Advances in Difference Equations
a SpringerOpen Journal

RESEARCH Open Access

Simulation of advection–diffusion–dispersion equations based on a composite time discretization scheme

Sunyoung Bu[1] and Soyoon Bak[2*]

*Correspondence: jiya525@knu.ac.kr
[2]Department of Mathematics, Kyungpook National University, Daegu, Republic of Korea
Full list of author information is available at the end of the article

Abstract
In this work, we develop a high-order composite time discretization scheme based on classical collocation and integral deferred correction methods in a backward semi-Lagrangian framework (BSL) to simulate nonlinear advection–diffusion–dispersion problems. The third-order backward differentiation formula and fourth-order finite difference schemes are used in temporal and spatial discretizations, respectively. Additionally, to evaluate function values at non-grid points in BSL, the constrained interpolation profile method is used. Several numerical experiments demonstrate the efficiency of the proposed techniques in terms of accuracy and computation costs, compare with existing departure traceback schemes.

Keywords: Time-discretization method; Semi-Lagrangian method; Advection–diffusion equation; Advection–dispersion equation; Burgers' equations; Korteweg-de Vries–Burgers' equation

1 Introduction

This study focuses on the advection–diffusion–dispersion equation described as

$$\begin{cases} \mathbf{u}_t + \mathcal{J}\mathbf{u} - \nu\Delta\mathbf{u} - \mu\nabla^3\mathbf{u} = 0, & t > t_s, \mathbf{x} \in \Omega, \\ \mathbf{u}(t_s,\mathbf{x}) = \mathbf{u}_0(\mathbf{x}), & \mathbf{x} \in \Omega, \\ \mathbf{u}(t,\mathbf{x}) = \mathbf{q}(t,\mathbf{x}), & t > t_s, \mathbf{x} \in \partial\Omega, \end{cases} \tag{1}$$

where $\mathbf{u} := u$ or $[u,v]^T$ denotes the solution of (1); ν is a positive kinematic viscosity; μ is a dispersive coefficient; $\mathcal{J} := u_x$ or $\left[\begin{smallmatrix} u_x & u_y \\ v_x & v_y \end{smallmatrix}\right]$; operator ∇^3 denotes third-order partial derivatives or their sum; $\mathbf{x} := x$ or $[x,y]^T$; x and y represent special coordinates in the longitudinal and transverse directions, respectively; and t denotes time. Moreover, $\mathbf{u}_0(\mathbf{x})$ and $\mathbf{q}(t,\mathbf{x})$ are the given initial and boundary conditions, and $\partial\Omega$ is the boundary of the computational domain $\Omega = [x_L,x_R]$ or $[x_L,x_R] \times [y_L,y_R]$.

© The Author(s) 2020. This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Many physical phenomena can be described by the advection–diffusion or advection–dispersion equations, including various nonlinear time-dependent partial differential equations (PDEs), such as the Burgers-type and Korteweg–de Vries (KdV) type equations. In particular, the Burgers-type problems are simple models to describe various physical problems, such as turbulence flow, approximate theories of shock waves, vorticity transportation, and dispersion in porous media [9, 11, 13]. Many powerful methods to simulate Burgers-type problems have been proposed [2, 4, 8, 10, 12, 17, 20–23, 32, 35, 36] and have drawn research interest to find methodologies for nonlinear time-dependent PDEs in many fields, among which the characteristic methods are very attractive [14].

As one of the characteristic methods, the semi-Lagrangian method is one of the most popular numerical methods to simulate flow problems in computational fluid dynamics [1, 4, 6, 7, 27, 31, 33]. Among them, the backward semi-Lagrangian method (BSL) describes the movement of particles in the Lagrangian description and resets the positions of the particles to the Eulerian grid point at each time level to reduce the accumulation and collision of particles. Moreover, it is well known that BSL exhibits good stability, allowing the use of larger temporal steps than the spatial grid size by solving equations implicitly along characteristic curves of particles in the reverse direction to time evolution [3, 5, 30, 31]. Therefore, this paper proposes a robust high-order BSL algorithm for solving nonlinear advection–diffusion–dispersion equations. To do this, we first observe the abstract model (1) in the Lagrangian description, in which individual fluid particles are followed through time. Suppose that the position of a particle at time t is described by a function $\mathbf{x}^*(t)$. In both the Lagrangian and the Eulerian descriptions, the velocity of the particle at time t is satisfied as follows:

$$\frac{d\mathbf{x}^*(t)}{dt} = \mathbf{u}\bigl(t, \mathbf{x}^*(t)\bigr). \tag{2}$$

Then the model equation (1) can be expressed as

$$\frac{D}{Dt}\mathbf{u}\bigl(t, \mathbf{x}^*(t)\bigr) = \nu \triangle \mathbf{u}\bigl(t, \mathbf{x}^*(t)\bigr) + \mu \nabla^3 \mathbf{u}\bigl(t, \mathbf{x}^*(t)\bigr) \tag{3}$$

about the material derivative. By solving (2) at the same time as (3), we obtain the solution $\mathbf{u}$ of the model equation.

In the conventional BSL, several approximate methods are required to solve (3) simultaneously with (2): time and spatial discretization methods for (3), a departure traceback method for nonlinear ordinary differential equation (2), and an interpolation scheme to obtain the solution at non-Eulerian grid points. These approximate methods affect the overall time-to-space accuracy of the BSL. Additionally, a high-order stable temporal discretization method is critical to provide an accurate numerical approximation for the model problem, and it potentially allows for a sparser time step size than lower-order methods. Nevertheless, most current methods focus on developing spatial discretization schemes, since the strong nonlinearity of (3) is coupled with the unknown solution to (2) (or the original problem (1)), which is a significant difficulty for BSLs.

The main contributions of this paper are as follows. We propose an adaptable high-order BSL with good stability to solve nonlinear advection–diffusion–dispersion equations. It is possible to design a high-order departure traceback method, which simultaneously finds the departure points for the three previous time steps in the reverse-flow direction. The departure traceback method combines the methods of classical collocation and error correction, which is called the deferred or defect correction in some literature. The proposed departure traceback method is A-stable, verified by plotting, and determining stability function limits. Also, we use the third-order backward differentiation formula (BDF3) and fourth-order finite difference schemes for diffusion–dispersion problem discretization in the temporal and spatial domains, respectively. Moreover, the constrained interpolation profile (CIP) method [26] is used to evaluate function values at non-grid points. Several numerical experiments demonstrate that the proposed scheme provides third-order accuracy in both temporal and spatial for nonlinear advection–diffusion or –dispersion problems. In terms of computational costs, the proposed scheme is more efficient compared with current methods, including third-order AB (AB3) and fourth-order AM (AM4) [15], since it is not an iterative approach. The proposed scheme allows a more massive time step, which has a particularly significant impact on stiffer initial conditions of smaller viscosity.

The remainder of this paper is structured as follows. Section 2 provides temporal and spatial semi-discretization systems to solve a diffusion–dispersion problem (3). Section 3 designs the departure traceback method to find particle traces along characteristic curves in the reverse time direction by solving the nonlinear initial value problem (IVP) and discusses the stability of the proposed method. Section 4 provides several numerical examples to demonstrate the robustness and adaptability of the proposed scheme as well as the physical behaviors of Burgers-type equations. Finally, Sect. 5 summarizes and concludes the paper.

2 Semi-discretization system of diffusion–dispersion problem

This section provides the implementation of the temporal and spatial discretization method to obtain a semi-discretization system for (3). We find approximate solutions u and $\mathbf{u}$ at $t = t_{m+1}$ on each spatial grid point by assuming that the approximate values u_i^k and $\mathbf{u}_{i,j}^k$ of $u(t_k, \mathbf{x}_i)$ and $\mathbf{u}(t_k, \mathbf{x}_{i,j})$ for $k \leq m$ and particle departure points $\mathbf{x}_{i,j}^*(t_{m-\kappa})$, $\kappa = 0, 1, 2$, along the characteristic curve arriving at the grid point at target time $t = t_{m+1}$ have already been calculated at all grid points where a finite set of mesh points $\{\mathbf{x}_i\}$ or $\{\mathbf{x}_{i,j}\}$, $(i = 1, 2, \dots, J_x, j = 1, 2, \dots, J_y)$ is equally spaced.

For the two-dimensional (2D) case, we first evaluate (3) at $(t, \mathbf{x}) = (t_{m+1}, \mathbf{x}_{i,j})$ and then apply BDF3 to approximate the material derivative on the left side of (3). Then we obtain the Helmholtz equation

$$\mathbf{u}(t_{m+1}, \mathbf{x}_{i,j}) - \bar{\nu}\triangle\mathbf{u}(t_{m+1}, \mathbf{x}_{i,j}) - \bar{\mu}\nabla^3\mathbf{u}(t_{m+1}, \mathbf{x}_{i,j}) = \mathbf{g}_{i,j}^{m+1}(\mathbf{u}) + \mathcal{O}(h^4) \tag{4}$$

with

$$\mathbf{g}_{i,j}^{m+1}(\mathbf{u}) := \frac{18\mathbf{u}(t_m, \mathbf{x}_{i,j}^*(t_m)) - 9\mathbf{u}(t_{m-1}, \mathbf{x}_{i,j}^*(t_{m-1})) + 2\mathbf{u}(t_{m-2}, \mathbf{x}_{i,j}^*(t_{m-2}))}{11},$$

where $\bar{\nu} = \nu\frac{6h}{11}$ and $\bar{\mu} = \mu\frac{6h}{11}$ for a uniform temporal step size h.

The one-dimensional (1D) Helmholtz equation can be obtained in the same way as the 2D Helmholtz equation.

To obtain a semi-discretization system for (4), we use finite difference weight matrices ((17) and (18)) for diffusion (or Laplace) and dispersion terms in (3), respectively. We set $\nabla^3 = \frac{\partial^3}{\partial x^3}$ and $\nabla^3 = \frac{\partial^3}{\partial x^3} + \frac{\partial^3}{\partial y^3}$ for description simplicity and obtain the following systems depending on the problem dimensionality.

- 1D problems

$$
\begin{aligned}
&\mathcal{L}\mathbf{U}^{m+1} = \mathbf{g}^{m+1} + \bar{\nu}\mathbf{b}_2^{m+1} + \bar{\mu}\mathbf{b}_3^{m+1}, \qquad \mathcal{L} = \left(\mathcal{I}_x - \bar{\nu}\hat{\mathcal{D}}^{(2)} - \bar{\mu}\hat{\mathcal{D}}^{(3)}\right), \\
&\mathbf{U}^{m+1} = \left[u_1^{m+1}, u_2^{m+1}, \dots, u_{\widehat{J}_x}^{m+1}\right]^T, \\
&\mathbf{g}^{m+1} = \left[g^{m+1}(x_1), g^{m+1}(x_2), \dots, g^{m+1}(x_{\widehat{J}_x})\right]^T,
\end{aligned}
\tag{5}
$$

where $\mathbf{b}_2^{m+1}$ and $\mathbf{b}_3^{m+1}$ are the vectors induced from the boundary conditions.

- 2D problems

$$
\begin{aligned}
&\mathcal{L}\mathbf{U}^{m+1} = \mathbf{g}^{m+1}(u) + \bar{\nu}\mathbf{b}_2^{m+1} + \bar{\mu}\mathbf{b}_3^{m+1}, \qquad \mathcal{L}\mathbf{V}^{m+1} = \mathbf{g}^{m+1}(v) + \bar{\nu}\mathbf{b}_2^{m+1} + \bar{\mu}\mathbf{b}_3^{m+1} \\
&\mathcal{L} = \left(\mathcal{I}_y \otimes \mathrm{I}_x - \bar{\nu}\left(\mathcal{I}_y \otimes \hat{\mathcal{D}}_x^{(2)} + \hat{\mathcal{D}}_y^{(2)} \otimes \mathcal{I}_x\right) - \bar{\mu}\left(\mathcal{I}_y \otimes \hat{\mathcal{D}}_x^{(3)} + \hat{\mathcal{D}}_y^{(3)} \otimes \mathcal{I}_x\right)\right), \\
&\mathbf{U}^{m+1} = \left\{U^{m+1}, V^{m+1}\right\}, \\
&U^{m+1} = \left[u_{1,1}^{m+1}, u_{2,1}^{m+1}, \dots, u_{\widehat{J}_y,1}^{m+1}, u_{1,2}^{m+1}, \dots, u_{\widehat{J}_y,2}^{m+1}, \dots, u_{1,\widehat{J}_x}^{m+1}, \dots, u_{\widehat{J}_y,\widehat{J}_x}^{m+1}\right]^T, \\
&V^{m+1} = \left[v_{1,1}^{m+1}, v_{2,1}^{m+1}, \dots, v_{\widehat{J}_y,1}^{m+1}, v_{1,2}^{m+1}, \dots, v_{\widehat{J}_y,2}^{m+1}, \dots, v_{1,\widehat{J}_x}^{m+1}, \dots, v_{\widehat{J}_y,\widehat{J}_x}^{m+1}\right]^T, \\
&\mathbf{g}^{m+1} = \left[g_{1,1}^{m+1}, g_{2,1}^{m+1}, \dots, g_{\widehat{J}_y,1}^{m+1}, g_{1,2}^{m+1}, \dots, g_{\widehat{J}_y,2}^{m+1}, \dots, g_{1,\widehat{J}_x}^{m+1}, \dots, g_{\widehat{J}_y,\widehat{J}_x}^{m+1}\right]^T,
\end{aligned}
\tag{6}
$$

where $\mathbf{b}_2^{m+1}$ and $\mathbf{b}_3^{m+1}$ are vectors induced from the boundary conditions; $\mathcal{I}_x$ and $\mathcal{I}_y$ are identity matrices of size $\widehat{J}_x = J_x - 1$ and $\widehat{J}_y = J_y - 1$, respectively; and $\hat{\mathcal{D}}_x^{(k)}$ and $\hat{\mathcal{D}}_y^{(k)}$ are constructed from $\mathcal{D}_x^{(k)}$ and $\mathcal{D}_y^{(k)}$ $(k = 2, 3)$ using interior elements only. (See Appendix A for more details.)

Approximate values for particle departure points $\mathbf{x}^*(t_{m-\kappa})$, $(\kappa = 0, 1, 2)$ along the characteristic curve arriving at the grid point at target time $t = t_{m+1}$ are required to obtain full discretization systems for (5) and (6), which is discussed in Sect. 3. Moreover, since $\mathbf{x}_{i,j}^*(t_{m-\kappa})$, $(\kappa = 0, 1, 2)$ may not coincide with the grid points, an appropriate interpolation scheme is also required.

3 High-order *A*-stable departure traceback scheme for nonlinear IVP (2)

3.1 Proposed departure traceback scheme

This section develops the proposed third-order composite scheme without iteration to solve the strongly nonlinear IVP (2) by combining error correction [28, 30] and classical collocation methods. Let $\mathbf{x}_{i,j}^*(t)$ be the solution for the following IVP:

$$
\begin{cases}
\frac{d\mathbf{x}_{i,j}^*(t)}{dt} = \mathbf{u}(t, \mathbf{x}_{i,j}^*(t)), \quad t \in [t_{m-2}, t_{m+1}), \\
\mathbf{x}_{i,j}^*(t_{m+1}) = \mathbf{x}_{i,j},
\end{cases}
\tag{7}
$$

where $\mathbf{x}_{i,j}$ is an arbitrary grid point and $\mathbf{u}$ is the solution of the model problem (1).

To find the departure points by solving (2), we first consider the perturbed asymptotically first-order ordinary differential system

$$\begin{aligned}
&\boldsymbol{\psi}'_{i,j}(t) = \mathcal{F}\big(t, \boldsymbol{\psi}_{i,j}(t)\big) + \mathcal{O}\big(\boldsymbol{\psi}^2_{i,j}(t)\big), \quad t \le t_{m+1},\\
&\mathcal{F}\big(t, \boldsymbol{\psi}_{i,j}(t)\big) := \mathcal{J}_{i,j}(t)\boldsymbol{\psi}_{i,j}(t) + \mathbf{w}_{i,j}(t), \qquad \mathbf{w}_{i,j}(t) := \mathbf{u}\big(t, \mathbf{y}_{i,j}(t)\big) - \mathbf{u}^m_{i,j},\\
&\mathcal{J}_{i,j}(t) := \begin{bmatrix} u_x(t, \mathbf{y}_{i,j}(t)) & u_y(t, \mathbf{y}_{i,j}(t)) \\ v_x(t, \mathbf{y}_{i,j}(t)) & v_y(t, \mathbf{y}_{i,j}(t)) \end{bmatrix}
\end{aligned} \tag{8}$$

with initial value $\boldsymbol{\psi}_{i,j}(t_{m+1}) = \mathbf{0}_{2,1}$, where $\mathbf{0}_{r,s}$ denotes the zero matrix of size $r \times s$, and $\mathbf{y}_{i,j}(t)$ is a qth-order approximating polynomial for $\mathbf{x}^*_{i,j}(t)$ obtained by solving (7) that satisfies

$$\boldsymbol{\psi}_{i,j}(t) = \mathbf{x}^*_{i,j}(t) - \mathbf{y}_{i,j}(t) = \mathcal{O}\big(h^{q+1}\big).$$

We choose the first-order polygon $\mathbf{y}_{i,j}$ defined as

$$\mathbf{y}_{i,j}(t) := \mathbf{x}_{i,j} + (t - t_{m+1})\mathbf{u}^m_{i,j}, \quad \mathbf{u}^m_{i,j} = \big[u^m_{i,j}, v^m_{i,j}\big]^T, \tag{9}$$

which allows the proposed method to attain up to fourth-order accuracy eventually.

Remark 1 To construct higher-order methods, $\mathbf{y}_{i,j}(t)$ can be chosen as an explicit second-order polynomial,

$$\mathbf{y}_{i,j}(t) := \mathbf{x}_{i,j} + (t - t_{m+1})\big(2\mathbf{u}^m_{i,j} - \mathbf{u}^{m-1}_{i,j}\big) + \frac{(t - t_{m+1})^2}{2h}\big(\mathbf{u}^m_{i,j} - \mathbf{u}^{m-1}_{i,j}\big) + \mathcal{O}\big(h^3\big),$$

which implies that the proposed method could achieve sixth-order accuracy. To maintain fourth-order and above temporal accuracy overall, a stable method for fourth-order and above must be employed rather than BDf3 to discretize the total time derivative.

The proposed method is then completed by developing a method to solve equation (8) with initial condition $\boldsymbol{\psi}_{i,j}(t_{m+1}) = \mathbf{0}_{2,1}$ on $[t_{m-2}, t_{m+1}]$. By applying (22) with (23) to (8),

$$\Psi = \hat{h}\left(\begin{bmatrix} \frac{19}{72} & -\frac{5}{72} & \frac{1}{72} \\ \frac{4}{9} & \frac{1}{9} & 0 \\ \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{bmatrix} \otimes \mathcal{I}_2\right)\mathbb{F} + \hat{h}\begin{bmatrix} \frac{1}{8} \\ \frac{1}{9} \\ \frac{1}{8} \end{bmatrix} \otimes \mathcal{F}\big(t_{m+1}, \boldsymbol{\psi}_{i,j}(t_{m+1})\big) + \mathcal{O}\big(h^4\big), \tag{10}$$

where $\Psi = [[\boldsymbol{\psi}^m_{i,j}]^T, [\boldsymbol{\psi}^{m-1}_{i,j}]^T, [\boldsymbol{\psi}^{m-2}_{i,j}]^T]^T$; $\boldsymbol{\psi}^{m-\kappa}_{i,j}$ $(\kappa = 0, 1, 2)$ is an approximation of $\boldsymbol{\psi}_{i,j}(t_{m-\kappa})$; $\mathbb{F} := [\mathcal{F}(t_m, \boldsymbol{\psi}_{i,j}(t_m))^T, \mathcal{F}(t_{m-1}, \boldsymbol{\psi}_{i,j}(t_{m-1}))^T, \mathcal{F}(t_{m-2}, \boldsymbol{\psi}_{i,j}(t_{m-2}))^T]^T$, $\hat{h} := -3h$ and $\mathcal{I}_2$ is the identity matrix of size 2. However, since $\mathcal{F}(t_{m+1}, \boldsymbol{\psi}_{i,j}(t_{m+1}))$ contains the unknown solution $\mathbf{u}$ for (1) at time $t = t_{m+1}$ to be obtained, an approximation tool is required. Therefore, we use extrapolation to find

$$\Psi = \hat{h}\left(\begin{bmatrix} \frac{19}{72} & -\frac{5}{72} & \frac{1}{72} \\ \frac{4}{9} & \frac{1}{9} & 0 \\ \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{bmatrix} \otimes \mathcal{I}_2\right)\mathbb{F} + \hat{h}\begin{bmatrix} \frac{1}{8} \\ \frac{1}{9} \\ \frac{1}{8} \end{bmatrix} \otimes \boldsymbol{\mu}^{m+1}_p + \mathcal{O}\big(h^{2+p}\big), \quad p = 1, 2, \tag{11}$$

where

$$\mathcal{F}\big(t_{m+1},\boldsymbol{\psi}_{i,j}(t_{m+1})\big)=\boldsymbol{\mu}_p^{m+1}+\mathcal{O}\big(h^{2+p}\big),$$
$$\boldsymbol{\mu}_1^{m+1}:=2\mathbf{U}_{i,j}^{m}-3\mathbf{U}_{i,j}^{m-1}+\mathbf{U}_{i,j}^{m-2},\qquad \boldsymbol{\mu}_2^{m+1}:=3\mathbf{U}_{i,j}^{m}-6\mathbf{U}_{i,j}^{m-1}+4\mathbf{U}_{i,j}^{m-2}-\mathbf{U}_{i,j}^{m-3}.$$

We proceed by simplifying (11), and expressing in matrix form

$$\mathcal{A}\Psi=\mathcal{B},\tag{12}$$
$$\mathcal{A}=\begin{bmatrix}24\mathcal{I}+19h\mathcal{J}_m & -5h\mathcal{J}_m & h\mathcal{J}_m\\ 32h\mathcal{J}_m & 24\mathcal{I}+8h\mathcal{J}_m & \mathbf{0}_{2,2}\\ 27h\mathcal{J}_m & 27h\mathcal{J}_m & 24\mathcal{I}+9h\mathcal{J}_m\end{bmatrix},$$
$$\mathcal{B}=-h\begin{bmatrix}19\mathbf{w}_{i,j}(t_m)-5\mathbf{w}_{i,j}(t_{m-1})+\mathbf{w}_{i,j}(t_{m-2})+9\boldsymbol{\mu}_k^{m+1}\\ 32\mathbf{w}_{i,j}(t_m)+8\mathbf{w}_{i,j}(t_{m-1})+8\boldsymbol{\mu}_k^{m+1}\\ 27\mathbf{w}_{i,j}(t_m)+27\mathbf{w}_{i,j}(t_{m-1})+9\mathbf{w}_{i,j}(t_{m-2})+9\boldsymbol{\mu}_k^{m+1}\end{bmatrix},$$

where $\mathcal{I}$ is the 2×2 identity matrix. Finally, by combining (9) with (12) and truncating the asymptotic error term $\mathcal{O}(h^{2+p})$, the approximation $\mathbf{x}_{i,j}^{*,m-\kappa}$ of the three departure points can be approximated as follows:

$$\begin{aligned}&\mathbf{x}_{i,j}^{*}(t_{m-\kappa})=\mathbf{x}_{i,j}^{*,m-\kappa}+\mathcal{O}\big(h^{2+p}\big),\\ &\mathbf{x}_{i,j}^{*,m-\kappa}:=\mathbf{y}_{i,j}(t_{m-\kappa})+\boldsymbol{\psi}_{i,j}^{m-\kappa},\quad \kappa=0,1,2.\end{aligned}\tag{13}$$

3.2 Linear stability of the proposed departure traceback scheme

In this subsection, to examine the stability of the proposed departure traceback scheme, the scheme is applied to the Dahlquist problem $\frac{d\mathbf{x}^*(t)}{dt}=\lambda\mathbf{x}^*(t)$ with initial $\mathbf{x}^*(t_{m+1})=\mathbf{x}^{*,m+1}$ and an eigenvalue λ. For each departure position at the previous time steps $t=t_{m-\kappa}$ ($\kappa=0,1,2$), stability functions of the proposed departure traceback scheme hold in the following theorem.

Theorem 1 *The three departure points of the proposed departure traceback scheme* (13) *are all A-stable schemes.*

Proof The linear stability functions of the three departure points described in (13) hold in the following:

$$\begin{aligned}&\mathbf{x}^{*,m}=\mathcal{S}^{m}(z)\mathbf{x}^{*,m+1},\qquad \mathcal{S}^{m}(z)=\big(12-6z-z^2+z^3\big)/\mathcal{S}(z),\\ &\mathbf{x}^{*,m-1}=\mathcal{S}^{m-1}(z)\mathbf{x}^{*,m+1},\qquad \mathcal{S}^{m-1}(z)=\big(12+6z-z^2-z^3\big)/\mathcal{S}(z),\\ &\mathbf{x}^{*,m-2}=\mathcal{S}^{m-2}(z)\mathbf{x}^{*,m+1},\qquad \mathcal{S}^{m-2}(z)=\big(12+18z+11z^2+3z^3\big)/\mathcal{S}(z),\end{aligned}\tag{14}$$

where $\mathcal{S}(z)=12-18z+11z^2-3z^3$ and $z=-\lambda h$. Additionally, the stability regions $\mathrm{R}(\mathcal{S}^{m-\kappa}):=\{z\in\mathbb{C}\mid|\mathcal{S}^{m-\kappa}(z)|<1\}$ ($\kappa=0,1,2$) are plotted in Fig. 1. Since the regions of absolute stability $R(\mathcal{S}^{m-\kappa})$ contain the left half plane $\{z\in\mathbb{C}\mid\mathcal{R}(\mathcal{S}^{m-\kappa})(z)<0\}$, one can confirm that the proposed scheme is A-stable. □

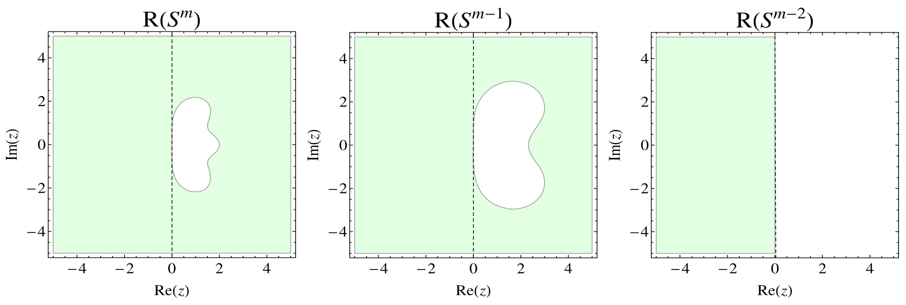

Figure 1 Regions of absolute stability

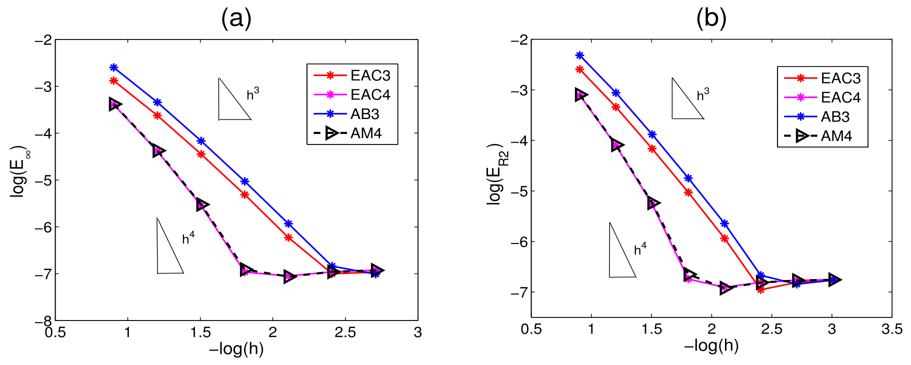

Figure 2 Verification of numerical temporal convergence for the four considered methods for Example 4.1

Remark that $\boldsymbol{\mu}_p^{m+1}$ ($p = 1, 2$) in (11) or (12) does not affect the linear stability of the proposed departure traceback scheme, since $\mathbf{u}(t, \mathbf{x}^*(t)) = \lambda\mathbf{x}^*(t)$ and its Jacobian is λ. As shown numerically in Fig. 2, the approximation order of $\boldsymbol{\mu}_p^{m+1}$ affects the temporal accuracy of the overall algorithm.

Moreover, the existing departure traceback schemes that will be compared with our method in the next section have the stability functions [15]

- AB3: $\mathbf{x}^{*,m-\kappa} = \mathcal{S}_{\mathrm{AB3}}^{m-\kappa}(z)\mathbf{x}^{*,m+1}$ ($\kappa = 0, 1, 2$) with $\mathcal{S}_{\mathrm{AB3}}^{m}(z) = \frac{6+2z}{6-4z+z^2}$, $\mathcal{S}_{\mathrm{AB3}}^{m-1}(z) = \frac{6+8z+5z^2}{6-4z+z^2}$, $\mathcal{S}_{\mathrm{AB3}}^{m-2}(z) = \frac{6+14z+16z^2}{6-4z+z^2}$.
- AM4: $\mathbf{x}^{*,m-\kappa} = \mathcal{S}_{\mathrm{AM4}}^{m-\kappa}(z)\mathbf{x}^{*,m+1}$ ($\kappa = 0, 1, 2$) with $\mathcal{S}_{\mathrm{AM4}}^{m}(z) = \frac{24-6z-7z^2}{24-30z+11z^2}$, $\mathcal{S}_{\mathrm{AM4}}^{m-1}(z) = \frac{4+18z-z^2-6z^3}{24-30z+11z^2}$, $\mathcal{S}_{\mathrm{AM4}}^{m-2}(z) = \frac{24+42z+29z^2-30z^3}{24-30z+11z^2}$.

Remark 2 Note that while all stability functions of the proposed method are A-stable, the stability functions of AB3 and AM4 are neither A-stable nor L-stable at $t = t_{m-\kappa}$, $\kappa = 1, 2$. The impoverished stability constrains the choice of time step size h, when the problem has small viscosity, hence losing the advantages of the BSL.

4 Numerical experiments of advection–diffusion or –dispersion equation

To assess the efficiency of the proposed method, we simulate six examples. To measure the accuracy, we calculate the computational errors using the maximum norm, L_2 norm and relative L_2 norm, respectively, defined by

$$E_\infty(t_k) = \|u^k(\mathbf{x}_i) - u_i^k\|_\infty, \qquad E_2(t_k) = \left(\Delta x \sum_i |u^k(\mathbf{x}_i) - u_i^k|^2\right)^{1/2},$$
$$E_{R2}(t_k) = \left(\sum_i |u^k(\mathbf{x}_i) - u_i^k|^2\right)^{1/2} / \left(\sum_i |u^k(\mathbf{x}_i)|^2\right)^{1/2},$$

where $u^k(\mathbf{x}_i)$ is the exact solution at $t = t_k$ and each grid point $\mathbf{x}_i$ and u_i^k is an approximation. $E_\infty(t_k)$ and $E_{R2}(t_k)$ are similarly defined for the 2D case. The numerical convergence for the proposed method, when the exact solution of the problem is available, is calculated as

$$rate := \log_2\left(E^k_{\Delta\tau} / E^k_{\frac{\Delta\tau}{2}}\right),$$

where E^k represents $E_\infty(t_k)$ or $E_{R2}(t_k)$ and $\Delta\tau$ represents h or Δx, respectively.

For convenience, we label the proposed departure traceback schemes and current schemes as follows.

- *EAC3*: Proposed method (13) with $\boldsymbol{\mu}_1^{m+1}$ in (11)
- *EAC4*: Proposed method (13) with $\boldsymbol{\mu}_2^{m+1}$ in (11)
- *AB3*: Third-order Adams–Bashforth method in [15]
- *AM4*: Fourth-order Adams–Moulton method in [15].

Remark 3 To find departure points $\mathbf{x}^*_{i,j}(t)$ at three previous time levels $t = t_{m-\kappa}$ ($\kappa = 0, 1, 2$) arriving on the grid points at target $t = t_{m+1}$, AB3 and AM4 are implicit schemes in our BSL, since they simultaneously find the departure points by solving the strongly nonlinear IVP (2) with initial $\mathbf{x}^*_{i,j}(t_{m+1}) = \mathbf{x}_{i,j}$ in the reverse direction to time. Therefore, an iteration method, such as the Newton-like iteration scheme, is required for the current departure traceback schemes. We employed the fixed iteration with maximum iteration $iter = 20$ and tolerance $tol = 10^{-7}$, considering computational costs.

4.1 Linear advection equation

This subsection provides experimental results to demonstrate convergence for the proposed BSL for a linear advection problem. To do this, we consider the linear advection problem

$$\begin{aligned} &\frac{\partial}{\partial t}\rho(t,\mathbf{x}) + \bar{\boldsymbol{v}}(t,\mathbf{x}) \cdot \nabla\rho(t,\mathbf{x}) = 0, \quad \mathbf{x} \in (-6,6)^2,\ 0 < t \le 1, \\ &\rho(0,\mathbf{x}) = \exp\left(-\|\mathbf{x}\|^2\right), \end{aligned} \tag{15}$$

where $\bar{\boldsymbol{v}}(t,\mathbf{x}) := \frac{1}{2}\mathbf{x}\tan(t)$ is a known velocity, and the analytical solution of (15) is $\rho(t,\mathbf{x}) = \exp(-\|\mathbf{x}\|^2\cos(t))$, $\|\mathbf{x}\|^2 = x^2 + y^2$. Using the semi-Lagrangian formulation, we rewrite (15) as $\rho_T(t,\mathbf{x}) = 0$, along the characteristic curve $\mathbf{x}^*(t;\mathbf{x},s)$ satisfying $\frac{d\mathbf{x}^*(t;\mathbf{x},s)}{dt} = \bar{\boldsymbol{v}}(t,\mathbf{x})$, which is a different type from the equation discussed previously. Specifically, the distribution function ρ is constant along the characteristic curve, i.e., $\rho(t_{m+1},\mathbf{x}_{i,j}) = \rho(t_m,\mathbf{x}^*_{i,j}(t_m))$, and hence, we only need the departure point $\mathbf{x}^*_{i,j}(t)$ at $t = t_m$.

Table 1 Computational costs *cpu* to obtain the results in Fig. 2 at $t = 1.0$

h	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}
EAC3	2.632	6.751	15.016	31.522	64.257	129.938	272.247
EAC4	2.816	6.774	15.122	31.687	65.195	131.261	273.875
AB3	1.889	4.381	9.327	19.668	39.724	81.652	160.135
AM4	2.023	4.772	10.171	21.252	43.359	88.768	180.081

To verify the temporal convergence order of each method, we measure E_∞ and E_{R2} and check the computational cost (*cpu*) for a fixed spatial resolution $J_x = J_y = 12 \times 100$ by varying the temporal step size h from 2^{-3} to 2^{-9}. The results (the errors versus temporal step size h with a logarithmic scale based on 10) are plotted in Fig. 2 and illustrated in Table 1. The figure shows that the EAC3 and AB3 exhibit third-order convergence rates and EAC4 and AM4 have fourth-order convergence rates. However, EAC3 and EAC4 have greater computational costs than AB3 and AM4, since EAC3 and EAC4 simultaneously compute all three departure points with high accuracy by solving the linear system (12), whereas AB3 and AM4 are designed to calculate with good stability only at $t = t_m$. These results seem that AB3 and AM4 are more efficient than the proposed methods for linear hyperbolic-type problems.

4.2 1D Burgers' equation

This section investigates the performance of the proposed method combined with BDF3 for the nonlinear advection–diffusion problem and demonstrates its superiority compared with AB3 and AM4. As an illustrative example, we consider a 1D Burgers' equation

$$u_t + uu_x = \nu u_x x, \quad x \in [0,1],\ t > 0 \tag{16}$$

with the following initial condition $u(0,x) = \frac{2\nu\pi \sin(\pi x)}{\sigma + \cos(\pi x)}$, $x \in [0,1]$ and the Dirichlet boundary condition. The analytic solution for this example is given as [32, 35]

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

where $\sigma > 1$ is a parameter determining the shape of the initial function, and σ and stiffness are inversely proportional.

We first numerically investigate temporal convergence for the four methods by measuring the errors E_∞ and E_{R2} for the smooth initial condition with $\sigma = 100$ and $\nu = 0.1$. The experiment is conducted by varying the time step h from 2^{-3} to 2^{-7} with spatial resolution $J_x = 400$, as shown in Table 2. All four methods, EAC3, EAC4, AB3, and AM4, combined with BDF3 achieve third-order time convergence. To check spatial convergence for the proposed method, simulation is performed until final time $t_f = 1.0$ for various spatial resolutions J_x from 2^3 to 2^7 with $h = 1/4000$ as shown in Table 3. The proposed EAC3 and EAC4 achieve at least third-order spatial accuracy. To demonstrate the superiority of EAC3, E_{R2} and *cpu* are obtained via the three methods, EAC3, AB3 and AM4, by using the time step size $h = 1/100, 1/200, 1/400$, and $1/800$ and the spatial resolution $J_x = 1000$ until $t = 1.0$; we display the *cpu* versus E_{R2} in Fig. 3. The figure clearly shows that EAC3 significantly outperforms the other methods, particularly as σ reduces (i.e., increasing initial stiffness or flow velocity). The initial condition becomes steeper and the solution speed increases as σ decreases from 100 to 1.2. Additionally, we measure the $E_\infty(t)$ and $E_2(t)$ by

Table 2 Temporal convergence rates for Example 4.2 with fixed spatial resolution $J_x = 400$

h	$\sigma = 100, \nu = 0.1$									
	EAC3					AB3				
	$E_\infty(t)$	$rate_t$	$E_{R2}(t)$	$rate_t$	cpu	$E_\infty(t)$	$rate_t$	$E_{R2}(t)$	$rate_t$	cpu
2^{-3}	8.74×10^{-7}	–	3.71×10^{-4}	–	0.07	8.70×10^{-7}	–	3.71×10^{-4}	–	0.14
2^{-4}	1.24×10^{-7}	2.82	5.26×10^{-5}	2.82	0.17	1.23×10^{-7}	2.82	5.25×10^{-5}	2.82	0.31
2^{-5}	1.62×10^{-8}	2.93	6.91×10^{-6}	2.93	0.34	1.61×10^{-8}	2.93	6.89×10^{-6}	2.93	0.65
2^{-6}	2.08×10^{-9}	2.97	8.84×10^{-7}	2.97	0.74	2.06×10^{-9}	2.97	8.78×10^{-7}	2.97	1.20
2^{-7}	2.62×10^{-10}	2.98	1.12×10^{-7}	2.98	1.38	2.64×10^{-10}	2.97	1.11×10^{-7}	2.99	2.38

h	EAC4					AM4				
	$E_\infty(t)$	$rate_t$	$E_{R2}(t)$	$rate_t$	cpu	$E_\infty(t)$	$rate_t$	$E_{R2}(t)$	$rate_t$	cpu
2^{-3}	7.14×10^{-7}	–	3.03×10^{-4}	–	0.08	7.09×10^{-7}	–	3.03×10^{-4}	–	0.15
2^{-4}	1.14×10^{-7}	2.64	4.87×10^{-5}	2.64	0.17	1.14×10^{-7}	2.63	4.87×10^{-5}	2.64	0.38
2^{-5}	1.57×10^{-8}	2.87	6.67×10^{-6}	2.87	0.34	1.61×10^{-8}	2.83	6.80×10^{-6}	2.84	0.87
2^{-6}	2.04×10^{-9}	2.94	8.69×10^{-7}	2.94	0.76	2.33×10^{-9}	2.79	9.64×10^{-7}	2.82	1.75
2^{-7}	2.60×10^{-10}	2.97	1.11×10^{-7}	2.97	1.45	4.72×10^{-10}	2.30	2.04×10^{-7}	2.24	3.72

Table 3 Spatial convergence rates for the proposed methods for Example 4.2 with fixed spatial resolution $h = 1/4000$

J_x	$\sigma = 100, \nu = 0.1$									
	EAC3					EAC4				
	$E_\infty(t)$	$rate_t$	$E_{R2}(t)$	$rate_t$	cpu	$E_\infty(t)$	$rate_t$	$E_{R2}(t)$	$rate_t$	cpu
2^3	8.11×10^{-4}	–	8.97×10^{-4}	–	0.72	8.10×10^{-4}	–	8.97×10^{-4}	–	0.71
2^4	1.33×10^{-4}	2.61	1.32×10^{-4}	2.77	1.34	1.33×10^{-4}	2.61	1.32×10^{-4}	2.77	1.34
2^5	9.97×10^{-6}	3.73	9.97×10^{-6}	3.73	2.64	9.97×10^{-6}	3.73	9.96×10^{-6}	3.73	2.65
2^6	6.72×10^{-7}	3.89	6.71×10^{-7}	3.89	6.11	6.71×10^{-7}	3.89	6.71×10^{-7}	3.89	6.17
2^7	7.69×10^{-8}	3.13	7.69×10^{-8}	3.13	12.80	7.69×10^{-8}	3.13	7.69×10^{-8}	3.13	13.01

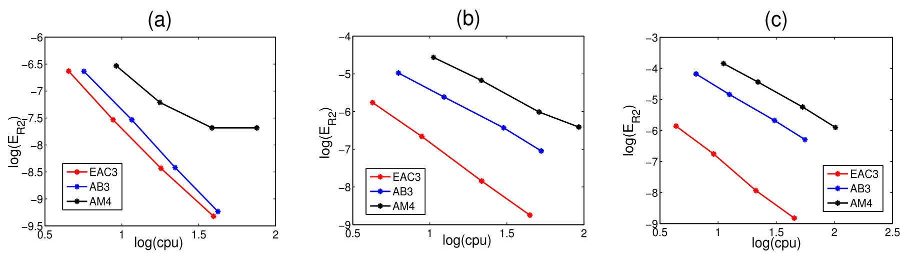

Figure 3 Accuracy and computational efficiency for Example 4.2 with (**a**) $\sigma = 100$, (**b**) $\sigma = 2$, and (**c**) $\sigma = 1.2$

EAC3 and other existing methods introduced in [29, 35]. The results are listed in Table 4 and show that EAC3 is more accurate than the compared methods presented in [29, 35].

4.3 Coupled Burgers' equations

As the third test problem, the coupled Burgers' equations are considered on the domain $[-20, 20]$ defined as

$$\begin{cases} u_t - u_{xx} - 2uu_x + (uv)_x = 0, \\ v_t - v_{xx} - 2vv_x + (uv)_x = 0, \end{cases}$$

Table 4 Comparison of $E_\infty(t)$ and $E_2(t)$ errors of EAC3 to the existing methods for Example 4.2 with $h = 0.01$ and $\nu = 0.005$, $\sigma = 100$ at $t = 1.0$

J_x	EAC3		Tamsir [35]		Mittal [29]	
	$E_\infty(t)$	$E_2(t)$	$E_\infty(t)$	$E_2(t)$	$E_\infty(t)$	$E_2(t)$
10	8.461×10^{-8}	1.416×10^{-8}	1.467×10^{-7}	6.330×10^{-8}	4.88×10^{-7}	3.455×10^{-7}
20	6.038×10^{-9}	9.009×10^{-10}	3.029×10^{-8}	1.014×10^{-8}	1.431×10^{-7}	1.012×10^{-7}
40	4.251×10^{-10}	4.654×10^{-11}	3.956×10^{-9}	1.207×10^{-9}	5.668×10^{-8}	4.003×10^{-8}
80	5.328×10^{-11}	4.149×10^{-12}	8.861×10^{-11}	1.322×10^{-10}	3.499×10^{-8}	4.002×10^{-8}

Table 5 Numerical accuracy of EAC3 for Example 4.3 at $t = 1.0$

$h = \Delta x$	u				v				*cpu*
	$E_\infty(t)$	*rate*	$E_{R2}(t)$	*rate*	$E_\infty(t)$	*rate*	$E_{R2}(t)$	*rate*	
2^{-3}	6.73×10^{-4}	–	1.65×10^{-3}	–	7.16×10^{-4}	–	1.73×10^{-3}	–	0.010
2^{-4}	9.41×10^{-5}	2.84	2.35×10^{-4}	2.81	1.00×10^{-4}	2.84	2.46×10^{-4}	2.81	0.011
2^{-5}	1.24×10^{-5}	2.92	3.15×10^{-5}	2.90	1.32×10^{-5}	2.92	3.30×10^{-5}	2.90	0.047
2^{-6}	1.59×10^{-6}	2.96	4.05×10^{-6}	2.96	1.70×10^{-6}	2.96	4.25×10^{-6}	2.96	0.563
2^{-7}	2.02×10^{-7}	2.98	5.14×10^{-7}	2.98	2.15×10^{-7}	2.98	5.39×10^{-7}	2.98	5.050

Table 6 Comparison of accuracy of three methods for different time steps in Example 4.3

t	u					
	EAC3		AB3		AM4	
	$E_\infty(t)$	$E_{R2}(t)$	$E_\infty(t)$	$E_{R2}(t)$	$E_\infty(t)$	$E_{R2}(t)$
0.5	2.76×10^{-7}	3.93×10^{-7}	1.91×10^{-6}	2.83×10^{-6}	4.74×10^{-3}	7.50×10^{-3}
1.0	2.01×10^{-7}	5.13×10^{-7}	1.40×10^{-6}	3.69×10^{-6}	7.40×10^{-3}	1.87×10^{-2}
2.0	6.67×10^{-8}	4.81×10^{-7}	5.45×10^{-7}	4.01×10^{-6}	5.14×10^{-3}	3.32×10^{-2}
3.0	1.86×10^{-8}	3.71×10^{-7}	1.97×10^{-7}	3.95×10^{-6}	2.46×10^{-3}	4.09×10^{-2}

t	v					
	EAC3		AB3		AM4	
	$E_\infty(t)$	$E_{R2}(t)$	$E_\infty(t)$	$E_{R2}(t)$	$E_\infty(t)$	$E_{R2}(t)$
0.5	2.89×10^{-7}	4.10×10^{-7}	1.89×10^{-6}	2.82×10^{-6}	4.74×10^{-3}	7.50×10^{-3}
1.0	2.14×10^{-7}	5.38×10^{-7}	1.40×10^{-6}	3.70×10^{-6}	7.40×10^{-3}	1.87×10^{-6}
2.0	7.07×10^{-8}	5.08×10^{-7}	5.48×10^{-7}	4.04×10^{-6}	5.14×10^{-3}	3.32×10^{-2}
3.0	1.99×10^{-8}	3.97×10^{-7}	1.98×10^{-7}	3.98×10^{-6}	2.46×10^{-3}	4.09×10^{-2}

where we consider the exact solution to be [4, 22, 24]

$$u_1(t,x) = u_2(t,x) = \exp(-t)\sin(x), \quad x \in [-\pi,\pi],\ t > 0.$$

Boundary and initial conditions are as used for the exact solution.

To demonstrate the robustness and superiority of the proposed method, we numerically estimate the convergence order of EAC3 and *cpu*. We measure convergence rates for both solutions u and v simultaneously at $t = 1.0$ for the same temporal and spatial step sizes $h = \Delta x$ from 2^{-3} to 2^{-7}. As shown in Table 5, EAC3 clearly exhibits the expected third-order convergence rate. To demonstrate the accuracy and superiority of EAC3, we implement the experiment of the example and compare the numerical results obtained by EAC3, AB3, and AM4. We compare the errors, E_∞ and E_{R2}, obtained by the three methods for different time levels $t = 0.5, 1.0, 2.0$, and 3.0 with $h = 2^{-7}$ and $J_x = 1024$. The results are listed in Table 6 and Fig. 4. EAC3 exhibits significantly superior accuracy, showing

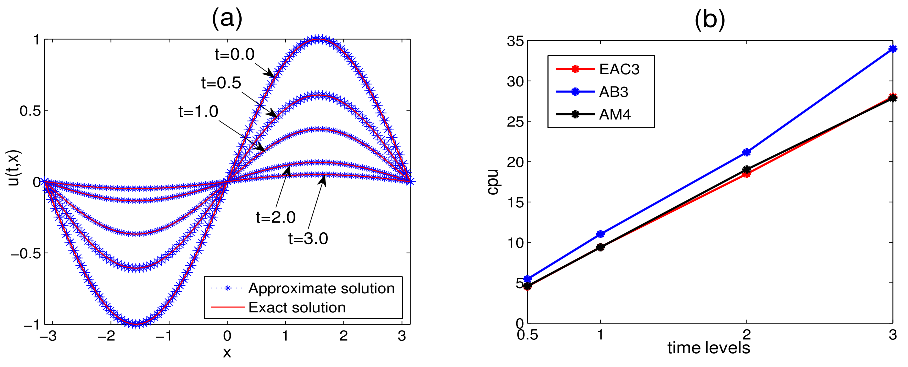

Figure 4 (**a**) Approximate and exact solution behaviors for different time steps, and (**b**) comparison of *cpu* for the three departure traceback schemes from Table 6

Table 7 Numerical accuracy for Example 4.4 using EAC3 with $h = 0.1$ and $J_x = 20$ on [–10, 10] for various values of ε, ν and η

ε	ν	η	$E_\infty(t)$			$E_{R2}(t)$		
			$t = 0.3$	$t = 0.6$	$t = 0.9$	$t = 0.3$	$t = 0.6$	$t = 0.9$
1.0	0.1	1.0	9.11×10^{-17}	3.52×10^{-15}	3.39×10^{-14}	9.75×10^{-15}	2.26×10^{-13}	2.09×10^{-12}
	0.1	0.1	3.23×10^{-10}	2.09×10^{-9}	3.95×10^{-9}	4.86×10^{-9}	3.03×10^{-8}	5.56×10^{-8}
	0.1	0.01	3.12×10^{-4}	2.12×10^{-3}	4.31×10^{-3}	2.89×10^{-4}	1.85×10^{-3}	3.63×10^{-3}
0.1	0.1	1.0	9.02×10^{-16}	3.49×10^{-14}	3.37×10^{-13}	9.62×10^{-15}	2.24×10^{-13}	2.08×10^{-12}
	0.1	0.1	3.23×10^{-9}	2.09×10^{-8}	3.95×10^{-8}	4.86×10^{-9}	3.30×10^{-8}	5.56×10^{-8}
	0.1	0.01	6.49×10^{-4}	4.42×10^{-3}	8.98×10^{-3}	2.89×10^{-4}	1.85×10^{-3}	3.63×10^{-3}

excellent agreement with the exact solutions and slightly superior computational costs compared with AB3 and AM4 (see Fig. 4(b)).

4.4 KdV–Burgers' equation

This section investigates the adaptability of the proposed method for the problem with dispersive flow. Consider the following KdV–Burgers equation:

$$u_t + \varepsilon u u_x - \nu u_{xx} + \eta u_{xxx} = 0, \quad x \in [x_L, x_R]$$

with the exact solitary wave solution as given by ([25])

$$u(t,x) = a_0\bigl(9 - 6\tanh\bigl(a_1(x - a_2 t)\bigr) - 3\tanh^2\bigl(a_1(x - a_2 t)\bigr)\bigr),$$

where $a_0 = \frac{\nu^2}{25\varepsilon\eta}$, $a_1 = \frac{\nu}{10\eta}$, $a_2 = \frac{6\nu^2}{25\eta}$; ε, and ν and η are arbitrary constants. The KdV–Burgers' equation applies to studying weak effects of dispersion, dissipation, and nonlinearity in waves propagating in a liquid-filled elastic tube [34].

To investigate the accuracy and efficiency of the proposed method, we solve the problem with fixed sparse grid sizes $h = 0.1$ and $J_x = 10$ by varying ε, ν, and η, as shown in Table 7. We measure errors E_∞ and E_{R2} at $t = 0.3, 0.6$, and 0.9, and the numerical solutions show very good agreement with the exact solutions. We observe the graphical representation of the time-progressive evolution for the traveling solution with small $\mu = 0.01, 0.001$. Hence, we employ $h = 0.02$, $J_x = 800$ and $h = 0.01$, $J_x = 1000$ as shown in Fig. 5. The figure shows

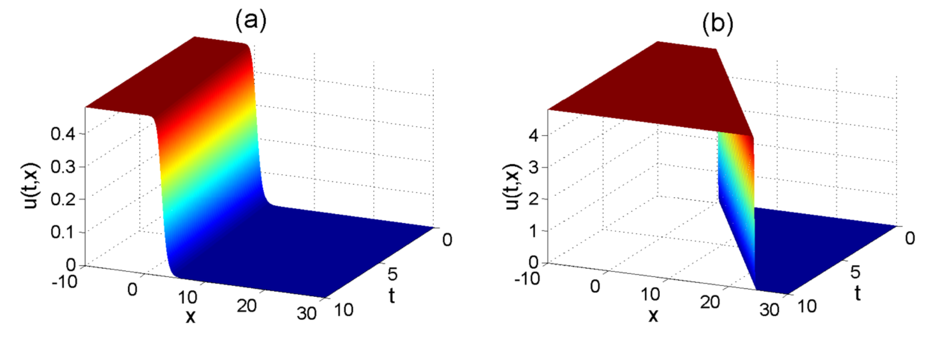

Figure 5 Graphical representation of the time-progressive evolution for traveling solution with (**a**) $\mu = 0.01$, $h = 0.02$, $J_x = 800$ and (**b**) $\mu = 0.001$, $h = 0.01$, $J_x = 1000$ during $t \le 10$

Table 8 Convergence rates of EAC3 for Example 4.5 at $t = 0.5$ and $t = 1.0$ with $\nu = 0.1$

$h = \Delta x = \Delta y$	$t = 0.5$					$t = 1.0$				
	$E_\infty(t)$	*rate*	$E_{R2}(t)$	*rate*	*cpu*	$E_\infty(t)$	*rate*	$E_{R2}(t)$	*rate*	*cpu*
1/50	1.89×10^{-3}	–	4.08×10^{-4}	–	0.039	3.87×10^{-4}	–	4.91×10^{-5}	–	0.073
1/100	3.02×10^{-4}	2.64	6.58×10^{-5}	2.63	0.255	6.40×10^{-5}	2.60	7.04×10^{-6}	2.80	0.512
1/200	4.21×10^{-5}	2.84	9.18×10^{-6}	2.84	2.097	9.13×10^{-6}	2.81	9.65×10^{-7}	2.87	4.147
1/400	5.64×10^{-6}	2.90	1.31×10^{-6}	2.81	22.033	1.20×10^{-6}	2.93	1.28×10^{-8}	2.92	41.431

that EAC3 works well providing good performance for the traveling solution with small $\mu = 0.01$ and 0.001.

4.5 2D Burgers' equation

In this subsection, to observe numerical accuracy in 2D nonlinear advection–diffusion equations, we perform the test problem using the 2D unsteady Burgers' equation,

$$u_t + u(u_x + u_y) = \nu \Delta u, \quad \mathbf{x} \in [-0.5, 0.5]^2,$$

with the analytic solution from [37]

$$u(t, \mathbf{x}) = \frac{1}{2} - \tanh\left(\frac{x + y - t}{2\nu}\right), \quad \mathbf{x} = (x, y).$$

Initial and boundary conditions are taken directly from the analytic solution.

We first examine the accuracy of EAC3 for $\nu = 0.1$ by varying the same time step and space grid sizes, $h = \Delta x = \Delta y$ from 1/50 to 1/400 for two different time levels. As shown in Table 8, EAC3 provides third-order convergence.

We also observe the behaviors of the numerical solutions obtained by EAC3 for different viscosities $\nu = 0.1, 0.01$, and 0.001 at $t = 1.0, 2.0$, and 3.0 using the time step, and spatial grid sizes $h = \Delta x = \Delta y = 0.01, 0.002$, and 0.0005 are used. Figure 6 shows that EAC3 has good performance and works well for small viscosity coefficients. Finally, by comparing the errors and computational cost *cpu* with the results of AB3, we demonstrate the efficiency of the proposed method at two different time levels $t = 0.05$ and 1.0. The results displayed in Table 9 show that the proposed method EAC3 provides more accurate solutions with lower computational cost than AB3, clearly indicating that EAC3 is more efficient than AB3.

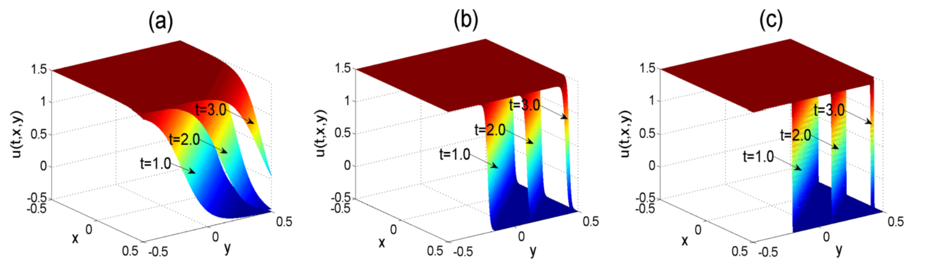

Figure 6 Time evolution of the numerical solution with (**a**) $\nu = 0.1$, $h = \Delta x = \Delta y = 0.01$; (**b**) $\nu = 0.01$, $h = \Delta x = \Delta y = 0.002$; (**c**) $\nu = 0.001$, $h = \Delta x = \Delta y = 0.0005$

Table 9 Comparison of numerical results from EAC3 and AB3 for Example 4.5

ν	h	$J_x = J_y$	EAC3			AB3		
			$E_\infty(0.05)$	$E_{R2}(0.05)$	*cpu*	$E_\infty(0.05)$	$E_{R2}(0.05)$	*cpu*
1.0	0.005	40	4.92×10^{-10}	4.89×10^{-10}	0.012	1.66×10^{-8}	1.38×10^{-8}	0.014
0.1	0.0025	80	1.85×10^{-6}	7.59×10^{-7}	0.066	3.45×10^{-5}	1.50×10^{-6}	0.079
0.01	0.00125	320	5.11×10^{-4}	6.96×10^{-5}	2.156	1.10×10^{-3}	1.33×10^{-4}	2.913

ν	h	$J_x = J_y$	EAC3			AB3		
			$E_\infty(1.0)$	$E_{R2}(1.0)$	*cpu*	$E_\infty(1.0)$	$E_{R2}(1.0)$	*cpu*
1.0	0.005	40	3.26×10^{-9}	1.74×10^{-9}	0.182	4.09×10^{-8}	2.05×10^{-8}	0.225
0.1	0.0025	160	1.16×10^{-6}	1.11×10^{-7}	5.124	4.52×10^{-6}	4.61×10^{-7}	6.428
0.01	0.00125	320	6.86×10^{-5}	7.52×10^{-7}	47.232	1.83×10^{-4}	1.16×10^{-6}	47.937

4.6 System of Burgers' equations

As the last test problem, consider the system of 2D Burgers' equations:

$$\mathbf{u}_t(t,\mathbf{x}) + \mathcal{J}\mathbf{u}(t,\mathbf{x}) - \nu\Delta\mathbf{u}(t,\mathbf{x}) = 0, \quad \mathbf{x} \in [0,1]^2$$

with analytic solution [19, 25, 35, 38]

$$u(t,\mathbf{x}) = \frac{3}{4} - \frac{1}{4(1+\exp(\frac{4y-4x-t}{32\nu}))}, \qquad v(t,\mathbf{x}) = \frac{3}{4} + \frac{1}{4(1+\exp(\frac{4y-4x-t}{32\nu}))}.$$

To observe the behaviors of the numerical solutions, the graphical results of the time evolution for the numerical solutions obtained by EAC3 are observed for small viscosity $\nu = 0.0001$ at three different time levels $t = 1.0, 2.0$, and 3.0 with $h = 0.001$ and $\Delta x = \Delta y = 1200$. The results depicted in Fig. 7 show good performance with small viscosity $\nu = 0.0001$.

To investigate the accuracy and efficiency of EAC3, we measure the errors of u, v, and *cpu* for different viscosities $\nu = 1.0, 0.1, 0.01$, and 0.005 for various h and $J_x = J_y$ at $t = 0.5$ and 2.0. Table 10 shows that the numerical solutions of EAC3 exhibit very good agreement with the exact solutions for various viscosity coefficients.

5 Conclusions

In this paper, an adaptable and stable BSL was developed to solve advection–diffusion–dispersion problems induced from fluid dynamics by introducing a high-order depar-

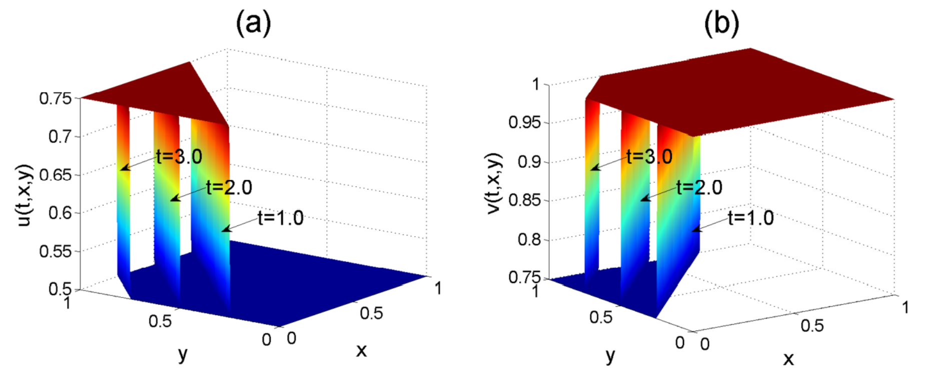

Figure 7 Time evolution of the solution (**a**) u (**b**) v with $\nu = 0.0001$, $h = 0.001$, $J_x = J_y = 1200$ for Example 4.6

Table 10 Numerical accuracy of Example 4.6 using EAC3 with different parameters

ν	h	$J_x = J_y$	u		v		cpu
			$E_\infty(0.5)$	$E_{R2}(0.5)$	$E_\infty(0.5)$	$E_{R2}(0.5)$	
1.0	0.1	20	8.11×10^{-6}	3.57×10^{-6}	8.11×10^{-6}	2.54×10^{-6}	0.005
0.1	0.025	80	5.58×10^{-5}	1.41×10^{-5}	5.58×10^{-5}	9.81×10^{-6}	0.093
0.01	0.0125	160	6.74×10^{-4}	9.98×10^{-5}	6.74×10^{-4}	6.67×10^{-5}	0.755
0.005	0.00625	320	6.85×10^{-4}	6.89×10^{-5}	6.85×10^{-4}	4.59×10^{-5}	7.092

ν	h	$J_x = J_y$	u		v		cpu
			$E_{R\infty}(2.0)$	$E_{R2}(2.0)$	$E_{R\infty}(2.0)$	$E_{R2}(2.0)$	
1.0	0.1	20	8.17×10^{-6}	3.62×10^{-6}	8.17×10^{-6}	2.56×10^{-6}	0.011
0.1	0.025	80	5.35×10^{-5}	1.64×10^{-5}	5.35×10^{-5}	1.06×10^{-5}	0.441
0.01	0.0125	160	6.86×10^{-4}	2.12×10^{-4}	6.86×10^{-4}	1.18×10^{-4}	3.282
0.005	0.00625	320	6.86×10^{-4}	1.40×10^{-4}	6.86×10^{-5}	7.76×10^{-5}	28.964

ture scheme, which is a A-stable with up to fourth-order accuracy. In particular, the scheme is designed to solve the strongly nonlinear IVP and to have high-order accuracy while having good stability for all approximation of the departure points at three previous times. It is experimentally shown to be more suitable for solving advection–diffusion–dispersion type problems than other conventional high-order departure traceback methods. Moreover, it is more efficient, compared with other methods, in that the proposed method does not need an iterative technique that is required to solve the strong nonlinear IVP. Through several numerical experiments, it was shown that the proposed scheme is superior to the conventional high-order departure traceback schemes and the latest solvers of Burgers' equation, in terms of accuracy and computational costs.

Appendix A: Finite difference method for space discretization

To implement the proposed BSL, we require fourth-order finite difference formulas of order k ($k = 1,2,3$) to approximate a function's partial derivatives. Therefore, this appendix introduces fourth-order finite difference matrices of size $(J_x + 1) \times (J_x + 1)$ from the discrete partial derivative operators for the Dirichlet boundary condition

[16]

$$\mathcal{D}_x^{(1)} = \frac{1}{12\Delta x}\begin{bmatrix} -25 & 48 & -36 & 16 & -3 & \cdots & \\ -3 & -10 & 18 & -6 & 1 & & \\ 1 & -8 & 0 & 8 & -1 & & \\ & \ddots & \ddots & \ddots & \ddots & & \\ & & 1 & -8 & 0 & 8 & -1 \\ & & -1 & 6 & -18 & 10 & 3 \\ & \cdots & 3 & -16 & 36 & -48 & 25 \end{bmatrix},$$

$$\mathcal{D}_x^{(2)} := \frac{1}{12\Delta x^2}\begin{bmatrix} 45 & -154 & 214 & -156 & 61 & -10 & \\ 10 & -15 & -4 & 14 & -6 & 1 & \\ -1 & 16 & -30 & 16 & -1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & -1 & 16 & -30 & 16 & -1 \\ & 1 & -6 & 14 & -4 & -15 & 10 \\ & -10 & 61 & -156 & 214 & -154 & 45 \end{bmatrix}, \quad (17)$$

and

$$\mathcal{D}_x^{(3)} = \frac{1}{8\Delta x^3}\begin{bmatrix} -49 & 232 & -461 & 496 & -307 & 104 & -15 & \cdots & \\ -15 & 56 & -83 & 64 & -29 & 8 & -1 & & \\ -1 & -8 & 35 & -48 & 29 & -8 & 1 & & \\ 1 & -8 & 13 & 0 & -13 & 8 & -1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & 1 & -8 & 13 & 0 & -13 & 8 & -1 \\ & & -1 & 8 & -29 & 48 & -35 & 8 & 1 \\ & & 1 & -8 & 29 & -64 & 83 & -56 & 15 \\ & \cdots & 15 & -104 & 307 & -496 & 461 & -232 & 49 \end{bmatrix}, \quad (18)$$

where the matrix elements were obtained from MATHEMATICA. Using these finite difference matrices [16], partial derivatives $\frac{\partial^k}{\partial x^k} f_i$ or $\frac{\partial^k}{\partial x^k} f_{i,j}$ ($k = 1, 2, 3$) of a given function f can be approximated as follows.

1D case

$$\frac{\partial^k}{\partial x^k} f_i = \left(\mathcal{D}_x^{(k)} \boldsymbol{f}\right)_i + \mathcal{O}\left(\Delta x^4\right),$$

2D case

$$\frac{\partial^k}{\partial x^k} f_{i,j} = \left(\mathcal{D}_x^{(k)} \boldsymbol{F}\right)_{i,j} + \mathcal{O}\left(\Delta x^4\right), \qquad \frac{\partial^k}{\partial y^k} f_{i,j} = \left(\boldsymbol{F}\left(\mathcal{D}_y^{(k)}\right)^T\right)_{i,j} + \mathcal{O}\left(\Delta y^4\right),$$

where $f_i = f(\mathbf{x}_i)$; $f_{i,j} = f(\mathbf{x}_{i,j})$; $\boldsymbol{f} = (f_i)_{(J_x+1)\times 1}$ and $\boldsymbol{F} = (f_{i,j})_{(J_x+1)\times(J_y+1)}$; Δx and Δy are uniform spatial grid sizes along the x- and y-directions, respectively; T denotes matrix transposition; and $(\boldsymbol{a})_i$ and $(\boldsymbol{A})_{i,j}$ denote the ith and (i,j)th components of vector $\boldsymbol{a}$ and matrix $\boldsymbol{A}$, respectively.

Appendix B: Collocation method for equidistant collocation nodes

This section reviews classical one-step collocation as a solver of the IVP,

$$\begin{cases} \boldsymbol{\phi}'(t) = \mathbf{f}(t, \boldsymbol{\phi}(t)), & t \in (t_m, t_{m+1}], \\ \boldsymbol{\phi}(t_m) = \boldsymbol{\phi}_m, \end{cases} \tag{19}$$

where $\mathbf{f}$ is assumed to satisfy the conditions guaranteeing the existence of a unique solution.

Suppose that we have an approximation to the solution for (19) at point t_m for a set of equally spaced points on the integration interval $t_s = t_0 < t_1 < \cdots < t_M = t_f$, where we set $t_m = t_s + mh$ $(m = 0, 1, 2, \ldots, M)$ and $h = (t_f - t_s)/M$. We employed the classical collocation method [18] to obtain the approximation $\boldsymbol{\varphi}(t_{m+1})$ for $\boldsymbol{\phi}(t_{m+1})$. To design the fourth-order scheme in the BSL, we used equidistant collocation nodes $t_{m,j} := t_m + c_j h$ $(j = 0, 1, 2, 3)$ with $c_j = \frac{j}{3}$. Consider the Lagrange form of the interpolating polynomial,

$$\begin{aligned} &\boldsymbol{\phi}'(t) = \sum_{k=0}^{3} \boldsymbol{\phi}'(t_{m,k}) \mathcal{L}_k(\xi) + \mathcal{R}(\xi), \\ &\mathcal{L}_k(\xi) = \prod_{j=0, j\neq k}^{3} \frac{\xi - c_j}{c_k - c_j}, \qquad \mathcal{R}(\xi) := \boldsymbol{\phi}'[t_{m,0}, \ldots, t_{m,3}] \prod_{j=0}^{3} (\xi - c_j), \end{aligned} \tag{20}$$

for some $t = t_m + \xi h$ $(0 \le \xi \le 1)$ and divided difference $\boldsymbol{\phi}'[t_{m,0}, \ldots, t_{m,3}]$.

By integrating and evaluating (20) at the collocation points,

$$\boldsymbol{\phi}(t_{m,j}) = \boldsymbol{\phi}(t_m) + h \sum_{k=0}^{3} \mathbf{f}(t_{m,k}) \int_0^{c_j} \mathcal{L}_k(\eta)\, d\eta + h \int_0^{c_j} \mathcal{R}(\eta)\, d\eta. \tag{21}$$

By truncating the last term in (21), we obtain a four-stage implicit collocation method (or implicit Runge–Kutta method),

$$\boldsymbol{\varphi}(t_{m+1}) = \boldsymbol{\varphi}(t_m) + h \sum_{k=0}^{3} b_k K_{m,k}, \quad K_{m,k} = \mathbf{f}\left(t_{m,k}, \boldsymbol{\varphi}(t_m) + h \sum_{j=0}^{3} a_{kj} K_{m,j} \right) \tag{22}$$

with Butcher tableau

$$\begin{array}{c|cccc} c_0 & a_{00} & a_{01} & a_{02} & a_{03} \\ c_1 & a_{10} & a_{11} & a_{12} & a_{13} \\ c_2 & a_{20} & a_{21} & a_{22} & a_{23} \\ c_3 & a_{30} & a_{31} & a_{32} & a_{33} \\ \hline & b_0 & b_1 & b_2 & b_3 \end{array} = \begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{8} & \frac{19}{72} & -\frac{5}{72} & \frac{1}{72} \\ \frac{2}{3} & \frac{1}{9} & \frac{4}{9} & \frac{1}{9} & 0 \\ 1 & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \\ \hline & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{array}, \tag{23}$$

where

$$a_{jk} := \int_0^{c_j} \mathcal{L}_k(\eta)\, d\eta, \qquad b_k = \int_0^{1} \mathcal{L}_k(\eta)\, d\eta, \quad j, k = 0, 1, 2, 3.$$

We remark that both the stage and the convergence orders of the collocation method (22) are four, which can be easily checked (see [18] for details).

Acknowledgements
The authors would like to thank the reviewers for their valuable comments and suggestions, which helped improve its presentation overall.

Funding
This work was supported by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (grant number NRF-2019R1F1A1058378).

Availability of data and materials
Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

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

Authors' contributions
All authors provided the basic idea of this work. The corresponding author S. Bak simulated the numerical examples and write the draft of the manuscript, and the other author S. Bu revised the manuscript. All authors read and approved the final manuscript.

Author details
[1]Department of Liberal Arts, Hongik University, Sejong, Republic of Korea. [2]Department of Mathematics, Kyungpook National University, Daegu, Republic of Korea.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 3 June 2019 Accepted: 10 March 2020 Published online: 20 March 2020

References
1. Allievi, A., Bermejo, R.: Finite element modified method of characteristics for the Navier–Stokes equations. Int. J. Numer. Methods Fluids **32**(4), 439–463 (2000)
2. Arora, G., Singh, B.K.: Numerical solution of Burgers' equation with modified cubic B-spline differential quadrature method. Appl. Math. Comput. **224**, 166–177 (2013)
3. Bak, S.: High-order characteristic-tracking strategy for simulation of a nonlinear advection–diffusion equations. Numer. Methods Partial Differ. Equ. **35**(5), 1756–1776 (2019)
4. 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)
5. Bak, S., Kim, P., Piao, X., Bu, S.: Numerical solution of advection–diffusion type equation by modified error correction scheme. Adv. Differ. Equ. **2018**, 432 (2018)
6. Bermúdez, A., Nogueiras, M.R., Vázquez, C.: Numerical analysis of convection-diffusion-reaction problems with higher order characteristics/finite elements. Part I: time discretization. SIAM J. Numer. Anal. **44**, 1829–1853 (2006)
7. Bermúdez, A., Nogueiras, M.R., Vázquez, C.: Numerical analysis of convection-diffusion-reaction problems with higher order characteristics/finite elements. Part II: fully discretized scheme and quadrature formulas. SIAM J. Numer. Anal. **44**, 1854–1876 (2006)
8. 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)
9. Burgers, J.M.: A mathematical model illustrating the theory of turbulence. Adv. Appl. Mech. **1**, 171–199 (1948)
10. Chen, Y., Zhang, T.: A weak Galerkin finite element method for Burgers' equation. J. Comput. Appl. Math. **348**, 103–119 (2019)
11. Cole, J.D.: On a quasi-linear parabolic equation occurring in aerodynamics. Q. Appl. Math. **9**, 225–236 (1951)
12. Egidi, N., Maponi, P., Quadrini, M.: An integral equation method for numerical solution of the Burgers' equation. Comput. Math. Appl. **76**, 35–44 (2018)
13. Esipov, S.E.: Coupled Burgers' equations: a model of poly-dispersive sedimentation. Phys. Rev. **52**, 3711–3718 (1995)
14. Ewing, R.E., Wang, H.: A summary of numerical methods for time-dependent advection-dominated partial differential equations. J. Comput. Appl. Math. **128**, 423–445 (2001)
15. Filbet, F., Prouveur, C.: High order time discretization for backward semi-Lagrangian methods. J. Comput. Appl. Math. **303**, 171–188 (2016)
16. Fornberg, B.: Calculation of weights in finite difference formulas. SIAM Rev. **40**(3), 685–691 (1998)
17. Gowrisankar, S., Natesan, S.: An efficient robust numerical method for singularly perturbed Burgers' equation. Appl. Math. Comput. **346**, 385–394 (2019)
18. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems. Springer Series in Computational Mathematics. Springer, Berlin (1996)
19. Haq, S., Ghafoor, A.: An efficient numerical algorithm for multi-dimensional time dependent partial differential equations. Comput. Math. Appl. **75**, 2723–2734 (2018)
20. Jiwari, R.: A harr wavelet quasilinearization approach for numerical simulation of Burgers' equation. Comput. Phys. Commun. **183**, 2413–2423 (2012)
21. Jiwari, R.: A hybrid numerical scheme for the numerical solution of the Burgers' equation. Comput. Phys. Commun. **188**, 59–67 (2015)
22. Jiwari, R., Kumar, S., Mittal, R.C.: Meshfree algorithms based on radial basis functions for numerical simulation and to capture shocks behavior of Burgers' type problems. Eng. Comput. **36**(4), 1142–1168 (2019)

23. Jiwari, R., Tomasiello, S., Tornabene, F.: A numerical algorithm for computational modelling of coupled advection–diffusion–reaction systems. Eng. Comput. **35**(3), 1383–1401 (2018)
24. Kaya, D.: An explicit solution of coupled viscous Burgers' equation by the decomposition method. Int. J. Math. Math. Sci. **27**, 675–680 (2001)
25. Khater, A.H., Temash, R.S., Hassan, M.M.: A Chebyshev spectral collocation method for solving Burgers'-types equations. J. Comput. Appl. Math. **222**, 333–350 (2008)
26. Kim, D., Song, O., Ko, H.: A semi-Lagrangian CIP fluid solver without dimensional splitting. Comput. Graph. Forum **27**, 467–475 (2008)
27. Kim, P., Kim, D., Piao, X., Bak, S.: A completely explicit scheme of Cauchy problem in BSLM for solving the Navier–Stokes equations. J. Comput. Phys. **401**, 109028 (2020)
28. Kim, P., Piao, X., Kim, S.D.: An error corrected Euler method for solving stiff problems based on Chebyshev collocation. SIAM J. Numer. Anal. **49**, 2211–2230 (2011)
29. Mittal, R.C., Jiwari, R.: A differential quadrature method for solving Burgers' type equation. Int. J. Numer. Methods Heat Fluid Flow **22**(7), 880–895 (2012)
30. Piao, X., Bu, S., Bak, S., Kim, P.: An iteration free backward semi-Lagrangian scheme for solving incompressible Navier–Stokes equations. J. Comput. Phys. **283**, 189–204 (2015)
31. Robert, A.: A stable numerical integration scheme for the primitive meteorological equations. Atmos.-Ocean **19**(1), 35–46 (1981)
32. Seydaoğlu, M., Erdoğan, U., Öziş, T.: Numerical solution of Burgers' equation with high order splitting methods. J. Comput. Appl. Math. **291**, 410–421 (2016)
33. Staniforth, A., Coté, J.: Semi-Lagrangian integration schemes for atmospheric models—a review. Mon. Weather Rev. **119**, 2206–2223 (1991)
34. Su, C.H., Gardner, C.S.: Korteweg–de Vries equation and generalizations. III. Derivation of the Korteweg–de Vries equation and Burgers equation. J. Math. Phys. **10**(3), 536–539 (1969)
35. Tamsir, M., Srivastava, V.K., Jiwari, R.: An algorithm based on exponential modified cubic B-spline differential quadrature method for nonlinear Burgers' equation. Appl. Math. Comput. **290**, 111–124 (2016)
36. Yadav, O.P., Jiwari, R.: Finite element analysis and approximation of Burgers'–Fisher equation. Numer. Methods Partial Differ. Equ. **33**(5), 1652–2677 (2017)
37. Zhang, J., Yan, G.: Lattice Boltzmann method for one and two-dimensional Burgers' equation. Physica A **387**, 4771–4786 (2008)
38. Zhang, X., Tian, H., Chen, W.: Local method of approximate particular solutions for two-dimensional unsteady Burgers' equations. Comput. Math. Appl. **66**, 2425–2432 (2014)

Submit your manuscript to a SpringerOpen journal and benefit from:

► Convenient online submission
► Rigorous peer review
► Open access: articles freely available online
► High visibility within the field
► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com