Research Article

Mamta Kapoor*

Numerical simulation of Burgers' equations *via* quartic HB-spline DQM

https://doi.org/10.1515/nleng-2022-0264
received September 30, 2022; accepted December 16, 2022

Abstract: *Via* modified quartic hyperbolic B-spline DQM, Burgers' equation is numerically approximated in the current study. Ten numerical instances are discussed, and the findings are compared with those already in existence and with exact results. Error norms are assessed, and findings are shown in tabular as well as graphical formats, to validate the resilience and applicability portion of established numerical system. Matrix stability analysis approach is used to discuss proposed scheme's stability. The current plan is robust, precise, and simple to put into action.

Keywords: Burgers' equations, DQM, modified quartic hyperbolic B-spline, SSP-RK43 scheme

1 Introduction

Burgers' equation is known as a simplified form of the incompressible Navier–Stokes equation. Burgers' equation is an important partial differential equation from the aspects of various physical applications, traffic flow, shock waves theory, investigating the shallow water waves, in the examination of chemical reaction-diffusion model of Brusselator, and many more. Burgers' equation helps in testing a variety of numerical algorithms.

Burgers' equation is a well-known parabolic partial differential equation nonlinear in nature, which deals with the modeling of a variety of physical flow phenomena of science and engineering, for instance, turbulence, shock waves simulation, boundary layer behavior, and mass transport. Because of its numerous applications, a large number of researchers have developed different numerical techniques to solve it. There is the existence of different explicit and implicit numerical approaches in the literature.

1.1 1D nonlinear Burgers' equation

One-dimensional Burgers' equation is notified as follows:

$$U_t + \alpha U U_x - \nu U_{xx} = 0. \tag{1}$$

1.2 2D nonlinear Burgers' equation

Two-dimensional Burgers' equation is considered as follows:

$$U_t = -UU_x - UU_y + \frac{1}{\mathrm{Re}}[U_{xx} + U_{yy}]. \tag{2}$$

Here, Re is the Reynolds Number. Burgers' equation represents a variety of scientific and engineering issues, and hence, it is important to analyze its solution from both analytical and numerical perspectives.

Bateman first proposed the 1D Burgers' equation and two steady type solutions in 1915. Burgers' equation has been used to simulate a number of technical and theoretical scientific issues, including heat conduction, elasticity, free turbulent traffic flow, and others. According to Burgers' investigations, the mentioned frame worked equation is regarded as his equation. Both the advection term UU_x and dissipation term νU_{xx} are present in this model equation.

Analytical solutions for Burgers' equation were provided by Burgers in 1948, Hopf in 1950, and Cole in 1951. Infinite series solutions were used to arrive at some Burgers' equation answers, but because of the low viscosity, these analytical solutions are not useful because the series solutions converged slowly. Numerous scholars produced a numerical approximation to Burgers' equation and contrasted it with existing analytical solution to ensure accuracy.

In 1D Burgers' equation, $\nu = 0$ changes equation's nature to hyperbolic. The characteristics of solutions to

* **Corresponding author: Mamta Kapoor,** Department of Mathematics, Lovely Professional University, Phagwara, Punjab, 144411, India, e-mail: mamtakapoor.78@yahoo.com

parabolic and hyperbolic equations are very different. Different types of equations can be used to find Burgers' equation. Generalized Burgers' equation will produce 1D Burgers' equation when $\alpha = 1$, $\delta = 1$, and $c = 0$.

Generalized Burgers' equation is specified as follows:

$$U_t + (c + \alpha U^{\delta})U_x - \nu U_{xx} = 0. \tag{3}$$

This equation, which deals with boundary layer equations and controls fluid flow, is identical to model equation when alternative parameters are utilized. Generalized Burgers' equation is converted to Modified Burgers' equation by substituting $c = 0$, $\alpha = 1$, and $\delta = 2$.

Modified Burgers' equation is specified as follows;

$$U_t + U^2 U_x - \nu U_{xx} = 0. \tag{4}$$

A noticeable amount of work is done upon Burgers' equation. Burgers' equation has been solved in both the approaches (analytical and numerical), alike Hopf-Cole transformation [1,2], least square quadratic B-spline termed FE regime [3], quadratic B-spline-based FEM [4], quadratic B-spline collocation approach [5], B-spline collocation approach [6], Sinc DQM [7], reproducing Kernel function approach [8], Quartic B-spline DQM [7], polynomial DQM [9,10], cubic B-spline DQM [10,11], and MCB-spline collocation approach [12]. Some other useful references are as follows [13–15].

1.3 DQM

DQM is a numerical approximation method for finding a numerical solution to PDEs, linear and nonlinear. DQM deals with approximating derivative of a function at specified node points, which is thought of as a sum (linear form) of functional values (which are considered in complete domain). DQM is preferred because it is accurate and simple to use. DQM has been widely used in a variety of physical problem applications. With weighted linear sums of functional values at all nodes taken into consideration, DQM experiments can approximate the derivative of functions in space at any given node location.

1D approximation by DQM is notified as follows:

$$U_x^{(1)} = \sum_{j=1}^{n} p_{ij}^{(1)} U(x_j), \tag{5}$$

$$U_x^{(2)} = \sum_{j=1}^{n} p_{ij}^{(2)} U(x_j). \tag{6}$$

2D approximation by DQM is notified as follows:

$$U_x^{(1)}(x_i, y_j, t) = \sum_{k=1}^{n} p_{ik}^{(1)} U(x_k, y_j, t), \tag{7}$$

$$U_{xx}^{(2)}(x_i, y_j, t) = \sum_{k=1}^{n} p_{ik}^{(2)} U(x_k, y_j, t). \tag{8}$$

$$U_y^{(1)}(x_i, y_j, t) = \sum_{k=1}^{m} q_{jk}^{(1)} U(x_i, y_k, t), \tag{9}$$

$$U_{yy}^{(2)}(x_i, y_j, t) = \sum_{k=1}^{m} q_{jk}^{(2)} U(x_i, y_k, t), \tag{10}$$

or in a more condensed approach.

$$U_x = \sum_{j=1}^{n} p_{ij} U_j,$$

$$U_{xx} = \sum_{j=1}^{n} q_{ij} U_j.$$

$$U_y = \sum_{j=1}^{n} p'_{ij} U_j,$$

$$U_{yy} = \sum_{j=1}^{n} q'_{ij} U_j.$$

Bellman *et al.* [16] introduced an efficient discretization regime to fetch approximated numerical solution with accuracy in 1972. With the current approach, the space coordinate, which is stated as a linear sum of weighted coefficients for all functional values in computing domain under consideration, may be used to approximate the partial derivative of any smooth function. Computation of weighting coefficients is the central idea of DQM. Bellman *et al.* [16], introduced two strategies for evaluating weighting coefficients. Quan and Chang [17,18] and Shu [19] enriched the concept of DQM by giving a recursive formula independent of sample of node points. DQM is able to produce extremely accurate solutions with only a small amount of computational work. Weighting coefficients are evaluated using various basis functions. The literature lists a number of basis functions that are used to create unique varieties of DQMs, including Bernstein polynomial, Sinc functions, Lagrange polynomials, and B-spline functions of various orders (cubic, quartic, quintic, and many others).

A number of DQMs are used in science and engineering to numerically approximate partial differential equations. Feng and Bert [20], used DQM for a nonlinear beam. Korkmaz and Dağ [21] implemented cosine DQM for numerical approximation of RLW equation. Mittal and Jiwari [22] implemented DQM for approximation of 2D Burgers' equation. Korkmaz and Dağ [9] implemented notion of Sinc DQM for fetching solution of Burgers' equation numerically. Some latest researches in this regard is mentioned in [13,23–43].

1.3.1 Main advantages of the study

The need of the hour is to create innovative numerical regimes. It is noted that there is still much scope for numerical study with reference to the 1D and 2D Burgers' 5equations. This article therefore promises to efficiently deliver findings with minimal errors. There is always need for additional study, despite the abundance of numerical techniques in the literature. Some significant results are achieved using currently available technique. Considering that the developed method has produced some positive outcomes, a large class of PDEs, partial-integro differential equations, fractional-order PDEs, and many others can all be effectively handled in this way.

1.3.2 Framework of the article

The present study is subdivided into different sections and subsections.

- In Section 2, a detailed explanation id provided regarding the MQHB-spline.
- In Subsection 2.1, fourth-order MQHB-spline is developed.
- In Subsection 2.2, modified values of QHB-spline are notified.
- In Subsection 2.3, the complete strategy of the developed regime is mentioned.
- In Subsection, 2.4, the formulae regarding the higher-order weighting coefficients are notified.

Table 1: Tabular values of quartic hyperbolic B-spline and derivatives

Tabular values	Ω_{m-2}	Ω_{m-1}	Ω_m	Ω_{m+1}	Ω_{m+2}	Ω_{m+3}
$B_m(\Omega)$	0	a_1	a_2	a_2	a_1	0
$B'_m(\Omega)$	0	a_3	a_4	$-a_4$	$-a_3$	0

- In Subsection 2.5, the error formulae are mentioned.
- In Section 3, implementation of the developed regime is mentioned in one and two dimensions.
- In Section 4, 10 examples are assessed.
- In Section 5, graphical and tabular discussion of the results are discussed in detail.
- Section 6 is about the stability analysis of the regime.
- In Section 7, concluding remarks of this research are provided.

2 Numerical method (MQHB-spline DQM)

2.1 Development of QHB-spline

The development of the quaric hyperbolic B-spline is as follows:

$$H_m(\Omega) = \frac{1}{M} \times \begin{cases} 1.\ [\sinh(\Omega - \Omega_{m-2})]^4, \quad [\Omega_{m-2}, \Omega_{m-1}) \\ 2.\ [\sinh(\Omega - \Omega_{m-2})]^3 \sinh(\Omega_m - \Omega) \\ \quad + [\sinh(\Omega - \Omega_{m-2})]^2 \sinh(\Omega_{m+1} - \Omega) \sinh(\Omega - \Omega_{m-1}) \\ \quad + \sinh(\Omega - \Omega_{m-2}) \sinh(\Omega_{m+2} - \Omega)[\sinh(\Omega - \Omega_{m-1})]^2 \\ \quad + \sinh(\Omega_{m+3} - \Omega)[\sinh(\Omega - \Omega_{m-1})]^3, \quad [\Omega_{m-1}, \Omega_m) \\ 3.\ [\sinh(\Omega - \Omega_{m-2})]^2[\sinh(\Omega_{m+1} - \Omega)]^2 \\ \quad + \sinh(\Omega - \Omega_{m-2}) \sinh(\Omega_{m+2} - \Omega) \sinh(\Omega - \Omega_{m-1}) \sinh(\Omega_{m+1} - \Omega) \\ \quad + \sinh(\Omega - \Omega_{m-2})[\sinh(\Omega_{m+2} - \Omega)]^2 \sinh(\Omega - \Omega_m) \\ \quad + \sinh(\Omega_{m+3} - \Omega)[\sinh(\Omega - \Omega_{m-1})]^2 \sinh(\Omega_{m+1} - \Omega) \\ \quad + \sinh(\Omega_{m+3} - \Omega) \sinh(\Omega - \Omega_{m-1}) \sinh(\Omega_{m+2} - \Omega) \sinh(\Omega - \Omega_m) \\ \quad + [\sinh(\Omega_{m+3} - \Omega)]^2[\sinh(\Omega - \Omega_m)]^2, \quad [\Omega_m, \Omega_{m+1}) \\ 4.\ \sinh(\Omega - \Omega_{m-2})[\sinh(\Omega_{m+2} - \Omega)]^3 \\ \quad + \sinh(\Omega_{m+3} - \Omega) \sinh(\Omega - \Omega_{m-1})[\sinh(\Omega_{m+2} - \Omega)]^2 \\ \quad + [\sinh(\Omega_{m+3} - \Omega)]^2 \sinh(\Omega - \Omega_m) \sinh(\Omega_{m+2} - \Omega) \\ \quad + [\sinh(\Omega_{m+3} - \Omega)]^3 \sinh(\Omega - \Omega_{m+1}), \quad [\Omega_{m+1}, \Omega_{m+2}) \\ 5.\ [\sinh(\Omega_{m+3} - \Omega)]^4, \quad [\Omega_{m+2}, \Omega_{m+3}) \\ 6.0, \quad \text{elsewhere}, \end{cases} \tag{11}$$

where $M = \sinh(h)\sinh(2h)\sinh(3h)\sinh(4h)$.

Derivative function of quartic hyperbolic B-spline is provided as follows:

$$
H'_m(\Omega) = \frac{1}{M} \times \begin{cases}
(1)\ 4[\sinh(\Omega - \Omega_{m-2})]^3 \cosh(\Omega - \Omega_{m-2}),\ [\Omega_{m-2}, \Omega_{m-1}) \\
(2)\ 3[\sinh(\Omega - \Omega_{m-2})]^2 \cosh(\Omega - \Omega_{m-2}) \sinh(\Omega_m - \Omega) \\
\quad - [\sinh(\Omega - \Omega_{m-2})]^3 \cosh(\Omega_m - \Omega) \\
\quad + 2\sinh(\Omega - \Omega_{m-2}) \cosh(\Omega - \Omega_{m-2}) \sinh(\Omega_{m+1} - \Omega) \sinh(\Omega - \Omega_{m-1}) \\
\quad - [\sinh(\Omega - \Omega_{m-2})]^2 \cosh(\Omega_{m+1} - \Omega) \sinh(\Omega - \Omega_{m-1}) \\
\quad + [\sinh(\Omega - \Omega_{m-2})]^2 \sinh(\Omega_{m+1} - \Omega) \cosh(\Omega - \Omega_{m-1}) \\
\quad + \cosh(\Omega - \Omega_{m-2}) \sinh(\Omega_{m+2} - \Omega)[\sinh(\Omega - \Omega_{m-1})]^2 \\
\quad - \sinh(\Omega - \Omega_{m-2}) \cosh(\Omega_{m+2} - \Omega)[\sinh(\Omega - \Omega_{m-1})]^2 \\
\quad + 2\sinh(\Omega - \Omega_{m-2}) \sinh(\Omega_{m+2} - \Omega) \sinh(\Omega - \Omega_{m-1}) \cosh(\Omega - \Omega_{m-1}) \\
\quad - \cosh(\Omega_{m+3} - \Omega)[\sinh(\Omega - \Omega_{m-1})]^3 \\
\quad + 3\sinh(\Omega_{m+3} - \Omega)[\sinh(\Omega - \Omega_{m-1})]^2 \cosh(\Omega - \Omega_{m-1}),\ [\Omega_{m-1}, \Omega_m) \\
(3)\ 2\sinh(\Omega - \Omega_{m-2}) \cosh(\Omega - \Omega_{m-2})[\sinh(\Omega_{m+1} - \Omega)]^2 \\
\quad - 2[\sinh(\Omega - \Omega_{m-2})]^2 \sinh(\Omega_{m+1} - \Omega) \cosh(\Omega_{m+1} - \Omega) \\
\quad + \cosh(\Omega - \Omega_{m-2}) \sinh(\Omega_{m+2} - \Omega) \sinh(\Omega - \Omega_{m-1}) \sinh(\Omega_{m+1} - \Omega) \\
\quad - \sinh(\Omega - \Omega_{m-2}) \cosh(\Omega_{m+2} - \Omega) \sinh(\Omega - \Omega_{m-1}) \sinh(\Omega_{m+1} - \Omega) \\
\quad + \sinh(\Omega - \Omega_{m-2}) \sinh(\Omega_{m+2} - \Omega) \cosh(\Omega - \Omega_{m-1}) \sinh(\Omega_{m+1} - \Omega) \\
\quad - \sinh(\Omega - \Omega_{m-2}) \sinh(\Omega_{m+2} - \Omega) \sinh(\Omega - \Omega_{m-1}) \cosh(\Omega_{m+1} - \Omega) \\
\quad + \cosh(\Omega - \Omega_{m-2})[\sinh(\Omega_{m+2} - \Omega)]^2 \sinh(\Omega - \Omega_m) \\
\quad - 2\sinh(\Omega - \Omega_{m-2}) \sinh(\Omega_{m+2} - \Omega) \cosh(\Omega_{m+2} - \Omega) \sinh(\Omega - \Omega_m) \\
\quad + \sinh(\Omega - \Omega_{m-2})[\sinh(\Omega_{m+2} - \Omega)]^2 \cosh(\Omega - \Omega_m) \\
\quad - \cosh(\Omega_{m+3} - \Omega)[\sinh(\Omega - \Omega_{m-1})]^2 \sinh(\Omega_{m+1} - \Omega) \\
\quad + 2\sinh(\Omega_{m+3} - \Omega) \sinh(\Omega - \Omega_{m-1}) \cosh(\Omega - \Omega_{m-1}) \sinh(\Omega_{m+1} - \Omega) \\
\quad - \sinh(\Omega_{m+3} - \Omega)[\sinh(\Omega - \Omega_{m-1})]^2 \cosh(\Omega_{m+1} - \Omega) \\
\quad - \cosh(\Omega_{m+3} - \Omega) \sinh(\Omega - \Omega_{m-1}) \sinh(\Omega_{m+2} - \Omega) \sinh(\Omega - \Omega_m) \\
\quad + \sinh(\Omega_{m+3} - \Omega) \cosh(\Omega - \Omega_{m-1}) \sinh(\Omega_{m+2} - \Omega) \sinh(\Omega - \Omega_m) \\
\quad - \sinh(\Omega_{m+3} - \Omega) \sinh(\Omega - \Omega_{m-1}) \cosh(\Omega_{m+2} - \Omega) \sinh(\Omega - \Omega_m) \\
\quad + \sinh(\Omega_{m+3} - \Omega) \sinh(\Omega - \Omega_{m-1}) \sinh(\Omega_{m+2} - \Omega) \cosh(\Omega - \Omega_m) \\
\quad - 2\sinh(\Omega_{m+3} - \Omega) \cosh(\Omega_{m+3} - \Omega)[\sinh(\Omega - \Omega_m)]^2 \\
\quad + 2[\sinh(\Omega_{m+3} - \Omega)]^2 \sinh(\Omega - \Omega_m) \cosh(\Omega - \Omega_m),\ [\Omega_m, \Omega_{m+1}) \\
(4)\ \cosh(\Omega - \Omega_{m-2})[\sinh(\Omega_{m+2} - \Omega)]^3 \\
\quad - 3\sinh(\Omega - \Omega_{m-2})[\sinh(\Omega_{m+2} - \Omega)]^2 \cosh(\Omega_{m+2} - \Omega) \\
\quad - \cosh(\Omega_{m+3} - \Omega) \sinh(\Omega - \Omega_{m-1})[\sinh(\Omega_{m+2} - \Omega)]^2 \\
\quad + \sinh(\Omega_{m+3} - \Omega) \cosh(\Omega - \Omega_{m-1})[\sinh(\Omega_{m+2} - \Omega)]^2 \\
\quad - 2\sinh(\Omega_{m+3} - \Omega) \sinh(\Omega - \Omega_{m-1}) \sinh(\Omega_{m+2} - \Omega) \cosh(\Omega_{m+2} - \Omega) \\
\quad - 2\sinh(\Omega_{m+3} - \Omega) \cosh(\Omega_{m+3} - \Omega) \sinh(\Omega - \Omega_m) \sinh(\Omega_{m+2} - \Omega) \\
\quad + [\sinh(\Omega_{m+3} - \Omega)]^2 \cosh(\Omega - \Omega_m) \sinh(\Omega_{m+2} - \Omega) \\
\quad - [\sinh(\Omega_{m+3} - \Omega)]^2 \sinh(\Omega - \Omega_m) \cosh(\Omega_{m+2} - \Omega) \\
\quad - 3[\sinh(\Omega_{m+3} - \Omega)]^2 \cosh(\Omega_{m+3} - \Omega) \sinh(\Omega - \Omega_{m+1}) \\
\quad + [\sinh(\Omega_{m+3} - \Omega)]^3 \cosh(\Omega - \Omega_{m+1}),\ [\Omega_{m+1}, \Omega_{m+2}) \\
(5)\ -4[\sinh(\Omega_{m+3} - \Omega)]^3 \cosh(\Omega_{m+3} - \Omega),\ [\Omega_{m+2}, \Omega_{m+3}) \\
(6)\ 0, \quad \text{elsewhere},
\end{cases} \tag{12}
$$

where $M = \sinh(h)\sinh(2h)\sinh(3h)\sinh(4h)$.

$$
a_1 = \frac{[\sinh(h)]^4}{M}, \quad a_2 = \frac{2[\sinh(h)]^2[\sinh(2h)]^2 + \sinh(3h)[\sinh(h)]^3}{M},
$$

$$
a_3 = \frac{2[\sinh(h)]^2 \sinh(2h)}{M}, \; a_4 = \frac{2\sinh(2h)\cosh(2h)[\sinh(h)]^2 - \cosh(3h)[\sinh(h)]^3 + 3\sinh(3h)[\sinh(h)]^2 \cosh(h)}{M}
$$

(Table 1).

2.2 Modified value of QHB-spline

Improvised value of quartic hyperbolic B-spline is fetched from the following set of formulae [44–46].

$$\begin{aligned} MQH_1(\Omega) &= QH_1(\Omega) + 2QH_0(\Omega), \\ MQH_2(\Omega) &= QH_2(\Omega) - QH_0(\Omega), \\ MQH_j(\Omega) &= QH_j(\Omega), j = 3, 4, 5, \ldots, K-2, \\ MQH_{K-1}(\Omega) &= QH_{K-1}(\Omega) - QH_{K+1}(\Omega), \\ MQH_K(\Omega) &= QH_K(\Omega) + 2QH_{K+1}(\Omega). \end{aligned} \tag{13}$$

2.3 Evaluation of weighting coefficients

Weighting coefficients can be easily obtained by implementing modified values of quartic hyperbolic B-spline in the discretization formula of DQM {considered $\xi = x$}.

$$MQH_k^{(1)}(x_i) = \sum_{j=1}^{n} p_{ij}^{(1)} MQH_k(x_j). \tag{14}$$

At grid point x_1:

$$\begin{bmatrix} MQH_1^{(1)}(x_1) \\ MQH_2^{(1)}(x_1) \\ MQH_3^{(1)}(x_1) \\ \vdots \\ MQH_n^{(1)}(x_1) \end{bmatrix} = \begin{bmatrix} MQH_1(x_1) & MQH_1(x_2) & MQH_1(x_3) & \ldots & \ldots & \ldots & MQH_1(x_n) \\ MQH_2(x_1) & MQH_2(x_2) & MQH_2(x_3) & \ldots & \ldots & \ldots & MQH_2(x_n) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ MQH_n(x_1) & MQH_n(x_2) & MQH_n(x_3) & \ldots & \ldots & \ldots & MQH_n(x_n) \end{bmatrix} \begin{bmatrix} a_{11}^{(1)} \\ a_{12}^{(1)} \\ a_{13}^{(1)} \\ \vdots \\ a_{1n}^{(1)} \end{bmatrix}. \tag{15}$$

At grid point x_2:

$$\begin{bmatrix} MQH_1^{(1)}(x_2) \\ MQH_2^{(1)}(x_2) \\ MQH_3^{(1)}(x_2) \\ \vdots \\ MQH_n^{(1)}(x_2) \end{bmatrix} = \begin{bmatrix} MQH_1(x_1) & MQH_1(x_2) & MQH_1(x_3) & \ldots & \ldots & \ldots & MQH_1(x_n) \\ MQH_2(x_1) & MQH_2(x_2) & MQH_2(x_3) & \ldots & \ldots & \ldots & MQH_2(x_n) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ MQH_n(x_1) & MQH_n(x_2) & MQH_n(x_3) & \ldots & \ldots & \ldots & MQH_n(x_n) \end{bmatrix} \begin{bmatrix} a_{21}^{(1)} \\ a_{22}^{(1)} \\ a_{23}^{(1)} \\ \vdots \\ a_{2n}^{(1)} \end{bmatrix}. \tag{16}$$

At grid point x_3:

$$\begin{bmatrix} MQH_1^{(1)}(x_3) \\ MQH_2^{(1)}(x_3) \\ MQH_3^{(1)}(x_3) \\ \vdots \\ MQH_n^{(1)}(x_3) \end{bmatrix} = \begin{bmatrix} MQH_1(x_1) & MQH_1(x_2) & MQH_1(x_3) & \ldots & \ldots & \ldots & MQH_1(x_n) \\ MQH_2(x_1) & MQH_2(x_2) & MQH_2(x_3) & \ldots & \ldots & \ldots & MQH_2(x_n) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ MQH_n(x_1) & MQH_n(x_2) & MQH_n(x_3) & \ldots & \ldots & \ldots & MQH_n(x_n) \end{bmatrix} \begin{bmatrix} a_{31}^{(1)} \\ a_{32}^{(1)} \\ a_{33}^{(1)} \\ \vdots \\ a_{3n}^{(1)} \end{bmatrix}. \tag{17}$$

$$\vdots$$

At grid point x_n:

$$\begin{bmatrix} MQH_1^{(1)}(x_n) \\ MQH_2^{(1)}(x_n) \\ MQH_3^{(1)}(x_n) \\ \vdots \\ MQH_n^{(1)}(x_n) \end{bmatrix} = \begin{bmatrix} MQH_1(x_1) & MQH_1(x_2) & MQH_1(x_3) & \ldots & \ldots & \ldots & MQH_1(x_n) \\ MQH_2(x_1) & MQH_2(x_2) & MQH_2(x_3) & \ldots & \ldots & \ldots & MQH_2(x_n) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ MQH_n(x_1) & MQH_n(x_2) & MQH_n(x_3) & \ldots & \ldots & \ldots & MQH_n(x_n) \end{bmatrix} \begin{bmatrix} a_{n1}^{(1)} \\ a_{n2}^{(1)} \\ a_{n3}^{(1)} \\ \vdots \\ a_{nn}^{(1)} \end{bmatrix}, \tag{18}$$

Table 2: L_2 and L_∞ at $t = 0.1$, 0.5, and 1.0

	$t = 0.1$		$t = 0.5$		$t = 1.0$	
	L_2	L_∞	L_2	L_∞	L_2	L_∞
$v = 10^{-3}$	5.4008×10^{-5}	7.8420×10^{-5}	2.6847×10^{-4}	3.8849×10^{-4}	5.3324×10^{-4}	7.6707×10^{-4}
$v = 10^{-4}$	5.4100×10^{-7}	7.8607×10^{-7}	2.7034×10^{-6}	3.9270×10^{-6}	5.4028×10^{-6}	7.8454×10^{-6}
$v = 10^{-5}$	5.4110×10^{-9}	7.8626×10^{-9}	2.7053×10^{-8}	3.9309×10^{-8}	5.4102×10^{-8}	7.8611×10^{-8}

where

$$\begin{bmatrix} MQH_1^{(1)}(x_1) \\ MQH_2^{(1)}(x_1) \\ MQH_3^{(1)}(x_1) \\ \vdots \\ MQH_n^{(1)}(x_1) \end{bmatrix} = \begin{bmatrix} -a_4 \\ a_3 + a_4 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

$$\begin{bmatrix} MQH_1^{(1)}(x_2) \\ MQH_2^{(1)}(x_2) \\ MQH_3^{(1)}(x_2) \\ \vdots \\ MQH_n^{(1)}(x_2) \end{bmatrix} = \begin{bmatrix} -2a_3 - a_4 \\ a_3 + a_4 \\ a_3 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

$$\begin{bmatrix} MQH_1^{(1)}(x_3) \\ MQH_2^{(1)}(x_3) \\ MQH_3^{(1)}(x_3) \\ \vdots \\ MQH_n^{(1)}(x_3) \end{bmatrix} = \begin{bmatrix} -a_3 \\ -a_4 \\ a_4 \\ a_3 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

$$\begin{bmatrix} MQH_1^{(1)}(x_4) \\ MQH_2^{(1)}(x_4) \\ MQH_3^{(1)}(x_4) \\ \vdots \\ MQH_n^{(1)}(x_4) \end{bmatrix} = \begin{bmatrix} 0 \\ -a_3 \\ -a_4 \\ a_4 \\ a_3 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

$$\begin{bmatrix} MQH_1^{(1)}(x_5) \\ MQH_2^{(1)}(x_5) \\ MQH_3^{(1)}(x_5) \\ \vdots \\ MQH_n^{(1)}(x_5) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -a_3 \\ -a_4 \\ a_4 \\ a_3 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

$$\begin{bmatrix} MQH_1^{(1)}(x_{n-2}) \\ MQH_2^{(1)}(x_{n-2}) \\ MQH_3^{(1)}(x_{n-2}) \\ \vdots \\ MQH_n^{(1)}(x_{n-2}) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ -a_3 \\ -a_4 \\ a_4 \\ a_3 \\ 0 \end{bmatrix}.$$

$$\begin{bmatrix} MQH_1^{(1)}(x_{n-1}) \\ MQH_2^{(1)}(x_{n-1}) \\ MQH_3^{(1)}(x_{n-1}) \\ \vdots \\ MQH_n^{(1)}(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ -a_3 \\ -a_4 \\ a_4 \\ a_3 \end{bmatrix}.$$

$$\begin{bmatrix} MQH_1^{(1)}(x_n) \\ MQH_2^{(1)}(x_n) \\ MQH_3^{(1)}(x_n) \\ \vdots \\ MQH_n^{(1)}(x_n) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ -a_3 \\ a_4 + 2a_3 \end{bmatrix}.$$

2.4 Higher-order weighting coefficients

After evaluating weighting coefficients, which approximates first-order partial derivatives and second-order weighting coefficients can be easily calculated *via* following formulae [19,44–46]:

$$q_{ij} = 2p_{ij}\left[p_{ii} - \frac{1}{(x_i - x_j)}\right],$$

$$q_{ii} = -\sum_{j=1, j\neq i}^{n} q_{ij}.$$

By utilizing the aforementioned formulas, higher-order weighting coefficients can be evaluated.

$$w_{ij}^{(m)} = m\left[p_{ij}w_{ii}^{(m-1)} - \frac{w_{ij}^{(m-1)}}{(x_i - x_j)}\right],$$

$$w_{ii}^{(m)} = -\sum_{j=1, j\neq i}^{n} w_{ij}^{(m)}.$$

In two dimension, weighting coefficients can be evaluated by following the set of formulae,

$$w_{ij}^{(n)} = n\left[p_{ij}^{x} w_{ii}^{(n-1)} - \frac{w_{ij}^{(n-1)}}{(x_i - x_j)}\right],$$
$$w_{ii}^{(n)} = -\sum_{j=1, j\neq i}^{n} w_{ij}^{(n)}.$$
$$\bar{w}_{ij}^{(m)} = m\left[a_{ij}^{y} \bar{w}_{ii}^{(m-1)} - \frac{\bar{w}_{ij}^{(m-1)}}{(y_i - y_j)}\right],$$
$$\bar{w}_{ii}^{(m)} = -\sum_{j=1, j\neq i}^{n} \bar{w}_{ij}^{(m)}.$$

2.5 Error norms

In 1D:

$$L_2 = \sqrt{h\sum |U^{\text{Exact}} - U^{\text{Numerical}}|^2},$$
$$L_\infty = \max|U^{\text{Exact}} - U^{\text{Numerical}}|.$$

In 2D:

$$\text{Relative error} = \sqrt{\frac{\sum\sum (u_{ij} - \bar{u}_{ij})^2}{\sum\sum u_{ij}^2}},$$
$$\text{RMS error} = \sqrt{\frac{\sum\sum (u_{ij} - \bar{u}_{ij})^2}{N \times M}},$$
$$L_2 = \sqrt{\sum\sum |U^{\text{Exact}} - U^{\text{Numerical}}|^2},$$
$$L_\infty = \max|U^{\text{Exact}} - U^{\text{Numerical}}|,$$

where u_{ij} represents exact solution and $\bar{u}_{ij}$ represents numerical solution.

3 Implementation of scheme

3.1 1D Burgers' equation

Implementing the formulae of DQM approximation in 1D Burgers' equation,

$$U_t + \alpha U U_x - \nu U_{xx} = 0,$$
$$U_t = -\alpha U U_x + \nu U_{xx},$$
$$\frac{\mathrm{d}U_i}{\mathrm{d}t} = -\alpha_i U_i \sum_{j=1}^{n} p_{ij}^{(1)} U(x_j) + \nu \sum_{j=1}^{n} p_{ij}^{(2)} U(x_j).$$

3.2 2D Burgers' equation

By implementing the DQM approximation in 2D Burgers' equation,

$$U_t = -UU_x - UU_y + \frac{1}{\text{Re}}[U_{xx} + U_{yy}],$$
$$\frac{\mathrm{d}U_i}{\mathrm{d}t} = -U\sum_{j=1}^{n} p_{ij} U_j - U\sum_{j=1}^{n} p_{ij}' U_j + \frac{1}{\text{Re}}\left[\sum_{j=1}^{n} q_{ij} U_j + \sum_{j=1}^{n} q_{ij}' U_j\right].$$

3.3 SSP-RK 43 scheme

SSP-RK 43 regime [46–48] is considered as TVD time discretization scheme and is implemented to solve the system of ODE'E2 process of RK method have order m is provided as follows:

$$u^{(0)} = u^n,$$
$$u^{(i)} = \sum_{k=1}^{j-1} [\alpha_{j,k} u^{(k)} + \Delta t \beta_{j,k} L(u^{(k)})],$$
$$u^{(n+1)} = u^{(m)},$$

where $\alpha_{j,k} > 0$. For the consistent system,

$$U_t = -f(u)_x,$$
$$\frac{\mathrm{d}U}{\mathrm{d}t} = L(U),$$

and

$$u^{(n+1)} = u^n + \Delta t L(u^n).$$

4 Examples and discussion

This section deals with 10 numerical examples. Of 10 examples, first eight examples are provided regarding 1D Burgers' equation and last two examples are provided regarding 2D Burgers' equation.

Example 1. Considered 1D Burgers' equation with $\alpha = 1$ in domain [0,2] [49,50].

$$u(x,t) = (2\pi\nu) \times \frac{[\sin(\pi x)\exp(-\pi^2\nu^2 t) + 4\sin(2\pi x)\exp(-4\pi^2\nu^2 t)]}{[4 + \cos(\pi x)\exp(-\pi^2\nu^2 t) + 2\cos(2\pi x)\exp(-4\pi^2\nu^2 t)]}.$$
$$\text{Domain} = [0, 2],\ t \geq 0, \tag{19}$$

Table 3: Comparison of errors at $t = 0.1$

Comparison of errors								
$t = 0.1$								
	MCB-DQM [44]		Mittal and Jain [12]		MTCB-DQM [49]		Present	
	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞
10^{-2}	3.41×10^{-3}	3.89×10^{-3}	3.55×10^{-3}	4.41×10^{-3}	1.60×10^{-3}	1.76×10^{-3}	2.9879×10^{-4}	3.0136×10^{-4}
10^{-3}	3.55×10^{-5}	4.09×10^{-5}	3.72×10^{-5}	4.60×10^{-5}	1.64×10^{-5}	1.84×10^{-5}	2.9901×10^{-6}	3.0218×10^{-6}
10^{-4}	3.56×10^{-7}	4.11×10^{-7}	3.74×10^{-7}	4.62×10^{-7}	1.65×10^{-7}	1.85×10^{-7}	2.9908×10^{-8}	3.0376×10^{-8}

Boundary conditions: $u(0, t) = 0$ and $u(2, t) = 0$.

In Table 2, L_2, and L_∞ are discussed at $t = 0.1$, 0.5, and 1.0, respectively, for different values of ν. It is noticed that while changing values of ν from 10^{-3} to 10^{-5}, scheme gave reduced errors. It can be said that on changing values of ν from 10^{-3} to 10^{-5}, scheme is showing converging results. In Table 3, error norms are compared with [12,44,49] at $t = 0.1$. Present errors are much better than the existing results. In Table 4, errors are compared with available results of [12,44,49] at $t = 1.0$. Present results are better than existing results at mentioned values of ν. In Figure 1, comparison between exact and numerical solution is provided at a series of time levels for $\nu = 10^{-2}$, $\Delta t = 0.00001$.

Example 2. Considered Burgers' equation with $\alpha = 1$. Given domain is [0, 1]. $t \geq 0$ [50].

$$u(x, t) = (2\pi\nu)\frac{[\sin(\pi x)\exp(-\pi^2\nu t)]}{[\sigma + \cos(\pi x)\exp(-\pi^2\nu t)]}, \quad \sigma > 1. \tag{20}$$

Initial condition:

$$u(x, 0) = (2\pi\nu) \times \frac{[\sin(\pi x)]}{[\sigma + \cos(\pi x)]}, \quad \sigma > 1. \tag{21}$$

Boundary conditions: $u(0, t) = 0$, $u(1, t) = 0$.

In Table 5, errors are provided at $t = 0.001$, 0.005, and 0.008, respectively, for different values of ν and σ. In Figure 2, solutions are matched at $t = 0.001$, 0.003, 0.005, and $t = 0.008$ for $\nu = 0.005$ and $\sigma = 100$.

Example 3. Considered 1D Burgers' equation where $\alpha = 1$ from [12], where the mentioned equation has following exact solution.

$$u(x, t) = \frac{\nu}{(1 + \nu t)}\left[x + \tan\left(\frac{x}{(2 + 2\nu t)}\right)\right]. \tag{22}$$

Domain = [0.5, 1.5], $t \geq 0$.

Initial condition:

$$u(x, 0) = \nu\left[x + \tan\left(\frac{x}{(2)}\right)\right]. \tag{23}$$

Boundary conditions:

$$u(0.5, t) = \frac{\nu}{(1 + \nu t)}\left[0.5 + \tan\left(\frac{0.5}{(2 + 2\nu t)}\right)\right], \tag{24}$$

and

Table 4: Comparison of errors at $t = 1.0$

Comparison of errors								
$t = 1.0$								
	MCB-DQM [44]		Mittal and Jain [12]		MTCB-DQM [49]		Present	
	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞
10^{-2}	2.63×10^{-2}	2.92×10^{-2}	2.66×10^{-2}	3.13×10^{-3}	1.29×10^{-2}	1.24×10^{-2}	2.9839×10^{-3}	3.1398×10^{-3}
10^{-3}	3.45×10^{-4}	3.93×10^{-4}	3.59×10^{-4}	4.45×10^{-4}	1.60×10^{-4}	1.76×10^{-4}	2.9877×10^{-5}	3.0038×10^{-5}
10^{-4}	3.55×10^{-6}	4.09×10^{-6}	3.72×10^{-6}	4.61×10^{-6}	1.64×10^{-6}	1.84×10^{-6}	2.9905×10^{-7}	3.0358×10^{-7}

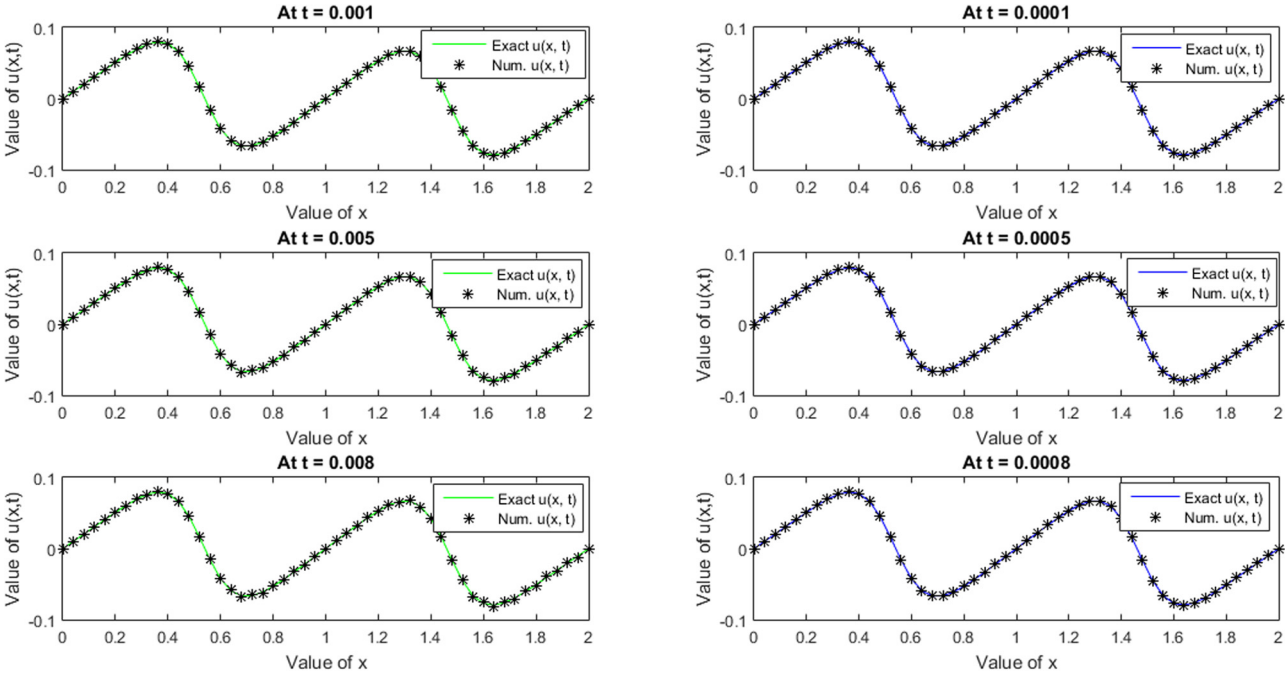

Figure 1: Solutions for $N = 51$, $\nu = 10^{-2}$, $\Delta t = 0.00001$.

$$u(1.5, t) = \frac{\nu}{(1 + \nu t)}\left[1.5 + \tan\left(\frac{1.5}{(2 + 2\nu t)}\right)\right]. \quad (25)$$

In Table 6, present errors are matched with [12] at mentioned values of t. Present results are almost near to compared results. In Table 7, present error norms are matched with [12,51] at $t = 1.68, 2.1, 2.66$, and 3.3, respectively. Present numerical regime has produced better results. In Figure 3, exact and numerical solutions are compared at mentioned values of t for $\Delta t = 0.0002$ and $\nu = 0.00006666$.

Example 4. In present example [12], Eq. (1) is considered with $\alpha = 1$, where the exact solution is notified as follows:

$$u(x, t) = \frac{c}{\alpha} + \frac{\nu k b}{\alpha \cosh(k(x - ct) + \mu)} - \frac{\nu k \sinh(k(x - ct))}{\alpha \cosh(k(x - ct) + \mu)}. \quad (26)$$

Initial condition:

$$u(x, 0) = \frac{c}{\alpha} + \frac{\nu k b}{\alpha \cosh(k(x) + \mu)} - \frac{\nu k \sinh(k(x))}{\alpha \cosh(k(x) + \mu)}. \quad (27)$$

Boundary conditions: $u(0, t) = 0$ and $u(l_1, t) = 0$. $b = \sqrt{\mu^2 - 1}$.

In Table 8, L_2 and L_∞ error norms are matched with [12], for different values of c, μ, and σ with $\Delta t = 0.01$. In Figure 4, graphical comparison of solutions is given at different values of t for $\Delta t = 0.01$, $\nu = 10^{-5}$, $c = 10^{-5}$, $\mu = 2$, and $K = 1$.

Example 5. Considered initial and boundary conditions are as per [12,52,53].

Table 5: L_2 and L_∞ error norms

n	$\Delta t = 0.0001$, $\nu = 0.005$, $\sigma = 100$, $t = 0.001$		$\Delta t = 0.0001$, $\nu = 0.005$, $\sigma = 100$, $t = 0.005$		$\Delta t = 0.0001$, $\nu = 0.005$, $\sigma = 100$, $t = 0.008$	
10	1.2027×10^{-8}	2.0064×10^{-8}	5.9992×10^{-8}	1.0003×10^{-7}	9.5813×10^{-8}	1.5970×10^{-7}
30	2.4941×10^{-7}	3.8321×10^{-7}	1.2571×10^{-6}	1.9155×10^{-6}	2.0347×10^{-6}	3.0655×10^{-6}
50	8.6607×10^{-7}	1.2621×10^{-6}	5.6842×10^{-6}	1.3991×10^{-5}	1.1311×10^{-5}	3.8291×10^{-5}

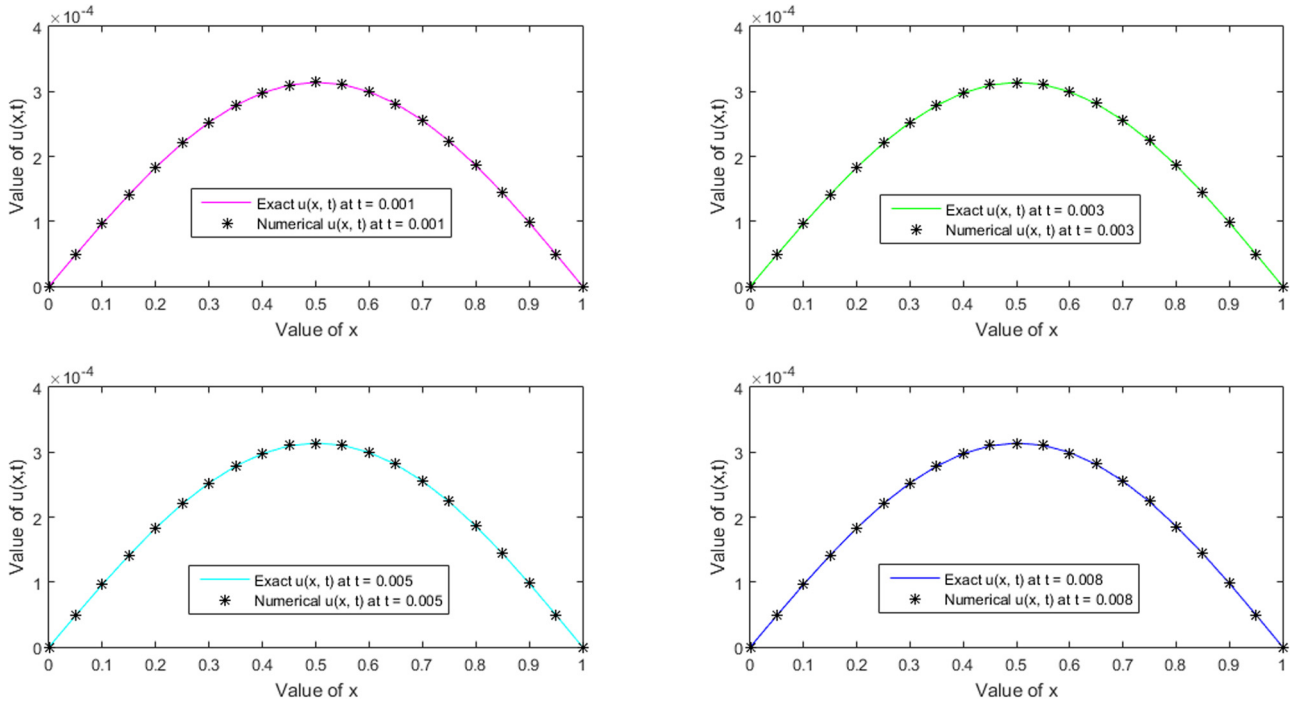

Figure 2: Solutions for $N = 21$, $\Delta t = 0.0001$, $\nu = 0.005$, and $\sigma = 100$.

Table 6: Compatibility of error norms with [12], for $\Delta t = 0.0001$, $\nu = 0.002$

Comparability of errors at different levels of time				
	Mittal and Jain [12]		Present method	
Time	L_2	L_∞	L_2	L_∞
1.68	3.21×10^{-6}	1.72×10^{-5}	4.0676×10^{-5}	8.1122×10^{-5}
2.1	4.31×10^{-6}	2.23×10^{-5}	5.0355×10^{-5}	1.0045×10^{-4}
2.66	5.88×10^{-6}	2.92×10^{-5}	6.2974×10^{-5}	1.2567×10^{-4}
3.3	7.78×10^{-6}	3.73×10^{-5}	7.7009×10^{-5}	1.5372×10^{-4}

Exact solution:

$$u(x, t) = \frac{2\nu\pi \exp(-\pi^2\nu t)\sin(\pi x)}{\alpha + \exp(-\pi^2\nu t)\cos(\pi x)}. \tag{28}$$

Initial condition:

$$u(x, 0) = \frac{2\nu\pi \sin(\pi x)}{\alpha + \cos(\pi x)}. \tag{29}$$

Boundary conditions: $u(0, t) = 0$, $u(1, t) = 0$.

In Figure 5, graphical compatibility of solutions is given at different values of t for $\Delta t = 0.0001$, $\alpha = 10$, and $\nu = 0.001$. In Figure 6, absolute error is notified at values of t with $\nu = 0.001$, $\alpha = 10$, and $\Delta t = 0.0001$. Absolute error approaches to zero at boundary points.

Example 6. In this example, considered particular solution is taken from [12,54,55].

$$u(x, t) = \frac{[\alpha + \mu + (\mu - \alpha \exp(\eta))]}{1 + \exp(\eta)}, \tag{30}$$

Table 7: Compatibility of errors with [12] and [51]

Comparison of L_2 and L_∞ error norms						
	Mittal and Jain [12]		Raslan [51]		Present method	
Time	L_2	L_∞	L_2	L_∞	L_2	L_∞
1.68	2.40×10^{-7}	1.48×10^{-6}	2.41×10^{-4}	1.63×10^{-3}	4.2305×10^{-9}	8.4238×10^{-9}
2.1	3.27×10^{-7}	1.97×10^{-6}	2.41×10^{-4}	1.63×10^{-3}	5.2876×10^{-9}	1.0529×10^{-8}
2.66	4.52×10^{-7}	2.65×10^{-6}	2.41×10^{-4}	1.63×10^{-3}	6.6967×10^{-9}	1.3335×10^{-8}

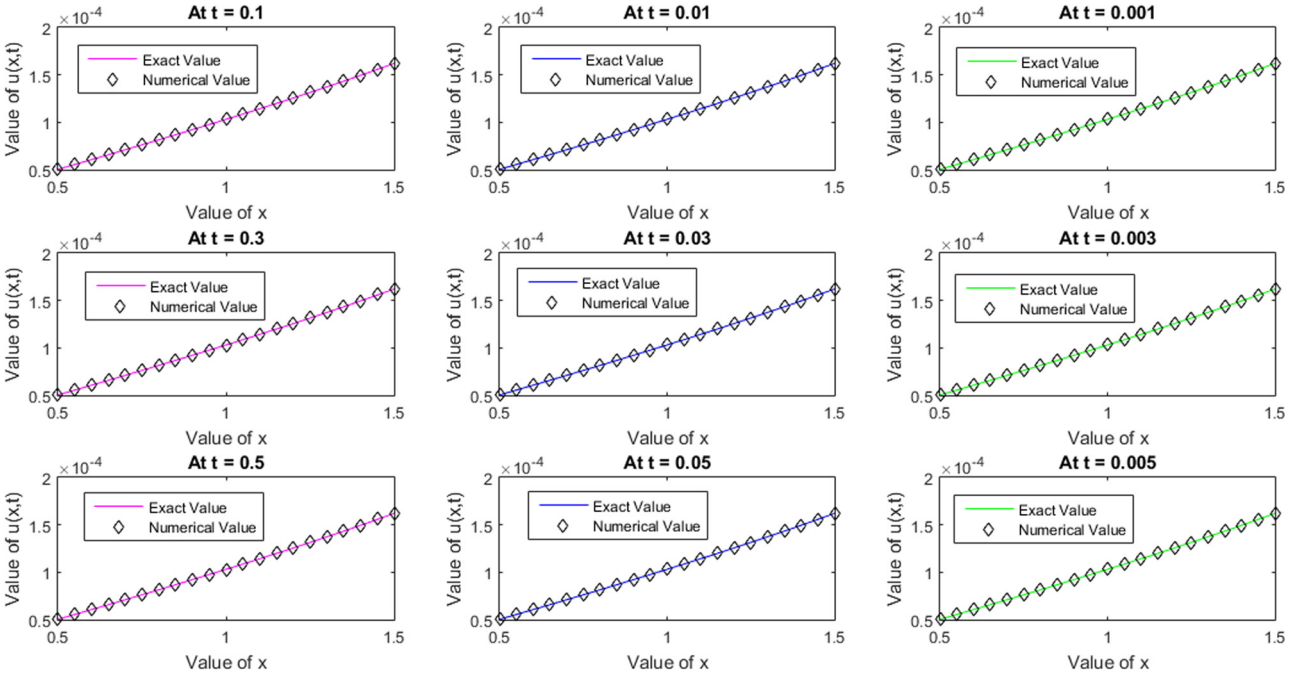

Figure 3: Comparison of solutions for $N = 21$, $\Delta t = 0.0002$ and $\nu = 0.00006666$.

where,

$$\eta = \frac{\alpha(x - \mu t - \beta)}{\nu}.$$

Initial condition:

$$u(x, 0) = \frac{[\alpha + \mu + (\mu - \alpha \exp(\eta))]}{1 + \exp(\eta)}, \tag{31}$$

where

$$\eta = \frac{\alpha(x - \beta)}{\nu}.$$

Boundary conditions:

$u(0, t) = 1$, $u(1, t) = 0.2$.

In Figure 7, graphical plot is provided for likeliness of exact and obtained solutions at mentioned values of t with $\Delta t = 0.0001$, $\alpha = 0.4$, $\nu = 0.01$, $\beta = 0.125$, and $\mu = 0.6$. In Table 9, solutions are matched at $t = 0.0001$, $t = 0.0005$, and $t = 0.0008$, respectively, for $\Delta t = 0.0001$, $\alpha = 0.4$, $\nu = 0.01$, $\beta = 0.125$, and $\mu = 0.6$.

Example 7. In present example, I.C. and B.C.s are considered as per [12,52,55,56].

Table 8: Comparability of error norms with [12], for $\Delta t = 0.01$, at $t = 0.05$ for diverse values of c, μ, and ν

Comparison of errors						
		Mittal and Jain [12]			Present	
c	μ	ν	L_2	L_∞	L_2	L_∞
10^{-5}	2	10^{-5}	9.35×10^{-6}	2.36×10^{-6}	5.6977×10^{-6}	1.4604×10^{-5}
10^{-5}	3	10^{-5}	5.67×10^{-6}	1.51×10^{-6}	5.2589×10^{-6}	1.2809×10^{-5}
10^{-5}	−2	10^{-5}	1.04×10^{-5}	2.73×10^{-6}	6.3124×10^{-6}	1.4604×10^{-5}
75×10^{-5}	2	75×10^{-5}	6.09×10^{-4}	1.06×10^{-4}	4.2734×10^{-4}	1.0953×10^{-3}
75×10^{-5}	3	75×10^{-5}	6.06×10^{-4}	1.06×10^{-4}	3.9442×10^{-4}	9.6072×10^{-4}
75×10^{-5}	−2	75×10^{-5}	6.05×10^{-4}	1.05×10^{-4}	4.7344×10^{-4}	1.0953×10^{-3}
75×10^{-5}	−3	75×10^{-5}	6.08×10^{-4}	1.06×10^{-4}	4.5699×10^{-4}	1.0796×10^{-3}

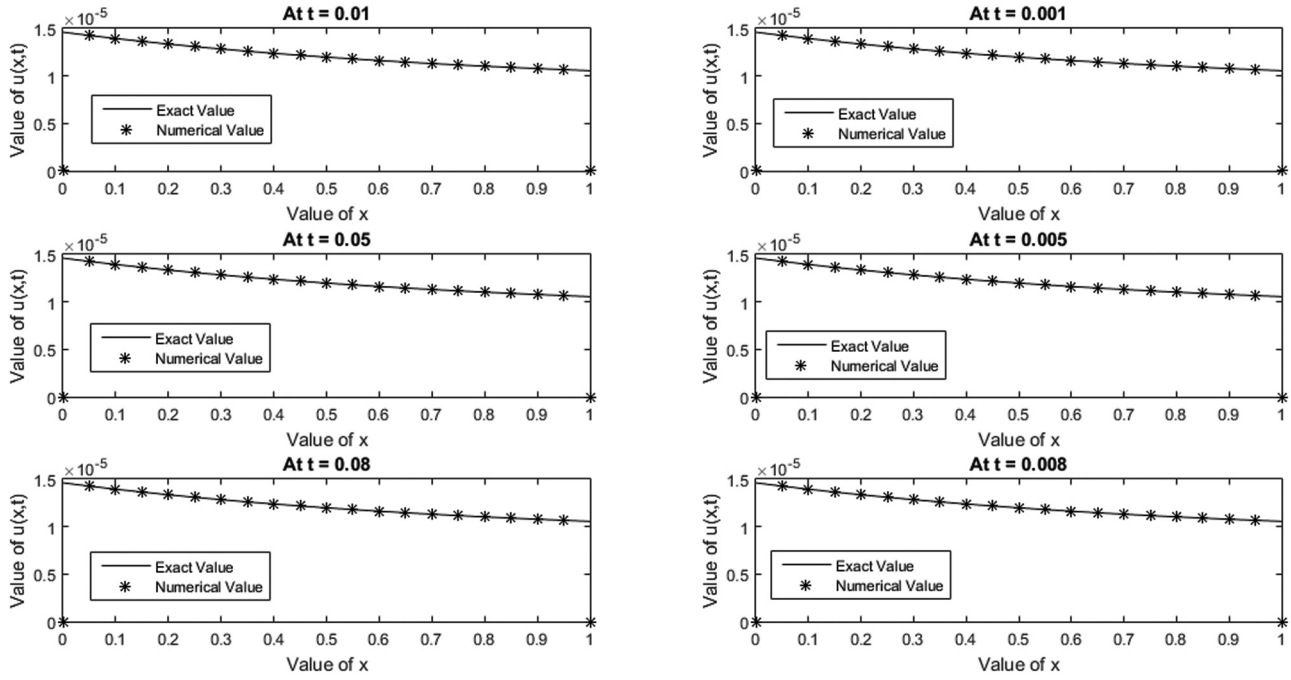

Figure 4: Pictorial depiction of solutions for $N = 21$, $\Delta t = 0.01$, $v = 10^{-5}$, $c = 10^{-5}$, $\mu = 2$, and $K = 1$.

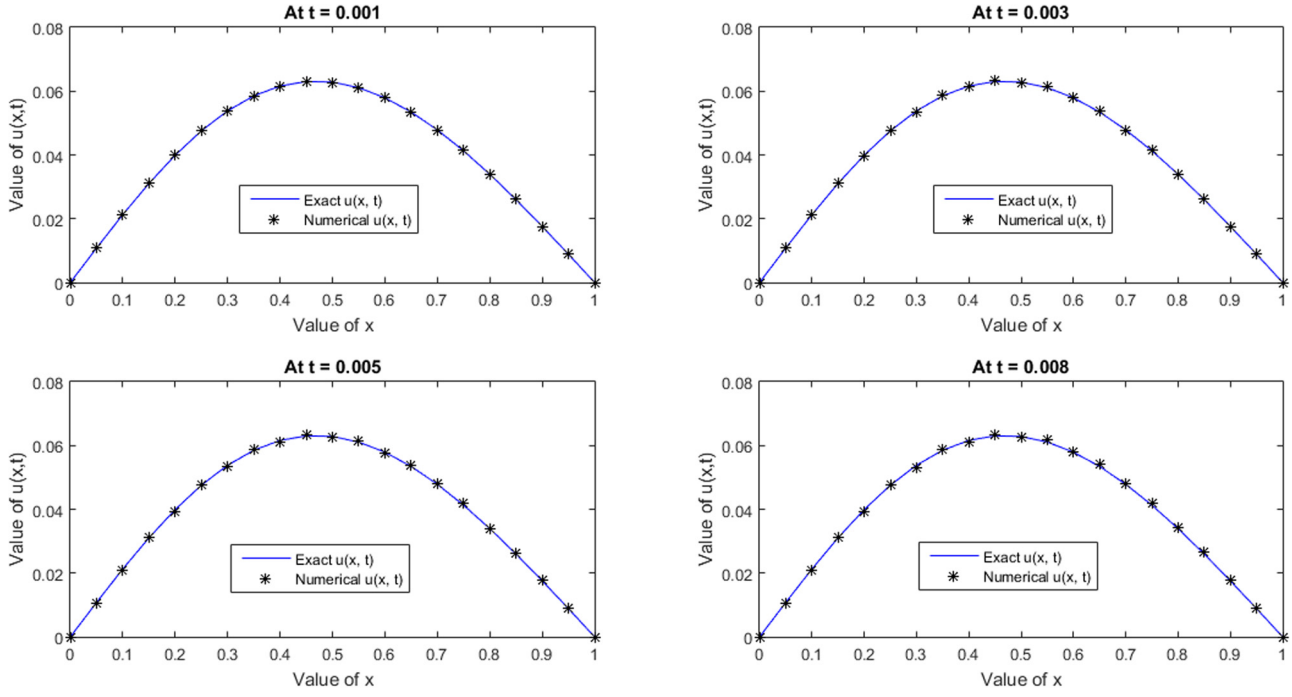

Figure 5: Graphical presentation of solutions for $N = 31$, $t = 0.0001$, $\alpha = 10$, and $v = 0.001$.

Initial condition:

$u(x, 0) = 4x(1 - x)$.

Boundary conditions: $u(0, t) = 0$, $u(1, t) = 0$.

In Figure 8, numerical solutions are provided at a series of time levels with $\Delta t = 0.0001$, $\alpha = 1$, and $v = 0.01$.

Example 8. I.C. and B.C.s are taken as per [9, 12].

Initial condition:

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

Boundary conditions:

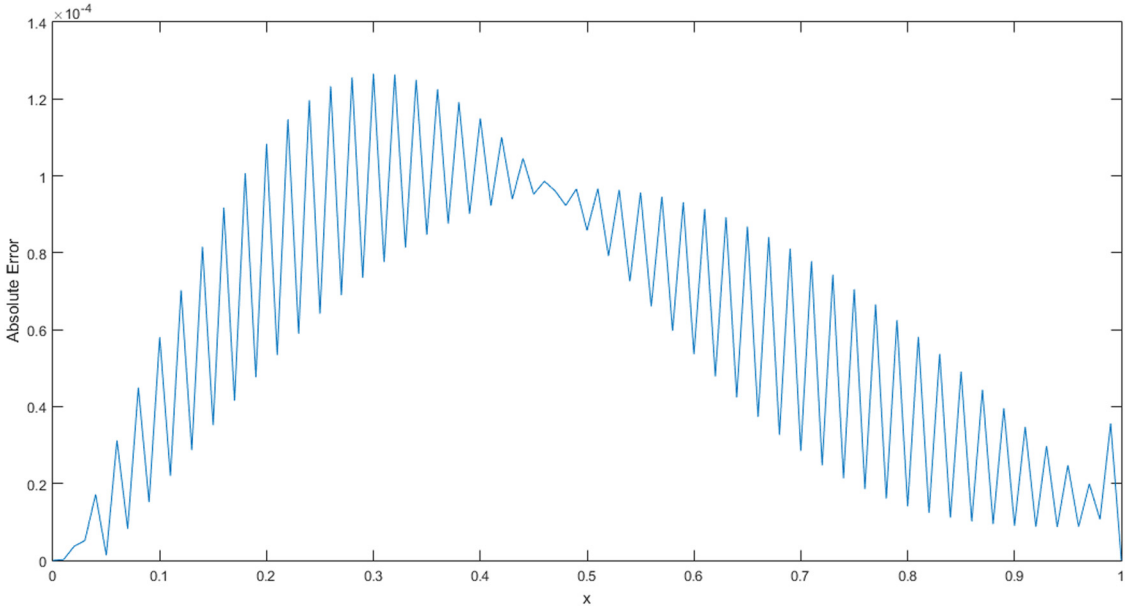

Figure 6: Absolute error at $t = 0.0005$, $\nu = 0.001$, $\alpha = 10$, $N = 101$, and $\Delta t = 0.0001$.

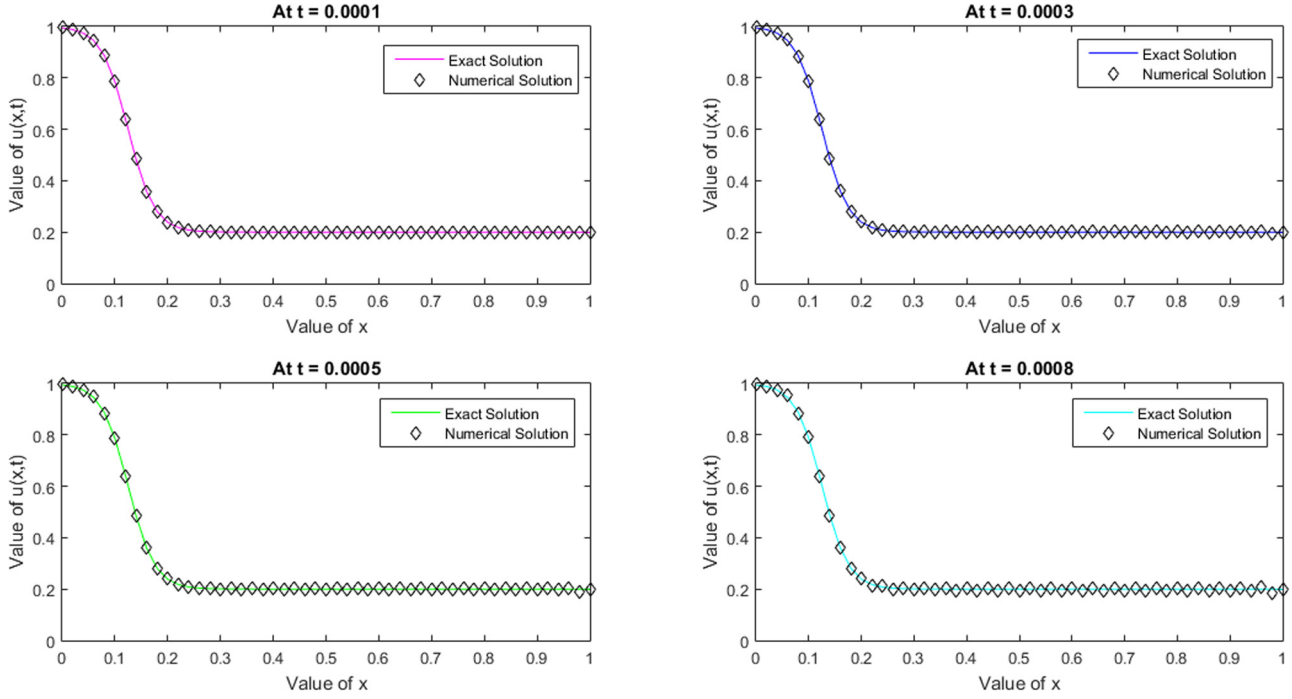

Figure 7: Plot for solutions for $N = 101$, $\Delta t = 0.0001$, $\alpha = 0.4$, $\nu = 0.01$, $\beta = 0.125$, and $\mu = 0.6$.

$u(a, t) = 0$, $u(b, t) = 0$, where $a = -1$ and $b = 1$.

In Figure 9, numerical approximations are provided at $t = 0.001$, 0.003, 0.05, and 0.008, respectively, with $\Delta t = 0.0001$, $\alpha = 1$, and $\nu = 0.01$.

Example 9. In the present example, exact solution is taken from [49]. Considered 2D Burgers' equation is expressed as follows:

Table 9: Comparability of solutions at $t = 0.0001$, $t = 0.0005$, and $t = 0.0008$, respectively, for $\Delta t = 0.0001$, $\alpha = 0.4$, $v = 0.01$, $\beta = 0.125$, $\mu = 0.6$

x	Exact	Numerical	Exact	Numerical	Exact	Numerical
	$t = 0.0001$		$t = 0.0005$		$t = 0.0008$	
1.00×10^{-1}	7.85×10^{-1}	7.85×10^{-1}	7.87×10^{-1}	7.85×10^{-1}	7.88×10^{-1}	7.85×10^{-1}
2.00×10^{-1}	2.38×10^{-1}	2.38×10^{-1}	2.38×10^{-1}	2.38×10^{-1}	2.39×10^{-1}	2.38×10^{-1}
3.00×10^{-1}	2.01×10^{-1}	2.01×10^{-1}	2.01×10^{-1}	2.01×10^{-1}	2.01×10^{-1}	2.01×10^{-1}

$$u_t = -uu_x - uu_y + \frac{1}{\mathrm{Re}}[u_{xx} + u_{yy}], \tag{32}$$

where $(x, y) \in [0, 1] \times [0, 1]$.

Exact solution:

$$u(x, y, t) = \frac{1}{1 + \exp(\frac{\mathrm{Re}(x+y-t)}{2})}. \tag{33}$$

Initial condition:

$$u(x, y, 0) = \frac{1}{1 + \exp\left(\frac{\mathrm{Re}(x+y)}{2}\right)}. \tag{34}$$

Boundary conditions:

$$u(0, y, t) = \frac{1}{1 + \exp\left(\frac{\mathrm{Re}(y-t)}{2}\right)}, \tag{35}$$

$$u(1, y, t) = \frac{1}{1 + \exp\left(\frac{\mathrm{Re}(1+y-t)}{2}\right)}, \tag{36}$$

$$u(x, 0, t) = \frac{1}{1 + \exp\left(\frac{\mathrm{Re}(x-t)}{2}\right)}, \tag{37}$$

$$u(x, 1, t) = \frac{1}{1 + \exp\left(\frac{\mathrm{Re}(x+1-t)}{2}\right)}. \tag{38}$$

In Figure 10, obtained and exact solutions are matched at $t = 0.001$ and $t = 0.005$, respectively, with $\Delta t = 0.0001$ and Re = 10. In Figure 11, a good match of solutions is obtained at $t = 0.0001$ for Re = 2 and 3, respectively. In Tables 10 and 11, error norms are provided for Re = 100 and Re = 200, respectively, at different values of t. In Table 12, relative and RMS error norms are discussed at $t = 0.0001$ and $t = 0.0005$, respectively, at mentioned grid points with Re = 10 for Example 9.

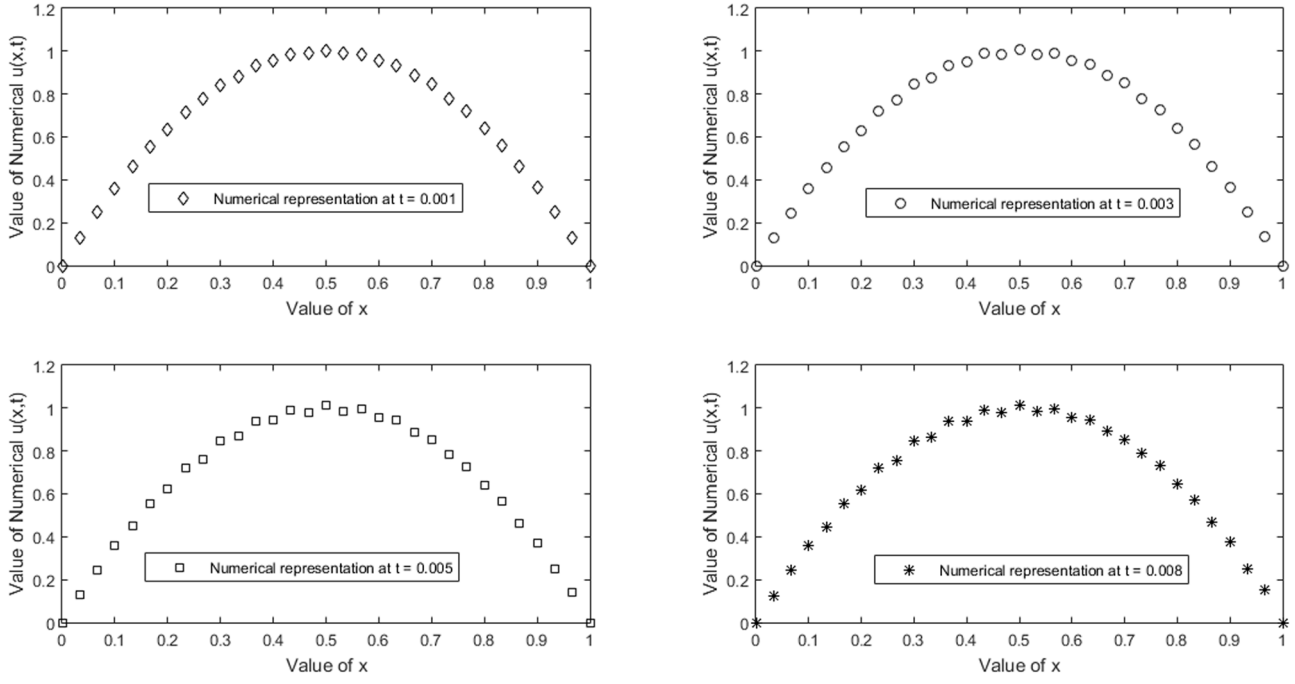

Figure 8: Graphical representation of numerical solution, where $N = 101$, $\Delta t = 0.0001$, $\alpha = 1$, and $v = 0.01$.

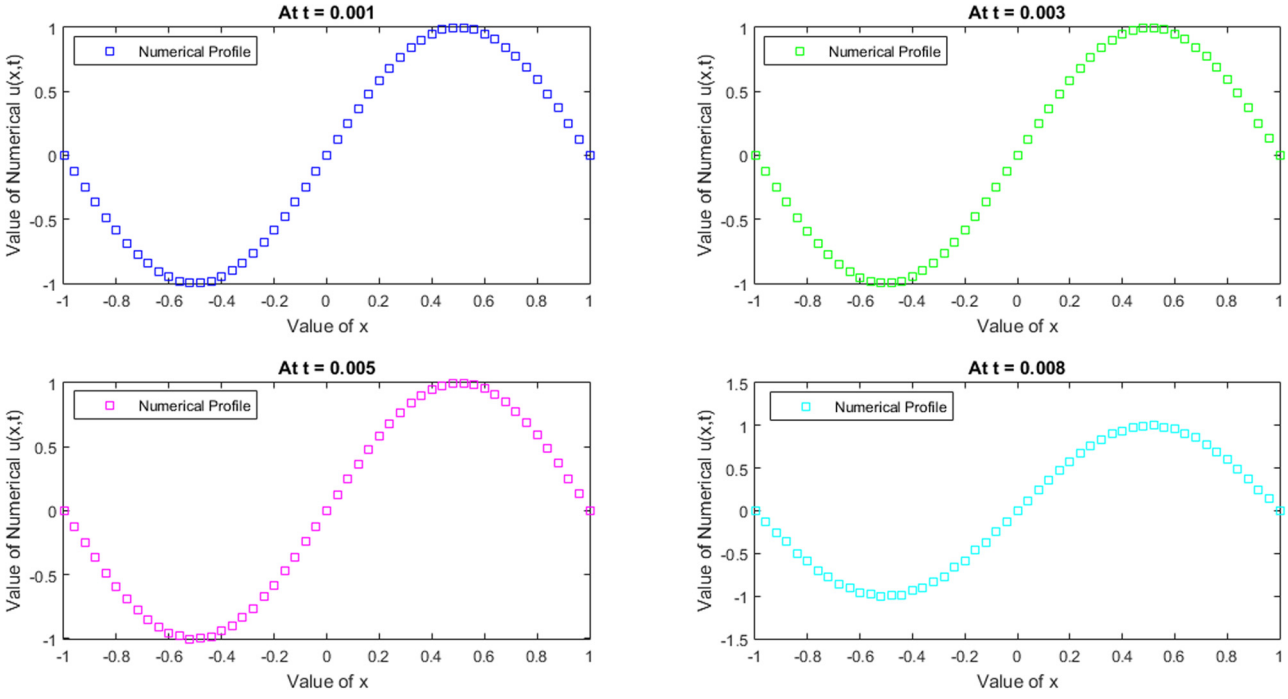

Figure 9: Numerical solutions for $N = 101$, $\Delta t = 0.0001$, $\alpha = 1$, and $\nu = 0.01$.

Example 10. In present example, considered exact solution is taken from [49]. Considered equation in 2D is

$$u_t = -uu_x - uu_y + \frac{1}{\mathrm{Re}}[u_{xx} + u_{yy}], \tag{39}$$

where $(x, y) \in [-0.5, 0.5] \times [-0.5, 0.5]$.

Exact solution:

$$u(x, y, t) = 0.5 - \tanh\left[\frac{\mathrm{Re}(x + y - t)}{2}\right]. \tag{40}$$

Initial condition:

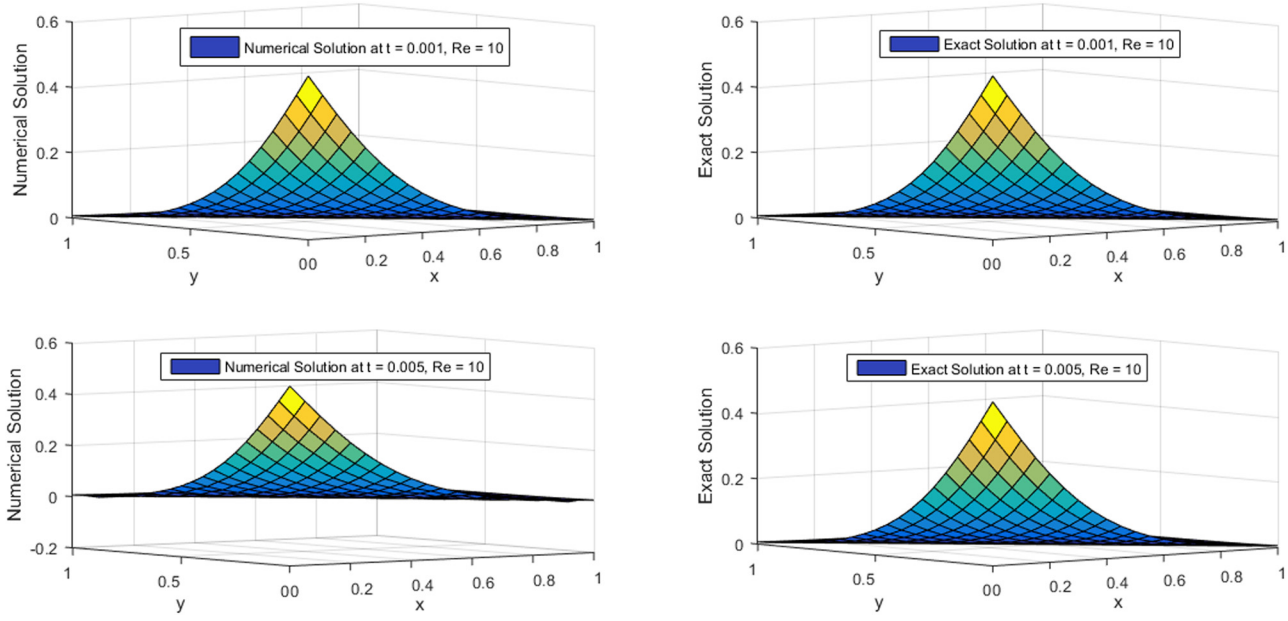

Figure 10: Graphical presentation of solutions at $t = 0.001$ and $t = 0.005$, respectively, for $N = 21$, $\Delta t = 0.0001$ with $\mathrm{Re} = 10$.

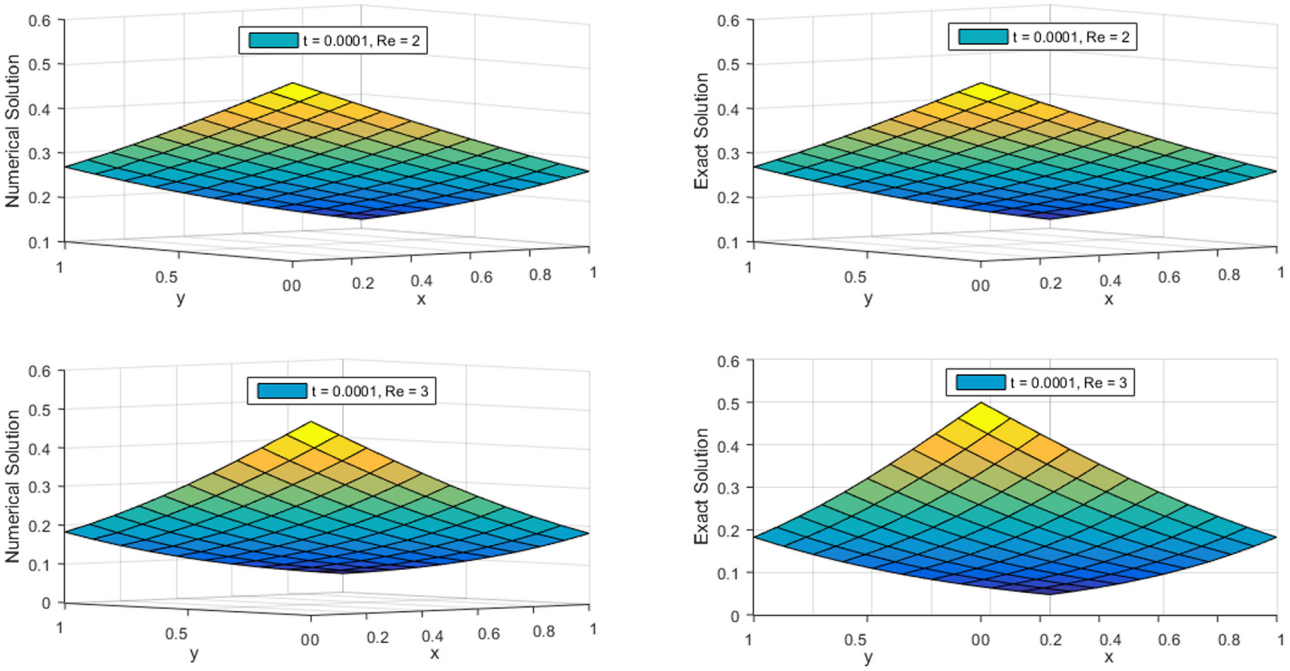

Figure 11: Graphical plot at $t = 0.0001$, for Re = 2 and 3 for $N = 51$, $\Delta t = 0.0001$.

Table 10: Error norms

	Re = 100, t = 0.001		Re = 100, t = 0.0001	
$N \times N$	L_2	L_∞	L_2	L_∞
10 × 10	5.3132×10^{-7}	4.0958×10^{-6}	1.0140×10^{-7}	7.9055×10^{-7}
20 × 20	8.5025×10^{-6}	6.3644×10^{-5}	2.7045×10^{-6}	3.2442×10^{-5}
30 × 30	1.4638×10^{-5}	2.0526×10^{-4}	4.9755×10^{-6}	8.5671×10^{-5}

Table 11: Error norms for different Reynolds number

	Re = 200, t = 0.0001		Re = 200, t = 0.0005	
$N \times N$	L_2	L_∞	L_2	L_∞
10 × 10	2.1485×10^{-10}	1.6963×10^{-9}	6.5657×10^{-10}	5.1837×10^{-9}
20 × 20	1.6090×10^{-7}	2.3318×10^{-6}	4.6294×10^{-7}	6.5750×10^{-6}
30 × 30	1.3658×10^{-6}	2.4035×10^{-5}	3.4420×10^{-6}	5.2652×10^{-5}

Table 12: Relative and RMS errors at $t = 0.0001$ and $t = 0.0005$, respectively, with Re = 10 regarding Example 9

N	t = 0.0001		t = 0.0005	
	Relative error	RMS error	Relative error	RMS error
10	1.3308×10^{-4}	2.7973×10^{-4}	4.0676×10^{-4}	8.2798×10^{-4}
20	1.4416×10^{-3}	2.3694×10^{-3}	4.2987×10^{-3}	7.2092×10^{-3}
30	4.9392×10^{-3}	6.8693×10^{-3}	1.5385×10^{-2}	2.3181×10^{-2}

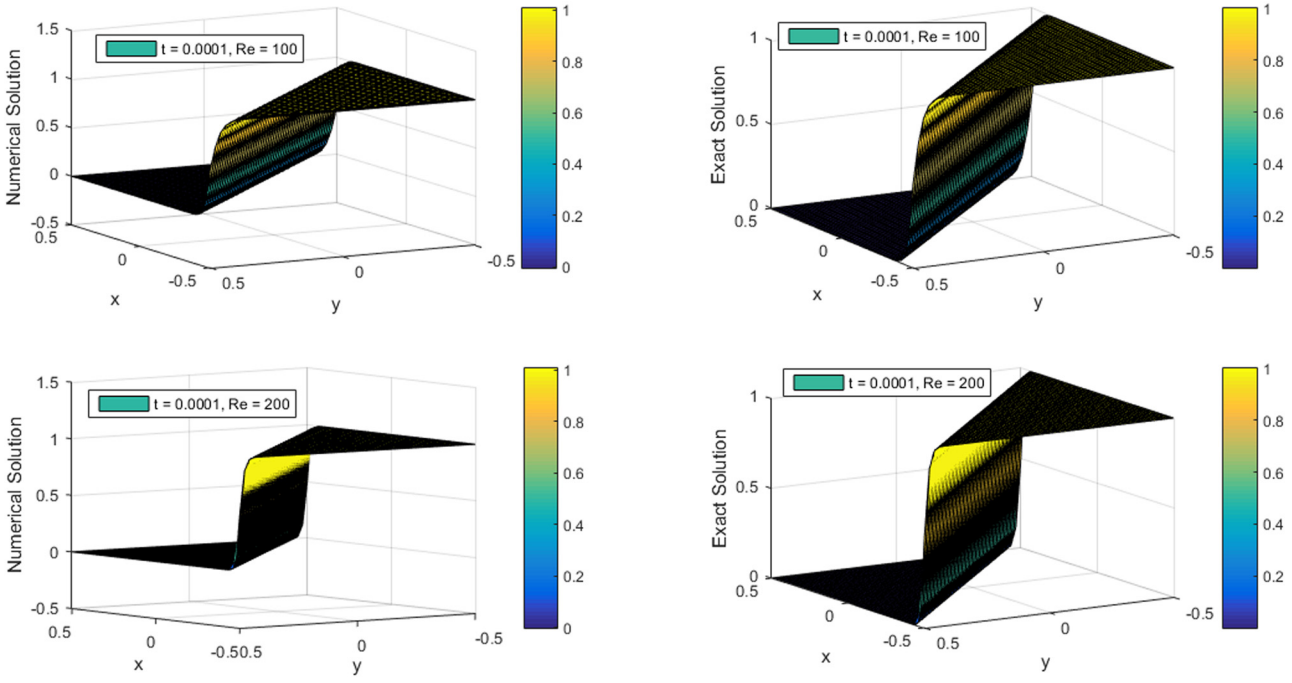

Figure 12: Comparability of fetched and exact solutions at $t = 0.0001$ for Re = 100 and 200.

$$u(x, y, 0) = 0.5 - \tanh\left[\frac{\mathrm{Re}(x + y)}{2}\right]. \quad (41)$$

Boundary conditions:

$$u(-0.5, y, t) = 0.5 - \tanh\left[\frac{\mathrm{Re}(-0.5 + y - t)}{2}\right], \quad (42)$$

$$u(0.5, y, t) = 0.5 - \tanh\left[\frac{\mathrm{Re}(0.5 + y - t)}{2}\right], \quad (43)$$

$$u(x, -0.5, t) = 0.5 - \tanh\left[\frac{\mathrm{Re}(x - 0.5 - t)}{2}\right], \quad (44)$$

$$u(x, 0.5, t) = 0.5 - \tanh\left[\frac{\mathrm{Re}(x + 0.5 - t)}{2}\right]. \quad (45)$$

In Figure 12, approximations are matched for Re = 100 and 200, respectively, at $t = 0.0001$. In Table 13, error norms are provided for Re = 10 and Re = 20, $t = 0.0001$, and $t = 0.0005$. In Table 14, relative and RMS error norms are discussed at $t = 0.0001$ and $t = 0.0005$, respectively, at mentioned grid points with Re = 200 for Example 10.

5 Graphical and tabular interpretation

The main analysis of this research lies in the graphical and tabular interpretation, which claims the robustness and the validity of the proposed regime. In Table 2, errors are fetched where it is notified that on decreasing the values of the coefficient of viscosity, the errors will be reduced, and it means that the convergence of the regime will occur. Table 3 shows that the obtained errors are

Table 13: Error norms at $t = 0.0001$ and $t = 0.0005$ for Re = 10 and 20, respectively, for $N = 101$, $\Delta t = 0.00001$

	Re = 10, t = 0.0001		Re = 20, t = 0.0005	
$N \times N$	L_2	L_∞	L_2	L_∞
11 × 11	9.2928×10^{-4}	1.0254×10^{-3}	4.2718×10^{-3}	4.7124×10^{-3}
21 × 21	4.1567×10^{-3}	3.9772×10^{-3}	1.8789×10^{-2}	1.9198×10^{-2}
31 × 31	1.0591×10^{-2}	9.6674×10^{-3}	4.8486×10^{-2}	5.4268×10^{-2}

Table 14: Relative and RMS errors at $t = 0.0001$ and $t = 0.0005$, respectively, with Re = 200 regarding Example 10

N	$t = 0.0001$		$t = 0.0005$	
	Relative error	**RMS error**	**Relative error**	**RMS error**
11	3.7093×10^{-4}	7.0978×10^{-4}	1.8013×10^{-3}	3.4516×10^{-3}
21	6.7542×10^{-4}	1.7867×10^{-3}	3.0869×10^{-3}	8.1721×10^{-3}
31	1.0436×10^{-3}	3.3544×10^{-3}	4.6568×10^{-3}	1.4977×10^{-2}

much better that [12,44,49] at the different values of viscosity as well as the reduced errors are claim for the numerical convergence. *Via* Table 3, L_2 and L_∞ errors are far better than [12,44,49] for different ν values. In this aspect also, convergence is claimed with the reduction of errors. Table 7 shows that the present errors are notified as much better than that shown in [12,51] at the diversified time levels. In Table 6, as well, the present errors are reduced at some parameters as compared to [12]. Table 9 shows that the compatibility of the exact and numerical results is reported.

Validating the compatibility of the results *via* the graphs is an important tool in the numerical analysis. Keeping the same aspect in mind, the exact and approximate results are matched for a wide range of the time levels and at a set of different parameters. In Figure 1, results are matched for a wide range of time levels such as; $t = 0.001$ to 0.008. Figure 2 shows that exact and numerical results are matched at $t = 0.001$, 0.003, 0.005, and 0.008. Graphical compatibility of the results is notified *via* Figure 3 as well, for a wide range of time levels. Pictorial depiction of the compatibility of the results is claimed in Figures 4 and 5.

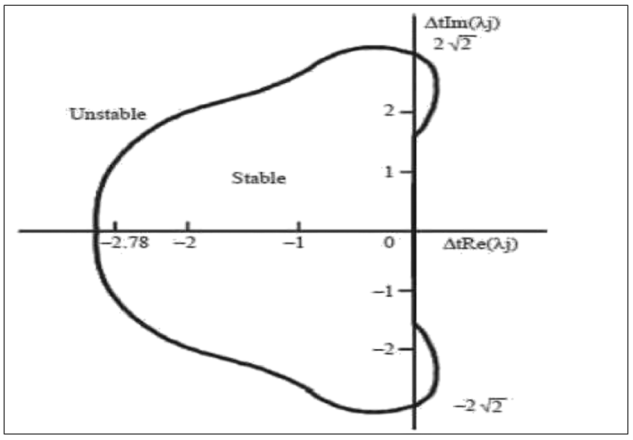

Figure 14: Stability region.

In Figure 6, absolute error is reported. It is notified that at different values of x in [0,1], first, the absolute error increased in some fixed pattern but after approaching at $x = 0.5$, the absolute error got decreased in the fixed pattern. In Figure 7, graphical match of the results is created. In Figures 8 and 9, only numerical results are drawn, as the exact solution is not provided in the associated

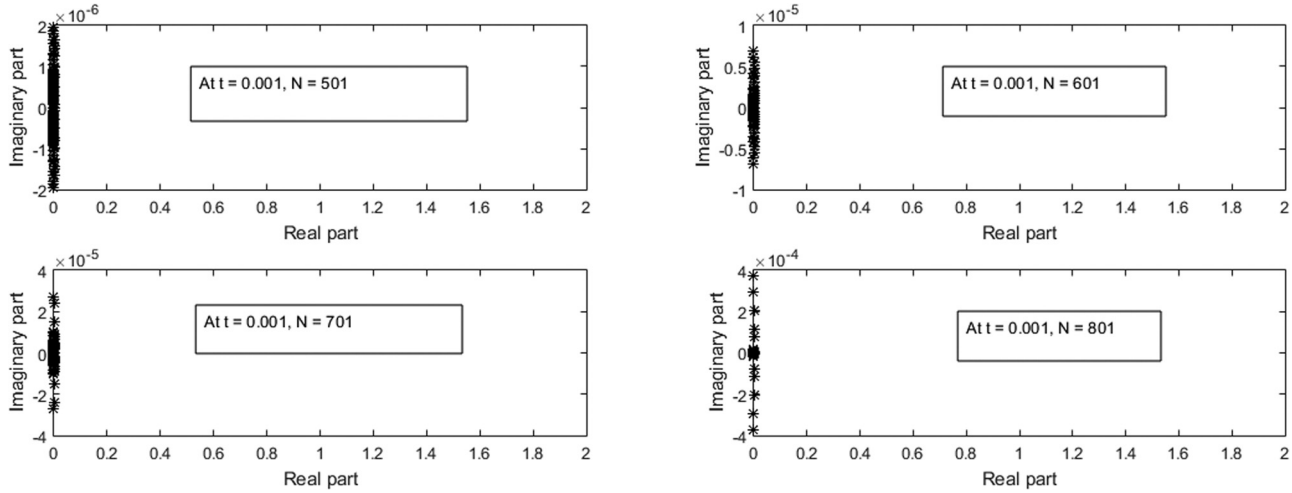

Figure 13: Stability of developed regime.

example. In Figures 10–12, 3D graphs are notified at the different values of t *via*, which the compatibility of the results is claimed.

6 Stability by matrix stability analysis method

For investigating stability of novel scheme, the given PDEs are converted into system of ODE, and thereafter, proposed scheme's stability is checked by using Matrix method [49,57]. By using the approximated values of partial derivatives in 1D and 2D nonlinear Burgers' equations and along with considering the nonlinear term as constant, the following system of equations will be followed.

$$\frac{\mathrm{d}U_i}{\mathrm{d}t} = -\alpha_i U_i \sum_{j=1}^{n} a_{ij}^{(1)} U(x_j) + \nu \sum_{j=1}^{n} a_{ij}^{(2)} U(x_j),$$

$$\frac{\mathrm{d}U_{i,j}}{\mathrm{d}t} = -U_{i,j} \sum_{j=1}^{n} a_{ij} U(x_i, y_j) - U_{i,j} \sum_{j=1}^{n} a'_{ij} U(x_i, y_j) + \frac{1}{\mathrm{Re}} \left[\sum_{j=1}^{n} b_{ij} U(x_i, y_j) + \sum_{j=1}^{n} b'_{ij} U(x_i, y_j) \right].$$

$$\frac{\mathrm{d}U(x_i, t)}{\mathrm{d}t} = AU + F[U(x, t)],$$

$$\frac{\mathrm{d}U(x_i, y_j, t)}{\mathrm{d}t} = BU + G[U(x, y, t)],$$

where $F[U(x, t)]$ and $G[U(x, y, t)]$ are the associated nonlinear terms present in 1D nd 2D nonlinear Burgers' equation, respectively. In the system developed for 1D Burgers' equation, stability rely upon eigen values of matrix A.

$$A = \nu \sum_{j=1}^{n} a_{ij}^{(2)}.$$

In developed system for 2D Burgers' equation, stability depends upon the eigen values of matrix B.

$$B = \frac{1}{\mathrm{Re}} \left[\sum_{j=1}^{n} b_{ij} + \sum_{j=1}^{n} b'_{ij} \right].$$

The system of equations is said to be stable as pre the following conditions:

(i) **Real** λ: $-2.78 < \Delta t\lambda < 0$,
(ii) **Imaginary** λ: $-2\sqrt{2} < \Delta t\lambda < 2\sqrt{2}$,
(iii) **Complex** λ: $\Delta t\lambda$ lie within region R as per given figure.

Stability of the proposed scheme is discussed in Figure 13 for Examples 1. Figure 14 represents the stability criteria to be followed.

7 Conclusion

To solve complex natured PDEs with a nonlinear nature, a novel regime named MQHB-spline DQM is established in the current study. In this regime, weighting coefficients and associated derivatives are computed using MQHB-spline as basis function. SSP-RK43 strategy is used to deal with obtained ODE system. Ten numerical examples are examined to demonstrate effectiveness and efficacy of suggested approach. Error norms (L_2 and L_∞) are mostly taken into consideration when comparing with previous work and exact results. It is clear from comparisons to results in literature that suggested regime yields superior and tolerable outcomes. This method is practical for use and economical in terms of complexity of data. To obtain a more accurate numerical approximation of complex-natured PDEs, developed regime will be useful.

This study's primary goal is to create a unique regime that delivers better results with fewer errors. There are numerous partial differential equations that cannot be solved analytically, including fractional equations, integro differential equations, partial integro differential equations, and many others. As a result, creating new, effective numerical regimes is currently necessary.

Funding information: The author states no funding involved.

Author contributions: The author has accepted responsibility for the entire content of this manuscript and approved its submission.

Conflict of interest: Not applicable.

Data availability statement: All data is included within the manuscript.

References

[1] Cole JD. On a quasi-linear parabolic equation occurring in aerodynamics. Q Appl Math. 1951;9(3):225–36.
[2] Fletcher CA. Generating exact solutions of the two-dimensional Burgers' equations. IJNMF. 1983;3:213–6.
[3] Kutluay S, Esen A, Dag I. Numerical solutions of the Burgers' equation by the least-squares quadratic B-spline finite element method. J Comput Appl Math. 2004;167(1):21–33.
[4] Özis T, Esen A, Kutluay S. Numerical solution of Burgers' equation by quadratic B-spline finite elements. Appl Math Comput. 2005;165(1):237–49.
[5] Saka B, Dağ İ. Quartic B-spline collocation method to the numerical solutions of the Burgers' equation. Chaos Solitons Fractals. 2007;32(3):1125–37.

[6] Dag I, Irk D, Sahin A. B-spline collocation methods for numerical solutions of the Burgers' equation. Math Problem Eng. 2005;2005:521–38.
[7] Korkmaz A, Aksoy AM, Dag I. Quartic B-spline differential quadrature method. Int J Nonlinear Sci. 2011;11(4):403–11.
[8] Xie SS, Heo S, Kim S, Woo G, Yi S. Numerical solution of one-dimensional Burgers' equation using reproducing kernel function. J Comput Appl Math. 2008;214(2):417–34.
[9] Korkmaz A, Dağ İ. II. Shock wave simulations using sinc differential quadrature method. Eng Comput. 2011;28(6):654–74.
[10] Korkmaz A, Dag I. Polynomial based differential quadrature method for numerical solution of nonlinear Burgers' equation. J Franklin Inst. 2011;348(10):2863–75.
[11] Korkmaz A, Dağ İ. Cubic B-spline differential quadrature methods for the advection-diffusion equation. Int J Numer Meth Heat Fluid Flow. 2012;22(8):1021–36.
[12] Mittal R, Jain R. Numerical solutions of nonlinear Burgers' equation with modified cubic B-splines collocation method. Appl Math Comput. 2012;218(15):7839–55.
[13] Kumar V, Singh S, Koksal ME. A composite algorithm for numerical solutions of two-dimensional coupled Burgers' equations. J Math. 2021;2021:1–13.
[14] Kumar V, Kaur L, Kumar A, Koksal ME. Lie symmetry based-analytical and numerical approach for modified Burgers-KdV equation. Results Phys. 2018;8:1136–42.
[15] Hanacc Duruk E, Koksal ME, Jiwari R. Analyzing similarity solution of modified fisher equation. J Math. 2022;2022:1–9.
[16] Bellman R, Kashef B, Casti J. Differential quadrature: a technique for the rapid solution of nonlinear partial differential equations. J Comput Phys. 1972;10(1):40–52.
[17] Quan J, Chang C. New insights in solving distributed system equations by the quadrature method-I. Analysis. Comput Chem Eng. 1989;13(7):779–88.
[18] Quan J, Chang CT. New insights in solving distributed system equations by the quadrature method-II. Numerical experiments. Comput Chem Eng. 1989;13(9):1017–24.
[19] Shu C. Differential quadrature and its application in engineering. London, UK: Springer; 2012.
[20] Feng Y, Bert C. Application of the quadrature method to flexural vibration analysis of a geometrically nonlinear beam. Nonlinear Dyn. 1992;3(1):13–8.
[21] Korkmaz A, Dağ İ. Numerical simulations of boundary-forced RLW equation with cubic B-spline-based differential quadrature methods. Arabian J Sci Eng. 2013;38(5):1151–60.
[22] Mittal R, Jiwari R. Differential quadrature method for two-dimensional Burgers' equations. Int J Comput Meth Eng Sci Mech. 2009;10(6):450–9.
[23] Yusuf U, Yağmurlu M, Başhan A. Numerical solutions and stability analysis of modified Burgers equation via modified cubic B-spline differential quadrature methods. Sigma J Eng Natural Sci. 2019;37(1):129–42.
[24] Tamsir M, Dhiman N. DQM based on the modified form of CTB shape functions for coupled Burgers' equation in 2D and 3D. Int J Math Eng Manag Sci. 2019;4(4):1051.
[25] Hashmi MS, Wajiha M, Yao SW, Ghaffar A, Inc M. Cubic spline based differential quadrature method: a numerical approach for fractional Burger equation. Results Phys. 2021;26:104415.
[26] Başhan A. An efficient approximation to numerical solutions for the Kawahara equation via modified cubic B-spline differential quadrature method. Mediterranean J Math. 2019;16(1):1–19.
[27] Msmali A, Tamsir M, Ahmadini AAH. Crank-Nicolson-DQM based on cubic exponential B-splines for the approximation of nonlinear Sine-Gordon equation. Ain Shams Eng J. 2021;12(4):4091–7.
[28] Başhan A. Highly efficient approach to numerical solutions of two different forms of the modified Kawahara equation via contribution of two effective methods. Math Comput Simul. 2021;179:111–25.
[29] Aziz AA, Ngarisan NS, Baki NA. Solution of finite difference method and differential quadrature method in Burgers equation. J Ocean Mech Aerospace-Sci Eng. 2019;63(3):1–4.
[30] Tamsir M, Dhiman N, Gill FS, Robin S. Approximation of 3D convection diffusion equation using DQM based on modified cubic trigonometric B-splines. J Comput Meth Sci Eng. 2020;20(4):1357–66.
[31] Arora G, Joshi V, Mittal R. A spline-based differential quadrature approach to solve sine-gordon equation in one and two dimension. Fractals. 2022;30(7):2250153.
[32] Mohamed NA, Rashed AS, Melaibari A, Sedighi HM, Eltaher MA. Effective numerical technique applied for Burgers' equation of (1+1)-, (2+1)-dimensional, and coupled forms. Math Meth Appl Sci. 2021;44(13):10135–53.
[33] Rasoulizadeh MN, Rashidinia J. Numerical solution for the Kawahara equation using local RBF-FD meshless method. J King Saud Univ-Sci. 2020;32(4):2277–83.
[34] Hussain M. Hybrid radial basis function methods of lines for the numerical solution of viscous Burgers' equation. Comput Appl Math. 2021;40(4):1–49.
[35] Tamsir M, Huntul M. A numerical approach for solving Fisheras reaction-diffusion equation via a new kind of spline functions. Ain Shams Eng J. 2021;12(3):3157–65.
[36] Izadi F, Saberi Najafi H, Refahi Sheikhani A. The numerical solution of Fisher equation: a nonstandard finite difference in conjunction with Richtmyer formula. Comput Meth Differ Equ. 2020;8(2):330–46.
[37] Başhan A, Karakoç SBG, Geyikli T. B-spline differential quadrature method for the modified Burgers' equation. Cankaya Univ J Sci Eng. 2015;12(1):1–13.
[38] Bassshan A, Karakoc SBG, Geyikli T. Approximation of the KdVB equation by the quintic B-spline differential quadrature method. Kuwait J Sci. 2015;42(2):67–92.
[39] Geyikli T, Karakoç SG. Subdomain finite element method with quartic B-splines for the modified equal width wave equation. Comput Math Math Phys. 2015;55(3):410–21.
[40] Karakoc SBG, Geyikli T, Bashan A. A numerical solution of the modified regularized long wave MRLW equation using quartic B-splines. TWMS J Appl Eng Math. 2013;3(2):231–44.
[41] Bhowmik SK, Karakoc SB. Numerical approximation of the generalized regularized long wave equation using Petrov-Galerkin finite element method. Numer Meth Partial Differ Equ. 2019;35(6):2236–57.
[42] Gazi Karakoc SB, Geyikli T. Petrov-Galerkin finite element method for solving the MRLW equation. Math Sci. 2013;7(1):1–10.
[43] Karakoç SBG, Başhan A, Geyikli T. Two different methods for numerical solution of the modified Burgers' equation. Scientif World J. 2014;2014.

[44] Arora G, Singh BK. Numerical solution of Burgers' equation with modified cubic B-spline differential quadrature method. Appl Math Comput. 2013;224:166–77.
[45] Kapoor M, Joshi V. Solution of non-linear Fisher's reaction-diffusion equation by using Hyperbolic B-spline based differential quadrature method. J Phys Conf Ser. 2020;1531:012064.
[46] Kapoor M, Joshi V. Numerical regime Uniform Algebraic Hyperbolic tension B-spline DQM for the solution of Fisheras Reaction-Diffusion equation. 3rd International Conference on Applied Research in Engineering, Science and Technology; 2020 Nov 20–22; Paris, France. Diamond Scientific Publishing, 2020. p. 44–63.
[47] Mittal R, Kumar S. Numerical study of Fisher's equation by wavelet Galerkin method. Int J Comput Math. 2006;83(3):287–98.
[48] Spiteri RJ, Ruuth SJ. A new class of optimal high-order strong-stability-preserving time discretization methods. SIAM J Numer Anal. 2002;40(2):469–91.
[49] Arora G, Joshi V. A computational approach using modified trigonometric cubic B-spline for numerical solution of Burgers' equation in one and two dimensions. Alexandr Eng J. 2018;57(2):1087–98.
[50] Tamsir M, Srivastava VK, Jiwari R. An algorithm based on exponential modified cubic B-spline differential quadrature method for nonlinear Burgers' equation. Appl Math Comput. 2016;290:111–24.
[51] Raslan K. A collocation solution for Burgers' equation using quadratic B-spline finite elements. Int J Comput Math. 2003;80(7):931–8.
[52] Asaithambi A. Numerical solution of the Burgers' equation by automatic differentiation. Appl Math Comput. 2010;216(9):2700–8.
[53] Rahman K, Helil N, Yimin R. Some new semi-implicit finite difference schemes for numerical solution of Burgers' equation. 2010 International Conference on Computer Application and System Modeling (ICCASM 2010); 2010 Oct 22–24; Taiyuan, China. IEEE, 2010. p. 451–55.
[54] Dogan A. A Galerkin finite element approach to Burgers' equation. Appl Math Comput. 2004;157(2):331–46.
[55] Xu M, Wang RH, Zhang JH, Fang Q. A novel numerical scheme for solving Burgers' equation. Appl Math Comput. 2011;217(9):4473–82.
[56] Aksan E. Quadratic B-spline finite element method for numerical solution of the Burgers' equation. Appl Math Comput. 2006;174(2):884–96.
[57] Mittal R, Dahiya S. Numerical simulation of three-dimensional telegraphic equation using cubic B-spline differential quadrature method. Appl Math Comput. 2017;313:442–52.