyshev–Gauss–Lobatto grid points to find the numerical solutions of the equation two dimensional telegraph equation (1). The Chebyshev–Gauss–Lobatto grid points are

$$x_i = a + \frac{1}{2} \left( 1 - \cos \left( \frac{(i - 1)\pi}{N - 1} \right) \right) (b - a), \quad i = 1, 2, \ldots, N.$$  

5. Numerical experiments

In this section, the proposed numerical scheme applied on several test problems to show the efficiency and accuracy of the numerical method. The examples are chosen such that their exact solutions are known and already discussed in literature. The numerical computations have been done with the help of software DEV C++ and Matlab. In each example, we have calculated the relative errors and root mean square (RMS) errors given by the following formulas

$$\text{Relative error} = \sqrt{\frac{\sum_{i=1}^{N}(u_{i}(t) - \bar{u}_{i}(t))^2}{\sum_{i=1}^{N}u_{i}(t)^2}},$$

$$\text{RMS error} = \sqrt{\frac{\sum_{i=1}^{N}(u_{i}(t) - \bar{u}_{i}(t))^2}{N \times N}},$$

where $u_{i}$ and $\bar{u}_{i}$ denote the exact and numerical solution of the problem respectively.

Example 1 [16]. In this example, we consider the second order telegraph equation (1) with domain $0 \leq x, y \leq 1$ and $x = y = 1$. The initial and boundary conditions are given by

$$u(x, y, 0) = x^2 + y^2,$$

$$u_t(x, y, 0) = x^2 + y^2 + 1,$$

$$\begin{align*}
  u(0, y, t) &= y^2 + t, \quad 0 \leq y \leq 1, \quad x = 0, \\
  u(1, y, t) &= 1 + y^2 + t, \quad 0 \leq y \leq 1, \quad x = 1, \\
  u(x, 0, t) &= x^2 + t, \quad 0 \leq x \leq 1, \quad y = 0, \\
  u(x, 1, t) &= 1 + x^2 + t, \quad 0 \leq x \leq 1, \quad y = 0.
\end{align*}$$

The exact solution is given by

$$u(x, y, t) = x^2 + y^2 + t, \quad t \geq 0.$$  

Table 1 presents RMS errors, relative errors and CPU time in seconds of Example 1 and a comparison between the PDQM and the two numerical schemes discussed in [16]. It is found that the present scheme gives better relative errors than [16]. The Figs. 1 and 2 show the numerical and exact solutions at time $t = 1.0$ and $t = 3.0$ respectively. The Figures show that the numerical results are in good agreements with the exact solutions.

Example 2 [11]. Consider the second order telegraph equation (1) in the region $0 \leq x, y \leq 1$. The exact solution is given by

$$u(x, y, t) = e^{-t} \sinh(x) \sinh(y), \quad t \geq 0,$$
The initial and boundary conditions are given by

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

Table 1 presents RMS errors, relative errors and CPU time of Example 1 and a comparison of relative errors with other numerical schemes at different times.

<table>
<thead>
<tr>
<th>(T)</th>
<th>PDQM RMS</th>
<th>PDQM rel. error</th>
<th>MLWS-MLS rel. error [16]</th>
<th>MLPG-MLS rel. error [16]</th>
<th>CPU Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>9.26110 \times 10^{-5}</td>
<td>6.77083 \times 10^{-5}</td>
<td>3.202 \times 10^{-5}</td>
<td>9.800 \times 10^{-5}</td>
<td>5</td>
</tr>
<tr>
<td>1.0</td>
<td>8.03601 \times 10^{-5}</td>
<td>6.32769 \times 10^{-5}</td>
<td>9.045 \times 10^{-5}</td>
<td>7.014 \times 10^{-5}</td>
<td>9</td>
</tr>
<tr>
<td>2.0</td>
<td>9.05079 \times 10^{-5}</td>
<td>3.21873 \times 10^{-5}</td>
<td>7.120 \times 10^{-5}</td>
<td>4.071 \times 10^{-5}</td>
<td>18</td>
</tr>
<tr>
<td>3.0</td>
<td>8.89250 \times 10^{-5}</td>
<td>2.34093 \times 10^{-5}</td>
<td>2.012 \times 10^{-4}</td>
<td>9.001 \times 10^{-5}</td>
<td>27</td>
</tr>
<tr>
<td>5.0</td>
<td>9.00659 \times 10^{-5}</td>
<td>1.55602 \times 10^{-5}</td>
<td>...</td>
<td>...</td>
<td>45</td>
</tr>
<tr>
<td>10.0</td>
<td>8.98311 \times 10^{-5}</td>
<td>8.33705 \times 10^{-5}</td>
<td>...</td>
<td>...</td>
<td>90</td>
</tr>
</tbody>
</table>

The initial and boundary conditions are given by

\[
\begin{align*}
  u(x, y, 0) &= \sinh(x) \sinh(y), \\
  u_t(x, y, 0) &= C_0 \sinh(x) \sinh(y), \\
  u(0, y, t) &= 0, \quad 0 \leq y \leq 1, \quad x = 0, \\
  u(1, y, t) &= e^{-t} \sinh(1) \sinh(y), \quad 0 \leq y \leq 1, \quad x = 1, \\
  u(x, 0, t) &= 0, \quad 0 \leq x \leq 1, \quad y = 0, \\
  u(x, 1, t) &= e^{-t} \sinh(x) \sinh(1), \quad 0 \leq x \leq 1, \quad y = 1.
\end{align*}
\]

The initial and boundary conditions are given by

\[
\begin{align*}
  u(x, y, 0) &= \sinh(x) \sinh(y), \\
  u_t(x, y, 0) &= C_0 \sinh(x) \sinh(y), \\
  u(0, y, t) &= 0, \quad 0 \leq y \leq 1, \quad x = 0, \\
  u(1, y, t) &= e^{-t} \sinh(1) \sinh(y), \quad 0 \leq y \leq 1, \quad x = 1, \\
  u(x, 0, t) &= 0, \quad 0 \leq x \leq 1, \quad y = 0, \\
  u(x, 1, t) &= e^{-t} \sinh(x) \sinh(1), \quad 0 \leq x \leq 1, \quad y = 1.
\end{align*}
\]

Table 2 presents RMS errors, relative errors and CPU time in seconds of Example 2 for \(\alpha = 10, \beta = 5\) and \(\alpha = 10, \beta = 0\). The Figs. 3 and 4 are the surface plot of the numerical and exact solutions at time \(t = 1.0\) and \(t = 3.0\) respectively. It is seen from the Figures that the numerical results are very similar to the exact solutions.

Example 3 [16]. In this example, we consider the second order telegraph equation (1) with domain \(0 \leq x, y \leq 1\) and \(\alpha = \beta = 1\). The exact solution is given by

\[
  u(x, y, t) = \cos(t) \sin(x) \sin(y), \quad t \geq 0.
\]

The initial and boundary conditions are given by

\[
\begin{align*}
  u(x, y, 0) &= \sinh(x) \sinh(y), \\
  u_t(x, y, 0) &= C_0 \sinh(x) \sinh(y), \\
  u(0, y, t) &= 0, \quad 0 \leq y \leq 1, \quad x = 0, \\
  u(1, y, t) &= e^{-t} \sinh(1) \sinh(y), \quad 0 \leq y \leq 1, \quad x = 1, \\
  u(x, 0, t) &= 0, \quad 0 \leq x \leq 1, \quad y = 0, \\
  u(x, 1, t) &= e^{-t} \sinh(x) \sinh(1), \quad 0 \leq x \leq 1, \quad y = 1.
\end{align*}
\]
\[ u(x, y, 0) = \sin(x) \sin(y), \]
\[ u_t(x, y, 0) = 0, \]
\[ u(0, y, t) = 0, \quad 0 \leq y \leq 1, \quad x = 0, \]
\[ u(1, y, t) = \cos(t) \sin(1) \sin(y), \quad 0 \leq y \leq 1, \quad x = 1, \]
\[ u(x, 0, t) = 0, \quad 0 \leq x \leq 1, \quad y = 0, \]
\[ u(x, 1, t) = \cos(t) \sin(x) \sin(1), \quad 0 \leq x \leq 1, \quad y = 1. \]

The numerical results of the Example 3 presented in Table 3 in form of RMS errors and relative errors. A comparison is made via the Table 3 between the PDQM and the two numerical schemes discussed in [16]. It is found that the present scheme gives better relative errors than [16]. The Figs. 5 and 6 show the numerical and exact solutions at time \( t = 1.0 \) and \( t = 3.0 \) respectively. The Figures show that the numerical results are in good agreements with the exact solution.

Example 4 [12]. Consider the second order hyperbolic telegraph equation (1) in the region \( 0 \leq x, y \leq 1 \) and with the initial and boundary conditions are given by
\[ u(x, y, 0) = \sin(x) \sin(y), \]
\[ u_t(x, y, 0) = 0, \]
Fig. 3. Numerical (left) and exact (right) solutions of Example 2 with time step length $\Delta t = 0.001$ at $t = 1.0$.

Fig. 4. Numerical (left) and exact (right) solutions of Example 2 with time step length $\Delta t = 0.001$ at $t = 3.0$. 
The exact solution is given by
\[ u(x, y, t) = \cos(t) \sinh(x) \sinh(y). \]

The exact solution is given by
\[ u(x, y, t) = e^{-t(x+y)^2} \sin(\pi x) \sin(\pi y). \quad t \geq 0. \]
With initial and boundary conditions are
\[
\begin{aligned}
  u(x, y, 0) &= \sinh(\pi x) \sinh(\pi y), \\
  u_t(x, y, 0) &= (x + y) \sinh(\pi x) \sinh(\pi y), \\
  u(0, y, t) &= 0, \quad 0 \leq y \leq 1, \quad x = 0, \\
  u(1, y, t) &= 0, \quad 0 \leq y \leq 1, \quad x = 1, \\
  u(x, 0, t) &= 0, \quad 0 \leq x \leq 1, \quad y = 0, \\
  u(x, 1, t) &= 0, \quad 0 \leq x \leq 1, \quad y = 1.
\end{aligned}
\]

The numerical results of the Example 5 are presented in the Table 5 in form of relative and RMS errors. The results are better than the schemes presented in [16] except regular node distribution. The Figs. 9 and 10 compare the exact and numerical solutions with the exact solutions at time \( t = 1.0 \) and \( t = 3.0 \) and it can be seen that the numerical results are in good agreement with the exact solutions.
Consider the second order hyperbolic telegraph equation (1) in the region $0 \leq x, y \leq 1$, $\alpha = \beta = 1$ and the initial conditions

$u(x, y, 0) = \sin(\pi x) \sin(\pi y)$, 
$u_t(x, y, 0) = -\sin(\pi x) \sin(\pi y)$. 

**Example 6** [16]. Consider the second order hyperbolic telegraph equation (1) in the region $0 \leq x, y \leq 1$, $\alpha = \beta = 1$ and the initial conditions

$u(x, y, 0) = \sin(\pi x) \sin(\pi y)$, 
$u_t(x, y, 0) = -\sin(\pi x) \sin(\pi y)$. 

**Fig. 7.** Numerical (left) and exact (right) solutions of Example 4 with time step length $\Delta t = 0.001$ at $t = 1.0$.

**Fig. 8.** Numerical (left) and exact (right) solutions of Example 4 with time step length $\Delta t = 0.001$ at $t = 3.0$. 

The mixed boundary conditions are

\[ \partial_x u(0, y, t) = C_0 p e^{-t} \sin \pi y, \quad 0 \leq y \leq 1, \quad x = 0, \]

\[ u(1, y, t) = 0, \quad 0 \leq y \leq 1, \quad x = 1, \]

\[ u(x, 0, t) = 0, \quad 0 \leq x \leq 1, \quad y = 0, \]

\[ \frac{\partial}{\partial y} u(x, 1, t) = -C_0 p e^{-t} \sin \pi x, \quad 0 \leq x \leq 1, \quad y = 1. \]

The exact solution is given by

\[ u(x, y, t) = e^{-t} \sin(\pi x) \sin(\pi y), \quad t \geq 0. \]

The mixed boundary conditions are

\[ \begin{cases} \frac{\partial u}{\partial x}(0, y, t) = -C_0 p e^{-t} \sin \pi y, & 0 \leq y \leq 1, \quad x = 0, \\ u(1, y, t) = 0, & 0 \leq y \leq 1, \quad x = 1, \\ u(x, 0, t) = 0, & 0 \leq x \leq 1, \quad y = 0, \\ \frac{\partial u}{\partial y}(x, 1, t) = -C_0 p e^{-t} \sin \pi x, & 0 \leq x \leq 1, \quad y = 1. \end{cases} \]

The exact solution is given by

\[ u(x, y, t) = e^{-t} \sin(\pi x) \sin(\pi y), \quad t \geq 0. \]

Table 5 presents RMS errors, relative errors and CPU time in seconds of Example 5 at different times. It is found that the present scheme gives better relative errors than [16]. The Figs. 11 and 12 show the numerical and exact solutions at time \( t = 1.0 \) and \( t = 3.0 \) respectively. The Figures show that the numerical results are in good agreements with the exact solutions.

**Example 7** [16]. Consider the second order hyperbolic telegraph equation (1) in the region \( 0 \leq x, y \leq 1 \) with \( \alpha = \beta = 1 \). The initial and mixed boundary conditions are

\[ \begin{cases} u(x, y, 0) = \log(1 + x + y), \\ u_t(x, y, 0) = \frac{1}{1 + x + y}, \end{cases} \]

\[ \begin{cases} u(x, y, 0) = \log(1 + x + y), \\ u_t(x, y, 0) = \frac{1}{1 + x + y}, \end{cases} \]
The exact solution is given by
\[ u(x, y, t) = \log(1 + x + y + t), \quad 0 \leq x \leq 1, \quad y = 0, \]
\[ \frac{\partial u}{\partial x}(1, y, t) = \frac{1}{1 + t}, \quad 0 \leq y \leq 1, \quad x = 1, \]
\[ \frac{\partial u}{\partial y}(x, 0, t) = \frac{1}{1 + x + t}, \quad 0 \leq x \leq 1, \quad y = 0, \]
\[ u(x, 1, t) = \log(2 + x + t), \quad 0 \leq x \leq 1, \quad y = 1. \]

The exact solution is given by
\[ u(x, y, t) = \log(1 + x + y + t), \quad t \geq 0. \]

The RMS errors, relative errors and CPU time in seconds of Example 6 are presented in Table 7. The Table made a comparison between the PDQM and the two numerical schemes discussed in [16]. It is found that the present scheme gives better relative errors than [16]. The Figs. 13 and 14 show the numerical and exact solutions at time \( t = 1.0 \) and \( t = 3.0 \) respectively and show that the numerical results are in good agreements with the exact solutions.

### Table 6
RMS, relative errors and CPU time of Example 6 and a comparison of relative errors with other numerical schemes at different times.

<table>
<thead>
<tr>
<th>( T )</th>
<th>RMS</th>
<th>Relative error</th>
<th>MLWS-MLS rel. error [16]</th>
<th>MLPG-MLS rel. error [16]</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>( 2.02516 \times 10^{-5} )</td>
<td>( 1.00788 \times 10^{-4} )</td>
<td>( 7.040 \times 10^{-5} )</td>
<td>( 3.701 \times 10^{-5} )</td>
<td>6</td>
</tr>
<tr>
<td>1.0</td>
<td>( 1.11333 \times 10^{-5} )</td>
<td>( 9.15349 \times 10^{-4} )</td>
<td>( 9.088 \times 10^{-5} )</td>
<td>( 7.900 \times 10^{-5} )</td>
<td>11</td>
</tr>
<tr>
<td>2.0</td>
<td>( 5.36320 \times 10^{-6} )</td>
<td>( 1.19624 \times 10^{-4} )</td>
<td>( 4.820 \times 10^{-4} )</td>
<td>( 1.216 \times 10^{-4} )</td>
<td>22</td>
</tr>
<tr>
<td>3.0</td>
<td>( 2.18844 \times 10^{-7} )</td>
<td>( 1.32672 \times 10^{-5} )</td>
<td>( 1.400 \times 10^{-4} )</td>
<td>( 8.392 \times 10^{-4} )</td>
<td>32</td>
</tr>
<tr>
<td>5.0</td>
<td>( 2.96407 \times 10^{-7} )</td>
<td>( 1.32777 \times 10^{-5} )</td>
<td>( 3.701 \times 10^{-4} )</td>
<td>( 3.540 \times 10^{-4} )</td>
<td>54</td>
</tr>
<tr>
<td>10.0</td>
<td>( 5.04962 \times 10^{-11} )</td>
<td>( 3.35744 \times 10^{-6} )</td>
<td>( 3.701 \times 10^{-4} )</td>
<td>( 3.540 \times 10^{-4} )</td>
<td>108</td>
</tr>
</tbody>
</table>

In this paper numerical solutions of the second order two dimensional hyperbolic telegraph equation by PDQM are discussed. Many authors [11–16] have proposed the numerical schemes for the two dimensional hyperbolic telegraph equation. In this study, numerical results obtained by using Gauss–Lobatto–Chebyshev grid points are presented. In all problems the RMS errors, relative errors and CPU times are presented in Tables 1–7 with time step length \( \Delta t = 0.001 \) and \( M = N = 21 \). The numerical results are compared with results presented in [16]. It is found that the present numerical technique gives better relative errors and CPU time. The main advantage of the present scheme is that the scheme gives very accurate and similar results to the exact solutions by choosing less number of grid points and the problem can be solved up to big time. The good thing of the present technique is that it is easy to apply and gives us better accuracy in less numbers of grid points as comparison to the other numerical techniques. The multidimensional problems can be easily handled by the present numerical technique.
Fig. 11. Numerical (left) and exact (right) solutions of Example 6 with time step length $\Delta t = 0.001$ at $t = 1.0$.

Fig. 12. Numerical (left) and exact (right) solutions of Example 6 with time step length $\Delta t = 0.001$ at $t = 3.0$.

<table>
<thead>
<tr>
<th>$T$</th>
<th>RMS</th>
<th>Relative error</th>
<th>MLWS-MLS Rel. error [16]</th>
<th>MLPG-MLS Rel. error [16]</th>
<th>CPU Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>$4.91809 \times 10^{-5}$</td>
<td>$4.91550 \times 10^{-5}$</td>
<td>$7.939 \times 10^{-5}$</td>
<td>$9.991 \times 10^{-5}$</td>
<td>6</td>
</tr>
<tr>
<td>1.0</td>
<td>$5.39567 \times 10^{-5}$</td>
<td>$5.34954 \times 10^{-5}$</td>
<td>$9.098 \times 10^{-5}$</td>
<td>$7.198 \times 10^{-5}$</td>
<td>11</td>
</tr>
<tr>
<td>2.0</td>
<td>$4.95636 \times 10^{-5}$</td>
<td>$3.32086 \times 10^{-5}$</td>
<td>$8.705 \times 10^{-5}$</td>
<td>$8.784 \times 10^{-5}$</td>
<td>22</td>
</tr>
<tr>
<td>3.0</td>
<td>$4.96391 \times 10^{-5}$</td>
<td>$2.77699 \times 10^{-5}$</td>
<td>$9.931 \times 10^{-5}$</td>
<td>$4.801 \times 10^{-4}$</td>
<td>32</td>
</tr>
<tr>
<td>5.0</td>
<td>$4.42952 \times 10^{-5}$</td>
<td>$2.27784 \times 10^{-5}$</td>
<td>$7.302 \times 10^{-3}$</td>
<td>$9.498 \times 10^{-4}$</td>
<td>54</td>
</tr>
<tr>
<td>10.0</td>
<td>$4.32592 \times 10^{-5}$</td>
<td>$1.74125 \times 10^{-5}$</td>
<td>...</td>
<td>...</td>
<td>108</td>
</tr>
</tbody>
</table>
Fig. 13. Numerical (left) and exact (right) solutions of Example 6 with time step length $\Delta t = 0.001$ at $t = 1.0$.

Fig. 14. Numerical (left) and exact (right) solutions of Example 7 with time step length $\Delta t = 0.001$ at $t = 3.0$.

References


