Advances in Continuous and Discrete Models

RESEARCH Open Access

Algorithms for coupled Burgers' equations by sharing characteristic curves within BSLM

Soyoon Bak[1] and Yonghyeon Jeon[2*]

*Correspondence:
yhjeon@hongik.ac.kr
[2]Mechatronics Research Center, Hongik University, Sejong, 30016, South Korea
Full list of author information is available at the end of the article

Abstract

This paper introduces a new perspective of the traditional view on the velocity of each physical particle in the coupled Burgers' equation in the backward semi-Lagrangian method (BSLM). The proposed methods reduce the number of Cauchy problems to be solved by observing a single virtual characteristic curve with a velocity. This can drastically reduce the computational cost of determining the departure point. Then, we solve the derived system reflected by the single virtual characteristic curve. Moreover, an efficient strategy for the derived linear system of equations is provided. Four examples are tested to demonstrate the adaptability and efficiency of the proposed method. The test results show that the proposed method has third- and fourth-order accuracy in time and space, respectively. In addition, compared with the existing method of solving the problem along two particles with different velocities, we confirm that the proposed method significantly reduces computational cost while maintaining accuracy well.

Keywords: Backward semi-Lagrangian method; Characteristic curve; Coupled Burgers' equations; Nonlinear Cauchy problem

1 Introduction

Under the effect of gravity, the sedimentation of two types of particle concentrations in fluid suspensions and colloids is described in the following viscous coupled Burgers' equations [9]:

$$\begin{cases} u_t - \nu_1 u_{xx} + \alpha_1 u u_x + \beta_1 (uv)_x = 0, \\ v_t - \nu_2 v_{xx} + \alpha_2 v v_x + \beta_2 (uv)_x = 0, \end{cases} \quad x \in [a,b], \quad t \in (t_0, T], \tag{1}$$

with the following initial and boundary conditions:

$$\begin{aligned} &u(t_0,x) := u_0(x), \qquad v(t_0,x) := v_0(x), \quad x \in [a,b], \\ &\begin{cases} u(t,a) = u_a(t), \\ u(t,b) = u_b(t), \end{cases} \qquad \begin{cases} v(t,a) = v_a(t), \\ v(t,b) = v_b(t), \end{cases} \quad t \in (t_0, T]. \end{aligned} \tag{2}$$

Coefficients α_k and β_k $(k = 1, 2)$ are real constants that describe the interaction of gravity and thermal fluctuation (Péclet number) and the velocities of the moving particles, respectively [6]. In addition, the positive real constants, ν_1 and ν_2, represent the viscosity coefficients related to the Reynolds numbers of u and v, respectively. The rate at which a particular reaction proceeds mainly depends on the reactant concentration. Suppose the reactant concentration is increased. In that case, the number of molecules or ions will increase, thereby increasing the number of collisions between them and the reaction rate. Thus, the solutions to the model problem simultaneously represent particle velocity and concentration. The governing equations are of advection–diffusion type, but the nonlinear coupling terms of u and v are not a common feature of this type.

Over the last 10 years, various numerical methods have been developed to solve the viscous coupled Burgers' equations. For example, Kapoor and Joshi developed a differential quadrature method with uniform algebraic trigonometric tension B-spline [12], Başhan applied a mixed method with the finite difference and differential quadrature method based on third-order modified cubic B-spline functions [5], Hussein and Kashkool used a weak Galerkin finite element method [10], Zhang et al. introduced an improved backward substitution method [21], Abdullah et al. developed a numerical procedure based on the cubic B-spline and the Hermite formula [1], Mohanty and Sharma presented a high accuracy two-level implicit method based on cubic spline approximation [17], Bhatt and Khaliq modified exponential time-differencing Runge–Kutta method based on Padé approximation [7], Kumar and Pandit proposed a composite scheme based on finite difference and Haar wavelets [13], Jiwari and Alshomrani developed a new collocation method based on modified cubic trigonometric B-spline function [11], Mohanty et al. proposed a two-level implicit compact operator method [16]. Among the abovementioned methods, there is a backward semi-Lagrangian (BSLM) to solve the model problem. It solves the system on fixed Eulerian spatial grids at every time step by tracing in the reverse time direction the trace of a particle observed from a Lagrangian viewpoint. This BSLM is called a coupled backward semi-Lagrangian method (CBSLM), which is the first attempt at using the BSLM to solve the model equations [2]. The CBSLM inherits the following well-known advantages of the traditional BSLM. The BSLM guarantees unconditional stability by implicitly dealing with the particle trajectories, allowing the use of a larger time step size than that in Eulerian methods [3, 4, 18]. It does not require domain remeshing, unlike the Lagrangian method. The CBSLM has second-order accuracy in time and fourth-order accuracy in space, and it implicitly treats the particle trajectories without any iteration process.

In the previous BSLM task, CBSLM, the virtual characteristics of $\pi(t)$ and $\phi(t)$ (or particle trajectories), respectively, satisfy the following nonlinear Cauchy problems:

$$\frac{d\pi(t)}{dt} = s_1\big(t, \pi(t)\big), \qquad \frac{d\phi(t)}{dt} = s_2\big(t, \phi(t)\big), \tag{3}$$

and the model equations are reformulated by the general form of the Lagrangian formulations:

$$\frac{d}{dt}u\big(t, \pi(t)\big) = f_1\big(t, \pi(t)\big), \qquad \frac{d}{dt}v\big(t, \phi(t)\big) = f_2\big(t, \phi(t)\big), \tag{4}$$

where

$$\begin{aligned} f_1\big(t,\pi(t)\big) &= \nu_1 u_{xx}\big(t,\pi(t)\big) - \alpha_1 u\big(t,\pi(t)\big)u_x\big(t,\pi(t)\big) \\ &\quad - \beta_1 (uv)_x\big(t,\pi(t)\big) + s_1\big(t,\pi(t)\big)u_x\big(t,\pi(t)\big), \\ f_2\big(t,\phi(t)\big) &= \nu_2 v_{xx}\big(t,\phi(t)\big) - \alpha_2 v\big(t,\phi(t)\big)v_x\big(t,\phi(t)\big) \\ &\quad - \beta_2 (uv)_x\big(t,\phi(t)\big) + s_2\big(t,\phi(t)\big)v_x\big(t,\phi(t)\big). \end{aligned} \tag{5}$$

In the CBSLM [2], the particle's velocities are fixed as $s_1(t,\pi(t)) = \alpha_1 u(t,\pi(t)) + \beta_1 v(t,\pi(t))$ and $s_2(t,\phi(t)) = \alpha_2 v(t,\phi(t)) + \beta_2 u(t,\phi(t))$. The CBSLM process for simultaneously solving (3) and (4) (or model equations) can be divided into three parts: (i) finding the departure points at the previous time steps by solving the Cauchy problems; (ii) interpolating the function values at the departure points; and (iii) solving the systems to discretize (4). Among these processes, in most BSLMs, as well as in CBSLM, interpolation processes incur the highest computational cost. In CBSLM, interpolation is required even in the process of determining the departure points. In particular, a more significant number of departure points and interpolations are inevitably required to obtain high-order accuracy in time and space. Thus, we describe a high-order (third-order) accuracy version of CBSLM, CBSLM3, and propose a faster algorithm while maintaining the accuracy of CBSLM3. Hereafter, we denote the existing CBSLM as CBSLM2 to distinguish it from CBSLM3.

Our main goal is to develop a novel methodology, without losing accuracy, than both CBSLM2 and CBSLM3 that individually consider all characteristic curves of each particle. For this, we propose a new method for choosing the velocity functions. The proposed method observes a single virtual characteristic curve with velocity s. First, we propose the choice of three versions of s and then present their algorithms for solving the corresponding Lagrangian form. Additionally, to construct high-order algorithms, the third-order backward differentiation formula and fourth-order finite difference method are applied to discretize the Lagrangian formulations. We apply the third-order multistep error correction method to solve the nonlinear Cauchy problem. Note that the typical Lagrange interpolation is used for estimating the required function values at nongrid points in all proposed methods. In particular, we improve two parts among the three parts of BSLM by (i) handling the number of Cauchy problems, and (ii) providing an efficient strategy for solving a linear system of equations. To improve these parts, we reduce the number of the Cauchy problems to be solved by sharing the characteristic curve arriving at point x as only one. We additionally design an explicit formula that converts to a pentadiagonal one for the system of equations occurring at every time integration step. This formula is used to reduce the computational cost of solving the linear system of equations.

The remainder of this paper is organized as follows. In Sect. 2, we review and extend CBSM2 for solving (1) and the Cauchy problem solver with third-order accuracy, which individually determines all characteristic curves of each particle. In Sect. 3, we introduce three proposed schemes for solving the model problem and provide a fast solver for derived nonlinear systems. The numerical simulations are illustrated in Sect. 4 to demonstrate the efficiency of the proposed algorithms. Finally, the conclusions are summarized in the last section.

2 Preliminary results

In this section, we introduce CBSLM3, which is an extended version of CBSLM2 with a third-order temporal accuracy for solving the model equations. To do that, we first define time sequences $t_n := t_0 + nh$, $n = 0, 1, 2, \dots, N$, where $h := (T - t_0)/N$ is the time step size. Furthermore, the uniformly divided spatial grid sequence is defined as $x_j = a + j\Delta x$, $\Delta x = (b - a)/J$. The time variable evaluated function $u(t_n, x)$ is denoted by superscript $u^n(x)$.

2.1 BSLM with BDF

Now, we provide the BSLM algorithm with third-order temporal accuracy to obtain the solutions at time t_{n+1}. The first step of BSLM is evaluating $x = x_j$ at $t = t_{n+1}$ for the Lagrangian formulations (4) with (5). As mentioned in the Introduction, CBSLM chooses the fixed virtual velocities:

$$\begin{aligned} &s_1\big(t, \pi(t)\big) = \alpha_1 u\big(t, \pi(t)\big) + \beta_1 v\big(t, \pi(t)\big), \\ &s_2\big(t, \phi(t)\big) = \alpha_2 v\big(t, \phi(t)\big) + \beta_2 u\big(t, \phi(t)\big). \end{aligned}$$

Then, along with the characteristic curves $\pi(t)$ and $\phi(t)$ satisfying nonlinear Cauchy problems (3) with initial values $\pi_j(t_{n+1}) = x_j$ and $\phi_j(t_{n+1}) = x_j$, the Lagrangian formulations (4) are given by

$$\begin{aligned} &\frac{d}{dt}u^{n+1}(x_j) = f_1^{n+1}(x_j), \quad f_1^{n+1}(x_j) = \nu_1 u_{xx}^{n+1}(x_j) - \beta_1 v_x^{n+1}(x_j) u^{n+1}(x_j), \\ &\frac{d}{dt}v^{n+1}(x_j) = f_2^{n+1}(x_j), \quad f_2^{n+1}(x_j) = \nu_2 v_{xx}^{n+1}(x_j) - \beta_2 u_x^{n+1}(x_j) v^{n+1}(x_j). \end{aligned} \tag{6}$$

To obtain the solutions u and v at $(t, x) = (t_{n+1}, x_j)$, we employ the third-order backward differentiation formula for the total derivatives du/dt and dv/dt in (6). Then, we obtain the following system of nonlinear boundary value problems:

$$\begin{aligned} &\frac{11}{6h}u^{n+1}(x_j) - f_1^{n+1}(x_j) = r_1^{n+1}(x_j) + \mathcal{O}\big(\Delta t^3\big), \\ &\frac{11}{6h}v^{n+1}(x_j) - f_2^{n+1}(x_j) = r_2^{n+1}(x_j) + \mathcal{O}\big(\Delta t^3\big), \end{aligned} \tag{7}$$

where

$$\begin{cases} r_1^{n+1}(x_j) = 3u^n(\pi_j(t_n)) - \frac{3}{2}u^{n-1}(\pi_j(t_{n-1})) + \frac{1}{3}u^{n-2}(\pi_j(t_{n-2})), \\ r_2^{n+1}(x_j) = 3v^n(\phi_j(t_n)) - \frac{3}{2}v^{n-1}(\phi_j(t_{n-1})) + \frac{1}{3}v^{n-2}(\phi_j(t_{n-2})). \end{cases} \tag{8}$$

To discretize the nonlinear boundary value problems (7), we use the fourth-order finite difference for first and second partial derivatives with weight matrices $\mathcal{D}_1$ and $\mathcal{D}_2$ and vectors $\mathbf{d}_{2,u}$ and $\mathbf{d}_{2,v}$ given in the appendix. Then, we obtain the semidiscretized matrix system

$$\begin{aligned} &A_1\big(\hat{\mathbf{v}}_{\text{ex}}^{n+1}\big)\hat{\mathbf{u}}^{n+1} = \mathbf{r}_1^{n+1} + \nu_1 \mathbf{d}_{2,u} + \mathcal{O}\big(h^3 + \Delta x^4\big), \\ &A_2\big(\hat{\mathbf{u}}^{n+1}\big)\hat{\mathbf{v}}^{n+1} = \mathbf{r}_2^{n+1} + \nu_2 \mathbf{d}_{2,v} + \mathcal{O}\big(h^3 + \Delta x^4\big), \end{aligned} \tag{9}$$

where

$$\begin{aligned} &A_\kappa(\mathbf{w}) := \frac{11}{6h}\mathcal{I} - \nu_\kappa \mathcal{D}_2 + \beta_\kappa \operatorname{diag}(\mathcal{D}_1\mathbf{w} + \mathbf{d}_{1,w}), \quad \kappa = 1,2, \\ &\mathbf{r}_\kappa^{n+1} = \left[r_\kappa^{n+1}(x_1), \dots, r_\kappa^{n+1}(x_{J-1})\right]^{\mathrm{T}}, \qquad \hat{\mathbf{v}}_{\mathrm{ex}}^{n+1} := 3\hat{\mathbf{v}}^n - 3\hat{\mathbf{v}}^{n-1} + \hat{\mathbf{v}}^{n-2}. \end{aligned} \tag{10}$$

Here, the notation $\operatorname{diag}(\mathbf{w})$ denotes a diagonal matrix, the diagonal entries of which are those of vector $\mathbf{w}$ and vector $\mathbf{d}_{1,w}$ is defined in the appendix. The hat notation on vector $\hat{\mathbf{w}}$ means a vector without both ends of column vector $\mathbf{w}$.

Note that the departure points $\pi_j(t_{n-l})$ and $\phi_j(t_{n-l})$, $l = 0,1,2$, do not usually coincide with the spatial grid point; consequently, the function values at these points $u^{n-k}(\pi_j(t_{n-l}))$ and $v^{n-k}(\phi_j(t_{n-l}))$ require an appropriate interpolation. Herein, we use the typical Lagrange interpolation with sixth-order accuracy. The approximate vector, $\mathbf{R}_\kappa^{n+1}$, of $\mathbf{r}_\kappa^{n+1}$ is defined as follows:

$$\begin{aligned} &\mathbf{R}_\kappa^{n+1} := \left[R_\kappa^{n+1}(x_1), \dots, R_\kappa^{n+1}(x_{J-1})\right]^{\mathrm{T}}, \quad \kappa = 1,2, \\ &R_1^{n+1}(x_j) = 3\mathcal{L}\left[\mathbf{U}^n\right]\left(\pi_j^n\right) - \frac{3}{2}\mathcal{L}\left[\mathbf{U}^{n-1}\right]\left(\pi_j^{n-1}\right) + \frac{1}{3}\mathcal{L}\left[\mathbf{U}^{n-2}\right]\left(\pi_j^{n-2}\right), \quad j = 1, \dots, J-1, \\ &R_2^{n+1}(x_j) = 3\mathcal{L}\left[\mathbf{V}^n\right]\left(\phi_j^n\right) - \frac{3}{2}\mathcal{L}\left[\mathbf{V}^{n-1}\right]\left(\phi_j^{n-1}\right) + \frac{1}{3}\mathcal{L}\left[\mathbf{V}^{n-2}\right]\left(\phi_j^{n-2}\right), \quad j = 1, \dots, J-1, \end{aligned} \tag{11}$$

where $\mathcal{L}$ is the interpolation operator using typical Lagrange polynomials of the sixth degree and is defined in the appendix. After applying the Lagrange interpolation to approximate $\mathbf{r}_\kappa^{n+1}$ and dropping the truncation error, we achieved a fully-discretized system for the approximation $\mathbf{U}^{n+1}$ and $\mathbf{V}^{n+1}$ of $\mathbf{u}^{n+1}$ and $\mathbf{v}^{n+1}$, respectively, namely

$$\begin{aligned} &A_1\left(\hat{\mathbf{V}}_{\mathrm{ex}}^{n+1}\right)\hat{\mathbf{U}}^{n+1} = \mathbf{R}_1^{n+1} + \nu_1 \mathbf{d}_{2,u}, \\ &A_2\left(\hat{\mathbf{U}}^{n+1}\right)\hat{\mathbf{V}}^{n+1} = \mathbf{R}_2^{n+1} + \nu_2 \mathbf{d}_{2,v} \end{aligned} \tag{12}$$

with $\hat{\mathbf{V}}_{\mathrm{ex}}^{n+1} = 3\hat{\mathbf{V}}^n - 3\hat{\mathbf{V}}^{n-1} + \hat{\mathbf{V}}^{n-2}$. The final process for implementing this system involves the determination of the locations of departure points. This process is essential not only for CBSLM3 but also for the proposed methods.

2.2 A Cauchy problem solver finding departure points

In this section, we introduce a strategy for solving the Cauchy problems to complete the discretized system (12). Hereafter, we use the notation θ_j to represent a characteristic curve π_j or ϕ_j. Recall that the position $\theta_j(t_{n-l}) := \theta(t_{n+1}, x_j; t_{n-l})$ $(l = 0,1,2)$ of the particle arriving at x_j $(j = 1, \dots, J-1)$ on time level $t = t_{n+1}$ is referred to as the departure point. The trajectory of the departure point $\theta_j(t)$ satisfies the Cauchy problem of

$$\frac{d\theta_j(t)}{dt} = s\left(t, \theta_j(t)\right), \quad t < t_{n+1}; \qquad \theta_j(t_{n+1}) = x_j. \tag{13}$$

Herein, the particle velocity s is combined with the solutions, u and v of the model problem, which is strong nonlinear. To resolve this nonlinearity, we employ the multistep error correction method introduced in [2]. The mechanism of the error correction method starts with defining a linear local approximation

$$y(t) = x_j + (t - t_{n+1})S^n(x_j), \quad t < t_{n+1}, \tag{14}$$

where $S^n(x_j)$ is an approximation for $s(t_n, x_j)$. The local approximation leads a perturbation $\psi(t) := \theta_j(t) - y(t)$ to be corrected, which satisfies the following asymptotic initial value problem:

$$\begin{aligned} \frac{d\psi}{dt}(t) &= s\big(t, y(t)\big) + s_x\big(t_n, y(t_n)\big)\psi(t) - s(t_n, x_j) + \mathcal{O}\big(h^3\big), \quad t < t_{n+1} \\ &\approx s\big(t, y(t)\big) + \mathcal{L}_x\big[\mathbf{S}^n\big]\big(y(t_n)\big)\psi(t) - S^n(x_j) \end{aligned} \tag{15}$$

with the initial value $\psi(t_{n+1}) = 0$. Here, $\mathbf{S}^n$ is a column vector defined as $\mathbf{S}^n := [S^n(x_0), S^n(x_1), \ldots, S^n(x_J)]^T$ and $\mathcal{L}_x$ is introduced in the appendix as the derivative operator of $\mathcal{L}$.

To estimate numerical solutions for the asymptotic initial value problem (16), the Radau IIA-type Runge–Kutta method with third-order accuracy is expressed as presented below

$$\begin{array}{c|c} \mathbf{c} & \mathcal{A} \\ \hline & \mathbf{b}^{\mathrm{T}} \end{array} = \begin{array}{c|ccc} 1/3 & 23/36 & -4/9 & 5/36 \\ 2/3 & 7/9 & -2/9 & 1/9 \\ 1 & 3/4 & 0 & 1/4 \\ \hline & 3/4 & 0 & 1/4 \end{array} \tag{16}$$

The above is applied to the problem, inducing the following linear system:

$$\begin{aligned} \big(\mathcal{I}_3 + 3h\mathcal{L}_x\big[\mathbf{S}^n\big]\big(y(t_n)\big)\mathcal{A}\big)\boldsymbol{\psi} &= -3h\mathcal{A}\mathbf{g} + \mathcal{O}\big(h^4\big) \\ &\approx -3h\mathcal{A}\mathbf{G}, \end{aligned} \tag{17}$$

where

$$\begin{aligned} \boldsymbol{\psi} &= \big[\psi(t_n), \psi(t_{n-1}), \psi(t_{n-2})\big]^{\mathrm{T}}, \\ \mathbf{g} &= \big[s\big(t_n, y(t_n)\big) - s(t_n, x_j), s\big(t_{n-1}, y(t_{n-1})\big) - s(t_n, x_j), s\big(t_{n-2}, y(t_{n-2})\big) - s(t_n, x_j)\big]^{\mathrm{T}}, \\ \mathbf{G} &= \big[\mathcal{L}\big[\mathbf{S}^n\big]\big(y(t_n)\big) - S^n(x_j), \mathcal{L}\big[\mathbf{S}^{n-1}\big]\big(y(t_{n-1})\big) - S^n(x_j), \mathcal{L}\big[\mathbf{S}^{n-2}\big]\big(y(t_{n-2})\big) - S^n(x_j)\big]^{\mathrm{T}}. \end{aligned} \tag{18}$$

Finally, the definition of the perturbation ψ allows us to obtain the approximation θ_j^{n-l} of $\theta_j(t_{n-l})$ by

$$\theta_j^{n-l} \approx y(t_{n-l}) + \psi(t_{n-l}), \quad l = 0, 1, 2. \tag{19}$$

Remark 1 To find the approximation of departure points π_j^n and ϕ_j^n in (11), two linear systems of the form (17) are solved for π and ϕ, respectively, in CBSLM3. CBSLM3 requires $2 \times J$ Jacobian calculations and $6 \times J$ function evaluations at each time step to construct these systems. The proposed schemes, introduced in the next section, require exactly half of the interpolation process of CBSLM3 to construct the system by sharing a single virtual characteristic curve. In addition, the computational cost to solve this system is also reduced by half.

3 Sharing characteristic curve schemes for solving the model problem

In the introduction, we have described the general form of the Lagrangian formulations (4) with (5), for coupled Burgers' equations. The Lagrangian formulations depend on how

the velocity functions s_1 and s_2 are set. This section presents a sharing characteristic curve semi-Lagrangian (SC-SL) scheme by choosing the velocity functions. The proposed method observes a single virtual characteristic curve $\theta_j(t)$ with velocity s. Here, velocity s is chosen by considering the simplicity of the resulting Lagrangian formulations, (4) with (5). That is, the velocity s is chosen to reduce the computational cost by selecting a single virtual characteristic curve and to minimize the number of terms to be approximated in (5). We propose the choice of three versions of s in the following subsection and then present their algorithms for solving the Lagrangian formulations accordingly.

3.1 Velocity type 1: SC-SL$_1$

The first choice of the velocity $s(t,\theta_j(t))$ along the single characteristic curve $\theta_j(t)$ for each grid x_j is defined by

$$s\big(t,\theta_j(t)\big) = \alpha_1 u\big(t,\theta_j(t)\big) + \alpha_2 v\big(t,\theta_j(t)\big). \tag{20}$$

Based on this choice, the Lagrangian formulations can be defined as

$$\frac{d}{dt}u\big(t,\theta_j(t)\big) = f_1\big(t,\theta_j(t)\big), \qquad \frac{d}{dt}v\big(t,\theta_j(t)\big) = f_2\big(t,\theta_j(t)\big) \tag{21}$$

with

$$\begin{aligned} f_1\big(t,\theta_j(t)\big) &= \nu_1 u_{xx}\big(t,\theta_j(t)\big) + \gamma_{21} v\big(t,\theta_j(t)\big) u_x\big(t,\theta_j(t)\big) - \beta_1 v_x\big(t,\theta_j(t)\big) u\big(t,\theta_j(t)\big), \\ f_2\big(t,\theta_j(t)\big) &= \nu_2 v_{xx}\big(t,\theta_j(t)\big) + \gamma_{12} u\big(t,\theta_j(t)\big) v_x\big(t,\theta_j(t)\big) - \beta_2 u_x\big(t,\theta_j(t)\big) v\big(t,\theta_j(t)\big), \end{aligned}$$

where $\gamma_{12} := \alpha_1 - \beta_2$ and $\gamma_{21} := \alpha_2 - \beta_1$. After evaluating $t = t_{n+1}$ in (21), the third-order backward differentiation formula is applied to total derivatives in (21). From the fact that $\theta_j(t_{n+1}) = x_j$, we can obtain the following system of nonlinear boundary value problems:

$$\begin{aligned} \frac{11}{6h}u^{n+1}(x_j) - f_1^{n+1}(x_j) &= r_1^{n+1}(x_j) + \mathcal{O}\big(h^3\big), \\ \frac{11}{6h}v^{n+1}(x_j) - f_2^{n+1}(x_j) &= r_2^{n+1}(x_j) + \mathcal{O}\big(h^3\big). \end{aligned} \tag{22}$$

To discretize (22), the fourth-order finite difference weight matrices $\mathcal{D}_1$ and $\mathcal{D}_2$, defined in the appendix, are used for first and second partial derivatives. Then, the semidiscretized matrix system for (22) is obtained by

$$\begin{aligned} A_1\big(\hat{\mathbf{v}}^{n+1}\big)\hat{\mathbf{u}}^{n+1} &= \mathbf{r}_1^{n+1} + \nu_1 \mathbf{d}_{2,u} + \gamma_{21}\operatorname{diag}\big(\hat{\mathbf{v}}^{n+1}\big)\mathbf{d}_{1,u} + \mathcal{O}\big(h^3 + \Delta x^4\big), \\ A_2\big(\hat{\mathbf{u}}^{n+1}\big)\hat{\mathbf{v}}^{n+1} &= \mathbf{r}_2^{n+1} + \nu_2 \mathbf{d}_{2,v} + \gamma_{12}\operatorname{diag}\big(\hat{\mathbf{u}}^{n+1}\big)\mathbf{d}_{1,v} + \mathcal{O}\big(h^3 + \Delta x^4\big), \end{aligned} \tag{23}$$

where

$$\begin{aligned} A_\kappa(\mathbf{w}) &:= \frac{11}{6h}\mathcal{I} - \nu_\kappa \mathcal{D}_2 - \gamma_{3-\kappa,\kappa}\operatorname{diag}(\mathbf{w})\mathcal{D}_1 + \beta_\kappa \operatorname{diag}(\mathcal{D}_1\mathbf{w} + \mathbf{d}_{1,w}), \\ \mathbf{r}_\kappa^{n+1} &:= \big[r_\kappa^{n+1}(x_1),\dots,r_\kappa^{n+1}(x_{J-1})\big]^{\mathrm{T}}, \quad \kappa = 1,2. \end{aligned}$$

To approximate $r_\kappa^{n+1}(x_j)$, we use the Lagrange interpolation and drop the truncation errors in (23). Then, the fully-discretized system is obtained by

$$\begin{aligned} A_1\big(\hat{\mathbf{V}}_{\mathrm{ex}}^{n+1}\big)\hat{\mathbf{U}}^{n+1} &= \mathbf{R}_1^{n+1} + \nu_1\mathbf{d}_{2,u} + \gamma_{21}\,\mathrm{diag}\big(\hat{\mathbf{V}}_{\mathrm{ex}}^{n+1}\big)\mathbf{d}_{1,u}, \\ A_2\big(\hat{\mathbf{U}}^{n+1}\big)\hat{\mathbf{V}}^{n+1} &= \mathbf{R}_2^{n+1} + \nu_2\mathbf{d}_{2,v} + \gamma_{12}\,\mathrm{diag}\big(\hat{\mathbf{U}}^{n+1}\big)\mathbf{d}_{1,v} \end{aligned} \tag{24}$$

with

$$\begin{aligned} &\mathbf{R}_\kappa^{n+1} := \big[R_\kappa^{n+1}(x_1),\dots,R_\kappa^{n+1}(x_{J-1})\big]^{\mathrm{T}}, \quad \kappa = 1,2, \\ &R_1^{n+1}(x_j) = 3\mathcal{L}\big[\mathbf{U}^n\big]\big(\theta_j^n\big) - \frac{3}{2}\mathcal{L}\big[\mathbf{U}^{n-1}\big]\big(\theta_j^{n-1}\big) + \frac{1}{3}\mathcal{L}\big[\mathbf{U}^{n-2}\big]\big(\theta_j^{n-2}\big), \\ &R_2^{n+1}(x_j) = 3\mathcal{L}\big[\mathbf{V}^n\big]\big(\theta_j^n\big) - \frac{3}{2}\mathcal{L}\big[\mathbf{V}^{n-1}\big]\big(\theta_j^{n-1}\big) + \frac{1}{3}\mathcal{L}\big[\mathbf{V}^{n-2}\big]\big(\theta_j^{n-2}\big). \end{aligned}$$

3.2 Velocity type 2: SC-SL$_2$

For each x_j, the second choice of the velocity $s(t,\theta_j(t))$ is

$$s\big(t,\theta_j(t)\big) = (\alpha_1+\beta_2)u\big(t,\theta_j(t)\big). \tag{25}$$

Then, the Lagrangian formulations are given by

$$\frac{d}{dt}u\big(t,\theta_j(t)\big) = f_1\big(t,\theta_j(t)\big), \qquad \frac{d}{dt}v\big(t,\theta_j(t)\big) = f_2\big(t,\theta_j(t)\big) \tag{26}$$

with

$$\begin{aligned} f_1(t,x) &= \nu_1 u_{xx}(t,x) + \big(\beta_2 u(t,x) - \beta_1 v(t,x)\big)u_x(t,x) - \beta_1 v_x(t,x)u(t,x), \\ f_2(t,x) &= \nu_2 v_{xx}(t,x) + \big(\alpha_1 u(t,x) - \alpha_2 v(t,x)\big)v_x(t,x) - \beta_2 u_x(t,x)v(t,x). \end{aligned}$$

Similar to the process of deriving (24) from (21), the following fully-discretized system can be obtained from (26):

$$\begin{aligned} A_1\big(\hat{\mathbf{V}}_{\mathrm{ex}}^{n+1}\big)\hat{\mathbf{U}}^{n+1} &= \mathbf{R}_1^{n+1} + \nu_1\mathbf{d}_{2,u} + \mathrm{diag}\big(\beta_2\hat{\mathbf{U}}_{\mathrm{ex}}^{n+1} - \beta_1\hat{\mathbf{V}}_{\mathrm{ex}}^{n+1}\big)\mathbf{d}_{1,u}, \\ A_2\big(\hat{\mathbf{U}}^{n+1}\big)\hat{\mathbf{V}}^{n+1} &= \mathbf{R}_2^{n+1} + \nu_2\mathbf{d}_{2,v} + \mathrm{diag}\big(\alpha_1\hat{\mathbf{U}}^{n+1} - \alpha_2\hat{\mathbf{V}}_{\mathrm{ex}}^{n+1}\big)\mathbf{d}_{1,v}, \end{aligned} \tag{27}$$

where

$$\begin{aligned} A_1(\mathbf{w}) &= \frac{11}{6h}\mathcal{I} - \nu_1\mathcal{D}_2 - \mathrm{diag}\big(\beta_2\hat{\mathbf{U}}_{\mathrm{ex}}^{n+1} - \beta_1\mathbf{w}\big)\mathcal{D}_1 + \beta_1\,\mathrm{diag}(\mathcal{D}_1\mathbf{w} + \mathbf{d}_{1,w}), \\ A_2(\mathbf{w}) &= \frac{11}{6h}\mathcal{I} - \nu_2\mathcal{D}_2 - \mathrm{diag}\big(\alpha_1\mathbf{w} - \alpha_2\hat{\mathbf{V}}_{\mathrm{ex}}^{n+1}\big)\mathcal{D}_1 + \beta_2\,\mathrm{diag}(\mathcal{D}_1\mathbf{w} + \mathbf{d}_{1,w}). \end{aligned} \tag{28}$$

3.3 Velocity type 3: SC-SL$_3$

The last choice of the velocity $s(t,\theta_j(t))$ is

$$s\big(t,\theta_j(t)\big) = \alpha_1 u\big(t,\theta_j(t)\big) + \beta_1 v\big(t,\theta_j(t)\big). \tag{29}$$

Then, the Lagrangian formulations are given by

$$\frac{d}{dt}u\big(t,\theta_j(t)\big)=f_1\big(t,\theta_j(t)\big),\qquad \frac{d}{dt}v\big(t,\theta_j(t)\big)=f_2\big(t,\theta_j(t)\big), \tag{30}$$

with

$$\begin{aligned}&f_1(t,x)=\nu_1u_{xx}(t,x)-\beta_1v_x(t,x)u(t,x),\\&f_2(t,x)=\nu_2v_{xx}(t,x)-\big(-\gamma_{12}u(t,x)+\gamma_{21}v(t,x)\big)v_x(t,x)-\beta_2u_x(t,x)v(t,x).\end{aligned}$$

Similar to the process of deriving (24) from (21) in the previous subsection, we can obtain the following fully-discretized system from (30):

$$\begin{aligned}&A_1\big(\hat{\mathbf{V}}_{\mathrm{ex}}^{n+1}\big)\hat{\mathbf{U}}^{n+1}=\mathbf{R}_1^{n+1}+\nu_1\mathbf{d}_{2,u},\\&A_2\big(\hat{\mathbf{U}}^{n+1}\big)\hat{\mathbf{V}}^{n+1}=\mathbf{R}_2^{n+1}+\nu_2\mathbf{d}_{2,v}-\operatorname{diag}\big(-\gamma_{12}\hat{\mathbf{U}}_{\mathrm{ex}}^{n+1}+\gamma_{21}\hat{\mathbf{V}}^{n+1}\big)\mathbf{d}_{1,v},\end{aligned} \tag{31}$$

where

$$\begin{aligned}&A_1(\mathbf{w})=\frac{11}{6h}\mathcal{I}-\nu_1\mathcal{D}_2+\beta_1\operatorname{diag}(\mathcal{D}_1\mathbf{w}+\mathbf{d}_{1,w}),\\&A_2(\mathbf{w})=\frac{1}{6h}\mathcal{I}-\nu_2\mathcal{D}_2+\operatorname{diag}\big(-\gamma_{12}\mathbf{w}+\gamma_{21}\mathbf{V}^{n+1}\big)\mathcal{D}_1+\beta_2\operatorname{diag}(\mathcal{D}_1\mathbf{w}+\mathbf{d}_{1,w}).\end{aligned} \tag{32}$$

Remark 2 We set the virtual particle velocity s according to the amplitudes of the initial values of u and v. For example, if the amplitude of u is larger, then we choose SC-SL$_3$. In the opposite case, we can define the velocity $s(t,\theta_j(t))=\alpha_2v(t,\theta_j(t))+\beta_2u(t,\theta_j(t))$. Then, $\hat{\mathbf{U}}^{n+1}$, $\hat{\mathbf{V}}^{n+1}$, and the coefficients of the system (31) are naturally exchanged in the reversed order.

3.4 Solver of the nearly pentadiagonal system

In this subsection, we discuss a strategy for solving the nonlinear system of equations (9), (23), (31), and (27). The $A_1(\cdot)$ and $A_2(\cdot)$ matrices of these systems have different elements but the same size and shape. Representing those systems, we define the following system:

$$A\mathbf{w}=\mathbf{r}, \tag{33}$$

where the coefficient matrix and loading vector are given as

$$A=\begin{bmatrix}p_1&p_2&p_3&p_4&p_5&&&&\\b_2&c_2&d_2&e_2&0&&&\mathbf{0}&\\a_3&b_3&c_3&d_3&e_3&&&&\\&a_4&b_4&c_4&d_4&e_4&&&\\&&\ddots&\ddots&\ddots&\ddots&\ddots&&\\&&&a_{J-4}&b_{J-4}&c_{J-4}&d_{J-4}&e_{J-4}&\\&&&&a_{J-3}&b_{J-3}&c_{J-3}&d_{J-3}&e_{J-3}\\&\mathbf{0}&&&0&a_{J-2}&b_{J-2}&c_{J-2}&d_{J-2}\\&&&&q_5&q_4&q_3&q_2&q_1\end{bmatrix}, \tag{34}$$

$\mathbf{r}=[r_1,r_2,\ldots,r_{J-1}]^{\mathrm{T}}$.

The matrix A is named a nearly pentadiagonal matrix for convenience.

Now, we design an explicit formula for converting the nearly pentadiagonal system (33) to a fully pentadiagonal system, allowing it to be solved using an advanced Thomas algorithm [8]. First, we reduce the bandwidth of system (33) by 2. The formula is based on applying a part of Gaussian elimination to p_4, p_5, q_4, and q_5. Then, we can easily reduce the nearly pentadiagonal system (33) to a pentadiagonal system as follows:

$$\tilde{A}\mathbf{w} = \tilde{\mathbf{r}}, \tag{35}$$

where

$$\begin{aligned}
\tilde{A} &= \begin{bmatrix}
\tilde{p}_1 & \tilde{p}_2 & \tilde{p}_3 & & & & \\
b_2 & c_2 & d_2 & e_2 & & \mathbf{0} & \\
a_3 & b_3 & c_3 & d_3 & e_3 & & \\
& \ddots & \ddots & \ddots & \ddots & \ddots & \\
& & a_{J-3} & b_{J-3} & c_{J-3} & d_{J-3} & e_{J-3} \\
& \mathbf{0} & & a_{J-2} & b_{J-2} & c_{J-2} & d_{J-2} \\
& & & & \tilde{q}_3 & \tilde{q}_2 & \tilde{q}_1
\end{bmatrix}, \\
\tilde{\mathbf{r}} &= \begin{bmatrix}
r_1 - \frac{p_5}{e_3} r_3 - \frac{\hat{p}_4}{e_2} r_2 \\
r_2 \\
r_3 \\
\vdots \\
r_{J-3} \\
r_{J-2} \\
r_{J-1} - \frac{q_5}{a_{J-2}} r_{J-3} - \frac{\hat{q}_4}{a_{J-1}} r_{J-2}
\end{bmatrix}.
\end{aligned} \tag{36}$$

Here, the quantities $\hat{p}_k$, $\tilde{p}_k$, $\hat{q}_k$, and $\tilde{q}_k$, $k = 1, 2, 3, 4$, are defined as

$$\begin{aligned}
\{\hat{p}_1, \hat{p}_2, \hat{p}_3, \hat{p}_4\} &= \{p_1, p_2, p_3, p_4\} - \frac{p_5}{e_3}\{a_3, b_3, c_3, d_3\}, \\
\{\tilde{p}_1, \tilde{p}_2, \tilde{p}_3\} &= \{\hat{p}_1, \hat{p}_2, \hat{p}_3\} - \frac{\hat{p}_4}{e_2}\{b_2, c_2, d_2\}, \\
\{\hat{q}_4, \hat{q}_3, \hat{q}_2, \hat{q}_1\} &= \{q_4, q_3, q_2, q_1\} - \frac{q_5}{a_{J-3}}\{b_{J-3}, c_{J-3}, d_{J-3}, e_{J-3}\}, \\
\{\tilde{q}_3, \tilde{q}_2, \tilde{q}_1\} &= \{\hat{q}_3, \hat{q}_2, \hat{q}_1\} - \frac{\hat{q}_4}{a_{J-2}}\{b_{J-2}, c_{J-2}, d_{J-2}\}.
\end{aligned} \tag{37}$$

4 Numerical results and discussions

To verify the accuracy and computational costs of the proposed schemes, we test four coupled Burgers' systems. In particular, we will show the adaptability of the proposed method by dealing with cases where the two solutions u and v are the same and different. For this, we record the maximum error $\mathrm{E}_\infty(w)$ given by

$$\mathrm{E}_\infty(w) = \max_j \left| w(t_N, x_j) - W_j^N \right|,$$

where $w(t_N, x_j)$ and W_j^N represent the analytical and numerical solutions at $(t, x) = (t_N, x_j)$, respectively. To assess the efficiency of the numerical scheme, we count the computational time in seconds for solving the model problem over $t \in [0, t_N]$, denoted by CPU.

4.1 Example 1

The first example to be solved is the coupled Burgers' equations given by

$$u_t - u_{xx} - 2uu_x + 2.5(uv)_x = 0, \quad v_t - v_{xx} - 2vv_x + 2.5(uv)_x = 0, \quad t > 0, \ x \in \Omega. \tag{38}$$

The above problem admits the following analytical traveling wave solutions [15]:

$$\big[u(t,x), v(t,x)\big] = \left[\kappa_0\left(1 - \tanh\left(\frac{3\kappa_0}{2}(x - 3\kappa_0 t)\right)\right), \ \kappa_0\left(1 - \tanh\left(\frac{3\kappa_0}{2}(x - 3\kappa_0 t)\right)\right)\right],$$

where κ_0 is a real constant that determines the initial slope and the advection velocity of the solution. The initial and Dirichlet boundary conditions can be recovered from the above analytical solutions.

First, to observe the adaptability of the proposed method to various initial conditions and different sharpness of the traveling wave, we plot analytical and numerical solutions of u during $t \leq 5.0$ for different values of κ_0 in Fig. 1. The numerical solutions are obtained by SC-SL$_3$. For three cases $\kappa_0 = 0.5, 1.0$, and 2.0, the resolutions and spatial domains $(h, \Delta x, \Omega)$ are employed as $(0.1, 0.1, [-20, 20])$, $(0.05, 0.05, [-10, 30])$, and $(0.005, 0.005, [0, 40])$, respectively. The figure shows that the wave has a faster moving speed and a sharper gradient as κ_0 is larger. It can be observed that the numerical solutions obtained by the proposed SC-SL$_3$ matches the analytical solutions well.

To examine the temporal and spatial convergence rates of the proposed methods, we first measure the maximum errors $\mathrm{E}_\infty(u)$ and $\mathrm{E}_\infty(v)$. The errors are computed with $\kappa_0 = 0.5$ at $t = 1.0$ on $\Omega = [-20, 20]$. The temporal convergence rates are obtained by varying time step size $h = 2^{-(k+3)}$ $(k = 1, 2, 3, 4)$ with a fixed spatial grid size $\Delta x = 1/60$. The spatial convergence rates are also obtained by varying $\Delta x = 1/2^k$ $(k = 1, 2, 3, 4)$ with a fixed $h = 0.005$. The results are illustrated in Fig. 2 for u, where the black dotted lines denote the reference slope for intuitive comparison confirms. Similar results can be obtained for v; our results for v have been omitted. In the figures, all proposed methods have third-order temporal and fourth-order spatial convergence rates. However, SC-SL$_1$ has the lowest accuracy than other methods, whereas the other methods have similar accuracy.

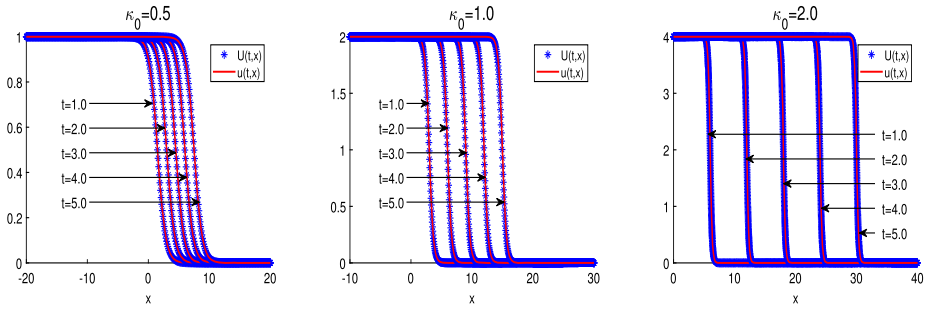

Figure 1 The behavior of the numerical solution for Example 1 obtained using SC-SL$_3$ with $(h, \Delta x, \kappa_0, \Omega) = (0.1, 0.1, 0.5, [-20, 20])$, $(0.05, 0.05, 1.0, [-10, 30])$, and $(0.005, 0.005, 2.0, [0, 40])$ at different times

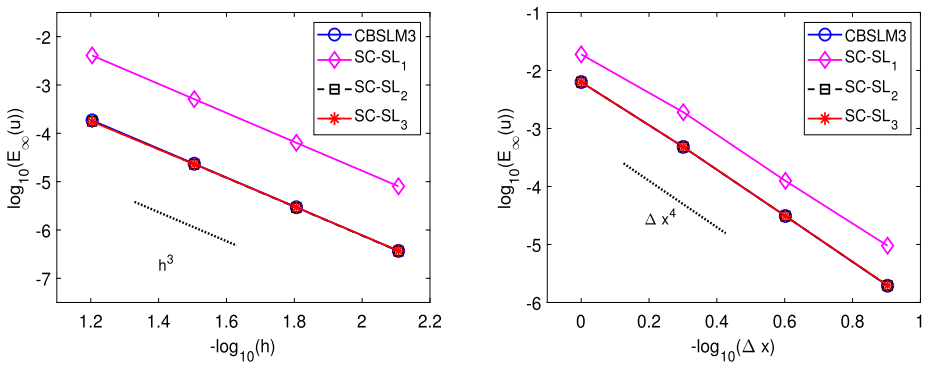

Figure 2 The temporal (left) and spatial (right) convergence rates of u for Example 1 with $\kappa_0 = 0.5$ at $t = 1.0$

Table 1 Error comparison of u at different time level with parameters $h = 1/1000$, $J = 320$ and $\kappa_0 = 0.1, 0.5$ for Example 1

κ_0	t	CBSLM3	SC-SL$_1$	SL-SC$_2$	SC-SL$_3$	CBSLM2 [2]	Kapoor [12]	Lai [14]
0.1	1.0	1.43e−10	4.26e−10	1.43e−10	1.43e−10	2.73e−10	2.58e−6	5.78e−7
	2.0	2.44e−10	2.08e−9	2.44e−10	2.44e−10	4.83e−10	2.88e−6	1.08e−6
	3.0	3.18e−10	2.83e−9	3.18e−10	3.18e−10	6.49e−10	3.18e−6	1.49e−6
	4.0	3.76e−10	3.44e−9	3.76e−10	3.76e−10	7.85e−10	4.49e−6	1.88e−6
	5.0	4.20e−10	3.94e−9	4.20e−10	4.20e−10	8.96e−10	3.83e−6	2.20e−6
0.5	1.0	4.55e−7	2.05e−6	4.55e−7	4.55e−7	4.83e−6	6.27e−7	6.75e−4
	2.0	5.53e−7	2.20e−6	5.53e−7	5.53e−7	5.24e−6	6.70e−7	8.17e−4
	3.0	4.40e−7	2.18e−6	4.40e−7	4.40e−7	5.38e−6	6.77e−7	8.64e−4
	4.0	4.28e−7	2.13e−6	4.28e−7	4.28e−7	5.42e−6	6.76e−7	8.82e−4
	5.0	4.16e−7	2.07e−6	4.16e−7	4.16e−7	5.43e−6	6.73e−7	8.91e−4

To investigate the accuracy of the CBSLM3 and SC-SLs, we compare the errors of numerical solutions obtained by the proposed methods and other methods [12, 14] including CBSLM2 [2]. In Table 1, the errors are listed at different time levels for two $\kappa_0 = 0.1$ and 0.5 on $\Omega = [-20, 20]$, where temporal step size and spatial resolution are employed as $(h, J) = (1/1000, 320)$. Here, $(h, J) = (1/1000, 320)$ is set the same as those used in the works of Kapoor et al. [12] and Lai et al. [14], and the results are used as those recorded in the literature [12, 14]. In this table, the CBSLM3 and SC-SLs are about three and four digits more accurate than those obtained by Kapoor [12] and Lai [14]. SC-SLs (s = 2, 3) are 2 and 10 times more accurate for $\kappa_0 = 0.1$ and 0.5, respectively, compared to those in CBLMS2. In addition, the accuracy of CBLSM3 is maintained, despite the use of a single characteristic curve. However, SC-SL$_1$ shows slightly insufficient accuracy, similar to the result of Fig. 2.

4.2 Example 2

For the second example, we consider the model equations on $(-\pi, \pi)$

$$u_t - 2uu_x + (uv)_x = u_{xx}, \qquad v_t - 2vv_x + (uv)_x = v_{xx} \tag{39}$$

with the analytical solutions [1, 2, 5, 15, 20, 21]

$$u(t,x) = v(t,x) = e^{-t}\sin(x), \quad x \in [-\pi, \pi], \quad t > 0. \tag{40}$$

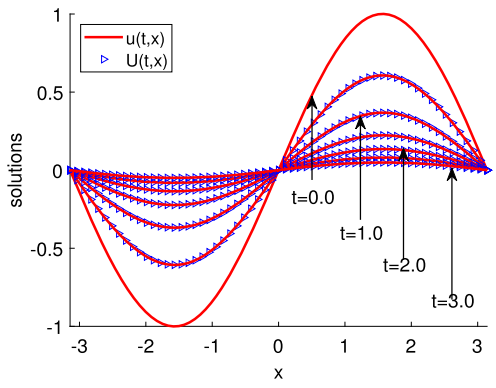

Figure 3 Comparison of the numerical and analytic solutions obtained by SC-SL_3

Table 2 Comparison of $E_\infty(u,t)$ for Example 2 with (h,J) at different times t

Method	(h,J)	$t=0.5$	$t=1.0$	$t=2.0$	$t=3.0$	$t=5.0$	$t=10.0$
CBSLM3 (CPU)	(0.01, 100)	4.27e−7 (0.014)	2.60e−7 (0.027)	5.12e−8 (0.053)	5.70e−9 (0.075)	6.30e−9 (0.130)	1.41e−10 (0.253)
SC-SL_1 (CPU)	(0.01, 100)	1.04e−5 (0.011)	2.67e−6 (0.022)	3.38e−7 (0.043)	1.10e−7 (0.063)	9.32e−9 (0.106)	4.53e−10 (0.205)
SC-SL_2 (CPU)	(0.01, 100)	4.03e−7 (0.011)	2.45e−7 (0.021)	4.56e−8 (0.042)	8.11e−9 (0.061)	6.66e−9 (0.108)	1.44e−10 (0.205)
SC-SL_3 (CPU)	(0.01, 100)	3.91e−7 (0.011)	2.38e−7 (0.021)	4.29e−8 (0.041)	9.36e−9 (0.061)	6.84e−9 (0.102)	1.45e−10 (0.204)
CBSLM2 [2] (CPU)	(0.01, 100)	1.01e−5 (0.010)	6.45e−6 (0.0189)	4.68e−6 (0.037)	3.25e−6 (0.052)	8.89e−7 (0.080)	1.36e−8 (0.158)
Mohanty [16]	(0.01, 100)	2.51e−6	3.04e−6	2.24e−6	1.24e−6	–	–
Zhang [21]	(0.01, 100)	2.53e−6	3.07e−6	2.26e−6	1.25e−6	–	–
Başhan [5]	(0.01, 190)	1.90e−6	2.25e−6	1.68e−6	9.62e−7	2.40e−7	1.67e−8

The initial and Dirichlet boundary conditions are imposed by the above analytical solutions. The solution in this example has the property of gradual diffusion over time.

In Fig. 3, we plot the behaviors of analytical and numerical solutions for u using SC-SL_3 at different time levels from $t = 0.0$ to $t = 3.0$ with an interval of 0.5. According to this figure, the solution to this problem begins with a sine function and decays over time, and the SC-SL_3 works well in this case. Moreover, the numerical solution U matches u well at different time levels.

To investigate the numerical accuracy of the proposed SC-SLs, we compare the errors obtained by SC-SLs and other methods introduced in [2, 5, 16, 21]. The methods in [16, 21] were compared only for the time levels at which the results were valid. The numerical results are obtained with spatial resolutions $J = 100$ when $h = 0.01$ and $J = 130$ when $h = 0.001$, except $J = 190$ for Başhan [5] with $h = 0.01$. Tables 2–3 report the results at different time levels $t \leq 10.0$. In particular, CPUs for CBSLM3, SC-SLs, and CBSLM2 [2] are also reported in Tables 2–3. The tables indicate that SC-SL_2 and SC-SL_3 achieved at least one- to two-digit higher accuracy than the other methods [2, 5, 16, 21] in the entire time intervals. Moreover, SC-SL_2 and SC-SL_3 can maintain the accuracy of CBSLM3 at a lower computational cost. In contrast, the proposed SC-SL_1 shows poor accuracy compared to CBSLM3, with the results being similar to those of Example 1. As shown

Table 3 Comparison of $E_\infty(u,t)$ for Example 2 with (h,J) at different times t

Method	(h,J)	$t=0.5$	$t=1.0$	$t=2.0$	$t=3.0$	$t=5.0$	$t=10.0$
CBSLM3 (CPU)	(0.001, 130)	1.81e−8 (0.153)	2.20e−8 (0.296)	1.63e−8 (0.602)	9.01e−9 (0.862)	2.04e−9 (1.423)	2.80e−11 (2.850)
SC-SL_1 (CPU)	(0.001, 130)	3.64e−8 (0.126)	3.11e−8 (0.247)	1.66e−8 (0.486)	8.91e−9 (0.737)	2.03e−9 (1.216)	2.95e−11 (2.407)
SC-SL_2 (CPU)	(0.001, 130)	1.82e−8 (0.121)	2.20e−8 (0.241)	1.63e−8 (0.487)	9.02e−9 (0.730)	2.04e−9 (1.206)	2.80e−11 (2.402)
SC-SL_3 (CPU)	(0.001, 130)	1.82e−8 (0.120)	2.20e−8 (0.237)	1.63e−8 (0.461)	9.02e−9 (0.695)	2.04e−9 (1.167)	2.80e−11 (2.323)
CBSLM2 [2] (CPU)	(0.001, 130)	1.18e−7 (0.120)	5.38e−8 (0.225)	3.21e−8 (0.427)	2.37e−8 (0.673)	6.86e−9 (1.082)	1.08e−10 (2.073)
Başhan [5]	(0.001, 130)	2.61e−6	1.57e−6	1.01e−6	5.47e−7	1.37e−7	1.02e−8

Tables 2–3, as time t increases, the error differences between SC-SL_1 and other SC-SLs (s = 2, 3) decrease because this is a special situation in which the solution scale converges to 0. According to the results of Examples 1 and 2, the results of SC-SL_1 will be excluded from the comparison.

We notice that the main observation point of the results obtained using the proposed method is the accuracy of the numerical results. Therefore, an important concern is whether the accuracy of the existing method, which does not share even if the virtual characteristic curve is shared, can be well maintained. Hence, we will examine the adaptability and numerical performance of the proposed method even for solutions with different behaviors in the following two examples.

4.3 Example 3

As the third example, the coupled Burgers' equation on the spatial domain Ω is as follows:

$$u_t + \alpha_1 u u_x + \beta_1 (uv)_x = \nu u_{xx}, \qquad v_t + \alpha_2 v v_x + \beta_2 (uv)_x = \nu v_{xx}, \tag{41}$$

where coefficients, α_k and β_k are arbitrary real constants. In this experiment, we take the analytical solution as follows [19]:

$$\begin{cases} u(t,x) = \kappa_1 (1 + \tanh(\frac{\kappa_0}{2\nu}(x + \kappa_0 t))), \\ v(t,x) = \frac{2\beta_2 - \alpha_1}{2\beta_1 - \alpha_2} u(t,x), \end{cases} \quad \text{if } 2\beta_1 \neq \alpha_2, \tag{42}$$

where $\kappa_1 = \frac{\kappa_0(2\beta_1 - \alpha_2)}{(\alpha_1\alpha_2 - 4\beta_1\beta_2)}$ and κ_0 is a nonzero constant. The initial and Dirichlet boundary conditions are imposed by the above analytical solutions.

To verify the effect of parameters and the adaptability of the proposed methods, we plot the behaviors of the numerical solutions in Figs. 4–5. The parameters considered herein are wavelet speed, viscosity, the difference in magnitude between the two solutions, and nonlinearity (coupling). The numerical experiments are tested according to viscosity $\nu = 1.0, 0.1$, and 0.001. We use the parameters $(\alpha_1, \alpha_2, \beta_1, \beta_2) = (2.0, 2.0, 0.2, 0.8)$ and employ the spatial domains $\Omega = [-10, 15]$, $[-10, 10]$, and $[-10, 10]$, considering the behaviors of the solutions. Fig. 4 depicts the numerical solution obtained by SC-SL_3 and compares it with the exact solution. The figure illustrates that the smaller the viscosity, the sharper the gradient of the solution. The proposed SC-SL_3 is observed to derive a fairly accurate solution even when ν becomes small. In addition, we observe numerical solutions for two

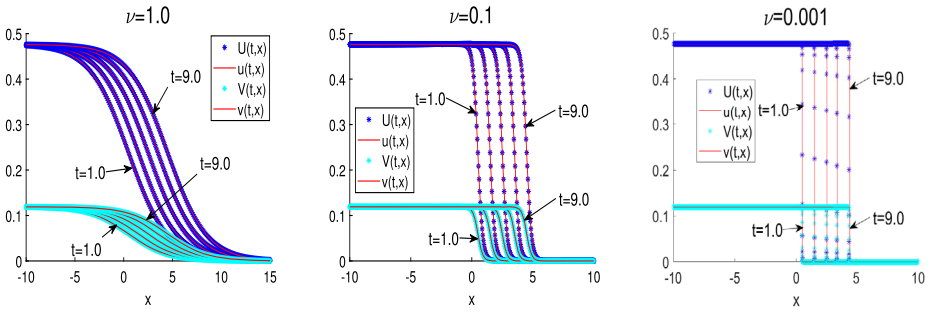

Figure 4 Behavior of the numerical solutions obtained by SC-SL$_3$ with $(h, \Delta x, \nu) = (0.1, 0.1, 1.0), (0.05, 0.05, 0.1)$, and $(0.002, 0.002, 0.001)$ and $\kappa_0 = -0.5$ at different times

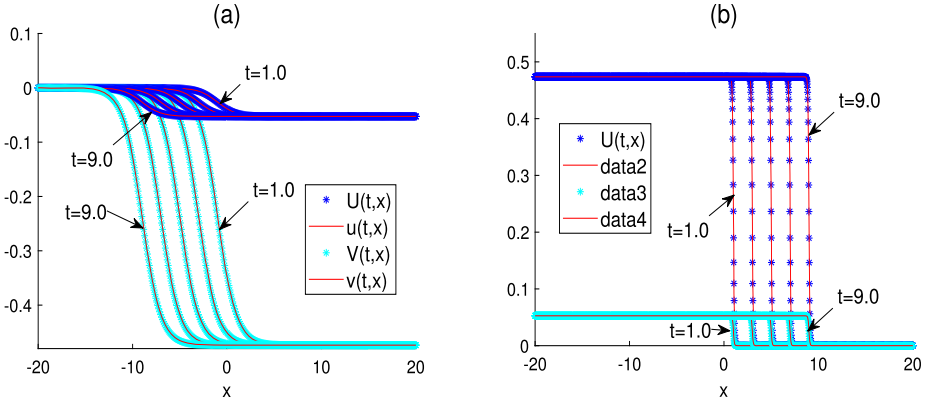

Figure 5 Behavior of the numerical solutions obtained by SC-SL$_3$ with (a) $(h, \Delta x, \nu, \kappa_0) = (0.1, 0.1, 1.0, 1.0)$ and (b) $(0.005, 1/50, 0.05, -2.0)$ at different times

different ν and κ_0. As shown in Fig. 5, when κ_0 is positive, the wave propagation direction is to the left, and when κ_0 is negative, the wave propagation direction is to the right. Therefore, the sign of κ_0 determines the advection direction of the solution. Finally, the ratio of β_2 and β_1 represents the difference between the magnitudes of u and ν.

For different parameter sets, we report the comparison of maximum errors $E_\infty(u)$ and $E_\infty(\nu)$ of numerical schemes CBSLM3, SC-SL$_3$, SC-SL$_2$, and CBSLM2 in Tables 4–6. Table 4 lists the errors and CPUs to compare the accuracy and efficiency of the four methods according to wavelet speed (magnitude of κ_0) and direction (a sign of κ_0) at different times. We test for cases where β_1 is different from β_2 as $\beta_1 = 0.2$ and $\beta_2 = 0.8$. As indicated in Fig. 5, this case makes the amplitudes of the behaviors of u and ν, and their function values different from each other. In particular, for $\kappa_0 = 1.0$, we set the virtual particle velocity for SC-SL$_3$ as $s(t, \theta_j(t)) = \alpha_2 \nu(t, \theta_j(t)) + \beta_2 u(t, \theta_j(t))$ because the amplitude of ν is larger. Then, we solve the reverse order of the original SC-SL$_3$ (refer to Remark 2). In Table 4, for $\kappa_0 = -1.0$ and 1.0, CBSLM3 is approximately 2-digits more accurate than CBSLM2. For $\kappa_0 = -2.0$, CBSLM3 is almost 1-digit more accurate than CBSLM2. The comparison of the accuracies shows that SC-SL$_2$ is more accurate than CBSLM2; however, for $|\kappa_0| = 1.0$ and 2.0, SC-SL$_2$ is approximately 1- and 0.4-digit less accurate than CBSLM3, respectively. In contrast, SC-SL$_3$ has almost the same error magnitude as CBSLM3. Considering

Table 4 Error comparison at different time levels with parameters $(h, \Delta x) = (0.01, 1/40)$ and $(\alpha_1, \alpha_2, \beta_1, \beta_2, \nu) = (2.0, 2.0, 0.2, 0.8, 1.0)$ for u and v on $\Omega = [-20, 20]$

t	$\kappa_0 = -1.0$											
	CBSLM3			SC-SL$_3$			SC-SL$_2$			CBSLM2 [2]		
	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU
1.0	5.18e−9	1.41e−9	0.778	5.17e−9	2.01e−9	0.429	4.17e−8	1.12e−8	0.430	1.05e−6	2.94e−7	0.461
2.0	7.79e−9	2.01e−9	1.599	7.74e−9	3.31e−9	0.869	5.88e−8	1.77e−8	0.891	1.43e−6	4.27e−7	0.821
3.0	9.51e−9	2.37e−9	2.378	9.41e−9	4.32e−9	1.319	6.95e−8	2.27e−8	1.350	1.61e−6	5.17e−7	1.167
5.0	1.17e−8	2.84e−9	3.898	1.14e−8	5.86e−9	2.153	8.37e−8	3.07e−8	2.212	1.77e−6	6.61e−7	1.934
7.0	1.31e−8	3.18e−9	5.513	1.27e−8	7.09e−9	3.093	9.49e−8	3.78e−8	3.105	1.88e−6	7.98e−7	2.717
9.0	1.43e−8	3.49e−9	7.022	1.37e−8	8.18e−9	3.922	1.05e−7	4.45e−8	3.933	1.99e−6	9.36e−7	3.453
t	$\kappa_0 = -2.0$											
	CBSLM3			SC-SL$_3$			SC-SL$_2$			CBSLM2 [2]		
	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU
1.0	1.36e−6	3.33e−7	0.768	1.34e−6	6.43e−7	0.430	9.78e−6	3.45e−6	0.446	5.36e−5	1.86e−5	0.401
2.0	1.74e−6	4.23e−7	1.590	1.67e−6	9.70e−7	0.873	1.28e−5	5.29e−6	0.885	6.02e−5	2.72e−5	0.815
3.0	2.02e−6	5.04e−7	2.335	1.89e−6	1.23e−6	1.295	1.56e−5	6.99e−6	1.325	6.67e−5	3.61e−5	1.160
5.0	2.64e−6	6.85e−7	4.011	2.34e−6	1.69e−6	2.190	2.25e−5	1.04e−5	2.236	8.18e−5	5.44e−5	1.955
7.0	3.41e−6	9.15e−7	5.469	2.91e−6	2.10e−6	3.052	3.16e−5	1.38e−5	3.069	9.83e−5	7.21e−5	2.695
9.0	4.23e−6	1.19e−6	7.081	3.55e−6	2.50e−6	3.915	4.09e−5	1.72e−5	3.918	1.17e−4	8.90e−5	3.485
t	$\kappa_0 = 1.0$											
	CBSLM3			SC-SL$_3$			SC-SL$_2$			CBSLM2 [2]		
	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU
1.0	5.18e−9	1.41e−9	0.797	5.17e−9	2.01e−9	0.442	4.17e−8	1.12e−8	0.443	1.05e−6	2.94e−7	0.442
2.0	7.79e−9	2.01e−9	1.577	7.74e−9	3.31e−9	0.864	5.88e−8	1.77e−8	0.869	1.43e−6	4.27e−7	0.790
3.0	9.51e−9	2.37e−9	2.328	9.41e−9	4.32e−9	1.324	6.95e−8	2.27e−8	1.327	1.61e−6	5.17e−7	1.236
5.0	1.17e−8	2.84e−9	3.891	1.14e−8	5.86e−9	2.203	8.37e−8	3.07e−8	2.207	1.77e−6	6.61e−7	1.977
7.0	1.31e−8	3.18e−9	5.433	1.27e−8	7.09e−9	3.056	9.49e−8	3.78e−8	3.086	1.88e−6	7.98e−7	2.713
9.0	1.43e−8	3.49e−9	7.127	1.37e−8	8.18e−9	3.947	1.05e−7	4.45e−8	3.961	1.99e−6	9.36e−7	3.464

Table 5 Error comparison at different time levels with parameters $(h, \Delta x) = (0.001, 0.01)$ and $(\alpha_1, \alpha_2, \beta_1, \beta_2, \kappa_0) = (2.0, 2.0, 0.2, 0.8, -1.0)$ for u and v on $\Omega = [-20, 20]$

t	$\nu = 0.1$											
	CBSLM3			SC-SL$_3$			SC-SL$_2$			CBSLM2 [2]		
	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU
1.0	1.34e−7	3.75e−8	40.109	1.34e−7	5.70e−8	21.290	2.29e−7	1.09e−7	21.392	2.83e−6	1.20e−6	19.923
2.0	1.41e−7	4.00e−8	79.928	1.40e−7	6.20e−8	42.555	3.16e−7	1.45e−7	43.115	4.04e−6	1.95e−6	39.938
3.0	1.49e−7	4.23e−8	120.273	1.46e−7	6.54e−8	63.541	4.03e−7	1.80e−7	64.403	5.33e−6	2.68e−6	59.636
5.0	1.64e−7	4.69e−8	199.060	1.57e−7	7.10e−8	100.729	5.82e−7	2.46e−7	107.506	8.05e−6	4.12e−6	99.475
7.0	1.78e−7	5.14e−8	278.519	1.69e−7	7.60e−8	147.976	7.61e−7	3.11e−7	148.760	1.09e−5	5.52e−6	138.652
9.0	1.93e−7	5.59e−8	358.730	1.80e−7	8.07e−8	191.171	9.41e−7	3.75e−7	192.853	1.37e−5	6.91e−6	179.023
t	$\nu = 0.01$											
	CBSLM3			SC-SL$_3$			SC-SL$_2$			CBSLM2 [2]		
	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU
1.0	2.32e−3	4.61e−4	39.790	2.29e−3	6.95e−4	21.367	4.72e−3	1.35e−3	26.467	1.64e−2	3.44e−3	20.147
2.0	2.83e−3	5.10e−4	79.383	2.82e−3	8.50e−4	42.610	7.78e−3	2.33e−3	53.094	2.93e−2	6.24e−3	39.745
3.0	3.34e−3	5.60e−4	123.809	3.34e−3	1.00e−3	64.103	1.17e−2	3.38e−3	79.825	4.21e−2	9.04e−3	60.158
5.0	4.36e−3	6.60e−4	200.306	4.40e−3	1.31e−3	102.442	1.97e−2	5.50e−3	132.757	6.76e−2	1.47e−2	99.885
7.0	5.37e−3	7.59e−4	274.728	5.44e−3	1.61e−3	145.512	2.76e−2	7.61e−3	185.927	9.25e−2	2.03e−2	140.867
9.0	6.39e−3	8.57e−4	362.107	6.49e−3	1.92e−3	191.648	3.56e−2	9.71e−3	192.787	1.17e−1	2.58e−2	178.660

the computational time, for all κ_0, SC-SL$_3$ is approximately 1.8-times as fast as CBSLM3 when calculating numerical solutions with the same accuracy.

To compare the results according to viscosity ν, we measure the maximum errors at different time levels in Table 5. We use two viscosities, ν = 0.1 and 0.01, and set up

Table 6 Error comparison at different time levels with parameters $(h, \Delta x) = (0.01, 1/40)$ and $(\alpha_1, \alpha_2, \kappa_0, \nu) = (2.0, 2.0, -1.0, 0.1)$ for u and v on $\Omega = [-20, 20]$

t	$\beta_1 = 10, \beta_2 = 2$											
	CBSLM3			SC-SL$_3$			SC-SL$_2$			CBSLM2 [2]		
	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU
1.0	3.01e−5	2.44e−6	0.762	3.32e−5	3.88e−6	0.431	4.02e−5	2.74e−6	0.443	1.32e−4	3.34e−5	0.436
2.0	4.94e−5	4.58e−6	1.541	5.97e−5	6.98e−6	0.873	6.69e−5	5.48e−6	0.902	3.25e−4	5.94e−5	0.845
3.0	6.91e−5	6.79e−6	2.368	8.71e−5	1.01e−5	1.292	9.37e−5	8.40e−6	1.334	5.47e−4	8.49e−5	1.212
5.0	1.10e−4	1.12e−5	3.912	1.42e−4	1.62e−5	2.166	1.48e−4	1.43e−5	2.256	1.00e−3	1.36e−4	1.975
7.0	1.50e−4	1.58e−5	5.452	1.98e−4	2.24e−5	3.034	2.01e−4	2.03e−5	3.125	1.46e−3	1.87e−4	2.748
9.0	1.91e−4	2.03e−5	7.021	2.53e−4	2.86e−5	3.745	2.56e−4	2.63e−5	4.085	1.92e−3	2.38e−4	3.504
t	$\beta_1 = 2, \beta_2 = 10$											
	CBSLM3			SC-SL$_3$			SC-SL$_2$			CBSLM2 [2]		
	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU	$E_\infty(u)$	$E_\infty(v)$	CPU
1.0	4.90e−6	2.40e−5	0.772	3.88e−6	3.32e−5	0.447	2.74e−6	4.02e−5	0.451	1.44e−5	3.15e−4	0.432
2.0	6.81e−6	4.38e−5	1.568	6.98e−6	5.97e−5	0.865	5.48e−6	6.69e−5	1.006	2.11e−5	5.52e−4	0.815
3.0	8.69e−6	6.37e−5	2.346	1.01e−5	8.71e−5	1.306	8.40e−6	9.37e−5	1.329	4.26e−5	7.83e−4	1.218
5.0	1.28e−5	1.04e−4	3.917	1.62e−5	1.42e−4	2.167	1.43e−5	1.48e−4	2.260	9.04e−5	1.24e−3	1.961
7.0	1.71e−5	1.45e−4	5.441	2.24e−5	1.98e−4	3.057	2.03e−5	2.01e−4	3.231	1.41e−4	1.70e−3	2.708
9.0	2.15e−5	1.86e−4	7.001	2.86e−5	2.53e−4	3.884	2.63e−5	2.56e−4	4.173	1.91e−4	2.16e−3	3.472

the parameters and resolutions as $(\alpha_1, \alpha_2, \beta_1, \beta_2, \kappa_0) = (2.0, 2.0, 0.2, 0.8, -1.0)$ and $(h, \Delta x) = (0.001, 0.01)$, respectively. For relatively mild viscosity $\nu = 0.1$, CBSLM3 and SC-SL$_3$ are approximately 2-digits more accurate than CBSLM2. When $\nu = 0.01$, CBSLM3 and SC-SL$_3$ are at least 1-digit more accurate than CBSLM2. Moreover, we remark that the errors of SC-SL$_2$ are less accurate than those of CBSLM3 and SC-SL$_3$, as shown in the previous table. Despite SC-SL$_3$ being almost twice as fast as CBSLM3, no significant error difference is observed between them in both cases. Considering the above results, we conclude that SC-SL$_3$ is the most efficient in terms of accuracy and computational cost.

Finally, we consider the case where solutions have different magnitudes and amplitudes. The pair of parameters β_1 and β_2 is employed as $(\beta_1, \beta_2) = (10, 2)$ and $(2, 10)$ with $(\alpha_1, \alpha_2, \kappa_0, \nu) = (2.0, 2.0, -1.0, 0.1)$. Herein, in the case where $(\beta_1, \beta_2) = (2, 10)$, we define the virtual particle velocity $s(t, \theta_j(t))$ as stated in Remark 2 and solve the derived reversed order system, as the amplitude of the initial value of v is greater than that of u. The numerical results of Table 6 show that SC-SL$_3$ maintains the accuracy as that of CBSLM3 despite using half its computational effort compared to CBSLM3. Thus, SC-SL$_3$ is more cost-effective than CBSLM3, considering two trajectories for the two distinct classes of particles.

4.4 Example 4

To investigate the proposed SC-SL$_3$, we consider a model with shock initial conditions in our last example:

$$u_t + 2uu_x + 0.2(uv)_x = \nu u_{xx}, \quad v_t - 2vv_x - 0.8(uv)_x = \nu v_{xx}, \quad t > 0, \ \ x \in [0, 1]. \tag{43}$$

This model has an initial condition as

$$\begin{aligned} u(0, x) &= \begin{cases} 9x, & \text{if } 0 \le x \le \frac{1}{10}, \\ 1 - x, & \text{if } \frac{1}{10} \le x \le 1, \end{cases} \\ v(0, x) &= \begin{cases} x, & \text{if } 0 \le x \le \frac{9}{10}, \\ 9(1 - x), & \text{if } \frac{9}{10} \le x \le 1, \end{cases} \end{aligned} \tag{44}$$

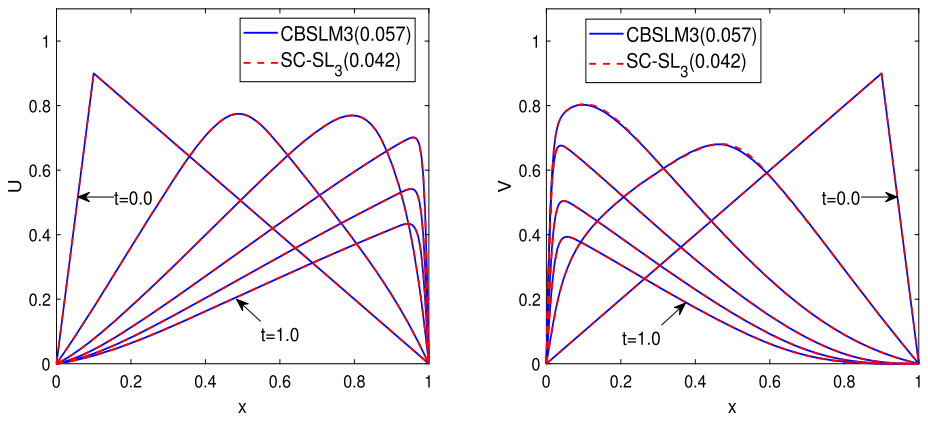

Figure 6 Behavior of the numerical solutions for $(\alpha_1, \alpha_2, \beta_1, \beta_2, \nu) = (2, -2, 0.2, -0.8, 0.01)$ obtained by CBSLM3 and SC-SL_3, and their CPUs with $(h, \Delta x) = (1/150, 1/200)$

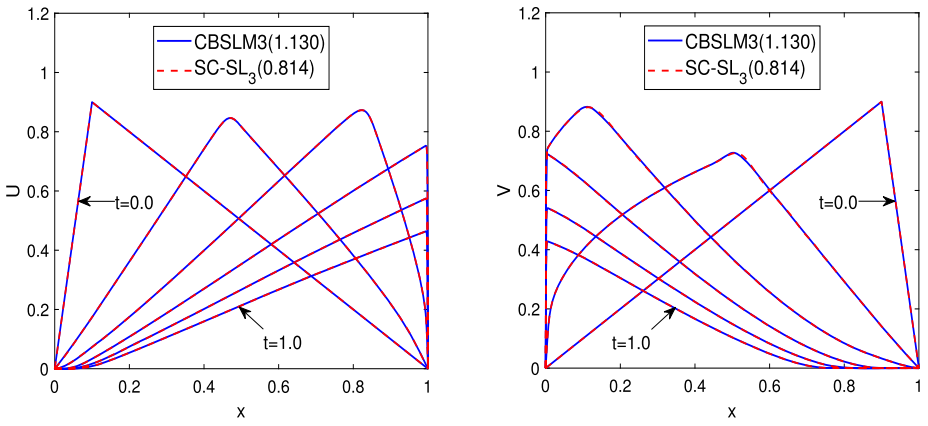

Figure 7 Behavior of the numerical solutions for $(\alpha_1, \alpha_2, \beta_1, \beta_2, \nu) = (2, -2, 0.2, -0.8, 0.0005)$ obtained using CBSLM3 and SC-SL_3, and their CPUs with $(h, \Delta x) = (1/500, 1/800)$

and the homogeneous boundary condition. Now, we examine the adaptability of the SC-SL_3 when solutions u and v move in opposite directions. In this case, the solutions u and v have strong nonlinearity with each other. In Figs. 6–8, we plot the behaviors of the numerical solutions using CBSLM3 and SC-SL_3 at different time levels from $t = 0.0$ to $t = 1.0$ with an interval of 0.2. Here, we note that the numerical solutions of u and v move in opposite directions, i.e., to the right and left, respectively. In addition, the CPUs for each method are described in the legends of the figures. In Fig. 4, we use the parameters $(\alpha_1, \alpha_2, \beta_1, \beta_2, \nu) = (2, -2, 0.2, -0.8, 0.01)$ and resolution $(h, \Delta x) = (1/150, 200)$. According to Fig. 6, the numerical solution of SC-SL_3 has the same performance as the numerical solution of CBLSM3 with less computational cost compared.

Fig. 7 is the result of the simulation conducted to verify the performance for the quite small viscosity $\nu = 0.0005$. The parameters and resolutions are set by $(\alpha_1, \alpha_2, \beta_1, \beta_2) = (2, -2, 0.2, -0.8)$ and $(h, \Delta x) = (1/500, 1/800)$, respectively. As can be observed in Fig. 7, the numerical solutions of SC-SL_3 have a good agreement with those of CBSLM3, even if

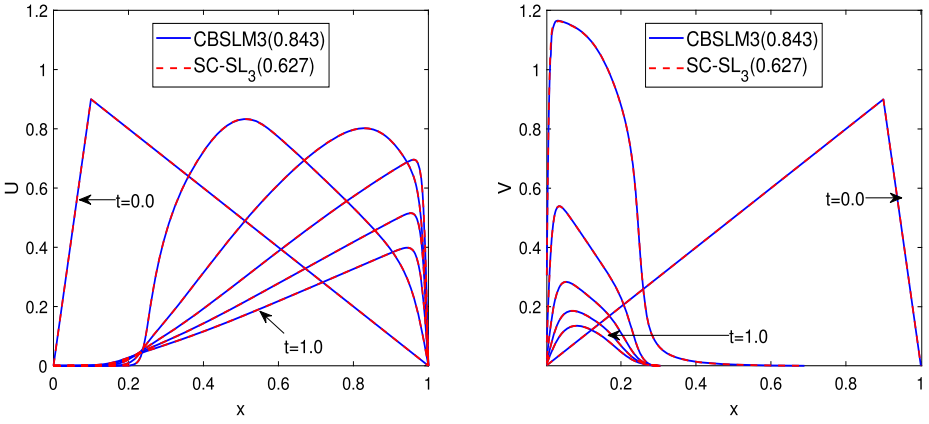

Figure 8 Behavior of the numerical solutions for $(\alpha_1, \alpha_2, \beta_1, \beta_2, \nu) = (2, -2, 2, -8, 0.01)$ obtained by CBSLM3 and SC-SL$_3$, and their CPUs with $(h, \Delta x) = (1/600, 1/600)$

the solutions have a sharp gradient. Thus, the efficiency of SC-SL$_3$ can also be observed in this simulation.

Finally, to simulate the case of a significant difference in the scales of u and v, we use the following parameters and resolutions $(\alpha_1, \alpha_2, \beta_1, \beta_2, \nu) = (2, -2, 2, 8, 0.01)$ and $(h, \Delta x) = (1/600, 1/600)$, respectively. In Fig. 8, the result of SC-SL$_3$ exhibits stable solution behavior that follows that of CBSLM3 well. Furthermore, SC-SL$_3$ requires fewer CPUs than CBSLM3 to obtain similar results in terms of accuracy as in the previous test.

5 Conclusion

This paper introduced three fast BSLMs for solving the viscous coupled Burgers' equations. The proposed methods simultaneously handle two Cauchy problems by observing a single virtual characteristic curve with a velocity. This can dramatically reduce the interpolation process and the computational cost of determining the departure point. We used four examples to assess the adaptability and efficiency of the proposed methods. In particular, we proved that SC-SL$_3$ displays a lower cost consumption than CBSLM3 while maintaining a similar accuracy to CBSLM3 for various types of solutions. This finding is expected to provide a reference about solving various other coupled equations with a new perspective that deviates from the traditional view of the velocity for each physical particle in the Burgers' equation coupled with the BSLM.

Appendix A: Finite difference methods

Many types of numerical methods exist for approximating spatial derivatives. Here, we list the fourth-order finite difference methods used in this study for the approximation. The finite difference method for a first-order spatial derivative of $w(x)$ at interior points is

$$w_x(x_i) = \frac{1}{12\Delta x}\big(w(x_{i-2}) - 8w(x_{i-1}) + 8w(x_{i+1}) - w(x_{i+2})\big) + \mathcal{O}\big(\Delta x^4\big), \quad i = 2, \ldots, J-2.$$

As the above formula is not applicable for $w_x(x_1)$ and $w_x(x_{J-1})$, we introduce exceptional formulas:

$$w_x(x_1) = \frac{1}{12\Delta x}\left(-3w(x_0) - 10w(x_1) + 18w(x_2) - 6w(x_3) + w(x_4)\right) + \mathcal{O}\left(\Delta x^4\right),$$
$$w_x(x_{M-1}) = \frac{1}{12\Delta x}\left(-w(x_{M-4}) + 6w(x_{M-3}) - 18w(x_{M-2}) + 10w(x_{M-1}) + 3w(x_M)\right) + \mathcal{O}\left(\Delta x^4\right).$$

Consequently, the matrix form of the finite difference method for first-order spatial derivatives is given by

$$\bar{\mathbf{w}}_x = \mathcal{D}_1\bar{\mathbf{w}} + \mathbf{d}_{1,w} + \mathcal{O}\left(\Delta x^4\right),$$

where

$$\mathcal{D}_1 = \frac{1}{12\Delta x}\begin{bmatrix} -10 & 18 & -6 & 1 & & & \mathbf{0} \\ -8 & 0 & 8 & -1 & & & \\ 1 & -8 & 0 & 8 & -1 & & \\ & \ddots & \ddots & \ddots & \ddots & & \\ & & 1 & -8 & 0 & 8 & -1 \\ & & & 1 & -8 & 0 & 8 \\ \mathbf{0} & & & -1 & 6 & -18 & 10 \end{bmatrix}, \qquad \mathbf{d}_{1,w} = \frac{1}{12\Delta x}\begin{bmatrix} -3w(x_0) \\ w(x_0) \\ 0 \\ \vdots \\ 0 \\ -w(x_J) \\ 3w(x_J) \end{bmatrix}.$$

Next, the finite difference method for second order spatial derivative of $w(x)$ at interior points x_i $(i = 2, \ldots, M-2)$ is given by

$$w_{xx}(x_i) = \frac{1}{12\Delta x^2}\left(-w(x_{i-2}) + 16w(x_{i-1}) - 30w(x_i) + 16w(x_{i+1}) - w(x_{i+2})\right) + \mathcal{O}\left(\Delta x^4\right).$$

Additionally, the formulas for $w_{xx}(x_1)$ and $w_{xx}(x_{J-1})$ are given as

$$w_{xx}(x_1) = \frac{1}{12\Delta x}\left(-3w(x_0) - 10w(x_1) + 18w(x_2) - 6w(x_3) + w(x_4)\right) + \mathcal{O}\left(\Delta x^4\right),$$
$$w_{xx}(x_{M-1}) = \frac{1}{12\Delta x}\left(-w(x_{M-4}) + 6w(x_{M-3}) - 18w(x_{M-2}) + 10w(x_{M-1}) + 3w(x_M)\right) + \mathcal{O}\left(\Delta x^4\right).$$

The finite difference method for a second-order spatial derivative also has the following matrix form:

$$\bar{\mathbf{w}}_{xx} = \mathcal{D}_2\bar{\mathbf{w}} + \mathbf{d}_{2,w} + \mathcal{O}\left(\Delta x^4\right),$$

where

$$\mathcal{D}_2 = \frac{1}{12\Delta x^2}\begin{bmatrix} -15 & -4 & 14 & -6 & 1 & 0 & \dots & 0 \\ 16 & -30 & 16 & -1 & 0 & \vdots & \ddots & \vdots \\ -1 & 16 & -30 & 16 & -1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & -1 & 16 & -30 & 16 & -1 & 0 \\ 0 & \dots & 0 & -1 & 16 & -30 & 16 & -1 \\ \vdots & \ddots & \vdots & 0 & -1 & 16 & -30 & 16 \\ 0 & \dots & 0 & 1 & -6 & 14 & -4 & -15 \end{bmatrix},$$

$$\mathbf{d}_{2,w} = \frac{1}{12\Delta x^2}\begin{bmatrix} 10w(x_0) \\ -w(x_0) \\ 0 \\ \vdots \\ 0 \\ -w(x_J) \\ 10w(x_J) \end{bmatrix}.$$

Appendix B: Lagrange interpolation

We briefly review a piecewise Lagrange interpolation of order six to approximate the function values and its Jacobian at $x \in [x_j, x_{j+1}]$ $(0 \le j < J-1)$. Suppose that $\eta(j) = (J-6)\mathbb{I}_{\{j|j>J-3\}} + (j-3)\mathbb{I}_{\{j|1<j<J-2\}}$ is an index function, where $\mathbb{I}_A$ denotes the indicator function for a set A. Then, for function values $w(x_{\eta(j)+i})$ $(0 \le i \le 6)$, the Lagrange interpolation polynomial, satisfying $\mathcal{L}[\mathbf{w}](x_{\eta(j)+i}) = w(x_{\eta(j)+i})$ $(0 \le i \le 6)$, is defined by

$$\mathcal{L}[\mathbf{w}](x) = \sum_{i=0}^{6} L_{i,6}(x)w(x_{\eta(j)+i}) \quad \text{and} \quad \mathcal{L}_x[\mathbf{w}](x) = \sum_{i=0}^{6} L'_{i,6}(x)w(x_{\eta(j)+i}),$$

where $\mathbf{w}$ is a vector comprising the function values $w(x_j)$ $(0 \le j \le J)$, and $L_{i,k}(x)$ is the Lagrange basis given by

$$L_{i,k}(x) = \prod_{m=0, m\neq i}^{k} \frac{x - \widetilde{x}_m}{\widetilde{x}_i - \widetilde{x}_m}, \quad \widetilde{x}_i = x_{\eta(j)+i}.$$

Further, $L'_{i,k}(x)$ is the derivative of $L_{i,k}(x)$.

Acknowledgements
Not applicable.

Funding
The fist author Bak was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Korea government (MSIT) (No. 2020R1C1C1A01004139) and the Ministry of Education (NRF-2022R1I1A1A01059167).

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author contributions
All authors contributed equally to the writing of this paper. All authors read and approved the final manuscript.

Author details
[1]Department of Mathematics, Kyungpook National University, Daegu, 41566, South Korea. [2]Mechatronics Research Center, Hongik University, Sejong, 30016, South Korea.

Received: 14 April 2023 Accepted: 15 September 2023 Published online: 09 October 2023

References

1. Abdullah, M., Yaseen, M., De la Sen, M.: Numerical simulation of the coupled viscous Burgers equation using the Hermite formula and cubic B-spline basis functions. Phys. Scr. **95**, 115216 (2020)
2. Bak, S., Kim, P., Kim, D.: A semi-Lagrangian approach for numerical simulation of coupled Burgers' equations. Commun. Nonlinear Sci. Numer. Simul. **69**, 31–44 (2019)
3. Bak, S.: High-order characteristic-tracking strategy for simulation of a nonlinear advection–diffusion equation. Numer. Methods Partial Differ. Equ. **35**, 1756–1776 (2019)
4. Bak, S.: A mixed approximate method to simulate generalized Hirota-Satsuma coupled KdV equations. Comput. Appl. Math. **41**, 102 (2022)
5. Başhan, A.: A numerical treatment of the coupled viscous Burgers' equation in the presence of very large Reynolds number. Physica A **545**, 123755 (2020)
6. Batchelor, G.K.: Sedimentation in a dilute polydisperse system of interacting spheres. Part 1. General theory. Comput. Math. Appl. **119**, 2711–2722 (1982)
7. Bhatt, H.P., Khaliq, A.Q.M.: Fourth-order compact schemes for the numerical simulation of coupled Burgers' equation. Comput. Phys. Commun. **200**, 117–138 (2016)
8. Engeln-Müllges, G., Uhlig, F.: Numerical Algorithms with C. Springer, Berlin (1996)
9. Esipov, S.E.: Coupled Burgers' equations: a model of polydispersive sedimentation. Phys. Rev. E **52**, 3711–3718 (1995)
10. Hussein, A.J., Kashkool, H.A.: Weak Galerkin finite element method for solving one-dimensional coupled Burgers' equations. J. Appl. Math. Comput. **63**, 265–293 (2020)
11. Jiwari, R., Alshomrani, A.S.: A new algorithm based on modified trigonometric cubic B-splines functions for nonlinear Burgers'-type equations. Int. J. Numer. Methods Heat Fluid Flow **27**, 1638–1661 (2017)
12. Kapoor, M., Joshi, V.: A new technique for numerical solution of 1D and 2D nonlinear coupled Burgers' equations by using cubic uniform algebraic trigonometric (UAT) tension B-spline based differential quadrature method. Ain Shams Eng. J. **12**, 3947–3965 (2021)
13. Kumar, M., Pandit, S.: A composite numerical scheme for the numerical simulation of coupled Burgers' equation. Comput. Phys. Commun. **185**, 809–817 (2014)
14. Lai, H., Ma, C.: A new lattice Boltzmann model for solving the coupled viscous Burgers' equation. Physica A **395**, 445–457 (2014)
15. Li, Q., Chai, Z., Shi, B.: A novel lattice Boltzmann model for the coupled viscous Burgers' equations. Appl. Math. Comput. **250**, 948–957 (2015)
16. Mohanty, R.K., Dai, W., Han, F.: Compact operator method of accuracy two in time and four in space for the numerical solution of coupled viscous Burgers' equations. Appl. Math. Comput. **256**, 381–393 (2015)
17. Mohanty, R.K., Sharma, S.: A new high-accuracy method based on off-step cubic polynomial approximations for the solution of coupled Burgers' equations and Burgers–Huxley equation. Eng. Comput. **30**, 3049–3066 (2021)
18. Park, S., Kim, P., Jeon, Y., Bak, S.: An economical robust algorithm for solving 1D coupled Burgers' equations in a semi-Lagrangian framework. Appl. Math. Comput. **428**, 127185 (2022)
19. Piao, X., Kim, P.: Comment on: "The modified extended tanh-function method for solving Burgers-type equations". Physica A **569**, 125771 (2021)
20. Rashid, A., Abbas, M., Ismail, A.I.M., Majid, A.A.: Numerical solution of coupled viscous Burgers equations by Chebyshev–Legendre pseudo-spectral method. Appl. Math. Comput. **245**, 372–381 (2014)
21. Zhang, Y., Lin, J., Reutskiy, S., Sun, H., Feng, W.: The improved backward substitution method for the simulation of time-dependent nonlinear coupled Burgers' equations. Results Phys. **18**, 103231 (2020)

Publisher's Note