Application of the Non-Polynomial Spline Approach to the Solution of the Burgers’ Equation

Mohamed A. Ramadan*,1, Talaat S. El-Danaf1 and Faisal E.I. Abd Alaal2

1Department of Mathematics, Faculty of Science, Menoufia University, Shebeen El-Koom, Egypt
2Graduate Student with the Department

Abstract: In this paper, we propose a non-polynomial spline based method to develop a numerical method for approximation to the Burgers’ equation. Applying the Von-Neumann stability analysis, we show that the proposed method is unconditionally stable. A numerical example is given to illustrate the applicability and the accuracy of the presented new method.

Keywords: Non-polynomial spline, nonlinear burgers’ equation, von neumann stability, accuracy.

INTRODUCTION

Consider the Burgers’ equation of the form:

\[ u_t + uu_x - \nu u_{xx} = 0, \quad a \leq x \leq b, \quad t \geq 0 \]  

subject to the conditions

\[ u(a,t) = \beta_1(t), \quad u(b,t) = \beta_2(t), \]

\[ u_x(a,t) = \lambda_1, \quad u_x(b,t) = \lambda_2, \quad t \geq 0 \]  

(2)

The last two additional conditions in (2) are true at the initial time for \( 0 < \nu < 1 \), so we suppose that it is true for any time; the initial condition takes the form:

\[ u(x,0) = f(x), \quad a \leq x \leq b \]  

(3)

The study of this Burgers’ equation is important due to its application in the approximate theory of flow through a shock wave propagating in a viscous fluid [1] and in the modeling of turbulence [2]. In the past few years a great deal of efforts has been expended to study the equation (1) as well as other forms of this partial differential equation, both theoretically and numerically, see for example, [3-11].

Recently, there is a wide use of non-polynomial splines based methods for approximating the solution of boundary value problems of different orders, see for example, [12-15]. However, the numerical analysis of literature contains little for using these non-polynomial splines dealing with the solution of partial differential equations [16,17].

In this paper, we are concerned with the problem of applying non-polynomial spline functions to develop a numerical method for obtaining approximation for the solution for non-linear Burgers’ equation (1). The non-polynomial spline function in this work has a trigonometric part, and a polynomial part of first degree.

Remark

The \( C^\infty \) - differentiability of the trigonometric part of non-polynomial spline compensates for the loss of smoothness inherited by polynomial splines.

The paper is organized as follows: In section 2, a new method which depends on the use of the non-polynomial splines is derived. In section 3, the stability analysis is theoretically discussed. Using Von Neuman method, for given values of specified parameters, the proposed method is shown to be unconditionally stable. Finally, in section 4, a numerical example is included to illustrate the practical implementation of the proposed method. The accuracy performance of the obtained numerical results is compared with the exact solution. Since we are using a new type of (moving) boundary conditions to improve the accuracy, therefore we can not make an accuracy comparison with other exciting methods for this problem. The numerical results show that our proposed method is a promising approach for solving different types of this nonlinear partial differential equation problem.

DERIVATION OF THE NUMERICAL METHOD FOR THE NON-POLYNOMIAL SPLINE APPROACH TO BURGERS’ EQUATION

To set up the non-polynomial spline method, select an integer \( N > 0 \) and time-step size \( k > 0 \).

With \( h = \frac{b-a}{N+1} \), the mesh points \( \{x_i,t_j\} \) are \( x_i = a + ih \), for each \( i = 0,1,...,N+1 \), and \( t_j = jk \), for each \( j = 0,1,... \).

Let \( U_i^j \equiv U(x_i,t_j) \) be an approximation to \( u_i^j \equiv u(x_i,t_j) \), obtained by the segment \( P(x_i,t_j) \) of the mixed spline function passing through the points \( (x_i,U_i^j) \) and \( (x_{i+1},U_{i+1}^j) \). Each segment has the form:
where, 
\[ a_i + d_i = U_i^j, \]
\[ a_i \cos \theta + b_i \sin \theta + c_i h + d_i = U_i^{j+1}, \]
\[ -a_i \omega^2 = S_i^j, \]
\[ -a_i \omega^2 \cos \theta - b_i \omega^2 \sin \theta = S_i^{j+1}, \]
where, 
\[ a_i \equiv a_i(t_j), \quad b_i \equiv b_i(t_j), \quad c_i \equiv c_i(t_j), \quad d_i \equiv d_i(t_j), \]
and \[ \theta = \omega h. \]

By solving the last four equations, we obtain the following expressions:

\[ a_i = -\frac{h^2}{\theta^2} S_i^j, \quad b_i = \frac{h^2}{\theta^2} \left( \cos \theta S_i^j - S_i^{j+1} \right), \]
\[ c_i = \frac{\left( U_i^{j+1} - U_i^j \right)}{h} + \frac{\left( S_i^{j+1} - S_i^j \right)}{\theta^2}, \quad d_i = \frac{h^2}{\theta^2} S_i^j + U_i^j, \]  

Using the continuity condition of the first derivative at \( x = x_j \), that is, \( P_i^1(x_j,t_j) = P_i^1(x_{j+1},t_j) \), we get the following relation:
\[ b_i \omega + c_i = -a_i \omega \sin \theta + b_i \omega \cos \theta + c_{i-1}. \]

Using (7), equation (8) gives us the following tridiagonal system:
\[ U_i^{j+1} - 2U_i^j + U_i^{j-1} = \alpha S_i^{j+1} + \beta S_i^j + \alpha S_i^{j-1}, \]
for \( i = 1, 2, \ldots, N \).

Where,
\[ \alpha = \frac{h^2}{\theta \sin \theta} - \frac{h^2}{\theta^2}, \quad \beta = -\frac{2h^2 \cos \theta}{\theta \sin \theta} + \frac{2h^2}{\theta^2}, \]
and
\[ S_i^j = \frac{\partial^2 U_i^j}{\partial x^2} = \frac{1}{h^2} \left( \frac{\partial U_i^j}{\partial t} + (U_i^j) \frac{\partial U_i^j}{\partial x} \right). \]

Replacing \( j \) by \( j+1/2 \), system (9) becomes:
\[ U_i^{j+1/2} - 2U_i^{j+1/2} + U_i^{j-1/2} = \]
\[ \alpha S_i^{j+1/2} + \beta S_i^{j/2} + \alpha S_i^{j-1/2}, \quad i = 1, 2, K, N. \]

where,
\[ U_i^{j+1/2} \equiv U(x_i,t_{j+1/2}), \quad t_{j+1/2} = \frac{t_j + t_{j+1}}{2}, \]
and,
\[ S_i^{j+1/2} = \frac{1}{h} \left( \frac{\partial U_i^{j+1/2}}{\partial t} + (U_i^{j+1/2}) \frac{\partial U_i^{j+1/2}}{\partial x} \right). \]

Using the finite difference method, we obtain
\[ U_i^{j+1} = \frac{U_i^{j+1} + U_i^j}{2}, \quad \frac{\partial}{\partial t} U_i^{j+1/2} = \frac{U_i^{j+1} - U_i^j}{h}, \]
and
\[ \frac{\partial}{\partial x} U_i^j = \frac{U_i^{j+1} - U_i^j}{h}. \]

Using these formulas allows us to express \( S_i^{j+1/2} \) as,
\[ S_i^{j+1/2} = \frac{1}{h} \left( U_i^{j+1} - U_i^j \right) + \]
\[ \left( \frac{U_i^{j+1/2} - U_i^{j-1/2}}{h} - \frac{U_i^{j+1} - U_i^j}{h} \right). \]

The use of (11) and (12) in equation (10) gives us the following system:
\[ A_i U_i^{j+1} + B_i U_i^{j+1/2} + C_i U_i^{j-1/2} + D_i U_i^{j-1} = \]
\[ A_i U_i^{j+1} + B_i U_i^j + C_i U_i^{j-1} + D_i U_i^{j-2}, \]
for each \( i = 1, 2, 3, \ldots, N - 1 \) \( j = 0, 1, 2, \ldots \)

where,
\[ D_i = \frac{2h}{\alpha \delta_i}, \quad D_i' = -\frac{2h}{\alpha \delta_i}, \]
\[ C_i = \frac{1}{2} + \frac{\alpha}{\alpha + 2h \delta_i}, \quad C_i' = \frac{1}{2} + \frac{\alpha}{\alpha - 2h \delta_i}, \]
\[ B_i = 1 + \frac{\beta \delta_i}{h}, \quad B_i' = 1 - \frac{\beta \delta_i}{h}, \]
\[ A_i = \frac{1}{2} + \frac{\alpha}{\alpha}, \quad A_i' = \frac{1}{2} + \frac{\alpha}{\alpha}. \]

and,
\[ \delta_i = (U_i^{j+1/2}). \]

System (13) consists of \( N-1 \) equations in the unknowns \( U_i^j, \quad i = 1, \ldots, N - 1 \). To get a solution to this system we need 3-additional equations. These equations are obtained from the conditions in (2). The first two parts in (2) are replaced by:
\[ U_i^j = \beta_i(t_j), \quad U_{N+1}^j = \beta_{N+1}(t_j), \quad j = 0, 1, \ldots \]

but the last part in (2) is discretized by the following equation:
\[ U_{N-2}^j + U_{N-1}^j - 13U_N^j + 11U_{N+1}^j = \]
\[ 8h \frac{\partial}{\partial x} U_{N+1}^j = 8h \lambda_2(t_j), \quad j \geq 0. \]
The last equation is true for any time. Writing Eqns. (13)-(15) in matrix form gives:

\[ QU^{j+1} = Q'U^j + r^{j+1} \]  \hspace{1cm} (16)

where,

\[ U^j = (U_{i-1}^j, U_i^j, U_{i+1}^j, \ldots , U_{N-1}^j, U_N^j, U_{N+1}^j)^T, \]

\[ Q = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
A_1 & B_1 & C_1 & D_1 & 0 & 0 & \cdots & 0 \\
0 & A_2 & B_2 & C_2 & D_2 & 0 & \cdots & 0 \\
0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & A_{N-2} & B_{N-2} & C_{N-2} & D_{N-2} & 0 \\
0 & \cdots & 0 & 0 & A_{N-1} & B_{N-1} & C_{N-1} & D_{N-1} \\
0 & \cdots & 0 & 0 & 1 & 1 & -13 & 11 \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix} \]

\[ Q' = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
A'_1 & B'_1 & C'_1 & D'_1 & 0 & 0 & \cdots & 0 \\
0 & A'_2 & B'_2 & C'_2 & D'_2 & 0 & \cdots & 0 \\
0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & A'_{N-2} & B'_{N-2} & C'_{N-2} & D'_{N-2} & 0 \\
0 & \cdots & 0 & 0 & A'_{N-1} & B'_{N-1} & C'_{N-1} & D'_{N-1} \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \]

and,

\[ r^j = \left( \beta_i(t_i), 0, 0, \ldots, 0, 0.8h^2 \lambda_2(t_i), 0 \right)^T \]

where \( r^j \) is an (N+2) dimensional column vector, while \( Q \) and \( Q' \) are (N+2)x(N+2) matrices. The initial condition \( u(x,0) = f(x) \), for each \( a \leq x \leq b \), implies that \( U_i^0 = f(x_i) \), for each \( i = 0, 1, \ldots, N+1 \). These values can be used in Eq. (13) to find the value of \( U_i^1 \), for each \( i = 0, 1, \ldots, N+1 \). If the procedure is reapplied once all the approximations \( U_i^1 \) are known, the values of \( U_i^2, U_i^3, \ldots \) can be obtained in a similar manner.

**Remark 1**

To cope with the nonlinear terms in System (13), the following steps are followed:

1. **At \( j = 0 \)**, we approximate \( \delta_i \) by \( \delta_i^0 \) calculated from \( U_i^0 \) only, that is,

\[ \delta_i = \delta_i^0 = U_i^0 \]  \hspace{1cm} for each \( i = 0, 1, \ldots, N \), which are needed to compute the elements of \( Q \) and \( Q' \). We then obtain \( U^1 \) from (16).

2. **At \( j = 1 \)**, we approximate \( \delta_i \) by \( \delta_i^1 \) calculated from \( 0.5 \left( U_i^0 + U_i^1 \right) \), that is,

\[ \delta_i = \delta_i^1 = \left( 0.5 \left( U_i^0 + U_i^1 \right) \right) \]  \hspace{1cm} for each \( i = 0, 1, \ldots, N \),

which are needed to compute the elements of \( Q \) and \( Q' \). We then obtain \( U_{i+1} \) from (16).

3. **At \( j = n \)**, we approximate \( \delta_i \) by \( \delta_i^n \) calculated from \( 0.5 \left( U_i^{n-1} + U_i^n \right) \). That is,

\[ \delta_i = \delta_i^n = \left( 0.5 \left( U_i^{n-1} + U_i^n \right) \right) \]  \hspace{1cm} for each \( i = 1, 2, \ldots, N \), which are needed to compute the elements of \( Q \) and \( Q' \). We then obtain \( U_i^{n+1} \) from (16).

**THE STABILITY ANALYSIS**

For stability analysis, we use the Von Neumann method. To do this, we must linearize the nonlinear term \( (u_x) \) of the Burgers’ equation (1) by making \( \delta_i^1 = \delta_i = \delta_i^n = d \) in the numerical scheme (16). According to the Von Neumann method we have:

\[ U_i^j = \zeta.exp(q_i \varphi h), \]  \hspace{1cm} (17)

where \( \varphi \) is the mode number, \( q = \sqrt{-1} \), \( h \) is the element size, and \( \zeta \) is the amplification factor of the scheme. Substitute Eq. (17) into Eq. (13) we get:

\[ \zeta^{j+1} \begin{bmatrix} A_i \exp((i-1)q_i \varphi h) + B_i \exp(iq_i \varphi h) + C_i \exp((i+1)q_i \varphi h) + D_i \exp((i+2)q_i \varphi h) \\ 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ 0 \\ A_i' \exp((i-1)q_i \varphi h) + B_i' \exp(iq_i \varphi h) + C_i' \exp((i+1)q_i \varphi h) + D_i' \exp((i+2)q_i \varphi h) \end{bmatrix} \]  \hspace{1cm} (18)

where,

\[ D_i = \frac{\alpha d}{2vh}, \quad D_i' = -\frac{\alpha d}{2vh}, \]

\[ C_i = \frac{1}{2} + \frac{\alpha d}{2vh} + \frac{\beta d}{2vh}, \quad C_i' = \frac{1}{2} + \frac{\alpha d}{2vh} - \frac{\beta d}{2vh}, \]  \hspace{1cm} (19)

\[ B_i = 1 + \frac{\beta d}{2vh} + \frac{\alpha d}{2vh}, \quad B_i' = -1 + \frac{\beta d}{2vh} - \frac{\alpha d}{2vh}, \]

\[ A_i = \frac{1}{2} + \frac{\alpha d}{2vh}, \quad A_i' = \frac{1}{2} + \frac{\alpha d}{2vh}. \]

Dividing both sides of Eq. (18) by \( \exp(iq_i \varphi h) \) we obtain:

\[ \zeta^{j+1} \begin{bmatrix} A_i \exp(-q_i \varphi h) + B_i + C_i \exp(q_i \varphi h) + D_i \exp(2q_i \varphi h) \\ 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ 0 \\ A_i' \exp(-q_i \varphi h) + B_i' + C_i' \exp(q_i \varphi h) + D_i' \exp(2q_i \varphi h) \end{bmatrix} = \]

\[ \zeta^j \begin{bmatrix} A_i \exp(-q_i \varphi h) + B_i + C_i \exp(q_i \varphi h) + D_i \exp(2q_i \varphi h) \\ 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix} \]  \hspace{1cm} (20)

Using the following Euler’s formula:

\[ \exp(q_i \varphi h) = \cos \varphi + q_i \sin \varphi \]  \hspace{1cm} \varphi = \varphi h \]  \hspace{1cm} (21)

Eq. (20) can be represented in the form:
\[
\begin{align*}
\zeta^{i+1} &= \left[ A_i (\cos \phi - q \sin \phi) + B_i + C_i (\cos \phi + q \sin \phi) + \right] \\
&\quad \left. D_i (\cos 2\phi + q \sin 2\phi) \right] \\
\zeta_i &= \left[ A_i^* (\cos \phi - q \sin \phi) + B_i^* + C_i^* (\cos \phi + q \sin \phi) + \right] \\
&\quad \left. D_i^* (\cos 2\phi + q \sin 2\phi) \right]
\end{align*}
\]

After simple calculations, we obtain:
\[
\zeta = \frac{X^* + qY^*}{X + qY}
\]

where,
\[
X^* = (A_i + C_i^*) \cos \phi + D_i \cos 2\phi + B_i,
\]
\[
Y^* = (C_i - A_i^*) \sin \phi + D_i^* \sin 2\phi
\]

Eqns. (19) and (23) together give:
\[
X' = \frac{1}{vk} (\beta + 2\alpha) - \frac{4\alpha}{vk} \sin^2 \phi - 2 \sin^2 \phi + \frac{d}{vh} (\beta - 2\alpha) \sin^2 \phi + \frac{\alpha d}{vh} \sin^2 \phi,
\]
\[
X = \frac{1}{vk} (\beta + 2\alpha) - \frac{4\alpha}{vk} \sin^2 \phi - 2 \sin^2 \phi - \frac{d}{vh} (\beta - 2\alpha) \sin^2 \phi - \frac{\alpha d}{vh} \sin^2 \phi
\]

\[
Y' = -\frac{\beta d}{2vh} \sin \phi - \frac{\alpha d}{2vh} \sin 2\phi
\]

\[
Y = \frac{\beta d}{2vh} \sin \phi + \frac{\alpha d}{2vh} \sin 2\phi
\]

The necessary and sufficient condition for (13) to be stable is:
\[
|\zeta| = \sqrt{X'^2 + Y'^2} \leq 1
\]

Simplifying the above inequality, we obtain:
\[
X'^2 \geq X''^2
\]

where \(Y'' = Y'^2\).

The last inequality gives us:
\[
\left[ \frac{1}{vk} (\beta + 2\alpha) - \frac{4\alpha}{vk} \sin^2 \phi \right] \\
\left[ 2 \sin^2 \phi - \frac{d}{vh} (\beta - 2\alpha) \sin^2 \phi - \frac{\alpha d}{vh} \sin^2 \phi \right] \geq 0
\]

For \(\beta > 2\alpha\), \(\beta > 0\), and \(\alpha > 0\) we obtain:
\[
\left[ \frac{1}{vk} (\beta + 2\alpha) - \frac{4\alpha}{vk} \sin^2 \phi \right] \geq 0
\]

which enables us to write (26) in the form:
\[
\left[ \frac{2 \sin^2 \phi}{2} - \frac{d}{vh} (\beta - 2\alpha) \sin^2 \phi - \frac{\alpha d}{vh} \sin^2 \phi \right] \geq 0
\]

Our system is conditionally stable. The condition of stability is \(\beta \geq 2\alpha\), \(\alpha \to 0\) and \(\beta \to 0\).

**NUMERICAL RESULTS**

We now obtain the approximate numerical solution of Burgers’ equation for one standard problem. The accuracy of our proposed numerical method is measured by computing the difference between the analytic and numerical solutions at each mesh point, and use these differences to compute the \(L_2\) and \(L_\infty\) error norms.

The analytic solution of the Burgers’ equation (1) [8] is given by:
\[
u(x,t) = \frac{(x/t)}{1 + (t/\sigma)^{1/2} \exp(x^2/4vt)}
\]

Where, \(0 \leq x \leq 1\), \(t \geq 1\)

Where \(\sigma = \exp(1/8\nu)\) make initial condition to be the equation (28) evaluated at \(t = 1\). The boundary conditions are:
\[
u(0,t) = 0, \quad u(1,t) = \frac{(1/t_j)}{1 + (t_j/\sigma)^{1/2} \exp(1/4vt_j)}
\]

Note that the second condition in Eq(28) is a type of moving boundary condition.

**Remark 2**

The additional conditions:
\[
u_x(a,t) = \frac{\partial}{\partial x} u(0,t_0) \quad \text{and} \quad u_x(b,t) = \frac{\partial}{\partial x} u(1,t_0)
\]

are true at the initial time for \(0 < \nu < 1\), so we suppose that the analytical solution (28) satisfies these conditions.

The obtained numerical results are summarized in the following tables for \(\Delta x = 0.025\) (Tables 1-4). Table 1 gives the numerical and exact solutions at time \(t = 3\).

**Table 1.**

<table>
<thead>
<tr>
<th>(x_i)</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0221546</td>
<td>0.0220390</td>
</tr>
<tr>
<td>0.2</td>
<td>0.0435601</td>
<td>0.0433457</td>
</tr>
<tr>
<td>0.3</td>
<td>0.0634297</td>
<td>0.0631439</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0809113</td>
<td>0.0805862</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0950889</td>
<td>0.0947549</td>
</tr>
<tr>
<td>0.6</td>
<td>0.1050300</td>
<td>0.1047120</td>
</tr>
<tr>
<td>0.7</td>
<td>0.1099090</td>
<td>0.1096260</td>
</tr>
<tr>
<td>0.8</td>
<td>0.1092040</td>
<td>0.1089760</td>
</tr>
<tr>
<td>0.9</td>
<td>0.1029440</td>
<td>0.1028100</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0918946</td>
<td>0.0918946</td>
</tr>
</tbody>
</table>
Table 2. $\nu = 0.05, \Delta t = 0.01, \alpha = 0.2 \cdot 10^{-6}$ and $\beta = 6 \cdot 10^{-4}$

<table>
<thead>
<tr>
<th>Time</th>
<th>$L_2$ error $\times 10^3$</th>
<th>$L_{\infty}$ error $\times 10^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.129983</td>
<td>0.203959</td>
</tr>
<tr>
<td>2.5</td>
<td>0.146748</td>
<td>0.235351</td>
</tr>
<tr>
<td>3</td>
<td>0.246727</td>
<td>0.334341</td>
</tr>
<tr>
<td>3.5</td>
<td>0.311609</td>
<td>0.428702</td>
</tr>
<tr>
<td>5</td>
<td>0.342211</td>
<td>0.482112</td>
</tr>
</tbody>
</table>

Table 3. $\nu = 0.5, \Delta t = 0.1, \alpha = 0.2 \cdot 10^{-6}$ and $\beta = 6 \cdot 10^{-4}$

<table>
<thead>
<tr>
<th>Time</th>
<th>$L_2$ error $\times 10^3$</th>
<th>$L_{\infty}$ error $\times 10^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.662543</td>
<td>0.933320</td>
</tr>
<tr>
<td>2.5</td>
<td>0.353418</td>
<td>0.497759</td>
</tr>
<tr>
<td>3</td>
<td>0.214160</td>
<td>0.301579</td>
</tr>
<tr>
<td>3.5</td>
<td>0.142256</td>
<td>0.200246</td>
</tr>
<tr>
<td>5</td>
<td>0.057475</td>
<td>0.0808847</td>
</tr>
</tbody>
</table>

Table 4. $\nu = 0.005, \Delta t = 0.05, \alpha = 0.2 \cdot 10^{-9}$ and $\beta = 4 \cdot 10^{-4}$

<table>
<thead>
<tr>
<th>Time</th>
<th>$L_2$ error $\times 10^3$</th>
<th>$L_{\infty}$ error $\times 10^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8.015390</td>
<td>2.187940</td>
</tr>
<tr>
<td>2.5</td>
<td>9.622790</td>
<td>2.618230</td>
</tr>
<tr>
<td>3</td>
<td>9.835130</td>
<td>2.848630</td>
</tr>
<tr>
<td>3.5</td>
<td>5.998950</td>
<td>1.912170</td>
</tr>
<tr>
<td>5</td>
<td>1.520370</td>
<td>0.228921</td>
</tr>
</tbody>
</table>

Remark

Using these new types of boundary conditions, (29) allows us to compute the approximate solution for large value of the time $t$ with an acceptable accuracy. In previous existing methods for Burgers' equation, numerical solutions are computed for the time $t$ with $t \ll 5$.

Next, we draw some of the obtained approximate solutions $U(x,t)$ for this test problem versus the distance $x$. Figs. (1) and (2) illustrate the behavior of the numerical solution at $\nu = 0.05, \Delta t = 0.01, \Delta x = 0.025$ but Figs. (3) and (4) illustrate the behavior of the numerical solution at $\nu = 0.005, \Delta t = 0.05, \Delta x = 0.025$ for some different times $t = \{2 \text{ and } 3\}$ respectively.

From Figs. (1), (3) at $t = 2$ and Figs. (2), (4) at $t = 3$, we can conclude that as $\nu$ in the dispersion term $\nu u_{xx}$, increases from 0.005 to 0.05, the effect of the nonlinearity term, the second term $\nu u_t$ of Burgers' equation (1), decreases.

CONCLUSION

In this paper, a numerical treatment for the Burgers' equation using non-polynomial spline is proposed. The stability analysis of the method is shown to be unconditionally
stable for given values of specified parameters. Namely for 
\( \beta \geq 2\alpha, \alpha \to 0 \) and \( \beta \to 0 \), our system is unconditionally stable. The obtained approximate numerical solutions are showed to maintain good accuracy.

REFERENCES


