Malaysian Journal of Mathematical Sciences 19(4): 1289–1304 (2025)
https://doi.org/10.47836/mjms.19.4.06

Malaysian Journal of Mathematical Sciences

Journal homepage: https://mjms.upm.edu.my

Numerical Solutions of The One-dimensional Burgers' Equation by A Preconditioned Successive Over Relaxation Method

Ghadamyari, S.[1] and Mojarrab, M.*[1]

[1]*Department of Mathematics, University of Sistan and Baluchestan, Zahedan, Iran*

E-mail: ma_mojarrab@math.usb.ac.ir
**Corresponding author*

Received: 7 January 2025
Accepted: 27 May 2025

Abstract

The Successive Over Relaxation (SOR) method is a well-known iterative method to solve the linear system $Ax = b$. One effective strategy for increasing the speed of convergence or even getting out of a possible recession is to use an effective preconditioner. With this perspective, in this paper, using a preconditioned version of the SOR method, we extract the numerical solutions of the one-dimensional Burgers' equation. At first, we discretize Burgers' equation by using some central and forward finite difference formulas. Then, we eliminate the non-linear term produced in the obtained differential equations by using an average formula called non-local arithmetic discretization scheme. Next, we examine the error analysis and conditional stability of the method. Finally, we modify the resulting system of linear equations with its preconditioned version. Theoretical and numerical results show that the present preconditioned method has increased the convergence rate significantly compared to the standard method and the resulting methods.

Keywords: Burgers' equation; finite difference scheme; preconditioner; linear system; SOR; semi approximate approach.

1 Introduction

The non-linear one-dimensional Burgers' equation with initial and boundary conditions consists of,

$$\begin{aligned} \frac{\partial \mathfrak{u}}{\partial t} + \mathfrak{u}\frac{\partial \mathfrak{u}}{\partial x} &= \vartheta \frac{\partial^2 \mathfrak{u}}{\partial x^2}, && (x,t) \in (\mathfrak{a},\mathfrak{b}) \times (0,\mathbb{T}], \\ \mathfrak{u}(x,0) &= \mathbb{f}(x), && x \in [\mathfrak{a},\mathfrak{b}], \\ \mathfrak{u}(\mathfrak{a},t) = \mathbb{f}_1(t), \quad \mathfrak{u}(\mathfrak{a},t) &= \mathbb{f}_2(t), && 0 \le t \le \mathbb{T}, \end{aligned} \tag{1}$$

where $\vartheta \geq 0$ is the kinematics viscosity parameter, which plays the role of maintaining the balance between convection and diffusion. In addition $\mathbb{f}(x)$, $\mathbb{f}_1(x)$ and $\mathbb{f}_2(x)$ are given sufficiently smooth functions of the variables. In (1), if $\vartheta = 0$, Burgers' equation is hyperbolic. Otherwise, it is parabolic. In 1915, Bateman presented this equation to express possible discontinuities in obtaining solutions to the equations of motion of a viscous fluid when the viscosity coefficient tends to zero [4]. Further, Burgers [7, 6], in his papers published in 1939 and 1948, introduced various aspects of turbulence, which led to the presentation of a mathematical model that confirmed this theory. So far, numerous numerical methods and algorithms have been presented to solve Burgers' equation. For example, the following can be mentioned. Higher-order accurate finite difference method [25], cubic spline quasi-interpolant scheme [23], a least-squares quadratic B–spline finite element method [10], modified cubic B–spline differential quadrature method [2], traveling and shock wave simulations in a viscous Burgers' equation with Periodic Boundary Conditions [13], full implicit finite difference schemes [15], UAT tension B–spline differential quadrature method [8], solving of Burgers' equation by Bernstein differential quadrature method [1], solving with quantum computing [17], a higher order implicit adaptive finite point method [21], a preconditiond SOR method [24], physics-based preconditioning of Jacobian free Newton Krylov for Burgers' equation using modified nodal integral method [9], applying barycentric interpolation method and fourth order cubic B- spline collocation method to solve the Burgers' and Fisher's equations [14, 19], finding travelling wave solutions and conservation laws of the Korteweg-de Vries-Burgers equation with power law nonlinearityand as a generalization of the Burgers' equation [12], and use of Keller-Box scheme in solving of Burgers' equation with nonlocal boundary condition [3].

In this work, we will solve the one-dimensional Burgers' equation numerically in two phases. In the first phase, Burgers' equation is discretized using a second-order implicit finite difference scheme along with a semi-approximate procedure. The output of this phase will be a system of approximate three-diagonal linear equations. The second phase includes applying the SOR projection method and a special preconditioner for it. In this phase, the iterative formula of the preconditioned SOR method is presented, and finally, the algorithm of the preconditioned SOR method is presented to solve the extracted system of linear equations. Numerical results show that the numerical solutions of (1) can be calculated using the proposed method method with optimal accuracy and in less CPU execution time than those of the standard method and Gauss-Seidel projection method.

The article is categorized in the following form: in Section 2, we present the governing equations. The formulation of the preconditioned SOR and the corresponding algorithm are the subject of Section 3. In Section 4, error analysis is done followed by Section 5 that contains stability analysis of the derived scheme for Burgers' equation. Then, we present our numerical experiments along with interpretation and conclusions in the text in Section 6. Finally, we derive a summary of the main results in Section 7.

2 Derivation of The System of Approximate Linear Equations

Consider (1) with initial and boundry conditions asummed on it. We rewrite (1) as below,

$$\frac{\partial \mathfrak{u}}{\partial t} + \mathbb{F}(x,t,\mathfrak{u})\frac{\partial \mathfrak{u}}{\partial x} = \vartheta \frac{\partial^2 \mathfrak{u}}{\partial x^2}, \tag{2}$$

where $\mathfrak{u}(x,t,\mathfrak{u})\dfrac{\partial \mathfrak{u}}{\partial x}$ is named as a nonlinear part. To begin with, we first create a finite grid network of points in the equation domain . Suppose,

$$\begin{aligned} x_k &= \mathfrak{a} + k\Delta x, \qquad k = 0,1,\ldots,\mathfrak{m}, \qquad \Delta x = \frac{\mathfrak{b}-\mathfrak{a}}{\mathfrak{m}}, \\ t_l &= l\Delta t, \qquad\qquad l = 0,1,\ldots,\mathfrak{n}, \qquad \Delta t = \frac{T}{\mathfrak{n}}, \end{aligned} \tag{3}$$

and $\mathfrak{u}(x_k,t_l)$ displayed with $\mathfrak{u}_{k,l}$ that can be determined by solving the corresponding linear system using an implicit finite difference approximation equation. Now, using the forward and central finite difference formulas on the (2), we will have at the $(l+1)$ level,

$$\frac{\mathfrak{u}_{k,l+1}-\mathfrak{u}_{k,l}}{\Delta t} + \mathbb{F}\left(x_k,t_{l+1},\mathfrak{u}_{k,l+1}\right)\frac{\mathfrak{u}_{k+1,l+1}-\mathfrak{u}_{k-1,l+1}}{2\Delta x} = \frac{\vartheta}{(\Delta x)^2}(\mathfrak{u}_{k-1,l+1}-2\mathfrak{u}_{k,l+1}+\mathfrak{u}_{k+1,l+1}). \tag{4}$$

Next, to linearize the nonlinear part of the appearance in (4), we apply a scheme called nonlocal arithmetic mean discretization scheme [18] which is as follows,

$$\mathbb{F}(x_k,t_{l+1},\mathfrak{u}_{k,l+1}) = \mathbb{F}\left(x_k,t_{l+1},\frac{\mathfrak{u}_{k-1,l+1}+\mathfrak{u}_{k+1,l+1}}{2}\right). \tag{5}$$

Substituting (5) into (4) and simplifying it, we will have

$$-\mathfrak{s}_k\mathfrak{u}_{k-1,l+1} + \mathfrak{q}_k\mathfrak{u}_{k,l+1} - \mathfrak{u}_{k+1,l+1} = \mathfrak{v}_{k,l}, \qquad k = 1,2,\ldots,\mathfrak{m}-1, \tag{6}$$

where

$$\begin{aligned} \mathfrak{s}_k &= \frac{(\frac{1}{2\Delta x})\mathbb{F}_{k,l+1} + \frac{\vartheta}{(\Delta x)^2}}{\alpha_k}, & \mathfrak{q}_k &= \frac{\frac{1}{\Delta t} + \frac{2\vartheta}{(\Delta x)^2}}{\alpha_k}, \\ \alpha_k &= \frac{\vartheta}{(\Delta x)^2} - \frac{1}{2\Delta x}\mathbb{F}_{k,l+1}, & \mathfrak{v}_{k,l} &= \frac{\mathfrak{u}_{k,l}}{\Delta t \times \alpha_k}, \\ \mathbb{F}_{k,l+1} &= \mathbb{F}\left(x_k,t_{l+1},\frac{\mathfrak{u}_{k-1,l+1}+\mathfrak{u}_{k+1,l+1}}{2}\right). \end{aligned}$$

So, at the level $(l+1)$ for different values k, the implicit approximation (6) generates a three diagonal linear system as follows,

$$\mathbb{A}\mathbb{U}_{l+1} = \mathbb{V}_{l+1}, \tag{7}$$

where

$$\mathbb{A} = \begin{bmatrix} \mathfrak{q}_1 & -1 & & & & \\ -\mathfrak{s}_2 & \mathfrak{q}_2 & -1 & & & \\ & -\mathfrak{s}_3 & \mathfrak{q}_3 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -\mathfrak{s}_{\mathbb{m}-2} & \mathfrak{q}_{\mathbb{m}-2} & -1 \\ & & & & -\mathfrak{s}_{\mathbb{m}-1} & \mathfrak{q}_{\mathbb{m}-1} \end{bmatrix},$$

$$\mathbb{U}_{l+1} = \left[\mathfrak{u}_{1,l+1}, \mathfrak{u}_{2,l+1}, \ldots, \mathfrak{u}_{\mathbb{m}-1,l+1}\right]^T,$$

$$\mathbb{V}_{l+1} = \left[\mathfrak{v}_{1,l+1} + \mathfrak{s}_1\mathfrak{u}_{0,l+1}, \mathfrak{v}_{2,l+1}, \ldots, \mathfrak{v}_{\mathbb{m}-1,l+1} + \mathfrak{u}_{\mathbb{m},l+1}\right]^T.$$

Now, it is possible to solve the three-diagonal system (7) using an iterative method. SOR is a well-known iterative method for solving systems of linear equations resulting from the discretization of nonlinear and differential equations [16, 22]. For this purpose, we consider a preconditioned SOR method (PSOR) [20]. Because as we know, combining an iterative method with a suitable preconditioner can strengthen the speed of convergence. In the next section, we will consider related formulations.

3 Formulation of PSOR

When $\mathbb{m}$ is large, the three-diagonal system (7) will be a large and sparse system, and in this situation, using an iterative method to derive an approximate solution is a common method. One of the solutions to ameliorate the convergence speed of an iterative method is to produce an equivalent system with the original system and solve it instead of the original system. This is done by multiplying a matrix from the left, right, or multiplying two matrices on the sides of the matrix of coefficients. The result of this operation should be such that the equivalent system is solved faster than the original system. This method is called preconditioning technique and the matrix used to produce the equivalent system is called a preconditioner.

Li and Evans [11] designed a preconditioner specific to the SOR method and showed that the solution of the preconditioned system is faster than the original system. This preconditioner is as follows,

$$\mathbb{P} = \mathbb{I} + \mathbb{S} = \begin{bmatrix} 1 & 1 & \ldots & \ldots & 0 & 0 \\ 0 & 1 & 1 & \ldots & 0 & 0 \\ 0 & 0 & 1 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \ldots & 0 & 1 & 1 \\ 0 & 0 & \ldots & 0 & 0 & 1 \end{bmatrix}, \tag{8}$$

where $\mathbb{I}$ is the identity matrix, and the entries of upper diagonal are the negative of the entries of upper diagonal of $\mathbb{A}$ in (7). In the following, we obtain the iterative formula of PSOR for (7). Suppose,

$$\mathbb{A}^*\mathbb{U}_{l+1} = \mathbb{V}^*_{l+1}, \tag{9}$$

is the preconditioned system corresponding to (7) so that, $\mathbb{A}^* = \mathbb{P} \times \mathbb{A}$ and $\mathbb{V}^*_{l+1} = \mathbb{P} \times \mathbb{V}_{l+1}$. Now, we consider the standard splitting for $\mathbb{A}^*$,

$$\mathbb{A}^* = \mathbb{D}^* - \mathbb{L}^* - \mathbb{F}^*, \tag{10}$$

where $-\mathbb{F}^*, -\mathbb{L}^*$, and $\mathbb{D}^*$ are strictly upper, strictly lower triangular and diagonal matrices corresponing to $\mathbb{A}^*$, respectively. According to (9) and (10), the iterative formula of PSOR is

$$\mathbb{U}_{l+1}^{(k+1)} = (\mathbb{D}^* - \omega\mathbb{L}^*)^{-1}\left[(1-\omega)\mathbb{D}^* + \omega\mathbb{F}^*\right]\mathbb{U}_{l+1}^{(k)} + \omega(\mathbb{D}^* - \omega\mathbb{L}^*)^{-1}\mathbb{V}_{l+1}^*, \tag{11}$$

where $\omega \in (0,1]$ is the relaxation parameter. It should be noted that in the implementation of the algorithm, it is not necessary to explicitly generate $\mathbb{A}^*$ and $\mathbb{V}_{l+1}^*$. The general structure of PSOR is algorithmized as follows.

Algorithm 1. PSOR steps for (9)

i. Set the starting value $\mathbb{U}_{l+1}^{(0)}, \epsilon$.

ii. For $l = 0, 1, \ldots, \mathbb{n} - 1$, do

a) Set $\mathbb{U}_{l+1}^{(0)} = 0$ and boundary conditions.

iii. For $\mathcal{K} = 0, 1, \ldots$, until convergence, do

b) Compute new value of $\mathbb{U}_{l+1}^{(\mathcal{K}+1)}$ by applying

b$_1$. $k = 1$,

$$\mathbb{u}_{k,l+1}^{(\mathcal{K}+1)} = (1-\omega)\mathbb{u}_{k,l+1}^{(\mathcal{K})} + \frac{\omega}{\mathbb{q}_k - \mathbb{s}_{k+1}}\left((1-\mathbb{q}_{k+1})\mathbb{u}_{k+1,l+1}^{(\mathcal{K})} + \mathbb{u}_{k+2,l+1}^{(\mathcal{K})} + \mathbb{V}_{k,l}^*\right).$$

b$_2$. For $k = 2, \ldots, \mathbb{m} - 3$,

$$\mathbb{u}_{k,l+1}^{(\mathcal{K}+1)} = (1-\omega)\mathbb{u}_{k,l+1}^{(\mathcal{K})} + \frac{\omega}{\mathbb{q}_k - \mathbb{s}_{k+1}}\left(\mathbb{s}_k\mathbb{u}_{k-1,l+1}^{(\mathcal{K}+1)} + (1-\mathbb{q}_{k+1})\mathbb{u}_{k+1,l+1}^{(\mathcal{K})} + \mathbb{u}_{k+2,l+1}^{(\mathcal{K})} + \mathbb{V}_{k,l}^*\right),$$

end For.

b$_3$. $k = \mathbb{m} - 2$,

$$\mathbb{u}_{k,l+1}^{(\mathcal{K}+1)} = (1-\omega)\mathbb{u}_{k,l+1}^{(\mathcal{K})} + \frac{\omega}{\mathbb{q}_k - \mathbb{s}_{k+1}}\left(\mathbb{s}_k\mathbb{u}_{k-1,l+1}^{(\mathcal{K}+1)} + (1-\mathbb{q}_{k+1})\mathbb{u}_{k+1,l+1}^{(\mathcal{K})} + \mathbb{V}_{k,l}^*\right).$$

b$_4$. $k = \mathbb{m} - 1$,

$$\mathbb{u}_{k,l+1}^{(\mathcal{K}+1)} = (1-\omega)\mathbb{u}_{k,l+1}^{(\mathcal{K})} + \frac{\omega}{\mathbb{q}_k}\left(\mathbb{s}_k\mathbb{u}_{k-1,l+1}^{(\mathcal{K}+1)} + \mathbb{V}_{k,l}^*\right).$$

c) If $\| \mathbb{U}_{l+1}^{(\mathcal{K}+1)} - \mathbb{U}_{l+1}^{(\mathcal{K})} \|_\infty < \epsilon$, set $l = l+1$ and go to **Step ii**, else set $\mathcal{K} = \mathcal{K}+1$ and go to **Step iii**.

end For $(\mathcal{K})$

end For (l)

iv. Present the approximate solution.

4 Error Analysis

A mathematical analysis of convergence specifies the accuracy with which the numerical scheme approximates the solution. Here, we assume that $\mathbb{U}_k$ and $\mathbb{u}_k$ represent the exact and approximate values, respectively, at the knot x_k and at the level $(l+1)$. We also assume that u and its derivatives remain bounded in the domain $(\mathbb{a}, \mathbb{b}) \times (0, \mathbb{T}]$. Here, we have the following theorem.

Theorem 4.1. *The error in the approximation of the Burgers' equation by (6) is of the order of* $O(\Delta t + h^2)$ *where* $h \equiv \Delta x$.

Proof. According to (1) at node x_k and at level $(l+1)$, we have

$$\frac{\partial \mathbb{U}_k}{\partial t} + \mathbb{U}_k \frac{\partial \mathbb{U}_k}{\partial x} = \vartheta \frac{\partial^2 \mathbb{U}_k}{\partial x^2}, \tag{12}$$

$$\frac{\partial \mathfrak{u}_k}{\partial t} + \mathfrak{u}_k \frac{\partial \mathfrak{u}_k}{\partial x} = \vartheta \frac{\partial^2 \mathfrak{u}_k}{\partial x^2}. \tag{13}$$

In this case, the point-wise error is as follows,

$$\begin{aligned}
&\left| \frac{\partial \mathbb{U}_k}{\partial t} - \frac{\partial \mathfrak{u}_k}{\partial t} + \left(\mathbb{U}_k \frac{\partial \mathbb{U}_k}{\partial x} - \mathfrak{u}_k \frac{\partial \mathfrak{u}_k}{\partial x} \right) - \vartheta \left(\frac{\partial^2 \mathbb{U}_k}{\partial x^2} - \frac{\partial^2 \mathfrak{u}_k}{\partial x^2} \right) \right| \\
&\quad \leq \left| \frac{\partial \mathbb{U}_k}{\partial t} - \frac{\partial \mathfrak{u}_k}{\partial t} \right| + \left| \mathbb{U}_k \frac{\partial \mathbb{U}_k}{\partial x} - \mathfrak{u}_k \frac{\partial \mathfrak{u}_k}{\partial x} \right| + \vartheta \left| \frac{\partial^2 \mathbb{U}_k}{\partial x^2} - \frac{\partial^2 \mathfrak{u}_k}{\partial x^2} \right| \\
&\quad = \left| \frac{\partial \mathbb{U}_k}{\partial t} - \frac{\partial \mathfrak{u}_k}{\partial t} \right| + \left| \mathbb{U}_k \frac{\partial \mathbb{U}_k}{\partial x} - \mathfrak{u}_k \frac{\partial \mathbb{U}_k}{\partial x} + \mathfrak{u}_k \frac{\partial \mathbb{U}_k}{\partial x} - \mathfrak{u}_k \frac{\partial \mathfrak{u}_k}{\partial x} \right| + \vartheta \left| \frac{\partial^2 \mathbb{U}_k}{\partial x^2} - \frac{\partial^2 \mathfrak{u}_k}{\partial x^2} \right| \\
&\quad \leq \left| \frac{\partial \mathbb{U}_k}{\partial t} - \frac{\partial \mathfrak{u}_k}{\partial t} \right| + |\mathbb{U}_k - \mathfrak{u}_k| \left| \frac{\partial \mathbb{U}_k}{\partial x} \right| + |u_k| \left| \frac{\partial \mathbb{U}_k}{\partial x} - \frac{\partial \mathfrak{u}_k}{\partial x} \right| + \vartheta \left| \frac{\partial^2 \mathbb{U}_k}{\partial x^2} - \frac{\partial^2 \mathfrak{u}_k}{\partial x^2} \right|.
\end{aligned} \tag{14}$$

Now, considering the numerical derivative formulas used in (4) and the nonlocal arithmetic mean discretization scheme in (5), it can be shown that,

$$\left| \frac{\partial \mathbb{U}_k}{\partial t} - \frac{\partial \mathfrak{u}_k}{\partial t} \right| = O\left(\Delta t\right), \tag{15}$$

$$|\mathbb{U}_k - \mathfrak{u}_k| = O\left(\frac{h^2}{2}\right), \tag{16}$$

$$\left| \frac{\partial \mathbb{U}_k}{\partial x} \right| = O\left(\frac{h^2}{6}\right), \tag{17}$$

$$\left| \frac{\partial^2 \mathbb{U}_k}{\partial x^2} - \frac{\partial^2 \mathfrak{u}_k}{\partial x^2} \right| = O\left(\frac{h^2}{12}\right). \tag{18}$$

So, from (14), we get point-wise error $\leq O(\Delta t + h^2)$ as $\mathbb{U}$ and its derivatives are bounded in the considered domain. Thus, we have $O(\Delta t + h^2)$ accuracy. □

5 Stability Analysis

In this section, we examine the stability of the proposed numerical scheme. Numerical stability of a scheme means that possible errors generated during the execution of the scheme should not be amplified, because otherwise, in practice, the final approximate solution may have a significant distance from the exact solution. For this purpose, we have used the von Neumann method. As discussed, according to (6), we have

$$-\mathfrak{s}_k \mathfrak{u}_{k-1,l+1} + \mathfrak{q}_k \mathfrak{u}_{k,l+1} - \mathfrak{u}_{k+1,l+1} = \mathfrak{v}_{k,l}, \qquad k = 1, 2, \ldots, \mathfrak{m} - 1, \tag{19}$$

where $\mathfrak{s}_k$, $\mathfrak{q}_k$, and $\mathfrak{v}_{k,l}$ have values as defined in Section 2. Substituting $\mathfrak{u}_{k,l} = A\xi^l \exp(ik\phi h)$, where $i = \sqrt{-1}$, A is amplitude, h is the step length and ϕ is the mode number, we get

$$\xi = \frac{-\mathfrak{s}_k + \mathfrak{q}_k - 1}{-(1 + \mathfrak{s}_k)\cos\phi h + \mathfrak{q}_k + i(\mathfrak{s}_k - 1)\sin\phi, h}, \tag{20}$$

which by definitions,

$$\begin{aligned} X_1 &= -\mathfrak{s}_k + \mathfrak{q}_k - 1, \\ X_2 &= -(1+\mathfrak{s}_k)\cos\phi h + \mathfrak{q}_k, \\ X_3 &= (\mathfrak{s}_k - 1)\sin\phi h. \end{aligned}$$

We will have

$$\xi = \frac{X_1}{X_2 + iX_3}. \tag{21}$$

For stability, we must have $|\xi| \leq 1$. In other words, we must have

$$\frac{X_1^2}{X_2^2 + X_3^2} \leq 1,$$

for the values as defined above. Hence, the proposed method is conditionally stable for Burgers' equation.

6 Numerical Results

In this part, we will solve numerically three examples of type (1) using MATLAB version 2023. The personal computer used has hardware specifications of 8GB RAM, 1TB storage, and an i7 core processor. In the Tables 1–3, PSOR implementation is compared with SOR and Gauss-Seidel (GS) methods. In each table, useful information includes $\mathfrak{m}(=\mathfrak{n})$ (grid size), Iter (average value of the number of iterations for the approximate calculation of $\mathfrak{u}_{kl}$), time (the average value of the approximate calculation time $\mathfrak{u}_{kl}$ in seconds) and Error (the average value of the error in the approximate calculation $\mathfrak{u}_{kl}$) are reported. The error tolerance is also considered $\epsilon = 10^{-10}$. The results presented in the tables are reported per optimal ω, which means that each method is implemented with certain values of ω at $(0, 1]$, starting from $\omega = 0.1$ and with a step length of 0.1, and the best implementation is reported considering the minimum run time. The absolute error function is also $e = \left|\mathfrak{u}(x,t^*)_{\text{exact}} - \mathfrak{u}(x,t^*)_{\text{approximate}}\right|$ where t^* is a number belonging to the domain t specified in each example.

As mentioned earlier, in this paper, the preconditioner presented in [11] is considered. Li and Evans [11] showed that if A in (7) is a Z-matrix, then, the convergence speed of PSOR will be higher than SOR and GS. It can be easily checked that all three examples presented have this characteristic.

Example 6.1. *We consider Burgers' equation* [2] *with the following exact solution,*

$$\mathfrak{u}(x,t) = \frac{\dfrac{x}{t}}{1 + \left(\dfrac{t}{t_0}\right)^{\frac{1}{2}} \exp\left(\dfrac{x^2}{4\vartheta t}\right)}.$$

The initial condition is given by,

$$\mathfrak{u}(x,1) = \frac{x}{1 + \exp\left(\dfrac{1}{4\vartheta}\left(x^2 - \dfrac{1}{4}\right)\right)}, \qquad t > 1.$$

The relevant numerical experiments are recorded in Table 1. In our implementations, $x \in [0, 1.2]$, $t \in [1, 2.2]$, $\vartheta = 0.5$ *and* $t_0 = \exp\left(\frac{1}{8\vartheta}\right)$ *is considered. The data in Table 1 show that PSOR has significantly reduced the convergence speed and the number of itereations compared to GS and SOR. In Figure 1, the 3D graphs of both analytical and approximate solutions are depicted which shows that the approximate solution closely estimates the analytical solution. In Figure 2, the 2D graphs of analytical and numerical solutions for certain values* $t(= 1.15, 1.30, 1.45)$ *and for* $\vartheta = 0.05$ *are plotted. The solutions of both sets are close to each other except near the boundaries. To enhance understanding, Figure 3 has been drawn. In Figure 3, the graphs of the exact and approximate solutions for certain values* $t(= 1.15, 1.30, 1.45)$ *and for* $\vartheta = 0.05$ *are depicted together. As observed, the solutions are close to each other except near the boundary, which is consistent with Figure 2.*

Table 1: Numerical experiments for Example 6.1.

	GS			*SOR*			*PSOR*		
m	Iter	time	Error	Iter	time	Error	Iter	time	Error
256	1186	0.51	$1.55e-04$	1423	0.66	$1.55e-04$	165	0.17	$1.55e-04$
512	2095	2.89	$6.59e-05$	2508	3.94	$6.59e-05$	320	0.78	$6.59e-05$
1024	3619	18.87	$2.80e-05$	4319	23.27	$2.80e-05$	604	4.77	$2.80e-05$
2048	6076	129.06	$1.18e-05$	7217	157.75	$1.18e-05$	1137	34.55	$1.18e-05$
4096	9809	835.09	$5.00e-06$	11572	1007	$5.00e-06$	2193	263.94	$5.00e-06$

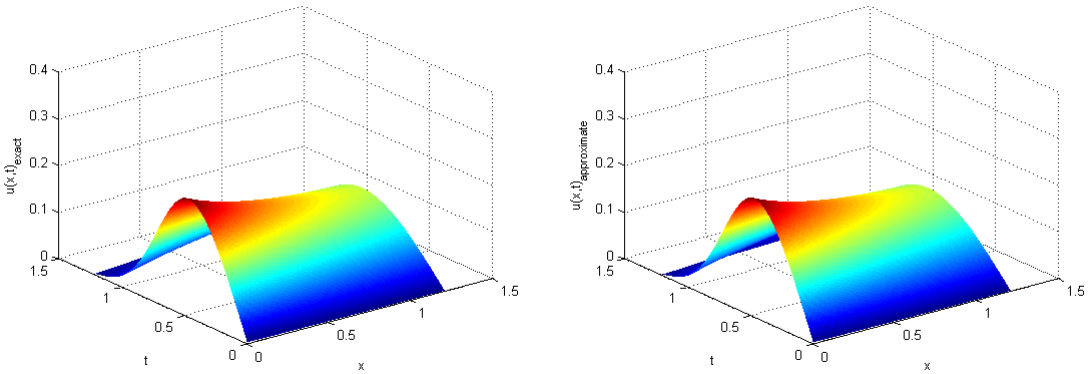

Figure 1: 3D plots of the exact solution (left) and the approximate solution (right) of Example 6.1 with $\vartheta = 0.05$.

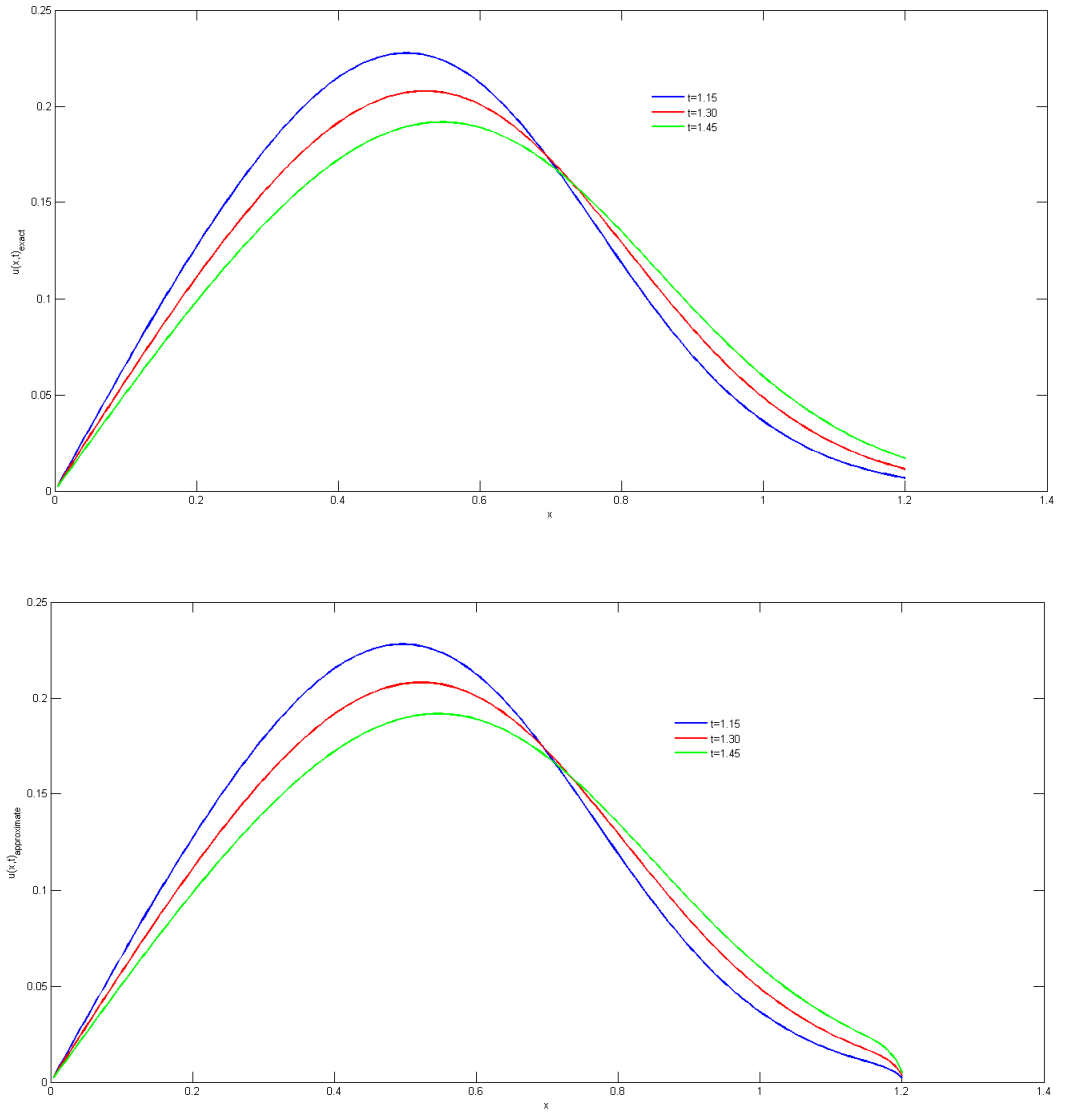

Figure 2: 2D plots of the exact solution (up) and the approximate solution (down) of Example 6.1 with $\vartheta = 0.05$ and for $t = 1.15, 1.30, 1.45$.

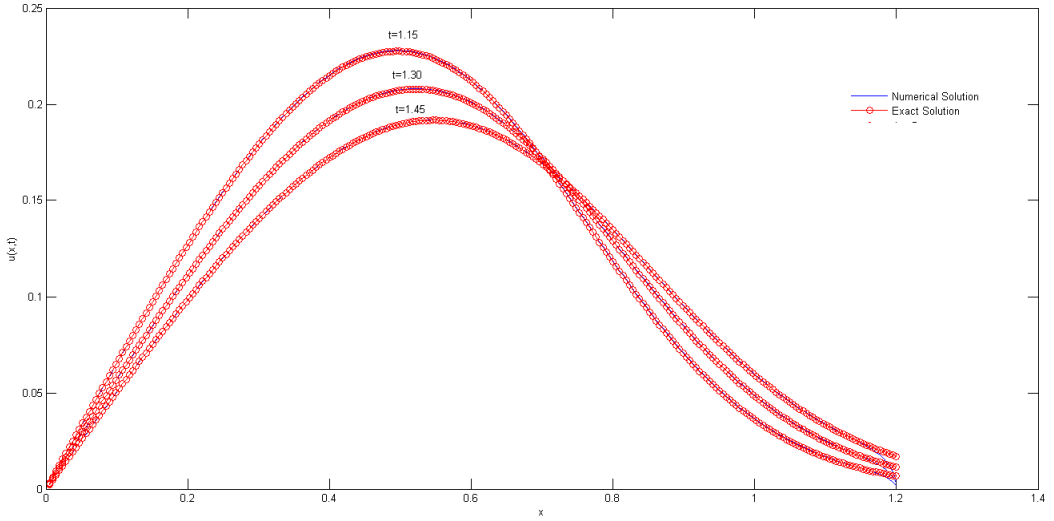

Figure 3: Compare the two-dimensional graphs of the exact and approximate solutions for Example 6.1 with $\vartheta = 0.05$ and for $t = 1.15, 1.30, 1.45$.

Example 6.2. *We consider Burgers' equation* [5] *with the following initial condition:*

$$\mathfrak{u}(x,0) = 2x, \quad t > 0,$$

and exact solution:

$$\mathfrak{u}(x,t) = \frac{2x}{1+2t}.$$

The corresponding results are shown in Table 2. In our implementations, $x \in [0,1]$, $t \in [0,1]$, *and* $\vartheta = 0.5$ *is considered. As can be seen in Table 2, both time and iteration factor have been impressively reduced in PSOR compared to GS and SOR. In Figure 4, the 3D graphs of both analytical and approximate solutions are depicted which shows that both solutions are relatively close. In Figure 5, the 2D graphs of exact and numerical solutions for certain values* $t(= 0.25, 0.50, 0.75)$ *and for* $\vartheta = 0.05$ *are plotted. The solutions of both sets are close to each other except near the end boundary. This is more evident for* $t = 0.25$ *and the difference decreases with increasing* t. *Figure 6 is obtained by superimposing the graphs of Figure 5. As can be seen, this figure accurately shows the process occurring in Figure 5 and confirms them.*

Table 2: Numerical experiments of Example 6.2.

	GS			SOR			$PSOR$		
$\mathbb{m}$	Iter	time	Error	Iter	time	Error	Iter	time	Error
256	1563	0.66	$3.35e-06$	1880	0.73	$3.35e-06$	185	0.16	$3.35e-06$
512	2811	4.09	$8.62e-07$	3374	4.53	$8.62e-07$	358	0.82	$8.62e-07$
1024	4969	26.93	$2.00e-07$	5949	32.03	$2.00e-07$	67198	5.21	$2.00e-07$
2048	8598	211.83	$5.58e-08$	10259	221.89	$5.58e-08$	1278	38.50	$5.58e-08$
4096	14470	1291	$1.41e-08$	17188	1526	$1.41e-08$	2406	285.58	$1.41e-08$

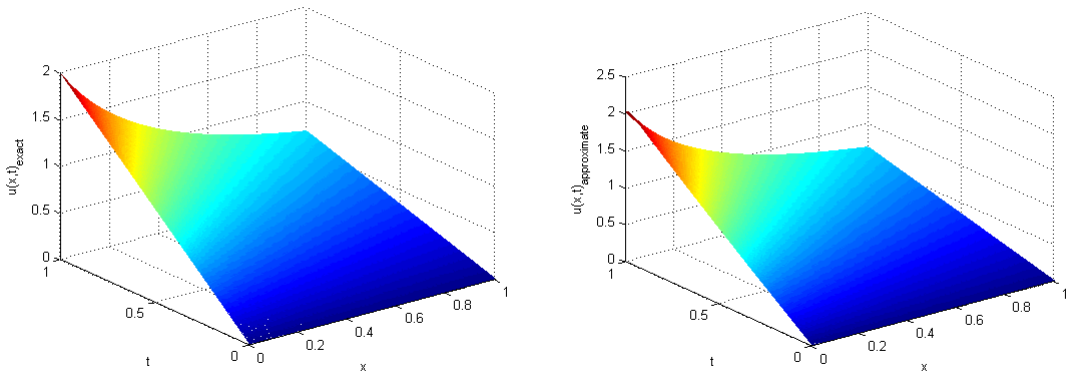

Figure 4: 3D plots of the exact solution (left) and the approximate solution (right) of Example 6.2 with $\vartheta = 0.05$.

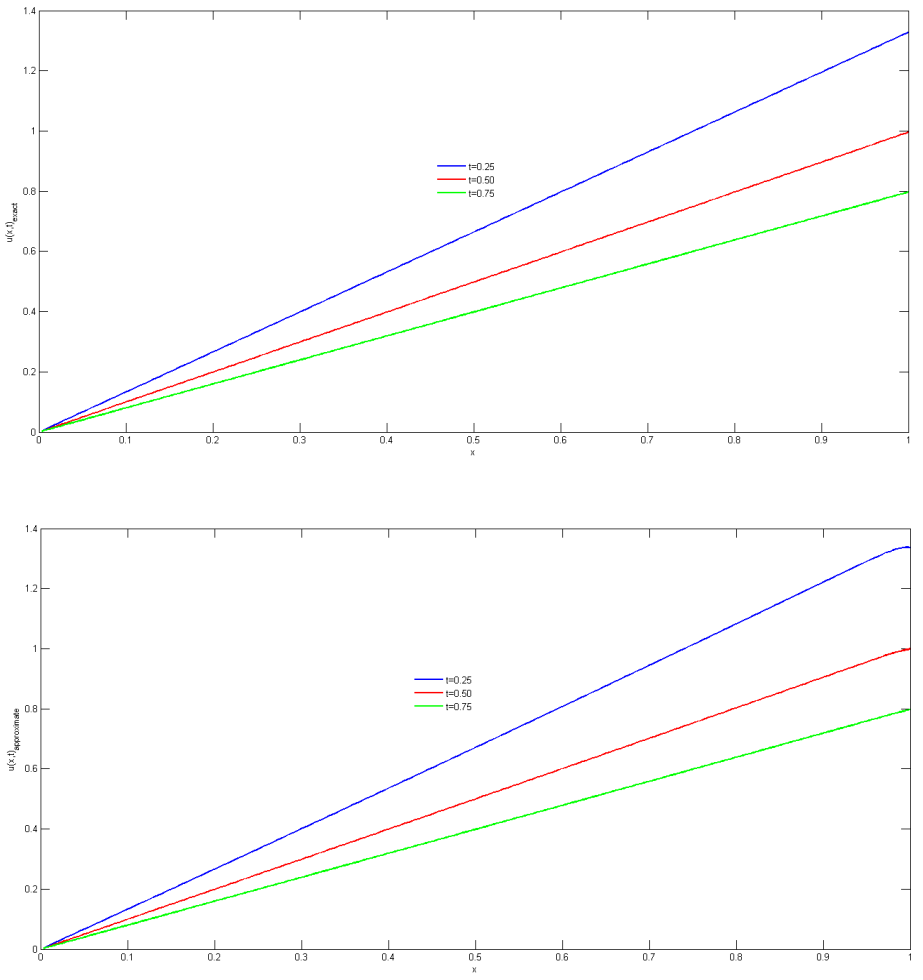

Figure 5: 2D plots of the exact solution (up) and the approximate solution (down) of Example 6.2 with $\vartheta = 0.05$ and for $t = 0.25, 0.50, 0.75$.

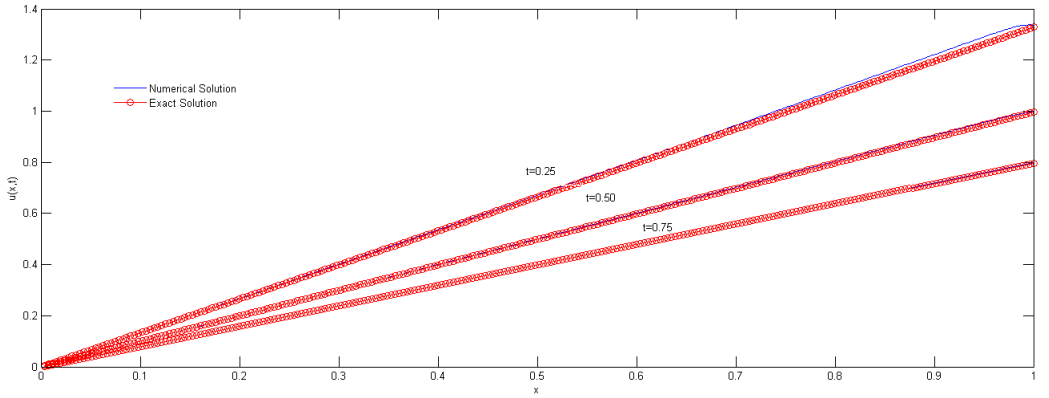

Figure 6: Compare the two-dimensional graphs of the exact and approximate solutions for Example 6.2 with $\vartheta = 0.05$ and for $t = 0.25, 0.50, 0.75$.

Example 6.3. *We consider Burgers' equation* [24] *with the following initial condition:*

$$\mathfrak{u}(x,1) = \frac{1}{2} - \frac{1}{2}\tanh\left(\frac{1}{4}x\right), \quad t > 0,$$

and exact solution:

$$\mathfrak{u}(x,t) = \frac{1}{2} - \frac{1}{2}\tanh\left(\frac{1}{4}\left(x - \frac{t}{2}\right)\right).$$

Table 3 contains the numerical results of Example 6.3. In our implementations, $x \in [0,1]$, $t \in [0,1]$, and $\vartheta = 0.5$ is considered. Comparing PSOR with SOR and GS, we find that this method has performed remarkably well in reducing the two fundamental factors of iteration and execution time. In Figure 7, the 3D graphs of both analytical and approximate solutions are depicted which shows that the approximate solution has well estimated the analytical solution. In addition, in Figure 8, the 2D graphs of exact and numerical solutions for certain values $t(= 0.25, 0.50, 0.75)$ and for $\vartheta = 0.05$ are plotted. The solutions of both sets are close to each across the range. To further confirm this, Figure 9 is plotted by superimposing the graphs of Figure 8. As we can see, the graphs of the exact and approximate solutions overlap completely across the range of x and for some certain values of t and ϑ.

Table 3: Numerical experiments of Example 6.3.

	GS			*SOR*			*PSOR*		
$\mathbb{m}$	Iter	time	Error	Iter	time	Error	Iter	time	Error
256	1266	0.54	$7.64e-07$	1516	0.62	$7.64e-07$	170	0.18	$7.64e-07$
512	2174	3.14	$1.91e-07$	2595	3.38	$1.91e-07$	327	0.76	$1.91e-07$
1024	3634	18.57	$4.77e-08$	4316	21.91	$4.77e-08$	620	4.31	$4.77e-08$
2048	5844	121.71	$1.19e-08$	6893	141.45	$1.19e-08$	1174	37.91	$1.19e-08$
4096	8863	723	$2.94e-09$	10337	861.53	$2.94e-09$	2249	272.63	$2.94e-09$

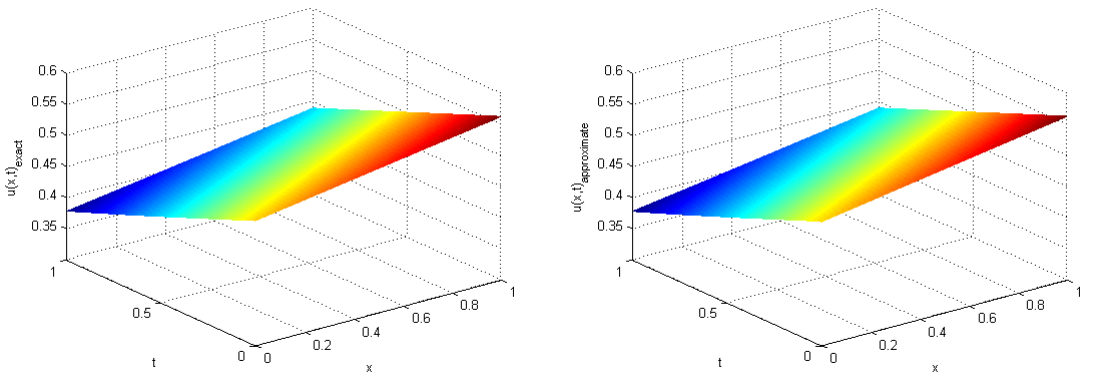

Figure 7: 3D plots of the exact solution (left) and the approximate solution (right) of Example 6.3 with $\vartheta = 0.05$.

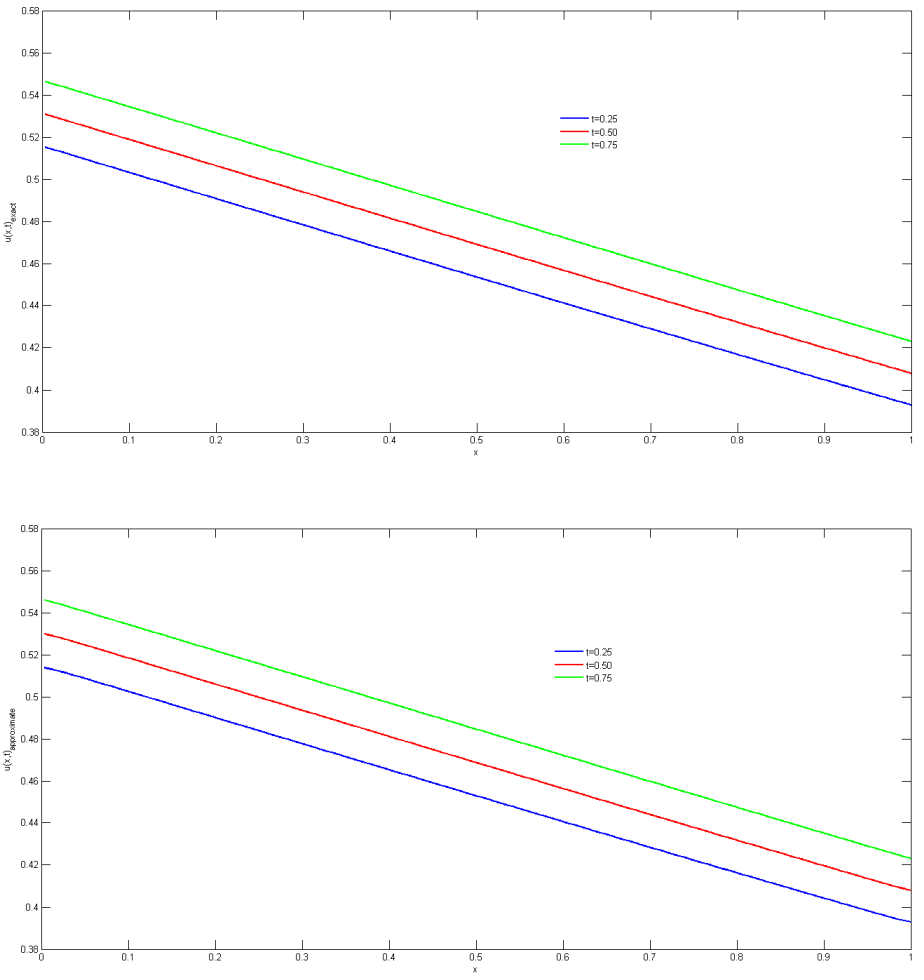

Figure 8: 2D plots of the exact solution (up) and the approximate solution (down) of Example 6.3 with $\vartheta = 0.05$ and for $t = 0.25, 0.50, 0.75$.

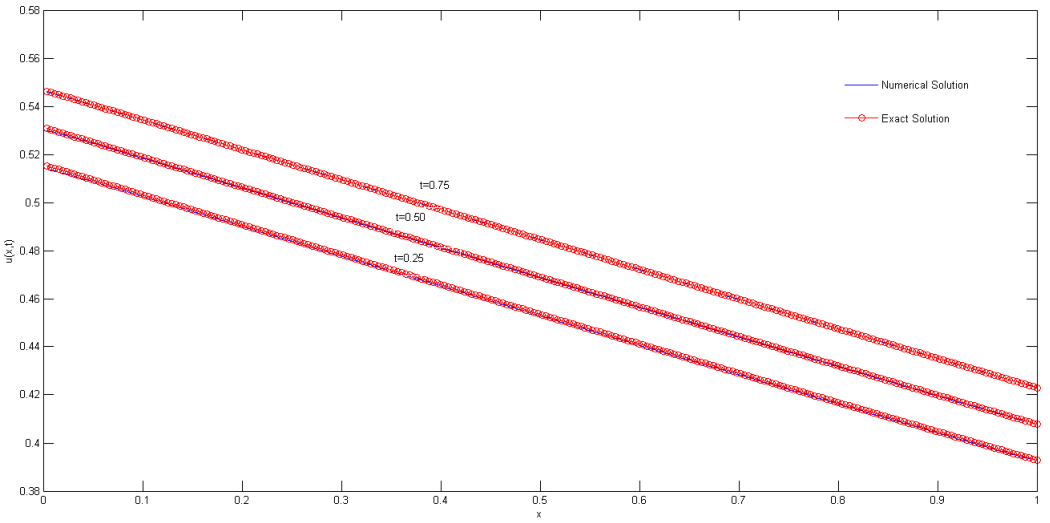

Figure 9: Compare the two-dimensional graphs of the exact and approximate solutions for Example 6.3 with $\vartheta = 0.05$ and for $t = 0.25, 0.50, 0.75$.

7 Discussion

In this research, we solved numerically the one-dimensional Burgers' equation using an improved SOR method. The method used to improve the SOR method was the use of a suitable preconditioner, the use of which leads to an increase in the speed of convergence and a decrease in the number of iterations in solving the system of approximate linear equations resulting from discretization of Burgers' equation by some finite difference formulas and nonlocal arithmetic mean scheme. The convergence and stability of the method were established and the method was found to be conditionally stable, second-order convergent in space, and first-order convergent in time. As observed, the obtained numerical results confirmed the theoretical results, meaning that by applying an effective preconditioner, we witnessed a reduction in the number of iterations and an increase in the convergence speed.

Acknowledgement The authors are grateful to anonymous referees for their remarks and suggestions that helped improve the manuscript.

Conflicts of Interest The authors do not receive any funds, grants, or any financial or non-financial benefits related to the publication of this article.

References

[1] Anisha & R. Rohila (2024). *National Conference on Emerging Trends in Science and Technology (NCETST-24)*. Srijan Samiti, Nashik, India.

[2] G. Arora & B. K. Singh (2013). Numerical solution of Burgers' equation with modified cubic B-spline differential quadrature method. *Applied Mathematics and Computation*, *224*, 166–177. https://doi.org/10.1016/j.amc.2013.08.071.

[3] A. Azad, E. Yaghoubi & A. Jafari (2024). Numerical solution of Burgers' equation with non-local boundary condition: Use of Keller-Box scheme. *Computational Methods for Differential Equations*, *12*(3), 425–437. https://doi.org/10.22034/cmde.2023.54380.2271.

[4] H. Bateman (1915). Some recent researches on the motion of fluids. *Monthly Weather Review*, *43*(4), 163–170. https://doi.org/10.1175/1520-0493(1915)43<163:SRROTM>2.0.CO;2.

[5] J. Biazar & H. Aminikhah (2009). Exact and numerical solutions for non-linear Burger's equation by VIM. *Mathematical and Computer Modelling*, *49*(7–8), 1394–1400. https://doi.org/10.1016/j.mcm.2008.12.006.

[6] J. M. Burgers (1939). Mathematical examples illustrating relations occurring in the theory of turbulent fluid motion. *Transaction of Royal Netherlands Academy of Science*, *17*(2), 1–53. http://www.dwc.knaw.nl/DL/publications/PU00011461.pdf.

[7] J. M. Burgers (1948). A mathematical model illustrating the theory of turbulence. *Advances in Applied Mechanics*, *1*, 171–199. https://doi.org/10.1016/S0065-2156(08)70100-5.

[8] M. Kapoor & V. Joshi (2021). A numerical regime for 1-D Burgers' equation using UAT tension B-spline differential quadrature method. *International Journal for Computational Methods in Engineering Science and Mechanics*, *22*(3), 181–192. http://dx.doi.org/10.1080/15502287.2021.1916175.

[9] N. Kumar, R. Majumdar & S. Singh (2019). Physics-based preconditioning of Jacobian free Newton Krylov for Burgers' equation using modified nodal integral method. *Progress in Nuclear Energy*, *117*, Article ID: 103104. https://doi.org/10.1016/j.pnucene.2019.103104.

[10] S. Kutluay, A. Esen & I. Dag (2004). Numerical solutions of the Burgers' equation by the least-squares quadratic B-spline finite element method. *Journal of Computational and Applied Mathematics*, *167*(1), 21–33. https://doi.org/10.1016/j.cam.2003.09.043.

[11] C. Li & D. J. Evans (1994). Improving the SOR method. *International Journal of Computer Mathematics*, *54*(3–4), 207–213. https://doi.org/10.1080/00207169408804352.

[12] I. E. Mhlanga & C. M. Khalique (2017). Travelling wave solutions and conservation laws of the Korteweg-de Vries-Burgers equation with power law nonlinearity. *Malaysian Journal of Mathematical Sciences*, *11*(S), 1–8.

[13] R. C. Mittal & R. Rohila (2018). Traveling and shock wave simulations in a viscous Burgers' equation with periodic boundary conditions. *International Journal of Applied and Computational Mathematics*, *4*, Article ID: 150. https://doi.org/10.1007/s40819-018-0582-y.

[14] R. C. Mittal & R. Rohila (2022). A numerical study of the Burgers' and Fisher's equations using Barycentric interpolation method. *International Journal of Numerical Methods for Heat & Fluid Flow*, *33*(2), 772–800. https://doi.org/10.1108/HFF-03-2022-0166.

[15] N. A. Mohamed (2019). Solving one-and two-dimensional unsteady Burgers' equation using fully implicit finite difference schemes. *Arab Journal of Basic and Applied Sciences*, *26*(1), 254–268. https://doi.org/10.1080/25765299.2019.1613746.

[16] F. A. Muhiddin & J. Sulaiman (2019). EGSOR iterative method for the fourth-order solution of one-dimensional convection-diffusion equations. *Malaysian Journal of Mathematical Sciences*, *13*(S), 15–26.

[17] F. Oz, R. K. S. S. Vuppala, K. Kara & F. Gaitan (2022). Solving Burgers' equation with quantum computing. *Quantum Information Processing*, *21*(1), Article ID: 30. https://doi.org/10.1007/s11128-021-03391-8.

[18] K. R. Raslan (2003). A collocation solution for Burgers equation using quadratic B-spline finite elements. *International Journal of Computer Mathematics*, *80*(7), 931–938. https://doi.org/10.1080/0020716031000079554.

[19] R. Rohila & R. C. Mittal (2018). Numerical study of reaction diffusion Fisher's equation by fourth order cubic B-spline collocation method. *Mathematical Sciences*, *12*(2), 79–89. https://doi.org/10.1007/s40096-018-0247-3.

[20] Y. Saad (2003). *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia.

[21] V. P. Shyaman, A. Sreelakshmi & A. Awasthi (2023). A higher order implicit adaptive finite point method for the Burgers' equation. *Journal of Difference Equations and Applications*, *29*(3), 235–269. https://doi.org/10.1080/10236198.2023.2197082.

[22] M. M. Xu, J. Sulaiman & L. H. Ali (2022). Refinement of SOR iterative method for the linear rational finite difference solution of second-order Fredholm integro-differential equations. *Malaysian Journal of Mathematical Sciences*, *16*(1), 105–117. https://doi.org/10.47836/mjms.16.1.09.

[23] M. Xu, R. H. Wang, J. H. Zhang & Q. Fang (2011). A novel numerical scheme for solving Burgers' equation. *Applied Mathematics and Computation*, *217*(9), 4473–4482. https://doi.org/10.1016/j.amc.2010.10.050.

[24] N. F. A. Zainal, J. Sulaiman, A. Saudi & N. A. Mat Ali (2023). Preconditioned successive over relaxation iterative method via semi-approximate approach for Burgers' equation. *Indonesian Journal of Electrical Engineering and Computer Science*, *29*(3), 1606–1613. http://doi.org/10.11591/ijeecs.v29.i3.pp1606-1613.

[25] T. Zhanlav, O. Chuluunbaatar & V. Ulziibayar (2015). Higher-order accurate numerical solution of unsteady Burgers' equation. *Applied Mathematics and Computation*, *250*, 701–707. https://doi.org/10.1016/j.amc.2014.11.013.