Research Paper
Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. 10, No. 4, 2022, pp. 837-859
DOI:10.22034/cmde.2022.47744.1997

Numerical solution of the hyperbolic telegraph equation using cubic B-spline-based differential quadrature of high accuracy

Athira Babu[1], **Bin Han**[2], **and Noufal Asharaf**[1,*]

[1]Department of Mathematics, Cochin University of Science and Technology, Kerala, India.

[2]Department of Mathematical and Statistical Sciences, University of Alberta, Edmonton, Alberta, Canada.

Abstract

By constructing a newly modified cubic B-splines having the optimal accuracy order four, we propose a numerical scheme for solving the hyperbolic telegraph equation using a differential quadrature method. The spatial derivatives are approximated by the differential quadrature whose weight coefficients are computed using the newly modified cubic B-splines. Our modified cubic B-splines retain the tridiagonal structure and achieve the fourth order convergence rate. The solution of the associated ODEs is advanced in the time domain by the SSPRK scheme. The stability of the method is analyzed using the discretization matrix. Our numerical experiments demonstrate the better performance of our proposed scheme over several known numerical schemes reported in the literature.

Keywords. Hyperbolic telegraph equation, Differential quadrature method, SSPRK scheme, Modified cubic B-spline basis functions, Discretization matrix.

1991 Mathematics Subject Classification. 65M22,65N22,35L10.

1. Introduction

In mathematical physics, hyperbolic equations have a significant role in understanding numerous physical mathematical phenomena, especially in the case of vibrations of structures such as beams, machines, buildings, etc. Many researchers have contributed to develop, analyze, and implement stable numerical methods to solve second-order hyperbolic equations in various ways. It is known that the more appropriate way to model reaction-diffusion is to use the hyperbolic telegraph equations than the ordinary diffusion equations in many contexts. The telegraph equation was proposed by Oliver Heaviside to depict current and voltage in an electric transmission line. The current and voltage often form wave patterns when a reflection of electromagnetic waves are present. In a double conductor, the flow of both electric current and voltage are modelled by the telegraph equation. In short, mathematical models in wave propagation, random walk theory, and heat diffusion equations are special cases of the telegraph equation. Also, the telegraph equation arises in the study of pulsating blood flow in arteries, parallel flows of viscous Maxwell fluids, and one-dimensional random motion of bugs along a hedge. For more details, see [7, 9, 19, 28, 43, 45] and references therein. The second-order one-dimensional hyperbolic telegraph equation has a dissipative form as follows:

$$\begin{aligned} &u_{tt}(x,t)+2\alpha u_t(x,t)+\beta^2 u(x,t)=u_{xx}(x,t)+f(x,t), \quad x\in\Omega:=[a,b],\ t\in(t_0,T],\\ &\text{IC: }\begin{cases}u(x,t_0)=f_1(x),\\ u_t(x,t_0)=f_2(x),\end{cases} \qquad \text{BC: }\begin{cases}u(a,t)=g_1(x),\\ u(b,t)=g_2(x).\end{cases} \end{aligned} \tag{1.1}$$

Received: 02 September 2021 ; Accepted: 14 January 2022.
∗ Corresponding author. Email: noufal@cusat.ac.in .

Its two-dimensional form is given by

$$\begin{aligned} &u_{tt}(x,y,t)+2\alpha u_t(x,y,t)+\beta^2 u(x,y,t)=u_{xx}(x,y,t)+u_{yy}(x,y,t)+f(x,y,t),\\ &\qquad\qquad (x,y)\in\Omega:=[a,b]\times[c,d]\,,\ t\in(t_0,T]\,,\\ &\text{IC}:\begin{cases}u(x,y,t_0) &=f_1(x,y),\\ u_t(x,y,t_0)&=f_2(x,y),\end{cases}\qquad \text{BC}:\begin{cases}u(a,y,t)=g_1(y,t), & u(b,y,t)=g_2(y,t),\\ u(x,c,t)=g_3(x,t), & u(x,d,t)=g_4(x,t),\end{cases}\end{aligned} \tag{1.2}$$

where α,β are strictly positive constants, Ω is the spatial domain and $(t_0,T]$ denotes the time interval. Several methods have been developed in the literature to numerically solve the hyperbolic telegraph equations. Lakestani in [20] used interpolating scaling functions, which are expressed in terms of Lagrange interpolating polynomials and Gauss-Legendre quadrature weights introduced by Shamsi and Razzagi [36]. In this collocation method, derivatives are arranged in terms of a block tridiagonal matrix for fast computation of the associated linear system. Motivated by [20], Bicer et al. in [8] solved the telegraph equation (1.1) using the Bernoulli collocation. B-spline collocation method is used to solve the problem in [46] and Sinc-Chebyshev collocation method is used to solve the time-fractional order telegraph equation in [42]. Meshfree schemes using the radial basis functions method were found to be effective and are widely used in interpolating scattered data. Such an approach is used by Dehghan [10] to solve the telegraph equation with the help of thin-plate splines as radial basis functions. Dehghan et al. in [11] numerically solved (1.2) based on a meshless local weak–strong (MLWS) method. Instead of using a pre-defined mesh for the discretized domain, they employed a system consisting of algebraic equations for the whole domain. Petrov-Galerkin weak form is applied locally at the nodes on the Neumann boundary for estimating the solution of the problem. Another meshless technique in [35] solves (1.2) by discretizing the spatial derivatives with the radial basis function differentiation matrix. Abbasbandy et al. in [1] provided a detailed comparison study between direct and indirect meshfree methods including popular meshfree methods based on the strong form equation, the approximate particular solutions method, nonsymmetric radial basis function collocation and their local forms. Lin et al. in [21] studied the two-dimensional telegraph equation for arbitrary domains using an accurate meshless collocation technique. Moving least square meshless method is used with radial basis functions to solve the problem in two-dimensions by Sepideh Niknam et al. in [31]. Aslefallah et al. in [5] considered the problem for an arbitrary domain and the Houbolt finite difference method to discretize the time domain. A combination of singular boundary and particular solution methods are employed to approximate the solution. A hybrid meshless method is proposed by Zhou et al. in [47] based on the Houbolt method where the time derivatives are approximated through finite differences. Hafez in [12] proposed a numerical technique based on shifted Jacobi collocation to solve linear and nonlinear hyperbolic telegraph equations with variable coefficients in one and two dimensions. Pandit et al. studied the efficiency of the reproducing kernel technique to numerically solve partial differential equations, especially in the case of one-dimensional telegraph equation in [32]. Also the numerical solutions of hyperbolic type partial differential equation using different approaches have been reported in [3, 18, 33]. Differential quadrature is a well-known method for approximating the derivatives at each grid point using sample values. Different strategies have been considered in the literature to obtain the weight coefficients if the functions can be effectively approximated by certain polynomial spaces. Pekmen et al. in [34] used both polynomial-based and Fourier-based differential quadrature algorithms for the discretization of the telegraph equation in one and two dimensions. The numerical solution is obtained by choosing Gauss-Chebyshev-Lobatto grid points in the spatial domain and equally spaced grid points in the time domain. Jiwari used the Lagrange interpolation technique and a differential quadrature technique based on modified cubic B-spline used to numerically solve hyperbolic partial differential equations in [17]. A lot of research in the literature employ many different variants of differential quadrature. For example, the differential quadrature technique is used to approximate the spatial derivatives, in which the weight coefficients are calculated through a linear system obtained from B-spline basis functions and then the Runge-Kutta method is used to progress in time (see [2, 4, 6, 15, 16, 24, 27, 36, 39, 40, 44]). In recent years, the polynomial space spanned by the cubic B-spline functions have been considered for the collocation method (see [22, 23, 25, 37]). In these works, a modified set of cubic B-splines are defined as a basis for the required polynomial space in a uniform grid. But this set of modified splines fails to preserve the optimal polynomial reproduction property near the boundary. Even though the cubic B-splines are competent to generate the polynomial space $\mathbb{P}_3$ (i.e., it has an accuracy order four), the current known modified set fails to generate the polynomial x^2 especially near the boundary of the domain. This disadvantage greatly

reduces the approximation ability of the modified cubic B-splines. In this paper, we propose a new modification of the standard cubic splines that retains the optimal accuracy order four and preserve the desired diagonally dominant tridiagonal structure. We follow a similar strategy as in [26] to numerically approximate the solution of the hyperbolic telegraph equation. Since our newly modified cubic B-splines can produce all polynomials of degree up to 3 in the whole computational domain, the approximation with such new basis functions gives a more accurate solution with the optimal approximation order four in the computational domain, as compared to the corresponding results in the literature. This polynomial reproduction property is found to be optimal, and hence the proposed scheme effectively performs in the computation process. In comparison with Jacobi collocation, the cubic B-spline-based differential quadrature is less computationally complex. Modified cubic B-splines are considered for discretization in a uniform mesh of the spatial domain. To illustrate the effectiveness of this simple procedure, we analyze the results of test problems and compare them with other known results in the literature. Since our modified set of splines effectively catch the polynomial space of degree up to three, the approximation accuracy is significantly higher than the modified cubic B-splines in [22, 23, 25, 37]. Also, at each time the solution converges to the exact solution stably. The stability of the scheme is studied and we observe that the eigenvalues of the discretization matrix lie on the left half-plane. The structure of this article is as follows. In section 2, we propose the numerical method to solve the telegraph equation in one- and two-dimensions. A cubic B-spline-based differential quadrature method is used to approximate the spatial derivatives. The newly modified cubic B-spline basis functions are introduced and the computation of weight coefficients using these basis functions are given in section 3. The stability analysis of our proposed scheme is given in section 4. In section 5, we illustrate the computational results for one- and two-dimensional telegraph equations and a detailed comparison with other known works in the literature. The conclusion of the paper is provided in section 6 with remarks about the main observations and advantages of the proposed method.

2. Methodology of the proposed numerical scheme

In this section, we explain our proposed numerical algorithm for solving hyperbolic telegraph equation in one and two dimensions.

2.1. The one-dimensional telegraph equation.

Consider the telegraph equation Eq. (1.1) with the given initial and boundary conditions defined in the spatial domain $\Omega = [a, b]$. Using the transformation $w = u_t$, it can be written in the following form:

$$\begin{aligned} u_t(x,t) &= w(x,t), \\ w_t(x,t) &= -2\alpha w(x,t) - \beta^2 u(x,t) + u_{xx}(x,t) + f(x,t). \end{aligned} \tag{2.1}$$

For a positive integer N, we uniformly partition the domain $[a, b]$ into $a = x_0 < x_1 < \cdots < x_N = b$ with the mesh size $h = \frac{b-a}{N}$. Using the differential quadrature technique, we could approximate the spatial derivative at each node point $x_i,\ i = 0, 1, \ldots, N$ as

$$u_{xx}(x_i,t) = \sum_{k=0}^{N} p_{ik}^{(2)}\, u(x_k,t), \tag{2.2}$$

where $p_{ik}^{(2)},\ i,k = 0, 1, \ldots, N$ are the weight coefficients for the second-order partial derivatives with respect to the space variable. The coefficients are computed by using the cubic B-spline based differential quadrature algorithm, which is explained in section 3. Eventually, we can represent Eq. (2.1) as a system of first-order ordinary differential equations

$$\begin{aligned} \frac{d\,u_i}{dt} &= w(x_i,t), \\ \frac{d\,w_i}{dt} &= -2\alpha w(x_i,t) - \beta^2 u(x_i,t) + \sum_{k=0}^{N} p_{ik}^{(2)} u(x_k,t) + f(x_i,t), \end{aligned} \tag{2.3}$$

for $i = 0, 1, \ldots, N$. Using the SSPRK-54 scheme (see [41]), we can solve the system of ordinary differential equations for the next time step. We start the computation with the initial and boundary conditions given by

$$IC: \begin{cases} u(x_i, 0) = f_1(x_i), \\ u_t(x_i, 0) = f_2(x_i), \end{cases} \qquad BC: \begin{cases} u(a, t) = g_1(t), \\ u(b, t) = g_2(t). \end{cases}$$

2.2. **The two-dimensional telegraph equation.** Now moving to the two-dimensional case, we consider the telegraph equation Eq. (1.2) with the accompanying initial and boundary conditions. Using a similar approach as in the one-dimensional case, we apply the transformation $w = u_t$ which yields a system as follows:

$$u_t(x, y, t) = w(x, y, t),$$
$$w_t(x, y, t) = -2\alpha w(x, y, t) - \beta^2 u(x, y, t) + u_{xx}(x, y, t) + u_{yy}(x, y, t) + f(x, y, t).$$

For selected positive integers N, M, we uniformly discretize the space domain Ω as $a = x_0 < x_1 < \cdots < x_N = b$ and $c = y_0 < y_1 < \cdots < y_M = d$. For $i = 0, 1, \ldots, N$ and $j = 0, 1, \ldots, M$, using the differential quadrature technique we represent the second-order partial derivatives in space variables as

$$u_{xx}(x_i, y_j, t) = \sum_{k=0}^{N} p_{ik}^{(2)}\, u(x_k, y_j, t), \quad u_{yy}(x_i, y_j, t) = \sum_{k=0}^{M} q_{jk}^{(2)}\, u(x_i, y_k, t).$$

This leads to the following system of ordinary differential equations

$$\begin{aligned} \frac{d\, u_{ij}}{dt} &= w(x_i, y_j, t), \\ \frac{d\, w_{ij}}{dt} &= -2\alpha w(x_i, y_j, t) - \beta^2 u(x_i, y_j, t) + \sum_{k=0}^{N} p_{ik}^{(2)} u(x_k, y_j, t) + \sum_{k=0}^{M} q_{jk}^{(2)} u(x_i, y_k, t) + f(x_i, y_j, t), \end{aligned} \tag{2.4}$$

for $i = 0, 1, \ldots, N$ and $j = 0, 1, \ldots, M$. This system can be solved for the next time step using SSPRK-43 scheme. The initial and boundary conditions are given by

$$IC: \begin{cases} u(x_i, , y_j, 0) = f_1(x_i, y_j), \\ u_t(x_i, y_j, 0) = f_2(x_i, y_j), \end{cases} \qquad BC: \begin{cases} u(a, y_j, t) = g_1(y_j, t), \quad u(b, y_j, t) = g_2(y_j, t), \\ u(x_j, c, t) = g_3(x_j, t), \quad u(x_j, d, t) = g_4(x_j, t). \end{cases}$$

3. Computation of weight coefficients using modified cubic B-splines

In this section we discuss how to compute the weight coefficients in the approximation (2.2) using the newly modified cubic B-splines. We use a cubic B-spline-based differential quadrature algorithm for this purpose. Using a newly modified cubic B-spline basis functions, we can achieve better accuracy in the approximation of a functions on a given computational domain.

3.1. **Modified cubic B-spline elements.** Let $h > 0$ and $x_j := a + jh$ for $j \in \mathbb{Z}$. The standard cubic B-spline $B_j(x)$ is defined by

$$B_j(x) := \frac{1}{h^3} \begin{cases} (x - x_{j-2})^3, & x \in [x_{j-2},\, x_{j-1}), \\ (x - x_{j-2})^3 - 4(x - x_{j-1})^3, & x \in [x_{j-1},\, x_j), \\ (x_{j+2} - x)^3 - 4(x_{j+1} - x)^3, & x \in [x_j,\, x_{j+1}), \\ (x_{j+2} - x)^3, & x \in [x_{j+1},\, x_{j+2}), \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

Note that the cubic spline B_j is supported inside $[x_{j-2}, x_{j+2}]$ and B_j is a nonnegative polynomial of degree 3 on every subinterval $[x_k, x_{k+1}]$ for $k = j - 2, \ldots, j + 1$. Moreover, $B_j \in C^2(\mathbb{R})$ is twice differentiable and B_j is symmetric about x_j. The function values, first-order derivatives and second-order derivatives of B_j at the knot points $\{x_k\}_{k\in\mathbb{Z}}$ are known and explicitly given in Table 1.

These cubic spline functions $\{B_{-1}, B_0, B_1, \ldots, B_N, B_{N+1}\}$ play the role of a basis in the space of spline functions for approximating the solutions of a differential equation on the domain $[a, b]$. Because the standard cubic spline

TABLE 1. Derivatives of cubic splines at each node

x	x_{j-2}	x_{j-1}	x_j	x_{j+1}	x_{j+2}
$B_j(x)$	0	1	4	1	0
$B_j'(x)$	0	$3/h$	0	$-3/h$	0
$B_j''(x)$	0	$6/h^2$	$-12/h^2$	$6/h^2$	0

B_j is supported on $[x_{j-2}, x_{j+2}]$ and $x_0 = a, x_N = b$, one can easily observe that the cubic splines B_{-1}, B_0, B_1 and B_{N-1}, B_N, B_{N+1} are not fully supported inside the problem domain $[a, b]$. Therefore, it is necessary to modify such standard cubic splines near the boundary. A modification using two boundary functions at each of the boundary points is proposed in [22]. More precisely, the basis functions used in these papers are $\{\tilde{B}_0, \tilde{B}_1\} \cup \{B_j \ : \ 2 \le j \le N-2\} \cup \{\tilde{B}_{N-1}, \tilde{B}_N\}$, where the modified boundary cubic splines $\tilde{B}_0, \tilde{B}_1$ near the boundary point a and the modified boundary cubic splines $\tilde{B}_{N-1}, \tilde{B}_N$ near the boundary point b are given by

$$\begin{aligned} \tilde{B}_0(x) &:= [B_0(x) + 2B_{-1}(x)]\,\chi_{[a,\,b]}(x), & \tilde{B}_N(x) &:= [B_N(x) + 2B_{N+1}(x)]\,\chi_{[a,\,b]}(x), \\ \tilde{B}_1(x) &:= [B_1(x) - B_{-1}(x)]\,\chi_{[a,\,b]}, & \tilde{B}_{N-1}(x) &:= [B_{N-1}(x) - B_{N+1}(x)]\,\chi_{[a,\,b]}(x). \end{aligned} \tag{3.2}$$

For simplicity of presentation, we shall also define $\tilde{B}_j(x) := B_j(x)$ for all $j = 2, \ldots, N-2$, where B_j is the standard interior cubic spline in Eq. (3.1). For a non-negative integer m, by $\mathbb{P}_{m-1}$ we denote the space of all polynomials of degree less than m. If $\{\varphi_j\}_{j=0}^N$ is a collection of compactly supported functions on $[a, b]$ and $\mathbb{P}_{m-1}\chi_{[a,\,b]} \subset \operatorname{span}\{\varphi_j\}_{j=0}^N$, then we say that $\{\varphi_j\}_{j=0}^N$ has *accuracy order* m on $[a, b]$, where $\operatorname{span}\{\varphi_j\}_{j=0}^N$ consists of all the elements of the form $\sum_{j=0}^N c_j\varphi_j$ for all $c_j \in \mathbb{R}$. It is well known that accuracy order is closely related to the performance of the associated numerical schemes.

Proposition 3.1. *The system $\{\tilde{B}_j \ : \ 0 \le j \le N\}$ in [22] with modified boundary cubic splines in Eq. (3.2) and the interior standard cubic splines $\tilde{B}_j := B_j$ for $2 \le j \le N-2$ has accuracy order two but cannot have accuracy order three.*

Proof. In fact, by direct calculation, we have $\frac{1}{6}\sum_{j=0}^N \tilde{B}_j(x) = 1$ and $\frac{h}{6}\sum_{j=0}^N j\tilde{B}_j(x) = x$ for all $x \in [a, b]$. On the other hand, for $x \in [a, a+h]$, we have

$$\begin{aligned} h^3\tilde{B}_0(x) &= x^3 - 3\,ax^2 + \left(3\,a^2 - 6\,h^2\right)x - a^3 + 6\,ah^2 + 6\,h^3, \\ h^3\tilde{B}_1(x) &= -2\,x^3 + 6\,ax^2 + (6\,h^2 - 6\,a^2)x + 2\,a^3 - 6\,ah^2, \\ h^3B_2(x) &= x^3 - 3\,ax^2 + 3\,a^2x - a^3, \end{aligned}$$

and $B_j(x) = 0$ for all $j > 2$ and $x \in [a, a+h]$. We claim that any combination of these three functions cannot generate the polynomials of degree two in the interval $[a, a+h]$. If we denote $p(x) = p_0x^3 + p_1x^2 + p_2x + p_3$ as the polynomial to be represented as a combination of these three functions with coefficients w_1, w_2, w_3, the corresponding matrix form $Aw = p$ will be

$$\begin{bmatrix} 1 & -2 & 1 \\ -3a & 6a & -3a \\ 3a^2 - 6h^2 & 6h^2 - 6a^2 & 3a^2 \\ -a^3 + 6ah^2 + 6h^3 & 2a^3 - 6ah^2 & -a^3 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}.$$

By removing the second row of A (one linearly dependent row) the resulting matrix has a nonzero determinant, ie.,

$$\det\left(\begin{bmatrix} 1 & -2 & 1 \\ 3a^2 - 6h^2 & 6h^2 - 6a^2 & 3a^2 \\ -a^3 + 6ah^2 + 6h^3 & 2a^3 - 6ah^2 & -a^3 \end{bmatrix}\right) = -36h^5 \neq 0.$$

Hence A has rank 3. In the case of the augmented matrix $A|p$,

$$\det(A|p) = \det\left(\begin{bmatrix} 1 & -2 & 1 & p_0 \\ -3a & 6a & -3a & p_1 \\ 3a^2-6h^2 & 6h^2-6a^2 & 3a^2 & p_2 \\ -a^3+6ah^2+6h^3 & 2a^3-6ah^2 & -a^3 & p_3 \end{bmatrix}\right) = -108ah^5p_0 - 36h^5p_1,$$

and $\det(A|p) = 0$ if and only if $p_1 = -3ap_0$. Thus the system $Aw = p$ has a solution only if $\text{Rank}(A) = \text{Rank}(A|p)$, ie., when $p_1 = -3ap_0$. In other words, the coefficients w_1, w_2, w_3 exist only for the polynomials of the form $p(x) = p_0(x^3 - 3ax^2) + p_2x + p_3$. Hence we can conclude that

$$\text{span}\left(1,\ x,\ x^3 - 3ax^2\right) = \text{span}(\{\tilde{B}_0(x), \tilde{B}_1(x), \tilde{B}_2(x)\}) \quad \text{in} \quad [a, a+h].$$

In particular, $x^2 \notin \text{span}(\{\tilde{B}_0(x), \tilde{B}_1(x), \tilde{B}_2(x)\})$ on $[a, a+h]$.

A similar thing happens at the right boundary $[b-h, b]$ also. In all other intervals we could find four linearly independent polynomials passing through and hence possess accuracy order four in $[a+h, b-h]$. But it lacks the optimal accuracy order on the boundary intervals. Combining the above argument, we conclude that the linear span of $\{\tilde{B}_j : 0 \le j \le N\}$ cannot produce all the second and third-degree polynomials throughout the interval $[a, b]$. This proves our claim. □

However, it is well known that the standard cubic B-splines have accuracy order four, that is, the standard cubic splines can reproduce all polynomials of degree three or less. Therefore, according to Proposition 3.1, the modified cubic splines Eq. (3.2) do not achieve the best possible accuracy order four of the standard cubic splines. The lower accuracy order of these modified cubic splines leads to a non-optimal performance of their associated numerical schemes. Therefore, it is necessary to make an appropriate modification near the boundary so that the modified cubic splines can preserve the optimal accuracy order.

In this paper, we aim at modifying the cubic splines near each boundary so that the newly modified cubic splines not only preserve the optimal accuracy order but also keep the tridiagonal matrix structure. These two properties are very important for us to use such modified cubic splines to solve the one- and two-dimensional telegraph equation. On one hand, as in [22], the associated matrix obtained here after discretization will be tridiagonal as well, which allows us to use the Thomas algorithm for effectively solving the corresponding linear system. On the other hand, our proposed modified cubic B-splines having the accuracy order four allow the associated numerical scheme to be effective and perform well. As demonstrated by our numerical experiments in section 5, the numerical schemes using our developed modified cubic B-splines having accuracy order four significantly outperforms the numerical schemes described in the literature.

Theorem 3.2. *Let N be a positive integer greater than seven and let $a = x_0 < x_1 < \cdots < x_N = b$, where $x_j := a + jh$ for all $j \in \mathbb{Z}$ and $h := \frac{b-a}{N}$. Define the modified cubic splines near the boundary by*

$$\begin{aligned}
\breve{B}_0 &:= B_0 + 4B_{-1}, & \breve{B}_N &:= B_N + 4B_{N+1}, \\
\breve{B}_1 &:= B_1 - \tfrac{7}{2}B_{-1} + \tfrac{5}{8}B_0, & \breve{B}_{N-1} &:= B_{N-1} - \tfrac{7}{2}B_{N+1} + \tfrac{5}{8}B_N, \\
\breve{B}_2 &:= B_2 + \tfrac{88}{37}B_{-1} - \tfrac{21}{37}B_0 - \tfrac{4}{37}B_1, & \breve{B}_{N-2} &:= B_{N-2} + \tfrac{88}{37}B_{N+1} - \tfrac{21}{37}B_N - \tfrac{4}{37}B_{N-1}, \\
\breve{B}_3 &:= B_3 - B_{-1} + \tfrac{1}{4}B_0 - \tfrac{1}{4}B_2, & \breve{B}_{N-3} &:= B_{N-3} - B_{N+1} + \tfrac{1}{4}B_N - \tfrac{1}{4}B_{N-2},
\end{aligned} \tag{3.3}$$

where the standard cubic splines B_j, $j \in \mathbb{Z}$ are defined in Eq. (3.1). For simplicity of presentation, we also define $\breve{B}_j := B_j$ for all $j = 4, \ldots, N-4$. Then the system $\{\breve{B}_j : j = 0, \ldots, N\}$ of the modified cubic splines has the accuracy order four on $[a, b]$.

Proof. Consider $x \in [a,b]$, we have

$$
\begin{aligned}
h^3\breve{B}_0(x) &= -x^3 + (3\,a + 6\,h)x^2 - (3\,a^2 + 12\,ah + 12\,h^2)x + (a^3 + 6\,a^2h + 12\,ah^2 + 8\,h^3),\\
h^3\breve{B}_1(x) &= \frac{1}{8}\Big[19\,x^3 - (57\,a + 90\,h)x^2 + (57\,a^2 + 180\,ah + 108\,h^2) - (19\,a^3 + 90\,a^2h + 108\,ah^2)\Big],\\
h^3\breve{B}_2(x) &= \frac{1}{37}\Big[-102\,x^3 + (306\,a + 378\,h)x^2 - (306\,a^2 + 756\,ah + 276\,h^2)x + (102\,a^3 + 378\,a^2h + 276\,ah^2)\Big],\\
h^3\breve{B}_3(x) &= \frac{1}{2}\Big[3\,x^3 - (9\,a + 9\,h)x^2 + (9\,a^2 + 18\,ah + 6\,h^2)x - (3\,a^3 + 9\,a^2h + 6\,ah^2)\Big].
\end{aligned}
\tag{3.4}
$$

The coordinate matrix of these four polynomials relative to the standard ordered basis $\{x^3, x^2, x, 1\}$ is given by

$$
B = \begin{bmatrix}
-1 & 3\,a + 6\,h & -(3\,a^2 + 12\,ah + 12\,h^2) & a^3 + 6\,a^2h + 12\,ah^2 + 8\,h^3\\
\frac{19}{8} & -\frac{1}{8}(57\,a + 90\,h) & \frac{1}{8}(57\,a^2 + 180\,ah + 108\,h^2) & -\frac{1}{8}(19\,a^3 + 90\,a^2h + 108\,ah^2)\\
-\frac{102}{37} & \frac{1}{37}(306\,a + 378\,h) & -\frac{1}{37}(306\,a^2 + 756\,ah + 276\,h^2) & \frac{1}{37}(102\,a^3 + 378\,a^2h + 276\,ah^2)\\
\frac{3}{2} & -\frac{1}{2}(9\,a + 9\,h) & \frac{1}{2}(9\,a^2 + 18\,ah + 6\,h^2) & -\frac{1}{2}(3\,a^3 + 9\,a^2h + 6\,ah^2)
\end{bmatrix}^T,
$$

where $\det(B) = 108h^6 \neq 0$. Since the coordinate matrix is invertible, the four polynomials in Eq. (3.4) are linearly independent and thus the polynomials $1, x, x^2, x^3$ are in the space spanned by these polynomials. A similar thing happens on the right boundary too. Since the standard cubic B-splines have accuracy order four, thus as a whole we have the system $\{\breve{B}_j \ : \ j = 0, \ldots, N\}$ has accuracy order four on $[a,b]$. This proves the result. In particular, for $a = 0$ and $x \in \Omega$ we have

$$
\begin{aligned}
1 &= \frac{1}{8}\left(\breve{B}_0(x) + \breve{B}_N(x)\right) + \frac{7}{37}\left(\breve{B}_1(x) + \breve{B}_{N-1}(x)\right) + \frac{5}{24}\left(\breve{B}_2(x) + \breve{B}_{N-2}(x)\right) + \frac{1}{6}\sum_{j=3}^{N-3}\breve{B}_j(x),\\
\frac{x}{h} &= \frac{8}{37}\breve{B}_1(x) + \frac{11}{24}\breve{B}_2(x) + \frac{1}{2}\breve{B}_3(x) + \frac{7}{6}\breve{B}_{N-3}(x) + \frac{13}{8}\breve{B}_{N-2}(x) + \frac{62}{37}\breve{B}_{N-1}(x) + \frac{5}{4}\breve{B}_N(x) + \frac{1}{6}\sum_{j=4}^{N-4} j\breve{B}_j(x),\\
\frac{x^2}{h^2} &= \frac{8}{37}\breve{B}_1(x) + \frac{35}{36}\breve{B}_2(x) + \frac{13}{9}\breve{B}_3(x) + \frac{73}{9}\breve{B}_{N-3}(x) + \frac{455}{36}\breve{B}_{N-2}(x) + \frac{548}{37}\breve{B}_{N-1}(x) + \frac{25}{2}\breve{B}_N(x) + \sum_{j=4}^{N-4}\frac{3j^2-1}{18}\breve{B}_j(x),\\
\frac{x^3}{h^3} &= \frac{8}{37}\breve{B}_1(x) + 2\breve{B}_2(x) + 4\breve{B}_3(x) + 56\breve{B}_{N-3}(x) + 98\breve{B}_{N-2}(x) + \frac{4832}{37}\breve{B}_{N-1}(x) + 125\breve{B}_N(x) + \frac{1}{6}\sum_{j=4}^{N-4}(j^3 - j)\breve{B}_j(x).
\end{aligned}
$$

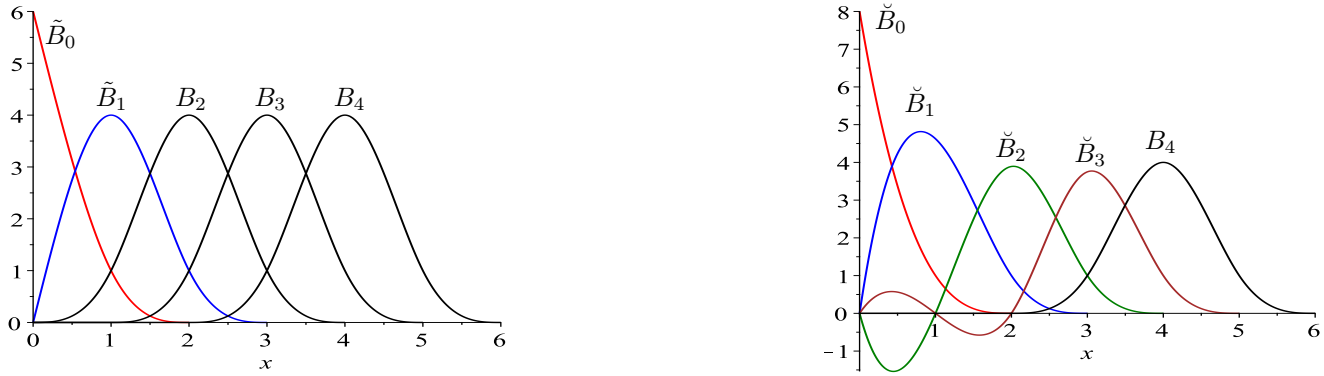
This proves our claim. □

See Figure 1 for the modified cubic splines in Proposition 3.1 with accuracy order two and our improved modified cubic splines in Theorem 3.2 with accuracy order four. For the system $\{\breve{B}_j \ : \ j = 0, \ldots, N\}$, we define

$$
\vec{B}(x) := \Big[\breve{B}_0(x), \breve{B}_1(x), \breve{B}_2(x), \breve{B}_3(x), \breve{B}_4(x), \ldots, \breve{B}_{N-4}(x), \breve{B}_{N-3}(x), \breve{B}_{N-2}(x), \breve{B}_{N-1}(x), \breve{B}_N(x)\Big].
$$

Then $[\vec{B}(x_0)^T, \vec{B}(x_1)^T, \ldots, \ldots, \vec{B}(x_{N-1})^T, \vec{B}(x_N)^T]^T$ is a diagonally dominant tridiagonal matrix:

$$
\begin{bmatrix}
\vec{B}(x_0)\\ \vec{B}(x_1)\\ \vec{B}(x_2)\\ \vec{B}(x_3)\\ \vec{B}(x_4)\\ \vec{B}(x_5)
\end{bmatrix}
=
\begin{bmatrix}
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots\\
1 & \frac{37}{8} & 0 & 0 & 0 & 0 & 0 & 0 & \ldots\\
0 & 1 & \frac{144}{37} & 0 & 0 & 0 & 0 & 0 & \ldots\\
0 & 0 & 1 & \frac{15}{4} & 1 & 0 & 0 & 0 & \ldots\\
0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & \ldots\\
0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & \ldots
\end{bmatrix},
\qquad
\begin{bmatrix}
\vec{B}(x_{N-5})\\ \vec{B}(x_{N-4})\\ \vec{B}(x_{N-3})\\ \vec{B}(x_{N-2})\\ \vec{B}(x_{N-1})\\ \vec{B}(x_N)
\end{bmatrix}
=
\begin{bmatrix}
\ldots & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0\\
\ldots & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0\\
\ldots & 0 & 0 & 0 & 1 & \frac{15}{4} & 1 & 0 & 0\\
\ldots & 0 & 0 & 0 & 0 & 0 & \frac{144}{37} & 1 & 0\\
\ldots & 0 & 0 & 0 & 0 & 0 & 0 & \frac{37}{8} & 1\\
\ldots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8
\end{bmatrix}.
$$



(A) (B)

FIGURE 1. (a) Modified cubic B-spline basis elements in Proposition 3.1 with accuracy order 2 (see [22, 23, 25]). $\tilde{B}_0$ and $\tilde{B}_1$ are modified boundary cubic splines at the boundary point 0 in Eq. (3.2), while B_2, B_3, B_4 are the standard cubic splines with $h = 1$ in Eq. (3.1). (b) Our improved modified cubic B-spline basis elements in Theorem 3.2 with the accuracy order 4. $\breve{B}_0, \ldots, \breve{B}_3$ are the modified boundary cubic splines at the boundary point 0 in Eq. (3.3).

Therefore, we can use the Thomas algorithm for finding its inverse. Also the matrices of the first and second-order derivatives $\vec{B}'$ and $\vec{B}''$ at each node point x_j are given below:

$$\begin{bmatrix} \vec{B}'(x_0) \\ \vec{B}'(x_1) \\ \vec{B}'(x_2) \\ \vec{B}'(x_3) \\ \vec{B}'(x_4) \\ \vec{B}'(x_5) \end{bmatrix} = \frac{1}{h} \begin{bmatrix} -12 & \frac{27}{2} & -\frac{276}{37} & 3 & 0 & 0\,0\,0\,\ldots \\ -3 & -\frac{15}{8} & \frac{174}{37} & -\frac{3}{2} & 0 & 0\,0\,0\,\ldots \\ 0 & -3 & \frac{12}{37} & 3 & 0 & 0\,0\,0\,\ldots \\ 0 & 0 & -3 & \frac{3}{4} & 3 & 0\,0\,0\,\ldots \\ 0 & 0 & 0 & -3 & 0 & 3\,0\,0\,\ldots \\ 0 & 0 & 0 & 0 & -3 & 0\,3\,0\,\ldots \end{bmatrix}, \quad \begin{bmatrix} \vec{B}'(x_{N-5}) \\ \vec{B}'(x_{N-4}) \\ \vec{B}'(x_{N-3}) \\ \vec{B}'(x_{N-2}) \\ \vec{B}'(x_{N-1}) \\ \vec{B}'(x_N) \end{bmatrix} = \frac{1}{h} \begin{bmatrix} \ldots\,0 & -3 & 0 & 3 & 0 & 0 & 0 & 0 \\ \ldots\,0 & 0 & -3 & 0 & 3 & 0 & 0 & 0 \\ \ldots\,0 & 0 & 0 & -3 & -\frac{3}{4} & 3 & 0 & 0 \\ \ldots\,0 & 0 & 0 & 0 & -3 & -\frac{12}{37} & 3 & 0 \\ \ldots\,0 & 0 & 0 & 0 & \frac{3}{2} & -\frac{174}{37} & \frac{15}{8} & 3 \\ \ldots\,0 & 0 & 0 & 0 & -3 & \frac{276}{37} & -\frac{27}{2} & 12 \end{bmatrix},$$

$$\begin{bmatrix} \vec{B}''(x_0) \\ \vec{B}''(x_1) \\ \vec{B}''(x_2) \\ \vec{B}''(x_3) \\ \vec{B}''(x_4) \\ \vec{B}''(x_5) \end{bmatrix} = \frac{1}{h^2} \begin{bmatrix} 12 & -\frac{45}{2} & \frac{756}{37} & -9 & 0 & 0 & 0\ldots \\ 6 & -\frac{33}{4} & \frac{144}{37} & 0 & 0 & 0 & 0\ldots \\ 0 & 6 & -\frac{468}{37} & 9 & 0 & 0 & 0\ldots \\ 0 & 0 & 6 & -\frac{27}{2} & 6 & 0 & 0\ldots \\ 0 & 0 & 0 & 6 & -12 & 6 & 0\ldots \\ 0 & 0 & 0 & 0 & 6 & -12 & 6\ldots \end{bmatrix}, \quad \begin{bmatrix} \vec{B}''(x_{N-5}) \\ \vec{B}''(x_{N-4}) \\ \vec{B}''(x_{N-3}) \\ \vec{B}''(x_{N-2}) \\ \vec{B}''(x_{N-1}) \\ \vec{B}''(x_N) \end{bmatrix} = \frac{1}{h^2} \begin{bmatrix} \ldots 6 & -12 & 6 & 0 & 0 & 0 & 0 \\ \ldots 0 & 6 & -12 & 6 & 0 & 0 & 0 \\ \ldots 0 & 0 & 6 & -\frac{27}{2} & 6 & 0 & 0 \\ \ldots 0 & 0 & 0 & 9 & -\frac{468}{37} & 6 & 0 \\ \ldots 0 & 0 & 0 & 0 & \frac{144}{37} & -\frac{33}{4} & 6 \\ \ldots 0 & 0 & 0 & -9 & \frac{756}{37} & -\frac{45}{2} & 12 \end{bmatrix}.$$

Looking upon the modification described in Eq. (3.3) and if we consider the case $x \in [x_0, x_1]$, there are four linearly independent polynomials as in Figure 1(b) that are capable to produce polynomials $1, x, x^2, x^3$ in this interval. It is a remarkable resolution for the issue of the modification Eq. (3.2). Polynomial reproduction property and high accuracy order of the basis $\{\breve{B}_j \ : \ j = 0, \ldots, N\}$ guarantee better numerical accuracy and performance.

3.2. **Method to compute the weight coefficients for differential quadrature.** We first derive the weight coefficients corresponding to the first-order derivative representation and then connect it to find the coefficients of second-order derivatives. Differential quadrature representation of the first-order derivative using the modified cubic splines $\{\breve{B}_j(x) : j = 0, \ldots, N\}$ is given by

$$\breve{B}'_j(x_i) = \sum_{k=0}^{N} p_{ik}^{(1)} \breve{B}_j(x_k), \quad i, j = 0, \ldots, N, \tag{3.5}$$

where $p_{ik}^{(1)},\ i,k=0,1,\ldots,N$ are the weight coefficients associated with the representation of the first-order derivatives. Equivalently, this can be written as a tridiagonal matrix system by Table 1 as follows:

$$\begin{bmatrix} 8 & 1 & & & & & & & & \\ & \frac{37}{8} & 1 & & & & & & & \\ & & \frac{144}{37} & 1 & & & & & & \\ & & & \frac{15}{4} & 1 & & & & & \\ & & & 1 & 4 & 1 & & & & \\ & & & & \ddots & \ddots & \ddots & & & \\ & & & & & 1 & 4 & 1 & & \\ & & & & & & 1 & \frac{15}{4} & & \\ & & & & & & & 1 & \frac{144}{37} & \\ & & & & & & & & 1 & \frac{37}{8} \\ & & & & & & & & & 1 & 8 \end{bmatrix} \begin{bmatrix} p_{i0}^{(1)} \\ p_{i1}^{(1)} \\ p_{i2}^{(1)} \\ p_{i3}^{(1)} \\ \vdots \\ p_{iN-3}^{(1)} \\ p_{iN-2}^{(1)} \\ p_{iN-1}^{(1)} \\ p_{iN}^{(1)} \end{bmatrix} = \begin{bmatrix} \breve{B}_0'(x_i) \\ \breve{B}_1'(x_i) \\ \breve{B}_2'(x_i) \\ \breve{B}_3'(x_i) \\ \breve{B}_4'(x_i) \\ \vdots \\ \breve{B}_{N-3}'(x_i) \\ \breve{B}_{N-2}'(x_i) \\ \breve{B}_{N-1}'(x_i) \\ \breve{B}_N'(x_i) \end{bmatrix},$$

for each $i=0,\ldots,N$. Let A denote the above $N+1$ order tridiagonal matrix. Then the above system can be represented in the matrix equation form as $AP^T=\breve{B}$, where the matrix P is given by

$$P=\begin{bmatrix} p_{00}^{(1)} & \cdots & p_{0N}^{(1)} \\ \vdots & \ddots & \vdots \\ p_{N0}^{(1)} & \cdots & p_{NN}^{(1)} \end{bmatrix},$$

and

$$\breve{B}=\begin{bmatrix} \breve{B}_0'(x_0) & \cdots & \breve{B}_0'(x_M) \\ \vdots & \ddots & \vdots \\ \breve{B}_N'(x_0) & \cdots & \breve{B}_N'(x_N) \end{bmatrix} = \frac{1}{h}\begin{bmatrix} -12 & -3 & & & & & & & & \\ \frac{27}{2} & -\frac{15}{8} & -3 & & & & & & & \\ -\frac{276}{37} & \frac{174}{37} & \frac{12}{37} & -3 & & & & & & \\ 3 & -\frac{3}{2} & 3 & \frac{3}{4} & -3 & & & & & \\ & & & 3 & 0 & -3 & & & & \\ & & & & \ddots & \ddots & \ddots & & & \\ & & & & & 3 & 0 & -3 & & \\ & & & & & & 3 & -\frac{3}{4} & -3 & \frac{3}{2} & -3 \\ & & & & & & & 3 & -\frac{12}{37} & -\frac{174}{37} & \frac{276}{37} \\ & & & & & & & & 3 & \frac{15}{8} & -\frac{27}{2} \\ & & & & & & & & & 3 & 12 \end{bmatrix}.$$

Since the tridiagonal coefficient matrix A is invertible, the matrix P is obtained by the matrix relation: $P^T=A^{-1}\breve{B}$. Weight coefficients corresponding to the second-order derivatives can be computed using the following recurrence relation as described in [38]:

$$p_{ij}^{(2)}=2\left[p_{ij}^{(1)}p_{ii}^{(1)}-\frac{p_{ij}^{(1)}}{x_i-x_j}\right],\quad i\neq j,\qquad p_{ii}^{(2)}=-\sum_{\substack{j=0\\ j\neq i}}^{N}p_{ij}^{(2)},\quad \text{for each } i,j=0,\ldots N, \tag{3.6}$$

where $p_{ij}^{(2)},\ i,j=0,1,\ldots,N$ are the weight coefficients associated with the representation of the second-order derivatives.

4. Stability analysis

In this section, we study the stability of our proposed scheme. This analysis is done with a focus on the eigenvalues of the discretization matrix obtained from the system of first-order ordinary differential equations Eq. (2.3). If all its eigenvalues have negative real parts, then we say that the discretized system is stable. Note that our proposed scheme

depends solely on this discretized system matrix. Similar approach can be carried out to handle the stability analysis both in one- and two-dimensions. For convenience we can express the system Eq. (2.3) in block matrix form with the boundary points excluded as follows:

$$\frac{d}{dt}\begin{bmatrix} w \\ u \end{bmatrix} = \begin{bmatrix} -2\alpha I & C \\ I & O \end{bmatrix}\begin{bmatrix} w \\ u \end{bmatrix} + \begin{bmatrix} F \\ \tilde{O} \end{bmatrix}, \tag{4.1}$$

where I and O are the identity and null matrices of size $N-1$ each, and

$$C = \begin{bmatrix} p_{11}^{(2)} - \beta^2 & p_{12}^{(2)} & \cdots & p_{1\,N-1}^{(2)} \\ p_{21}^{(2)} & p_{22}^{(2)} - \beta^2 & \cdots & p_{2\,N-1}^{(2)} \\ \vdots & & \ddots & \vdots \\ p_{N-1\,1}^{(2)} & p_{N-1\,2}^{(2)} & \cdots & p_{N-1\,N-1}^{(2)} - \beta^2 \end{bmatrix}.$$

Solution of Eq. (4.1) can be treated as a column vector,

$$\begin{bmatrix} w & u \end{bmatrix}^T = \begin{bmatrix} w_1 & w_2 & \ldots & w_{N-1} & u_1 & u_2 & \ldots & u_{N-1} \end{bmatrix}.$$

The column vector $\begin{bmatrix} \tilde{O} & F \end{bmatrix}^T$ in Eq. (4.1) includes the non-homogenous part and boundary conditions shown in Eq. (2.4), where $\tilde{O}$ is the zero column vector of length $N-1$. The removal of boundary grid points changes the entries of the column vector $F^T = \begin{bmatrix} f_1 & f_2 & \ldots & f_{N-1} \end{bmatrix}$, where

$$f_i = -\left(p_{i\,0}u_0 + p_{i\,N}u_N\right) + f(x_i, t), \quad i = 1, \ldots, N-1.$$

To establish the stability of the discretization matrix, we use the following well-known result:

Theorem 4.1. *[29] Consider real $n \times n$ matrices S and T and the $2n \times 2n$ block matrix $\begin{bmatrix} S & T \\ I & O \end{bmatrix}$ and assume that $S_{ii} = a < 0$, $T_{ii} < 0$ for all i and $S_{ij} = 0$ for all $i \neq j$. If T is stable (ie., all eigenvalues of T have negative real parts) and in addition $|a| > |\gamma|/\sqrt{-\rho}$ for all eigenvalues $\lambda = \rho + \gamma i$ of T, then the above block matrix is stable.*

Remark 4.2. *If T has no complex roots then the inequality follows trivially.*

Remark 4.3. *Note that A is centrosymmetric and $\breve{B}$ is skew-centrosymmetric, $P^{(1)}$ will be a skew-centrosymmetric matrix. The recurrence relation Eq. (3.6) guarantees $P^{(2)}$ a centrosymmetric matrix with negative diagonal entries. Hence C is also centrosymmetric.*

Remark 4.4. *The characteristic polynomial of C satisfies the Hurwitz matrix criterion for different discretizations and hence all its eigenvalues have a negative real part. This ensures the stability of matrix C.*

Since $\alpha > 0$, note that the scalar matrix $-2\alpha I$ has all the diagonal entries negative. For different discretizations N, we get $C_{ii} < 0$. However, all the eigenvalues of C are found to be negative real numbers for different discretizations (see Figure 2). Thus the scheme will be stable by the above theorem.

5. Results and Discussions

Using the proposed cubic B-spline-based differential quadrature method in the previous sections, we now numerically solve one-dimensional and two-dimensional hyperbolic telegraph equations Eq. (1.1) and Eq. (1.2) for different choices of the parameters α, β up to the required time bounds. From the initial numerical solution, a solution at the desired time step is naturally achieved by solving the obtained system recursively with a time step Δt. The computations in all our numerical examples are done on uniform meshes. The proposed numerical scheme is applied to solve several one-dimensional and two-dimensional hyperbolic telegraph equations with different spatial discretizations and time steps. The effectiveness of our proposed method is demonstrated by comparing our proposed scheme with several other well-known numerical methods in the literature. We analyze the accuracy of the approximated solution with the exact solution of these problems in terms of L_2, L_∞ norms as well as relative and root mean square errors.

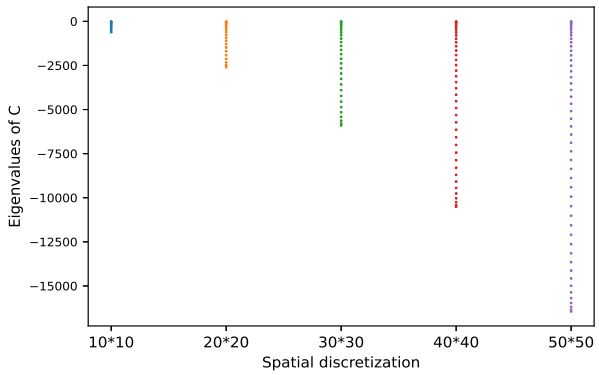
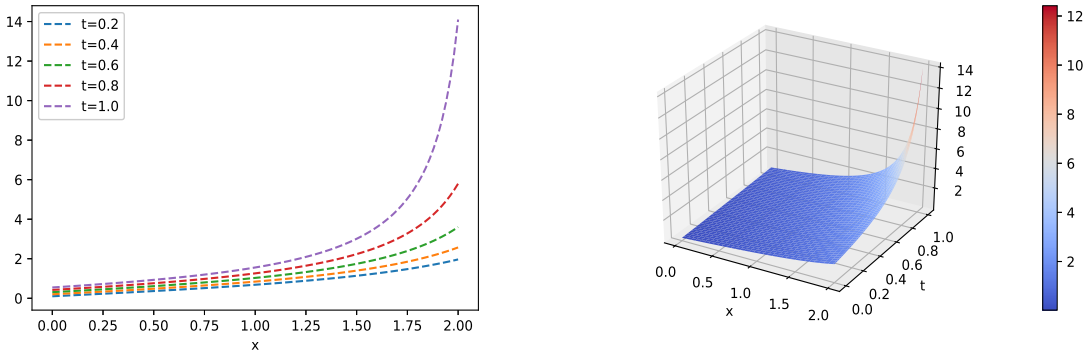


FIGURE 2. Eigenvalues of the matrix C for different spatial discretizations.

Example 5.1. Consider the one-dimensional telegraph equation Eq. (1.1) for $\alpha = 10$, $\beta = 5$ and $\Omega = (0, 2)$ with the initial and boundary conditions:

$$u(x,0) = \tan\left(\frac{x}{2}\right),\ u_t(x,0) = \frac{1}{2}\left(1 + \tan^2\left(\frac{x}{2}\right)\right),\ u(0,t) = \tan\left(\frac{t}{2}\right),\ u(2,t) = \tan\left(\frac{2+t}{2}\right),$$

respectively. The source term is $f(x,t) = \alpha(1 + \tan^2(\frac{x+t}{2})) + \beta^2 \tan(\frac{x+t}{2})$. The exact solution of this problem is $u(x,t) = \tan(\frac{x+t}{2})$. The computation is done with $N = 100$. The computed numerical solution is plotted by taking $\Delta t = 0.001$ for $t = 0.2, 0.4, 0.6, 0.8, 1.0$ s and the surface plot is also given up to $t = 1$ s in Figure 3. The L_2 and L_∞ errors are calculated and compared with results in [25, 30, 37] in Table 2. The tabulated data show that the computed numerical solutions with the proposed modified basis functions are much more accurate than these works. The convergence rate is calculated in Table 3, showing that the scheme has an overall fourth-order convergence.



(A) (B)

FIGURE 3. Example 5.1. (a) is for computed solutions with $N = 100$ and $\Delta t = 0.001$. (b) is for the surface plot.

TABLE 2. Example 5.1: L_2 and L_∞ errors comparison for $N = 100$ and for various Δt; total CPU time = 0.4 s.

Methods	t	L_2	L_∞	L_2	L_∞
		$\Delta t = 0.01$		$\Delta t = 0.001$	
	0.2	1.0975×10^{-7}	5.0027×10^{-7}	1.1462×10^{-7}	5.1120×10^{-7}
	0.4	3.4507×10^{-7}	1.5838×10^{-6}	3.5544×10^{-7}	1.6045×10^{-6}
Proposed	0.6	1.4459×10^{-6}	6.9892×10^{-6}	1.4699×10^{-6}	7.0288×10^{-6}
	0.8	1.1002×10^{-5}	5.7991×10^{-5}	1.1060×10^{-5}	5.7949×10^{-5}
	1.0	4.0339×10^{-4}	2.4742×10^{-3}	4.0174×10^{-4}	2.4582×10^{-3}
		$\Delta t = 0.0001$		$\Delta t = 0.001$	
	0.2	5.0305×10^{-5}	3.4797×10^{-5}	1.8782×10^{-4}	2.6332×10^{-4}
	0.4	9.5163×10^{-5}	5.3456×10^{-5}	4.8888×10^{-4}	6.9997×10^{-4}
[25]	0.6	2.2041×10^{-4}	9.4725×10^{-5}	9.4864×10^{-4}	1.4860×10^{-3}
	0.8	7.8267×10^{-4}	1.8792×10^{-4}	1.8743×10^{-3}	3.4057×10^{-3}
	1.0	7.9277×10^{-3}	5.8720×10^{-4}	5.0981×10^{-3}	1.1148×10^{-2}
		$\Delta t = 0.0001$		$\Delta t = 0.001$	
	0.2	2.13×10^{-5}	3.35×10^{-5}	1.85×10^{-4}	2.79×10^{-4}
	0.4	5.25×10^{-5}	7.92×10^{-5}	4.43×10^{-4}	6.35×10^{-4}
[30]	0.6	1.02×10^{-4}	1.63×10^{-4}	7.96×10^{-4}	1.17×10^{-3}
	0.8	2.39×10^{-4}	4.65×10^{-4}	1.47×10^{-3}	2.40×10^{-3}
	1.0	1.73×10^{-3}	5.59×10^{-4}	4.19×10^{-3}	9.51×10^{-3}
		$\Delta t = 0.001$			
[37]	0.2	9.99×10^{-6}	6.83×10^{-5}		
	0.4	7.07×10^{-6}	4.28×10^{-5}		

TABLE 3. Example 5.1: Convergence rate of the problem for with $\Delta t = 0.001$ at $t = 1$ s.

N	L_∞ error	CPU time (sec)	Convergence rate
2^7	1.19×10^{-3}	0.41	-
2^8	1.24×10^{-4}	0.52	3.26
2^9	1.02×10^{-5}	1.00	3.60
2^{10}	7.57×10^{-7}	4.41	3.75

Example 5.2. Consider the one-dimensional telegraph equation Eq. (1.1) for $\alpha = 10$, $\beta = 5$ and $\Omega = (0, 1)$ with the initial and boundary conditions:

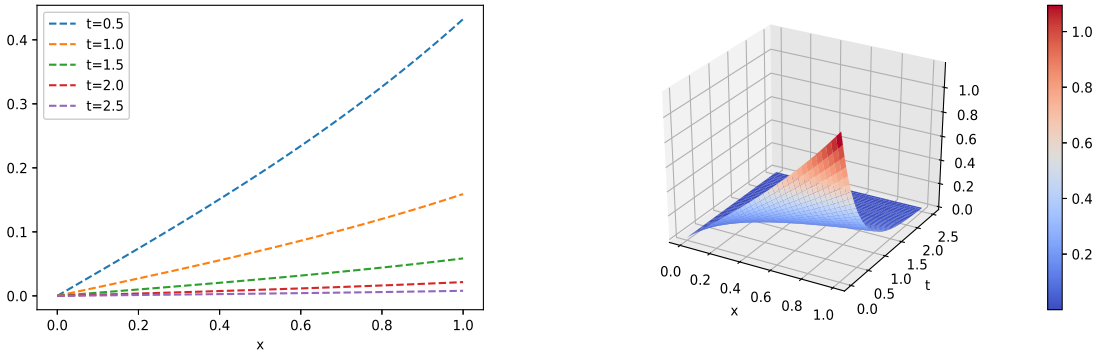
$$u(x,0) = \sinh(x), \quad u_t(x,0) = -2\sinh(x) \quad \text{and} \quad u(0,t) = 0, \quad u(1,t) = e^{-2t}\sinh(1),$$

respectively. The source term is $f(x,t) = e^{-2t}\sinh(x)\left(3 - 4\alpha + \beta^2\right)$. The exact solution of this problem is $u(x,t) = e^{-2t}\sinh(x)$. The numerical approximation using our proposed method yields much better accuracy than those in [14]. This can be observed from the error comparison in Table 4 and the plots in Figure 4. The convergence rate of the scheme has been discussed in Table 5. The results here show much better performance of the proposed scheme in terms of accuracy of the solution and its computational time.

Example 5.3. Consider the one-dimensional telegraph equation Eq. (1.1) for $\alpha = 6$, $\beta = 2$ and $\Omega = (0, 1)$ with the initial and boundary conditions:

$$u(x,0) = \sin(x), \quad u_t(x,0) = 0 \quad \text{and} \quad u(0,t) = 0, \quad u(1,t) = \cos(t)\sin(1),$$

respectively. The source term is $f(x,t) = -2\alpha\sin(t)\sin(x) + \beta^2\cos(t)\sin(x)$. The exact solution of this problem is $u(x,t) = \cos(t)\sin(x)$. The computation is done by taking $N = 100$ here. Computed numerical solutions are plotted for various time up to $t = 2$ s shown in Figure 5. The L_2 and L_∞ errors up to time $t = 5s$ are calculated and compared



(A) (B)

FIGURE 4. Example 5.2. (a) is for computed solutions with $N = 33$ and $\Delta t = 0.001$. (b) is for the surface plot.

TABLE 4. Example 5.2: L_2, L_∞ and RMS errors comparison for $N = 51$, $\Delta t = 0.01$; total CPU time $= 0.06$ s.

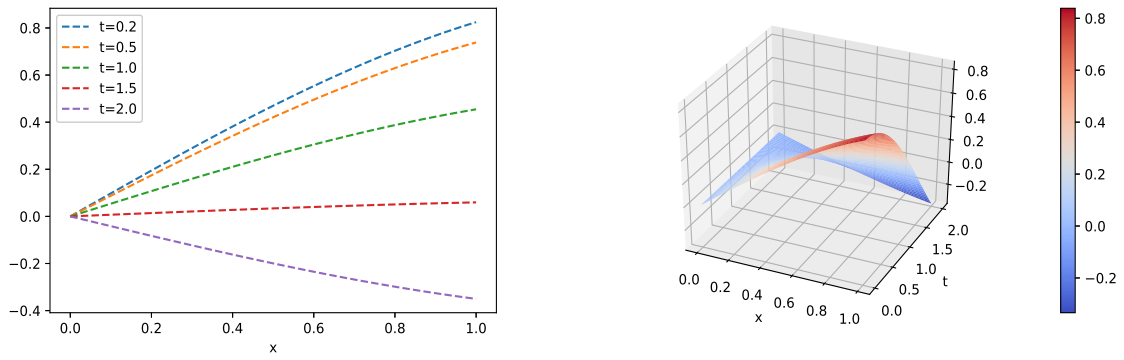
t	Method	L_2	L_∞	RMS
1	Proposed Method	1.20×10^{-9}	2.49×10^{-9}	1.19×10^{-9}
	[14]	8.35×10^{-5}	1.70×10^{-5}	1.19×10^{-5}
2	Proposed Method	2.47×10^{-10}	4.03×10^{-10}	2.45×10^{-10}
	[14]	2.37×10^{-5}	4.63×10^{-6}	3.38×10^{-6}

TABLE 5. Example 5.2: Convergence rate of the problem for with $\Delta t = 0.001$ at $t = 1$ s.

N	L_∞ error	CPU time (sec)	Convergence rate
2^3	2.00×10^{-6}	0.22	-
2^4	1.49×10^{-7}	0.23	3.74
2^5	1.00×10^{-8}	0.26	3.89
2^6	7.35×10^{-10}	0.30	3.76

with three known methods in [25, 30, 37], tabulated in Table 6 and Table 7. Comparison of root-mean-square error can be found in Table 6 and the convergence rate is tabulated in Table 8.

Example 5.4. Consider the one-dimensional telegraph equation Eq. (1.1) for $\alpha = \frac{1}{2}$, $\beta = 1$ and $\Omega = (0, 1)$ with the initial conditions $u(x, 0) = 0 = u_t(x, 0)$ and boundary conditions $u(0, t) = 0 = u(1, t)$. The source term is $f(x, t) = e^{-t}(2t^2 + (x - x^2)(t^2 - 2t + 2))$. The exact solution of this problem is $u(x, t) = e^{-t}(x - x^2)t^2$. We compute the numerical solution by using $N = 100$ grid points and time step $\Delta t = 0.001$. Figure 6 illustrates the behaviour of the solution we obtained as time progresses from $t = 1$ s to $t = 10$ s. The L_2, L_∞ and RMS errors are tabulated in Table 9 and Table 10 and compared with known results reported [10, 14, 25, 30, 37]. The polynomial in the exact solution of this problem is effectively captured by our modified basis functions near the boundary and therefore, yields significantly better numerical performance.



(A) (B)

FIGURE 5. Example 5.3. (a) is for computed solutions with $N = 100$ and $\Delta t = 0.001$. (b) is for the surface plot.

TABLE 6. Example 5.3: L_2, L_∞ and RMS errors comparison for $N = 100, \Delta t = 0.0001$ and at various times; total CPU time = 2.48 s.

t	L_2	L_∞	RMS	L_2	L_∞	RMS
	Proposed Method			[25]		
0.2	1.7141×10^{-10}	5.5379×10^{-10}	1.7056×10^{-10}	2.6903×10^{-6}	5.2412×10^{-6}	2.6775×10^{-6}
0.4	1.9603×10^{-10}	5.3219×10^{-10}	1.9505×10^{-10}	5.6183×10^{-6}	8.6095×10^{-6}	5.5904×10^{-6}
0.6	2.0429×10^{-10}	4.9235×10^{-10}	2.0328×10^{-10}	9.7545×10^{-6}	1.2529×10^{-5}	9.7061×10^{-6}
0.8	2.0231×10^{-10}	4.3383×10^{-10}	2.0131×10^{-10}	1.3774×10^{-5}	2.0274×10^{-5}	1.3706×10^{-5}
1.0	1.9219×10^{-10}	3.5991×10^{-10}	1.9123×10^{-10}	1.7347×10^{-5}	2.7555×10^{-5}	1.7261×10^{-5}
	[30]			[37]		
0.2	2.96×10^{-6}	4.63×10^{-6}	2.94×10^{-6}	2.33×10^{-8}	2.10×10^{-7}	1.49×10^{-8}
0.4	6.77×10^{-6}	1.01×10^{-5}	6.73×10^{-6}	4.55×10^{-8}	4.30×10^{-7}	3.04×10^{-8}

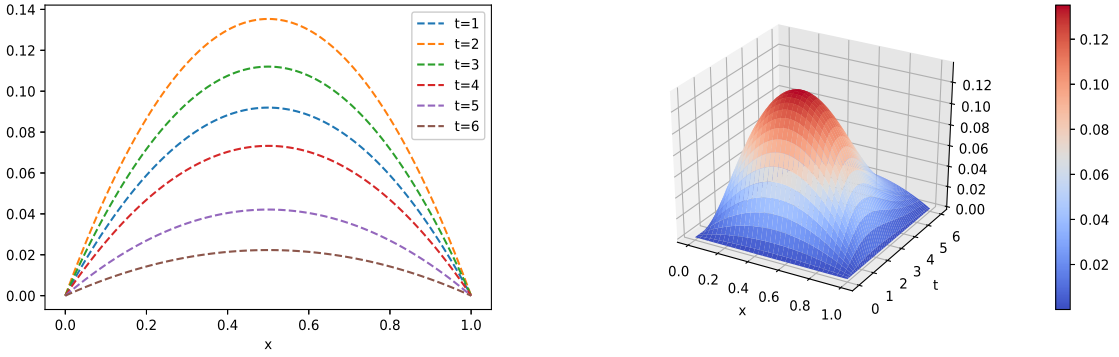
TABLE 7. Example 5.3: L_2 and L_∞ errors comparison for various N and $\Delta t = 0.001$; total CPU time = 0.8 s.

Methods	t	N=100		N=200	
		L_2	L_∞	L_2	L_∞
Proposed	0.2	1.7133×10^{-10}	5.5379×10^{-10}	1.1580×10^{-11}	3.7266×10^{-11}
	0.4	1.9599×10^{-10}	5.3220×10^{-10}	1.5794×10^{-11}	3.9851×10^{-11}
	0.6	2.0436×10^{-10}	4.9239×10^{-10}	2.0695×10^{-11}	4.3584×10^{-11}
	0.8	2.0254×10^{-10}	4.3389×10^{-10}	2.6223×10^{-11}	4.8410×10^{-11}
	1.0	1.9261×10^{-10}	3.6000×10^{-10}	3.2111×10^{-11}	5.4166×10^{-11}
[25]	0.2	3.6711×10^{-5}	7.9139×10^{-5}	3.4308×10^{-5}	6.8272×10^{-5}
	0.4	8.8989×10^{-5}	1.5967×10^{-4}	8.5756×10^{-5}	1.4935×10^{-4}
	0.6	1.3733×10^{-4}	2.3362×10^{-4}	1.3367×10^{-4}	2.2410×10^{-4}
	0.8	1.7913×10^{-4}	2.9820×10^{-4}	1.7532×10^{-4}	2.8978×10^{-4}
	1.0	2.1302×10^{-4}	3.5085×10^{-4}	2.0929×10^{-4}	3.4386×10^{-4}

Example 5.5. Consider the one-dimensional telegraph equation Eq. (1.1) for $\alpha = \beta = 1$ and $\Omega = (0, 1)$ with the initial conditions $u(x, 0) = x^2 = u_t(x, 0)$ and boundary conditions $u(0, t) = 0$ and $u(1, t) = e^t$. The source term is

TABLE 8. Example 5.3: Convergence rate of the problem for with $\Delta t = 0.001$ at $t = 1$ s.

N	L_∞ error	CPU time (sec)	Convergence rate
2^3	6.58×10^{-6}	0.15	-
2^4	4.58×10^{-7}	0.17	3.84
2^5	3.01×10^{-8}	0.20	3.92
2^6	1.96×10^{-9}	0.26	3.94
2^7	1.55×10^{-10}	0.35	3.66



(A) (B)

FIGURE 6. 5.4. (a) is for computed solutions with $N = 100$ and $\Delta t = 0.001$. (b) is for the surface plot.

TABLE 9. Example 5.4: L_2 and L_∞ errors comparison for $N = 100$, $\Delta t = 0.001$ and at various times; total CPU time = 1.12 s.

t	Proposed Method		[25]		[10]	
	L_2	L_∞	L_2	L_∞	L_2	L_∞
1	3.8017×10^{-12}	5.3167×10^{-12}	4.5526×10^{-5}	5.9153×10^{-5}	1.4386×10^{-4}	1.8479×10^{-5}
2	3.1440×10^{-12}	4.4222×10^{-12}	1.4307×10^{-5}	1.7864×10^{-5}	8.0879×10^{-5}	1.0713×10^{-5}
3	4.8213×10^{-13}	6.8479×10^{-13}	6.4273×10^{-6}	1.4309×10^{-5}	1.2944×10^{-4}	1.8161×10^{-5}
4	1.1137×10^{-12}	1.5544×10^{-12}	8.9203×10^{-6}	1.3529×10^{-5}	1.1845×10^{-4}	1.6489×10^{-5}
5	1.4593×10^{-13}	2.1010×10^{-13}	3.0161×10^{-6}	5.2032×10^{-6}	7.5545×10^{-5}	1.0455×10^{-5}

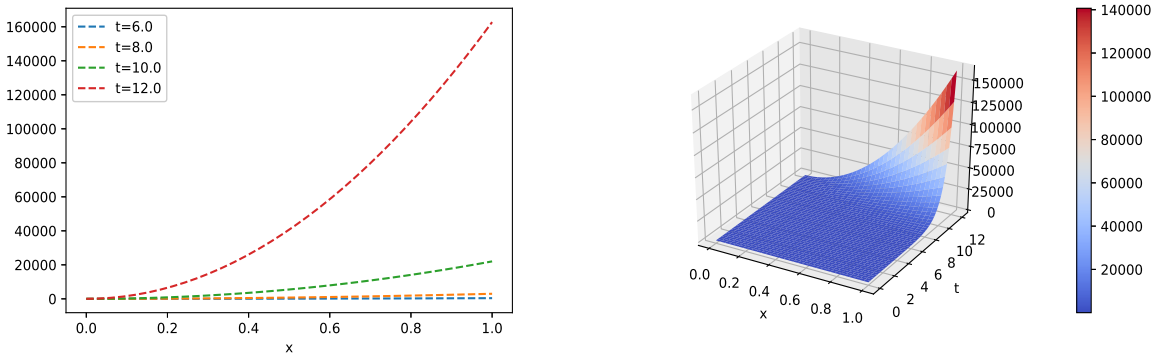
$f(x,t) = e^t(\beta^2 x^2 + 2\alpha x^2 + x^2 - 2)$. In this problem, the exact solution is $u(x,t) = x^2 e^t$, where the polynomial x^2 here is well captured by our proposed modified cubic B-splines and hence we obtain very accurate computed numerical solutions as shown in Table 11. L_2, L_∞ and RMS errors are calculated with $N = 32$ grid points for various Δt and the computed numerical solutions are drawn up to $t = 12$ s as in Figure 7.

Example 5.6. Consider the two-dimensional telegraph equation Eq. (1.2) with $\alpha = \beta = 1$, $\Omega = (0,1)^2$, and the source term $f(x,y,t) = 2(\cos(t) - \sin(t))\sin(x)\sin(y)$ such that the initial conditions and boundary conditions are given by

$$\text{IC}: \begin{cases} u(x,y,0) = \sin(x)\sin(y), \\ u_t(x,y,0) = 0, \end{cases} \qquad \text{BC}: \begin{cases} u(0,y,t) = 0, & u(1,y,t) = \cos(t)\sin(1)\sin(y), \\ u(x,0,t) = 0, & u(x,1,t) = \cos(t)\sin(x)\sin(1). \end{cases}$$

TABLE 10. Example 5.4: L_2 and L_∞ errors comparison for $N = 100$, $\Delta t = 0.005$ and for various time; total CPU time = 1.02 s.

t	Proposed Method			[14]		
	L_2	L_∞	RMS	L_2	L_∞	RMS
1	8.63×10^{-12}	1.03×10^{-11}	8.59×10^{-12}	7.54×10^{-6}	9.93×10^{-7}	7.58×10^{-7}
2	7.92×10^{-12}	8.84×10^{-12}	7.88×10^{-12}	1.57×10^{-5}	2.12×10^{-6}	2.58×10^{-6}
t	[30]			[37]		
	L_2	L_∞	RMS	L_2	L_∞	RMS
1	6.31×10^{-5}	8.76×10^{-5}	-	2.61×10^{-8}	1.67×10^{-7}	1.68×10^{-8}
2	2.34×10^{-5}	3.29×10^{-5}	-	6.10×10^{-8}	4.72×10^{-7}	4.74×10^{-8}



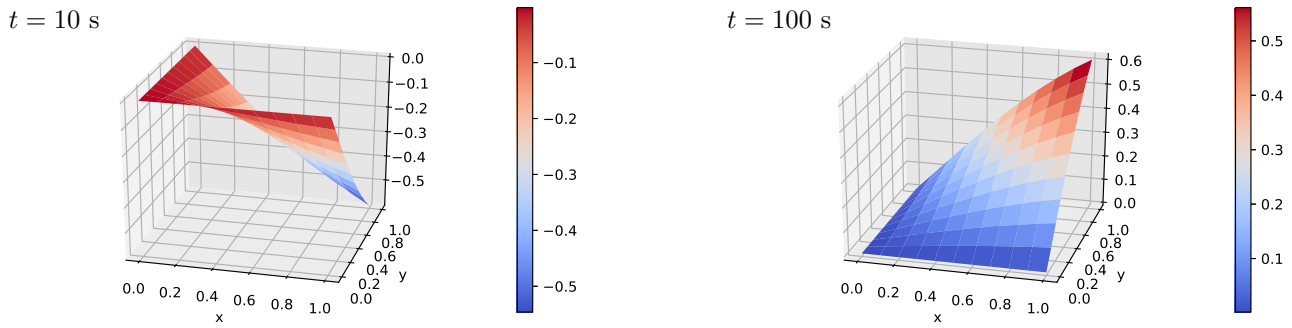
(A) (B)

FIGURE 7. Example 5.5. (a) is for computed solutions with $N = 33$ and $\Delta t = 0.01$. (b) is for the surface plot.

TABLE 11. Example 5.5: L_2, L_∞ and RMS errors comparison for various Δt and $N = 32$; total CPU time = 2.0 s.

t	$\Delta\ t = 0.01$			$\Delta\ t = 0.001$		
	L_2	L_∞	RMS	L_2	L_∞	RMS
2	5.2924×10^{-10}	7.9206×10^{-10}	5.2116×10^{-10}	3.0167×10^{-10}	5.6077×10^{-10}	2.9706×10^{-10}
4	3.9723×10^{-9}	5.9098×10^{-9}	3.9117×10^{-9}	2.5397×10^{-9}	4.7043×10^{-9}	2.5009×10^{-9}
6	2.9356×10^{-8}	4.3660×10^{-8}	2.8908×10^{-8}	1.9078×10^{-8}	3.5321×10^{-8}	1.8786×10^{-8}
8	2.1691×10^{-7}	3.2260×10^{-7}	2.1360×10^{-7}	1.4128×10^{-7}	2.6155×10^{-7}	1.3912×10^{-7}
10	1.6027×10^{-6}	2.3836×10^{-6}	1.5783×10^{-6}	1.0442×10^{-6}	1.9332×10^{-6}	1.0283×10^{-6}

The exact solution of this problem is $u(x, y, t) = \cos(t)\sin(x)\sin(y)$. Using the proposed scheme discussed in Section 2, we compute the numerical solution of this problem on a uniform mesh consisting of 100 points in each coordinate direction (i.e., $N = M = 10$) and then we progress the numerical solution up to $t = 5$ s using a time step $\Delta t = 0.01$. Figure 8 illustrates the computed numerical solution at time $t = 10, 100$ s. The L_2, L_∞ and relative errors are tabulated for different time steps $\Delta t = 0.01, 0.001$ and spatial discretizations $N = M = 10, 20$ when the numerical solution progresses up to time $t = 10$ s (see Table 12 and Table 13). In comparison with the results in [26], the proposed method is much more accurate than the method in [26]. The computed numerical solution seems to be very much close to the exact solution in terms of different norms.



(A) (B)

FIGURE 8. The computed numerical solutions in Example 5.6 using $N = M = 10$ and $\Delta t = 0.01$ at (a) $t = 10$ s, (b) $t = 100$ s.

TABLE 12. L_2, L_∞ and relative errors comparison in Example 5.6 for $\Delta t = 0.01$ and $N = M = 10$; total CPU time = 0.69 sec.

t	Proposed Method			[26]		
	L_2	L_∞	Relative error	L_2	L_∞	Relative error
1	1.0412×10^{-6}	2.2529×10^{-6}	6.8638×10^{-6}	9.9722×10^{-4}	2.2746×10^{-3}	5.9762×10^{-3}
2	4.7169×10^{-7}	1.5909×10^{-6}	4.0372×10^{-6}	1.0926×10^{-3}	2.8706×10^{-3}	8.5019×10^{-3}
3	1.4112×10^{-6}	3.8455×10^{-6}	5.0771×10^{-6}	2.2877×10^{-4}	6.0818×10^{-4}	7.4720×10^{-4}
5	3.0552×10^{-7}	1.0116×10^{-6}	3.8363×10^{-6}	1.1562×10^{-3}	2.9942×10^{-3}	1.2767×10^{-3}
7	1.1357×10^{-6}	3.0264×10^{-6}	5.3656×10^{-6}	7.2867×10^{-4}	1.8781×10^{-3}	3.1572×10^{-3}
10	1.2444×10^{-6}	3.3501×10^{-6}	5.2823×10^{-6}	5.8889×10^{-4}	1.5158×10^{-3}	2.2874×10^{-3}

TABLE 13. L_2, L_∞ and relative errors comparison in Example 5.6 for $\Delta t = 0.001$ and $N = M = 20$; total CPU time = 8.73 sec

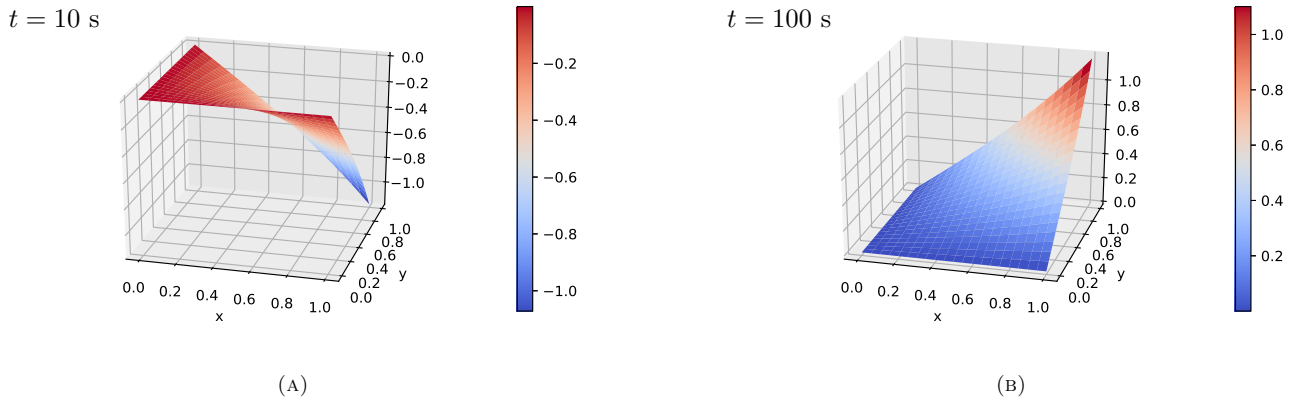
t	Proposed Method			[26]		
	L_2	L_∞	Relative error	L_2	L_∞	Relative error
1	7.1329×10^{-8}	1.5204×10^{-7}	4.7706×10^{-7}	9.8870×10^{-5}	2.4964×10^{-4}	6.2977×10^{-4}
2	3.6669×10^{-8}	1.1652×10^{-7}	3.1842×10^{-7}	1.2148×10^{-4}	3.2296×10^{-4}	1.0025×10^{-3}
3	1.0049×10^{-7}	2.7289×10^{-7}	3.6680×10^{-7}	3.7627×10^{-5}	9.9310×10^{-5}	1.3078×10^{-4}
5	2.3447×10^{-8}	7.6682×10^{-8}	2.9870×10^{-7}	1.2762×10^{-4}	3.3205×10^{-4}	1.5411×10^{-3}
7	7.9990×10^{-8}	2.0971×10^{-7}	3.8341×10^{-7}	6.7672×10^{-5}	1.7679×10^{-4}	3.0892×10^{-4}
10	8.7928×10^{-8}	2.3304×10^{-7}	3.7867×10^{-7}	5.1764×10^{-5}	1.3521×10^{-4}	2.1245×10^{-4}

Example 5.7. Consider the two-dimensional telegraph equation Eq. (1.2) with $\Omega = (0,1)^2$, the source term $f(x,y,t) = \sinh(x)\sinh(y)(-3\cos(t) - 2\alpha\sin(t) + \beta^2\cos(t))$ and the initial and boundary conditions:

$$\text{IC}: \begin{cases} u(x,y,0) = \sinh(x)\sinh(y), \\ u_t(x,y,0) = 0, \end{cases} \qquad \text{BC}: \begin{cases} u(0,y,t) = 0, \quad u(1,y,t) = \cos(t)\sinh(1)\sinh(y), \\ u(x,0,t) = 0, \quad u(x,1,t) = \cos(t)\sinh(x)\sinh(1). \end{cases}$$

The exact solution of this problem is $u(x,y,t) = \cos(t)\sinh(x)\sinh(y)$. The computation is done with a spatial discretization $N = M = 20$ and a time step $\Delta t = 0.001$. For $\alpha = 10, \beta = 5$, the computed numerical solution is

plotted at time $t = 10,100$s (see Figure 9). The numerical errors are computed up to $t = 10$ s for various values of α, β. Comparison of these results with [16, 26] is summarized in Table 14, showing that our proposed method performs much better than those in [16, 26] in the literature. Also, our algorithm takes less than 19 seconds on an Intel core i5 processor. Our proposed approach is capable to compute the solution up to a larger time with less computational cost.



(A) (B)

FIGURE 9. The computed numerical solutions in Example 5.7 for $\alpha = 10, \beta = 5, N = M = 20$ and $\Delta t = 0.001$ at (a) $t = 10$ s, (b) $t = 100$ s.

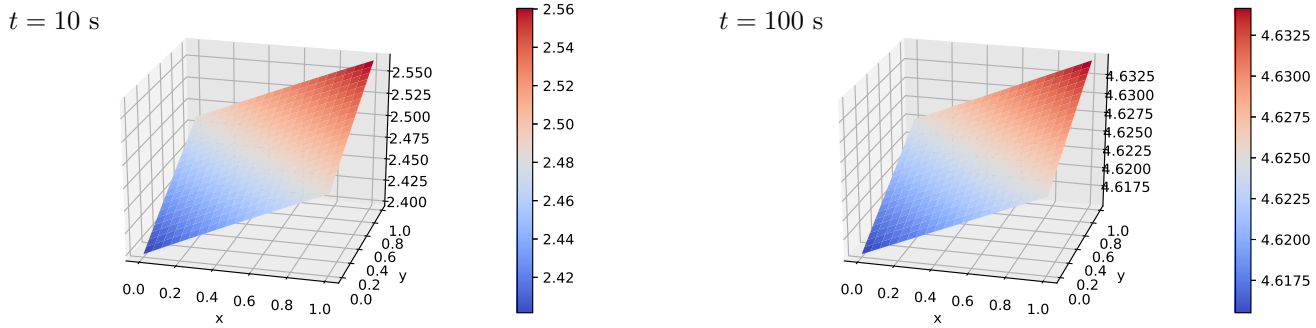
TABLE 14. L_2 and relative errors comparison in Example 5.7 for $\Delta t = 0.001$ and $N = M = 20$; total CPU time = 18.90 sec.

t	Proposed Method		[26]		[16]
	L_2	Relative error	L_2	Relative error	Relative error
$\alpha = 10,\ \beta = 5$					
0.5	9.0872×10^{-8}	2.4598×10^{-7}	1.0696×10^{-4}	1.3787×10^{-5}	2.0149×10^{-5}
1	6.9860×10^{-8}	3.0716×10^{-7}	1.7174×10^{-4}	3.5957×10^{-5}	4.6003×10^{-5}
2	2.8496×10^{-8}	1.6267×10^{-7}	1.6468×10^{-4}	4.4722×10^{-5}	4.4460×10^{-5}
3	9.8186×10^{-8}	2.3560×10^{-7}	8.9858×10^{-6}	1.0266×10^{-6}	6.4830×10^{-6}
5	1.7618×10^{-8}	1.4755×10^{-7}	1.7737×10^{-4}	7.0735×10^{-5}	7.3626×10^{-5}
7	8.9634×10^{-8}	2.8244×10^{-7}	1.4200×10^{-4}	2.1307×10^{-5}	-
10	9.5999×10^{-8}	2.7179×10^{-7}	1.2241×10^{-4}	1.6514×10^{-5}	2.4901×10^{-5}
$\alpha = 50,\ \beta = 5$					
0.5	5.8561×10^{-8}	1.5852×10^{-7}	9.8800×10^{-5}	1.2735×10^{-5}	2.8724×10^{-5}
1	5.8618×10^{-8}	2.5773×10^{-7}	1.6766×10^{-4}	3.5104×10^{-5}	7.0064×10^{-5}
2	2.0342×10^{-8}	1.1612×10^{-7}	1.7109×10^{-4}	4.6408×10^{-5}	6.7759×10^{-5}
3	6.6142×10^{-8}	1.5871×10^{-7}	1.7406×10^{-5}	1.9886×10^{-6}	1.3856×10^{-5}
5	2.9417×10^{-8}	2.4636×10^{-7}	1.8420×10^{-4}	7.3460×10^{-5}	1.2840×10^{-4}
7	7.6652×10^{-8}	2.4153×10^{-7}	1.3760×10^{-4}	2.0647×10^{-5}	-
10	7.8380×10^{-8}	2.2190×10^{-7}	1.1691×10^{-4}	1.5772×10^{-5}	4.4202×10^{-5}

Example 5.8. Consider the two-dimensional telegraph equation Eq. (1.2) with $\alpha = \beta = 1$, $\Omega = (0,1)^2$, the source term $f(x,y,t) = \frac{1}{(1+x+y+t)^2} + \frac{2\alpha}{1+x+y+t} + \beta^2 \log(1+x+y+t)$ and the initial and boundary conditions:

$$\text{IC}: \begin{cases} u(x,y,0) = \log(1+x+y), \\ u_t(x,y,0) = \frac{1}{1+x+y}, \end{cases} \qquad \text{BC}: \begin{cases} u(0,y,t) = \log(1+y+t), & u(1,y,t) = \log(2+y+t), \\ u(x,0,t) = \log(1+x+t), & u(x,1,t) = \log(2+x+t). \end{cases}$$

The exact solution of this problem is $u(x,y,t) = \log(1+x+y+t)$. This problem is solved on a uniform mesh with $N = M = 20$, and progresses with the time step $\Delta t = 0.001$. The computed numerical solutions at time $t = 10, 100$ s are illustrated in Figure 10. L_2, L_∞ and relative errors are computed for a time-bound $t = 10$ s and compared with the results in [11, 26] (see Table 15). It is interesting to observe that the numerical errors of our solutions is significantly reduced in this problem. This shows that the proposed method effectively computes the solution and can progress to a larger time.



(A) (B)

FIGURE 10. The computed numerical solutions in Example 5.8 for $\alpha = \beta = 1, N = M = 20$ and $\Delta t = 0.001$ at (a) $t = 10$ s, (b) $t = 100$ s.

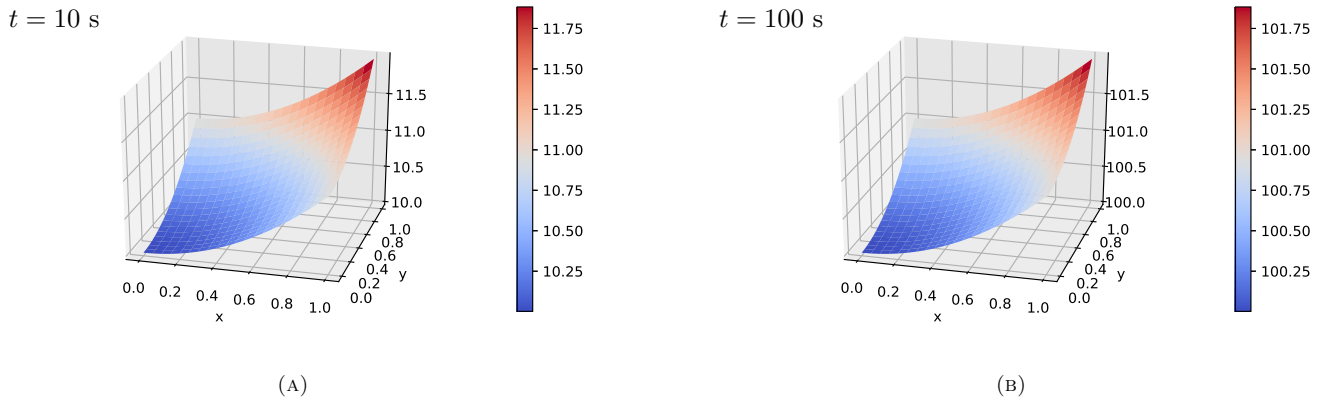
TABLE 15. L_2 and relative errors comparison in Example 5.8 for $\Delta t = 0.001$ and $N = M = 20$; total CPU time = 10.16 sec.

t	Proposed Method		[26]		[11]	
	L_2	RE	L_2	RE	RE(MLWS)	RE(MLPG)
1	9.1950×10^{-8}	8.3753×10^{-8}	1.5293×10^{-3}	1.3266×10^{-3}	9.098×10^{-5}	7.198×10^{-5}
2	4.4921×10^{-8}	3.2440×10^{-8}	4.6468×10^{-4}	3.1954×10^{-4}	8.705×10^{-4}	8.784×10^{-5}
3	1.1117×10^{-8}	6.9138×10^{-9}	2.1994×10^{-4}	1.3024×10^{-4}	9.931×10^{-4}	4.801×10^{-4}
4	2.6699×10^{-9}	1.3727×10^{-9}	2.7151×10^{-4}	1.4435×10^{-5}	4.703×10^{-3}	6.091×10^{-4}
5	2.5072×10^{-10}	1.1414×10^{-10}	1.7201×10^{-4}	8.4225×10^{-5}	7.302×10^{-3}	9.498×10^{-4}
10	1.1125×10^{-10}	4.4780×10^{-11}	7.7285×10^{-5}	2.9624×10^{-5}	-	-

Example 5.9. Consider the two-dimensional telegraph equation Eq. (1.2) with $\alpha = \beta = 1$, $\Omega = (0,1)^2$, the source term $f(x,y,t) = x^2 + y^2 + t - 2$, and the initial and boundary conditions:

$$\text{IC}: \begin{cases} u(x,y,0) = x^2 + y^2, \\ u_t(x,y,0) = 1, \end{cases} \qquad \text{BC}: \begin{cases} u(0,y,t) = y^2 + t, & u(1,y,t) = 1 + y^2 + t, \\ u(x,0,t) = x^2 + t, & u(x,1,t) = 1 + x^2 + t. \end{cases}$$

The exact solution is $u(x, y, t) = x^2 + y^2 + t$. This problem progresses for a time step $\Delta t = 0.01$. The computed numerical solution at time $t = 10, 100$ is illustrated in Figure 11 for a spatial discretization $N = M = 20$. L_∞ is calculated for $N = M = 10$ up to $t = 10$ s and compared with the results in [13] (see Table 16). The computational time shows the efficiency of the algorithm. It effectively captures the solution of the problem up to a large time-bound with the use of large time step Δt.



(A) (B)

FIGURE 11. The computed numerical solutions in Example 5.9 for $\alpha = \beta = 1, N = M = 20$ and $\Delta t = 0.01$ at (a) $t = 10$, (b) $t = 100$.

TABLE 16. Example 5.9 L_∞ error comparison for $\Delta t = 0.01$ and $N = M = 10$.

t	Proposed		[13]	
	L_∞	CPU time (s)	L_∞	CPU time (s)
1	2.1316×10^{-14}	0.03	1.2517×10^{-12}	0.27
2	4.2632×10^{-14}	0.05	2.2538×10^{-12}	0.29
3	6.3948×10^{-14}	0.07	2.0494×10^{-11}	0.31
5	1.0658×10^{-13}	0.12	1.3824×10^{-11}	0.35
10	2.1316×10^{-13}	0.21	4.7326×10^{-11}	0.47

6. CONCLUSION

In this article, we numerically solve the one- and two-dimensional hyperbolic telegraph equations using a modified cubic B-splines-based differential quadrature algorithm. Weight coefficients are determined using a newly modified set of cubic B-spline basis functions. This set of splines is capable to maintain the optimal polynomial reproduction property in the spatial domain and to ensure more accuracy near the boundaries. Also, the proposed modification of the cubic B-splines retains the tridiagonal matrix property, making the computation process effective. The convergence rate of the scheme is also discussed, and it is found that the algorithm possesses the fourth-order convergence. Thus the proposed method could achieve a better approximation than other known published methods in the literature. The spatial derivatives are approximated through the differential quadrature technique. The discretized telegraph equation is transformed into a system of first-order ordinary differential equations and is then solved by the SSPRK$-54, 43$ schemes. Eigenvalues of the discretization matrix are found to lie on the left half-plane and this guarantees the stability of the proposed method. Several test examples are solved by the proposed method. A comparative study is performed against certain published methods in the literature, especially in [10, 11, 13, 14, 16, 25, 26, 30, 37]. It is observed that the proposed scheme is efficient and computationally less complex.

Acknowledgment

Research of Bin Han supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada under grant RGPIN-2019-04276. Research of Noufal Asharaf supported under grant PL(UGC)1/SPG/SMNRI/2018-19 by Cochin University of Science and Technology, and Athira Babu under the Senior Research Fellowship of University Grants Commission.

References

[1] S. Abbasbandy, H. R. Ghehsareh, I. Hashim, and A. Alsaedi, *A comparison study of meshfree techniques for solving the two-dimensional linear hyperbolic telegraph equation*, Engineering Analysis with Boundary Elements, *47* (2014), 10-20.

[2] I. Ahmad, H. Ahmad, A. E. Abouelregal, P. Thounthong, and M. Abdel-Aty, *Numerical study of integer-order hyperbolic telegraph model arising in physical and related sciences*, The European Physical Journal Plus, *135*(9) (2020), 1-14.

[3] A. S. Alshomrani, S. Pandit, A. K. Alzahrani, M. S. Alghamdi, and R. Jiwari, *A numerical algorithm based on modified cubic trigonometric b-spline functions for computational modelling of hyperbolic-type wave equations*, Engineering Computations, 2017.

[4] G. Arora and V. Joshi, *Comparison of numerical solution of 1d hyperbolic telegraph equation using b-spline and trigonometric b-spline by differential quadrature method*, Indian Journal of Science and Technology, *9*(45) (2016), 1-8.

[5] M. Aslefallah and D. Rostamy, *Application of the singular boundary method to the two-dimensional telegraph equation on arbitrary domains*, Journal of Engineering Mathematics, *118*(1) (2019), 1-14.

[6] A. Babu, B. Han, and N. Asharaf, *Numerical solution of the viscous burgers' equation using localized differential quadrature method*, Partial Differential Equations in Applied Mathematics, 2021, 100044.

[7] J. Banasiak and J. R. Mika, *Singularly perturbed telegraph equations with applications in the random walk theory*, Journal of Applied Mathematics and Stochastic Analysis, *11*(1) (1998), 9-28.

[8] K. E. Biçer and S. Yalçinbaş, *Numerical solution of telegraph equation using bernoulli collocation method*, Proceedings of the National Academy of Sciences, India Section A: Physical Sciences, *89*(4) (2019), 769-775.

[9] G. Böhme, *Non-Newtonian fluid mechanics*, Elsevier, 2012.

[10] M. Dehghan and A. Shokri, *A numerical method for solving the hyperbolic telegraph equation*, Numerical Methods for Partial Differential Equations: An International Journal, *24*(4) (2008), 1080-1093.

[11] M. Dehghan and A. Ghesmati, *Combination of meshless local weak and strong (mlws) forms to solve the two dimensional hyperbolic telegraph equation*, Engineering Analysis with Boundary Elements, *34*(4) (2010), 324-336.

[12] R. M. Hafez, *Numerical solution of linear and nonlinear hyperbolic telegraph type equations with variable coefficients using shifted jacobi collocation method*, Computational and Applied Mathematics, *37*(4) (2018), 5253-5273.

[13] E. Hesameddini and E. Asadolahifard, *A new spectral galerkin method for solving the two dimensional hyperbolic telegraph equation*, Computers & Mathematics with Applications, *72*(7) (2016), 1926-1942.

[14] Z. Hong, Y. Wang, and H. Hao, *Adaptive monte carlo methods for solving hyperbolic telegraph equation*, Journal of Computational and Applied Mathematics, *345* (2019), 405-415.

[15] R. Jiwari, S. Pandit, and R. Mittal, *A differential quadrature algorithm for the numerical solution of the second-order one dimensional hyperbolic telegraph equation*, International Journal of Nonlinear Science, *13*(3) (2012), 259-266.

[16] R. Jiwari, S. Pandit, and R. Mittal, *A differential quadrature algorithm to solve the two dimensional linear hyperbolic telegraph equation with dirichlet and neumann boundary conditions*, Applied Mathematics and Computation, *218*(13) (2012), 7279-7294.

[17] R. Jiwari, *Lagrange interpolation and modified cubic b-spline differential quadrature methods for solving hyperbolic partial differential equations with dirichlet and neumann boundary conditions*, Computer Physics Communications, *193* (2015), 55-65.

[18] R. Jiwari, *Barycentric rational interpolation and local radial basis functions based numerical algorithms for multi-dimensional sine-gordon equation*, Numerical Methods for Partial Differential Equations, *37*(3) (2021), 1965-1992.

[19] P. Jordan and A. Puri, *Digital signal propagation in dispersive media*, Journal of Applied Physics, *85*(3) (1999), 1273-1282.

[20] M. Lakestani and B. N. Saray, *Numerical solution of telegraph equation using interpolating scaling functions*, Computers & Mathematics with Applications, *60*(7) (2010), 1964-1972.

[21] J. Lin, F. Chen, Y. Zhang, and J. Lu, *An accurate meshless collocation technique for solving two-dimensional hyperbolic telegraph equations in arbitrary domains*, Engineering Analysis with Boundary Elements, *108* (2019), 372-384.

[22] R. Mittal and R. Jain, *Numerical solutions of nonlinear burgers' equation with modified cubic b-splines collocation method*, Applied Mathematics and Computation, *218*(15) (2012), 7839-7855.

[23] R. C. Mittal and R. K. Jain, *Numerical solutions of nonlinear fisher's reaction–diffusion equation with modified cubic b-spline collocation method*, Mathematical Sciences, *7*(1) (2013), 1-10.

[24] R. Mittal, R. Jiwari, and K. K. Sharma, *A numerical scheme based on differential quadrature method to solve time dependent burgers' equation*, Engineering Computations, 2013.

[25] R. Mittal and R. Bhatia, *Numerical solution of second order one dimensional hyperbolic telegraph equation by cubic b-spline collocation method*, Applied Mathematics and Computation, *220* (2013), 496-506.

[26] R. Mittal and R. Bhatia, *A numerical study of two dimensional hyperbolic telegraph equation by modified b-spline differential quadrature method*, Applied Mathematics and Computation, *244* (2014), 976-997.

[27] R. Mittal and S. Dahiya, *Numerical simulation of three-dimensional telegraphic equation using cubic b-spline differential quadrature method*, Applied Mathematics and Computation, *313* (2017), 442-452.

[28] R. Mohanty, *New unconditionally stable difference schemes for the solution of multi-dimensional telegraphic equations*, International Journal of Computer Mathematics, *86*(12) (2009), 2061-2071.

[29] H. J. Nieuwenhuis and L. Schoonbeek, *Stability of matrices with negative diagonal submatrices*, Linear algebra and its applications, *353*(1-3) (2002), 183-196.

[30] T. Nazir, M. Abbas, and M. Yaseen, *Numerical solution of second-order hyperbolic telegraph equation via new cubic trigonometric b-splines approach*, Cogent Mathematics & Statistics, *4*(1) (2017), 1382061.

[31] S. Niknam and H. Adibi, *A numerical solution of two-dimensional hyperbolic telegraph equation based on moving least square meshless method and radial basis functions*, Computational Methods for Differential Equations, 2021.

[32] S. Pandit, M. Kumar, and S. Tiwari, *Numerical simulation of second-order one dimensional hyperbolic telegraph equation*, Computer Physics Communications, *187* (2015), 83-90.

[33] S. Pandit, R. Jiwari, K. Bedi, and M. E. Koksal, *Haar wavelets operational matrix based algorithm for computational modelling of hyperbolic type wave equations*, Engineering Computations, 2017.

[34] B. Pekmen and M. Tezer-Sezgin, *Differential quadrature solution of hyperbolic telegraph equation*, Journal of Applied Mathematics, 2012.

[35] D. Rostamy, M. Emamjome, and S. Abbasbandy, *A meshless technique based on the pseudospectral radial basis functions method for solving the two-dimensional hyperbolic telegraph equation*, The European Physical Journal Plus, *132*(6) (2017), 1-11.

[36] M. Shamsi and M. Razzaghi, *Numerical solution of the controlled duffing oscillator by the interpolating scaling functions*, Journal of electromagnetic waves and applications, *18*(5) (2004), 691-705.

[37] S. Sharifi and J. Rashidinia, *Numerical solution of hyperbolic telegraph equation by cubic b-spline collocation method*, Applied Mathematics and Computation, *281* (2016), 28-38.

[38] C. Shu. *Differential Quadrature and Its Application in Engineering*, Springer Science & Business Media, 2000.

[39] B. K. Singh and P. Kumar, *An algorithm based on a new dqm with modified extended cubic b-splines for numerical study of two dimensional hyperbolic telegraph equation*, Alexandria engineering journal, *57*(1) (2018), 175-191.

[40] B. K. Singh, J. P. Shukla, and M. Gupta, *Study of one dimensional hyperbolic telegraph equation via a hybrid cubic b-spline differential quadrature method*, International Journal of Applied and Computational Mathematics, *7*(1) (2021), 1-17.

[41] R. J. Spiteri and S. J. Ruuth, *A new class of optimal high-order strong-stability-preserving time discretization methods*, SIAM Journal on Numerical Analysis, *40*(2) (2002), 469-491.

[42] N. Sweilam, A. Nagy, and A. El-Sayed, *Sinc-chebyshev collocation method for time-fractional order telegraph equation*, Appl. Comput. Math, *19*(2) (2020), 162-174.
[43] P. R. Wallace, *Mathematical analysis of physical problems*. Courier Corporation, 1984.
[44] F. Wang and Y. Wang, *A coupled pseudospectral-differential quadrature method for a class of hyperbolic telegraph equations*, Mathematical Problems in Engineering, 2017.
[45] V. Weston and S. He, *Wave splitting of the telegraph equation in r3 and its application to inverse scattering*, Inverse Problems, *9*(6) (1993), 789.
[46] M. Zarebnia and R. Parvaz, *An approximation to the solution of one-dimensional hyperbolic telegraph equation based on the collocation of quadratic b-spline functions*, Computational Methods for Differential Equations, *9*(4) (2021), 1198-1213.
[47] Y. Zhou, W. Qu, Y. Gu, and H. Gao, *A hybrid meshless method for the solution of the second order hyperbolic telegraph equation in two space dimensions*, Engineering Analysis with Boundary Elements, *115* (2020), 21-27.