A differential quadrature method for numerical solutions of Burgers’-type equations

R.C. Mittal and Ram Jiwari

Department of Mathematics, Indian Institute of Technology Roorkee,
Roorkee, India

Abstract

Purpose – The purpose of this paper is to use the polynomial differential quadrature method (PDQM) to find the numerical solutions of some Burgers’-type nonlinear partial differential equations.

Design/methodology/approach – The PDQM changed the nonlinear partial differential equations into a system of nonlinear ordinary differential equations (ODEs). The obtained system of ODEs is solved by Runge-Kutta fourth order method.

Findings – Numerical results for the nonlinear evolution equations such as 1D Burgers’, coupled Burgers’, 2D Burgers’ and system of 2D Burgers’ equations are obtained by applying PDQM. The numerical results are found to be in good agreement with the exact solutions.

Originality/value – A comparison is made with those which are already available in the literature and the present numerical schemes are found give better solutions. The strong point of these schemes is that they are easy to apply, even in two-dimensional nonlinear problems.

Keywords Mathematics, Differential equations, 1D Burgers’ equation, Coupled Burgers’ equations, 2D Burgers’ equation, Differential quadrature method, Runge-Kutta method

Paper type Research paper

1. Introduction

The vast majorities of nonlinear partial differential equations (PDEs) arise in many fields of science, particularly in physics, engineering, chemistry and finance, and are fundamental for the mathematical formulation of continuum models. Systems of nonlinear PDEs have attracted much attention in studying evolution equations describing wave propagation. One such type of PDEs is Burgers’-type equations. The Burgers’-type equations are encountered in many fields such as the theory of shock waves, mathematical modeling of turbulent fluid and in continuous stochastic processes. The well known Burgers’ equation was first introduced by Bateman (1915) and he proposed the steady-state solution of the problem. In 1948, Burgers (1939, 1948) introduced this equation to capture some features of turbulent fluid in a channel caused by the interaction of the opposite effects of convection and diffusion, therefore it is popularly referred as “Burgers’ equation”. The structure of Burgers’ equation is roughly similar to that of Navier-Stoke’s equations due to the presence of the non-linear convection term and the occurrence of the diffusion term with viscosity coefficient. So this equation can be considered as a simplified form of the Navier-Stoke’s equations. The study of the general properties of the Burgers’ equation has attracted attention of

The authors gratefully acknowledge the valuable suggestions and observations of the anonymous referees to improve the quality of the paper.
scientific community due to its applications in the various fields such as gas dynamics, heat conduction, elasticity, etc.

The study of the solution of Burgers’ equation has been carried out for last half century and still it is an active area of research to develop some better numerical scheme to approximate its solution. Many researchers have used various methods to seek the numerical solutions of 1D Burgers’ equation (Mittal and Singhal, 1993; Kutluay et al., 1999; Ozis et al., 2003; Kadalbajoo and Awasthi, 2006). Similarly, the coupled and 2D Burgers’ equations are solved by many researchers with different numerical methods. Kaya (2001) presented the solution of the homogenous and inhomogeneous coupled Burgers’ equations in the form of convergent power series using decomposition method. Khater et al. (2008) proposed the Chebyshev spectral collocation method for solving the coupled Burgers’ equations. Dehghan et al. (2007) used the adomain-Pade technique to find the numerical results of coupled Burgers’ equations. Rashid and Ismail (2009) proposed a Fourier pseudospectral method for solving coupled viscous Burgers’ equations. Goyon (1996) proposed several multilevel ADI schemes while Wubs and de Goede (1992) discussed an explicit-implicit method for the solutions of 2D Burgers’ equations. Recently, Bahadir (2003) proposed a fully implicit finite-difference scheme and has compared results for two test problems with those of Jain and Holla. EI-Sayed and Kaya (2004) have applied the decomposition method for the solution of 2D Burgers’ equations. Liao (2009) proposed a fourth order finite-difference method for solving the system of 2D Burgers’ equations.

The purpose of this paper is to present the numerical solutions of Burgers’-type nonlinear equations defined on a bounded domain as:

P1: 1D Burgers’ equation:

\[
\frac{\partial u}{\partial t} + \alpha u \frac{\partial u}{\partial x} - \nu \frac{\partial^2 u}{\partial x^2} = 0, \tag{1}
\]

where \( \alpha \) and \( \nu \) are arbitrary constants.

P2: Coupled Burgers’ equations:

\[
\begin{align*}
\frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} - \eta u \frac{\partial u}{\partial x} - \alpha \frac{\partial (uv)}{\partial x}, \\
\frac{\partial v}{\partial t} &= \frac{\partial^2 v}{\partial x^2} - \xi v \frac{\partial v}{\partial x} - \beta \frac{\partial (uv)}{\partial x},
\end{align*} \tag{2}
\]

where \( \alpha, \beta, \eta \) and \( \xi \) are arbitrary constants.

P3: The 2D Burgers’ equation:

\[
\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + u \frac{\partial u}{\partial y} = \nu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{3}
\]

where \( \nu \) is an arbitrary constant.
The equations (1)-(3) are time dependent nonlinear equations. These equations have important applications in science and engineering. These types of equations are studied by a number of authors numerically (Dehghan and Salehi, 2010; Shokri and Dehghan, 2010; Dehghan and Ghesmati, 2010; Dehghan and Shokri, 2009, 2008; Dehghan and Shakeri, 2009; Dehghan and Mirzaei, 2008; Dehghan, 2007a, b, 2006).

Differential quadrature method (DQM) has been successfully applied to solve various linear and nonlinear 1D and 2D PDEs of many engineering, chemistry and physics problems (Korkmaz et al., 2010; Tomasiello, 2003, 2011, 1998; Al Kaisy et al., 2007; Korkmaz, 2010; Hsu, 2009; Korkmaz and Dag, 2008, 2009a, b, c; Mittal and Jiwari, 2009, 2011). In the present study, we have applied polynomial differential quadrature method (PDQM) to solve the 1D Burgers’, Coupled Burgers’ and 2D Burgers’ equations. The PDQM changed the nonlinear equations (1)-(4) into nonlinear ordinary differential equations (ODEs). The ODEs are solved by Runge-Kutta fourth order method (Pike and Roe, 1985). The accuracy and efficiency of the proposed method is demonstrated by several test examples.

2. Polynomial differential quadrature method

We consider the PDQM to approximate the solution of the problem. PDQM is an approximation to derivatives of a function at any grid points using weighted sum of all the functional values at certain points in the whole computational domain. Since the weighting coefficients are dependent only the spatial grid spacing, we assume uniformly distributed \( N \) grid points \( x_1 < x_2 < \cdots < x_N \) on the real axis. The differential quadrature discretization of the first and the second derivatives at a point \( x_i \) is given by the following equations:

\[
\begin{align*}
  u_x(x_i, t) &= \sum_{j=1}^{N} a_{ij} u(x_j, t), \\
  u_{xx}(x_i, t) &= \sum_{j=1}^{N} b_{ij} u(x_j, t),
\end{align*}
\]

(4)

where \( a_{ij} \) and \( b_{ij} \) represent the weighting coefficients (Chang, 2000), \( i = 1, 2, \ldots, N \). The following base functions are used to obtain weighting coefficients:

\[
g_k(x) = \frac{M(x)}{(x - x_k)M^{(1)}(x_k)}, \quad k = 1, 2, \ldots, N
\]

(5)

where:

\[
M(x) = (x - x_1)(x - x_2)\cdots(x - x_N).
\]

(6)

\[
M^{(1)}(x_i) = \prod_{k=1, k\neq i}^{N} (x_i - x_k)
\]

(7)

using the set of base functions given in equation (5), the weighting coefficients of the first order derivative are found as (Chang, 2000):

\[
a_{ij} = \frac{M^{(1)}(x_i)}{(x_i - x_j)M^{(1)}(x_j)}, \quad k = 1, 2, \ldots, N, \quad i \neq j
\]

(8)

\[
a_{ii} = -\sum_{j=1, j\neq i}^{N} a_{ij}, \quad i = 1, 2, \ldots, N
\]

(9)
and for weighting coefficients of the second order derivative, the formula is (Chang, 2000):

\[ b_{ij} = 2a_{ij}\left( a_{ii} - \frac{1}{x_i - x_j} \right), \quad i, j = 1, 2, \ldots, N, \quad i \neq j \]  

(10)

\[ b_{ii} = -\sum_{j=1, j \neq i}^{N} b_{ij}, \quad i = 1, 2, \ldots, N \]  

(11)

Similarly, Chang (2000) proposed the weighting coefficients of the higher order derivative in the explicit form:

\[ w^{(r)}_{ij} = r \left[ a_{ij}u_{ii}^{(r-1)} - \frac{w^{(r-1)}_{ii}}{x_i - x_j} \right] \quad \text{for } i \neq j \]  

(12)

for \( i, j = 1, 2, \ldots, N; \quad r = 2, 3, \ldots, N - 1 \)

\[ w^{(r)}_{ii} = -\sum_{j=1, j \neq i}^{N} w^{(r)}_{ij} \quad \text{for } i = j \]  

(13)

where \( a_{ij} \) are the weighting coefficients of the first order derivative mention in equations (8)-(9).

3. Numerical solutions by the PDQM

To illustrate the procedure, three examples related to the 1D Burgers’, coupled Burgers’ and 2D Burgers’ equations are given in the following.

3.1 1D Burgers’ equation

Let us first consider the 1D Burgers’ equation which has the form:

\[ \frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} + \alpha u \frac{\partial u}{\partial x} = 0, \quad (x, t) \in D \times [0, T], \]  

(14)

with the initial condition:

\[ u(x, 0) = \phi(x), \quad x \in D \]  

(15)

and the boundary conditions:

\[ u(x, t) = f(t), \quad (x, t) \in \delta D \times [0, T] \]  

(16)

where \( D = \{ x : a < x < b \} \) and \( \delta D \) is its boundary; \( \alpha \) and \( \nu \) are arbitrary constants.

The partial derivatives \( u_x \) and \( u_{xx} \) are approximated by PDQM, then the equations (14)-(16) changed into a system of ODEs as:

\[ \frac{du_i}{dt} = \nu \sum_{j=1}^{N} b_{ij}^{(1)} u_j - \alpha u_i \sum_{j=1}^{N} a_{ij}^{(1)} u_j, \quad (x_i, t) \in D \times [0, T], \]  

(17)
with the initial condition:

\[ u(x_i, 0) = \phi(x_i), \quad x_i \in D \]  \hfill (18)

and the boundary conditions:

\[ u(x_i, t) = f(t), \quad (x_i, t) \in \delta D \times [0, T] \]  \hfill (19)

The system of ODEs (17)-(19) is solved by Runge-Kutta fourth order method (Pike and Roe, 1985).

### 3.2 Coupled Burgers’ equations

The second instructive example to illustrate the PDQM is the homogeneous form of a coupled Burgers’ equation (Kaya, 2001) as follows:

\[
\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - \eta u \frac{\partial u}{\partial x} - \alpha \frac{\partial}{\partial x} (uv), \quad x \in D, \quad t \in [0, T]
\]

\[ \frac{\partial v}{\partial t} = \frac{\partial^2 v}{\partial x^2} - \xi v \frac{\partial v}{\partial x} - \beta \frac{\partial}{\partial x} (uv), \quad x \in D, \quad t \in [0, T] \]  \hfill (20)

with the initial conditions:

\[ u(x, 0) = g_1(x), \quad v(x, 0) = g_2(x), \quad x \in D \]  \hfill (22)

and the boundary conditions:

\[ u(x, t) = f_1(t), \quad v(x, t) = f_2(t), \quad (x, t) \in \delta D \times [0, T] \]

where \( D = \{x.a < x < b\} \) and \( \delta D \) is its boundary; \( \alpha, \beta, \eta \) and \( \xi \) are arbitrary constants.

Applying the PDQM to the problems (20)-(23), we have:

\[ \frac{d u_i}{d t} = \sum_{j=1}^{N} b_{ij}^{(1)} u_j - \eta u_i \sum_{j=1}^{N} a_{ij}^{(1)} u_j - \alpha \left( u_i \sum_{j=1}^{N} a_{ij}^{(1)} v_j + v_i \sum_{j=1}^{N} a_{ij}^{(1)} u_j \right), \quad x_i \in D, \] \hfill (24a)

\[ \frac{d v_i}{d t} = \sum_{j=1}^{N} b_{ij}^{(1)} v_j - \xi v_i \sum_{j=1}^{N} a_{ij}^{(1)} v_j - \beta \left( u_i \sum_{j=1}^{N} a_{ij}^{(1)} v_j + v_i \sum_{j=1}^{N} a_{ij}^{(1)} u_j \right), \quad x_i \in D, \] \hfill (24b)

with initial conditions:

\[ u(x_i, 0) = g_1(x_i), \quad v(x_i, 0) = g_2(x_i), \quad x_i \in D, \]  \hfill (25)

and the boundary conditions:

\[ u(x_i, t) = f_1(t), \quad v(x_i, t) = f_2(t), \quad (x_i, t) \in \delta D \times [0, T] \]  \hfill (26)

Again, the system of ODEs (24)-(26) is solved by Runge-Kutta fourth order method (Soliman, 2006).
3.3 2D Burgers’ equations

The third problem is 2D Burgers’ equations in the following form:

\[
\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + u \frac{\partial u}{\partial y} = \nu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)
\]

(27a)

with initial condition:

\[ u(x, y, 0) = g_3(x, y), \quad (x, y) \in D \]

(27b)

and the boundary conditions:

\[ u(x, y, t) = f_3(x, y, t), \quad x, y \in \partial D, \quad t > 0, \]

(27c)

where \( D = \{(x, y) : a < x < b, \quad a < y < b\} \) and \( \partial D \) is its boundary; and \( \nu = 1/R; \) \( R \) is the Reynolds number.

The PDQM changed the equation (27) into a system into the system of nonlinear ODEs as follows:

\[
u \left( \sum_{k=1}^{N} b^{(1)}_{i,k} u_{k,j} + \sum_{k=1}^{M} b^{(2)}_{j,k} u_{i,k} \right) - u_{i,j} \sum_{k=1}^{N} a^{(1)}_{i,k} u_{k,j} - u_{i,j} \sum_{k=1}^{M} a^{(2)}_{j,k} u_{i,k}
\]

(28a)

with initial conditions:

\[ u(i_1, j_1, 0) = g_3(i_1, j_1), \quad (i_1, j_1) \in D \]

(28b)

and the boundary conditions:

\[ u(i_1, j_1, t) = f_3(i_1, j_1, t), \quad (i_1, j_1) \in \partial D, \quad t > 0, \]

(28c)

where \( u' \) denote the derivatives of \( u \) with respect to \( t \) and \( u(i_1, j_1, t) \) is referred as \( u_{i,j} \). The term \( a^{(1)}_{i,j} \) and \( a^{(2)}_{i,j} \) are the weighting coefficients of first order partial derivatives with respect to \( x \) and \( y \) and similarly, \( b^{(1)}_{i,j}, b^{(2)}_{i,j} \) are the weighting coefficients of second order.

4. Numerical experiments

For numerical experiments, we have considered five examples for different Burgers’ equations P1-P3. In order to check to efficiency of the PDQM a comparison is made with numerical solutions and maximum error norm available in the literature. For describing the error, we define maximum error norm for \( u \) as follows:

\[ ||E(u)||_{\infty} = \max_{1 \leq j \leq N} \left| u(x_j, t) - u_N(x_j, t) \right| \]

(29)

where \( u(x_j, t) \) is exact solution of the problem and \( u_N(x_j, t) \) is numerical solution obtained by the proposed scheme.

**Example 1**

Consider the 1D Burgers’ equation (14) with initial and boundary conditions:

\[ u(x, 0) = \sin \pi x, \quad 0 \leq x \leq 1. \]

(30)

\[ u(0, t) = u(1, t) = 0, \quad t > 0. \]

(31)
The numerical solutions of Example 1 have been computed with $N = 21$, $\Delta t = 0.01$ for $\nu = 0.1$ and $\nu = 0.01$. The computed numerical solutions are compared with solutions obtained by numerical schemes proposed in Ozis et al. (2003), Kadalbajoo and Awasthi (2006) and Kutluay et al. (2004) and the exact solutions. The results are reported in Table I at different time levels. It is clear from the Table I that the solutions produced by the present method are in very good agreements with exact ones and are better in comparison to other methods. Figures 1 and 2 show the layer behavior of the computed solutions at different times $t = 0.2, 0.4, 0.6, 0.8, 1.0$.

**Example 2**
Consider the 1D Burgers’ equation (14) with the solitary solutions (Soliman, 2006):

$$u(x, t) = \frac{c}{\alpha} + \frac{2\nu}{\alpha} \tanh(x - ct),$$  \hspace{1cm} (32)

in the region $D = \{x: 0 < x < 1\}$. We obtain numerical solutions by PDQM. Computed maximum absolute errors are reported in Table II for various values of $\alpha$ and $\nu$ at different times with $N = 21$, $\Delta t = 0.01$ and $c = 0.1$. It is found that the errors are very small and negligible.

**Example 3**
We consider the coupled Burgers’ equations (20)-(21) with the initial and boundary conditions are taken as follows:

$$u(x, 0) = \sin(x), \quad v(x, 0) = \sin(x), \quad -\pi \leq x \leq \pi, \quad t > 0,$$

$$u(-\pi, t) = u(\pi, t) = 0, \quad v(-\pi, t) = v(\pi, t) = 0,$$  \hspace{1cm} (34)

<table>
<thead>
<tr>
<th>$x$</th>
<th>$t$</th>
<th>$\nu = 0.01$</th>
<th>$\Delta t = 0.0001$ (Kutluay et al., 2004)</th>
<th>$\Delta t = 0.01$ (Kadalbajoo and Awasthi, 2006)</th>
<th>PDQM</th>
<th>Exact solution</th>
<th>$\nu = 0.1$</th>
<th>$\Delta t = 0.01$ (Kadalbajoo and Awasthi, 2006)</th>
<th>PDQM</th>
<th>Exact solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.4</td>
<td>0.34819</td>
<td>0.34229</td>
<td>0.34192</td>
<td>0.3419</td>
<td>0.31429</td>
<td>0.30881</td>
<td>0.30889</td>
<td>0.30889</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.27536</td>
<td>0.26902</td>
<td>0.26896</td>
<td>0.26896</td>
<td>0.24373</td>
<td>0.24069</td>
<td>0.24074</td>
<td>0.24074</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>0.22752</td>
<td>–</td>
<td>0.22147</td>
<td>0.22148</td>
<td>0.19758</td>
<td>–</td>
<td>0.19568</td>
<td>0.19568</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.19375</td>
<td>0.18817</td>
<td>0.18819</td>
<td>0.18819</td>
<td>0.16391</td>
<td>0.16254</td>
<td>0.16256</td>
<td>0.16256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>0.07754</td>
<td>0.07511</td>
<td>0.07505</td>
<td>0.07511</td>
<td>0.02743</td>
<td>0.02720</td>
<td>0.02720</td>
<td>0.02720</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.50</td>
<td>0.4</td>
<td>0.66543</td>
<td>0.66797</td>
<td>0.66071</td>
<td>0.66071</td>
<td>0.56955</td>
<td>0.56963</td>
<td>0.56963</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.53525</td>
<td>0.53211</td>
<td>0.52942</td>
<td>0.52942</td>
<td>0.45169</td>
<td>0.44714</td>
<td>0.44721</td>
<td>0.44721</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>0.44526</td>
<td>–</td>
<td>0.43914</td>
<td>0.43914</td>
<td>0.36245</td>
<td>–</td>
<td>0.35924</td>
<td>0.35924</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.38047</td>
<td>0.37500</td>
<td>0.37442</td>
<td>0.37442</td>
<td>0.29437</td>
<td>0.29188</td>
<td>0.29192</td>
<td>0.29192</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>0.15362</td>
<td>0.15018</td>
<td>0.15014</td>
<td>0.15018</td>
<td>0.04057</td>
<td>0.04021</td>
<td>0.04021</td>
<td>0.04021</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.4</td>
<td>0.91201</td>
<td>0.93680</td>
<td>0.91026</td>
<td>0.91026</td>
<td>0.62592</td>
<td>0.62544</td>
<td>0.62544</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.77132</td>
<td>0.77724</td>
<td>0.76724</td>
<td>0.76724</td>
<td>0.49034</td>
<td>0.48715</td>
<td>0.48722</td>
<td>0.48721</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>0.65254</td>
<td>–</td>
<td>0.64740</td>
<td>0.64740</td>
<td>0.37713</td>
<td>–</td>
<td>0.37392</td>
<td>0.37392</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.56157</td>
<td>0.55833</td>
<td>0.55605</td>
<td>0.55605</td>
<td>0.29016</td>
<td>0.28744</td>
<td>0.28747</td>
<td>0.28747</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>0.22874</td>
<td>0.22485</td>
<td>0.22480</td>
<td>0.22481</td>
<td>0.01334</td>
<td>0.02978</td>
<td>0.02977</td>
<td>0.02977</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table I.** Comparison of numerical solutions of Example 1 for different values of $x$ and $t$. The errors are very small and negligible.
Figure 1. Numerical solutions of Example 1 at different times for $\nu = 0.1$

Figure 2. Numerical solutions of Example 1 at different times for $\nu = 0.01$
and the exact solutions are taken from Kutluay et al. (1999):

\[ u(x, t) = e^{-t} \sin(x), \quad v(x, t) = e^{-t} \sin(x), \quad -\pi \leq x \leq \pi, \quad t > 0, \quad (35) \]

The numerical results are computed with different time step length \( \Delta t \) and for various values of parameters \( \eta, \xi, \alpha, \beta \). The maximum absolute errors obtained with \( \Delta t = 0.01 \) up to \( T = 3 \) are reported in Table III. It is also shown in Table IV that absolute errors become half when time step is reduced to half. Figures 3-5 show the layer behavior of the computed solutions for \( u \) at different time.

**Example 4**

In this problem, we considered the coupled Burgers’ equations (20)-(21) with initial conditions:

\[ u(x, 0) = a_0 - 2A \left( \frac{2\alpha - 1}{4\alpha \beta - 1} \right) \tanh(Ax), \quad -10 \leq x \leq 10, \quad (36) \]

<table>
<thead>
<tr>
<th>( \alpha )</th>
<th>( \nu )</th>
<th>( t = 0.1 )</th>
<th>( t = 0.25 )</th>
<th>( t = 0.5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.01</td>
<td>( 3.20 \times 10^{-5} )</td>
<td>( 7.77 \times 10^{-5} )</td>
<td>( 1.53 \times 10^{-4} )</td>
</tr>
<tr>
<td>0.001</td>
<td></td>
<td>( 4.49 \times 10^{-7} )</td>
<td>( 9.06 \times 10^{-7} )</td>
<td>( 1.67 \times 10^{-6} )</td>
</tr>
<tr>
<td>0.0001</td>
<td></td>
<td>( 2.07 \times 10^{-8} )</td>
<td>( 2.37 \times 10^{-8} )</td>
<td>( 3.03 \times 10^{-8} )</td>
</tr>
<tr>
<td>0.00001</td>
<td></td>
<td>( 1.99 \times 10^{-9} )</td>
<td>( 1.99 \times 10^{-9} )</td>
<td>( 2.01 \times 10^{-9} )</td>
</tr>
</tbody>
</table>

Table II. Maximum absolute norm of Example 2 for various values of \( \alpha \) and \( \nu \) at different times

| \( t \) | \( ||E(u)||_\infty \) | \( ||E(v)||_\infty \) |
|---|---|---|
| 0.5 | \( 1.5168 \times 10^{-4} \) | \( 1.5168 \times 10^{-4} \) |
| 1.0 | \( 1.8397 \times 10^{-4} \) | \( 1.8397 \times 10^{-4} \) |
| 2.0 | \( 1.3525 \times 10^{-4} \) | \( 1.3525 \times 10^{-4} \) |
| 3.0 | \( 7.46014 \times 10^{-5} \) | \( 7.46014 \times 10^{-5} \) |

Table III. Maximum errors norm for Example 3 at different times

| \( \Delta t \) | \( ||E(u)||_\infty \) | \( ||E(v)||_\infty \) |
|---|---|---|
| 0.0200 | \( 3.70976 \times 10^{-3} \) | \( 3.70976 \times 10^{-3} \) |
| 0.0100 | \( 1.84705 \times 10^{-3} \) | \( 1.84705 \times 10^{-3} \) |
| 0.0050 | \( 9.21572 \times 10^{-4} \) | \( 9.21572 \times 10^{-4} \) |
| 0.0020 | \( 4.60280 \times 10^{-4} \) | \( 4.60280 \times 10^{-4} \) |
| 0.0010 | \( 1.83970 \times 10^{-4} \) | \( 1.83970 \times 10^{-4} \) |
| 0.0005 | \( 9.19422 \times 10^{-5} \) | \( 9.19422 \times 10^{-5} \) |

Table IV. Maximum absolute norm of \( u \) and \( v \) for \( \eta = -2, \xi = -2, \alpha = 1, \beta = 1 \) of Example 3 at \( t = 1 \).
Figure 3. Computed solutions $u$ at different time of Example 3 for $\eta = -2, \xi = -2, \alpha = 1, \beta = 1$

Figure 4. Computed solutions $u$ at different time of Example 3 for $\eta = 1, \xi = 2, \alpha = 1, \beta = 2$
\[ v(x, 0) = a_0 \left( \frac{2\beta - 1}{2\alpha - 1} \right) - 2A \left( \frac{2\alpha - 1}{4\alpha \beta - 1} \right) \tanh(Ax), \quad -10 \leq x \leq 10, \quad (37) \]

and the exact solutions are taken from Kaya (2001):

\[ u(x, t) = a_0 - 2A \left( \frac{2\alpha - 1}{4\alpha \beta - 1} \right) \tanh(A(x - 2At)), \quad -10 \leq x \leq 10, \quad t > 0 \quad (38) \]

\[ v(x, t) = a_0 \left( \frac{2\beta - 1}{2\alpha - 1} \right) - 2A \left( \frac{2\alpha - 1}{4\alpha \beta - 1} \right) \tanh(A(x - 2At)), \quad -10 \leq x \leq 10, \quad t > 0, \quad (39) \]

where \( A = a_0(4\alpha \beta - 1)/(4\alpha - 2) \) and \( a_0, \alpha, \beta \) are arbitrary constants. In Tables V and VI, we present a comparison between the numerical solution obtained by the present method and the method proposed in Khater et al. (2008) and Rashid and Ismail (2009)

**Table V.**
Comparison of numerical values of \( u \) for the Example 4 with those obtained by Khater et al. (2008) and Rashid (2009) with \( a_0 = 0.05, N = 21 \) and \( \Delta t = 0.01 \)

<table>
<thead>
<tr>
<th>( t )</th>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>Khater</th>
<th>Rashid</th>
<th>( |E(u)|_\infty )</th>
<th>Present method</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.1</td>
<td>0.3</td>
<td>1.44 \times 10^{-3}</td>
<td>9.619 \times 10^{-4}</td>
<td>4.173 \times 10^{-5}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.03</td>
<td>6.68 \times 10^{-4}</td>
<td>4.310 \times 10^{-4}</td>
<td>4.585 \times 10^{-5}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>0.1</td>
<td>0.3</td>
<td>1.27 \times 10^{-3}</td>
<td>1.153 \times 10^{-3}</td>
<td>8.275 \times 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.03</td>
<td>1.30 \times 10^{-3}</td>
<td>1.268 \times 10^{-3}</td>
<td>9.167 \times 10^{-5}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.3</td>
<td>1.30 \times 10^{-3}</td>
<td>1.268 \times 10^{-3}</td>
<td>9.167 \times 10^{-5}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.03</td>
<td>1.30 \times 10^{-3}</td>
<td>1.268 \times 10^{-3}</td>
<td>9.167 \times 10^{-5}</td>
<td></td>
</tr>
</tbody>
</table>
for different values of $t, \alpha$ and $\beta$. It is evident from the tables that the present method is more accurate than the methods proposed in Khater et al. (2008) and Rashid and Ismail (2009).

**Example 5**
Consider the 2D Burgers’ equation (27) with the exact solutions given by Dehghan et al. (2007):

$$u(x, y, t) = \frac{1}{1 + e^{R(y+x-t)/2}}$$

(40)

The initial and boundary conditions are taken from exact solution (44). The computational domain for this problem $D = \{(x, y): 0 < x < 1, 0 < y < 1\}$. The numerical solution by PDQM have been computed with $N = M = 21$ and time step length $\Delta t = 10^{-3}$ for $T = 0.5$ and $R = 100$. Results are reported in the Table VII. Figure 6 shows the concentration profile of $u$ at different time.

<table>
<thead>
<tr>
<th>5. Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>In this paper, a PDQM is presented to obtain numerical solutions of nonlinear Burgers’-type equations. The problem is reduced to a system of ODEs that is solved by the Runge-Kutta method of order four. Numerical results of 1D Burgers’, coupled Burgers’ and 2D Burgers’ equations are obtained and compared with those which are already available in the literature and found the present numerical schemes give better solutions.</td>
</tr>
</tbody>
</table>

**Table VI.**
Comparison of numerical values of $v$ for the Example 4 with those obtained by Khater et al. (2008) and Rashid (2009) with $a_0 = 0.05$, $N = 21$ with $\Delta t = 0.01$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>Khater</th>
<th>Rashid</th>
<th>Present method</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.1</td>
<td>0.3</td>
<td>$5.42 \times 10^{-4}$</td>
<td>$3.332 \times 10^{-4}$</td>
<td>$5.418 \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.03</td>
<td>$1.20 \times 10^{-3}$</td>
<td>$1.148 \times 10^{-3}$</td>
<td>$2.826 \times 10^{-5}$</td>
</tr>
<tr>
<td>1.0</td>
<td>0.1</td>
<td>0.3</td>
<td>$1.29 \times 10^{-3}$</td>
<td>$1.162 \times 10^{-3}$</td>
<td>$1.074 \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.03</td>
<td>$2.35 \times 10^{-3}$</td>
<td>$1.638 \times 10^{-3}$</td>
<td>$5.673 \times 10^{-5}$</td>
</tr>
<tr>
<td>3.0</td>
<td>0.1</td>
<td>0.3</td>
<td>–</td>
<td>–</td>
<td>$3.119 \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.03</td>
<td>–</td>
<td>–</td>
<td>$1.663 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

**Table VII.**
Comparison between exact and numerical solutions of Example 5 when $R = 100$ for $t = 0.5$
In addition, the results are very useful for its physical value in plasma physics, fluid mechanics, solid state physics and engineering science. Moreover, the method is easily applicable for 2D nonlinear problems which are difficult to solve by other numerical techniques.

References


**Corresponding author**
Ram Jiwari can be contacted at: ram1maths@gmail.com

To purchase reprints of this article please e-mail: reprints@emeraldinsight.com
Or visit our web site for further details: www.emeraldinsight.com/reprints