Journal of Computational Physics: X 12 (2021) 100098

Contents lists available at ScienceDirect

Journal of Computational Physics: X

www.elsevier.com/locate/jcpx

A recursive system-free single-step temporal discretization method for finite difference methods

Youngjun Lee [a], Dongwook Lee [a,*], Adam Reyes [b]

[a] Department of Applied Mathematics, The University of California, Santa Cruz, CA, United States
[b] Department of Physics and Astronomy, The University of Rochester, NY, United States

ARTICLE INFO

Article history:
Received 26 February 2021
Received in revised form 14 June 2021
Accepted 28 June 2021
Available online 2 July 2021

Keywords:
Picard integration formulation
Jacobian-free and Hessian-free
Recursive
Cauchy-Kowalewski procedure
High-order method
Finite difference method

ABSTRACT

Single-stage or single-step high-order temporal discretizations of partial differential equations (PDEs) have shown great promise in delivering high-order accuracy in time with efficient use of computational resources. There has been much success in developing such methods for finite volume method (FVM) discretizations of PDEs. The Picard Integral formulation (PIF) has recently made such single-stage temporal methods accessible for finite difference method (FDM) discretizations. PIF methods rely on the so-called Lax-Wendroff procedures to tightly couple spatial and temporal derivatives through the governing PDE system to construct high-order Taylor series expansions in time. Going to higher than third order in time requires the calculation of *Jacobian-like* derivative tensor-vector contractions of an increasingly larger degree, greatly adding to the complexity of such schemes. To that end, we present in this paper a method for calculating these tensor contractions through a recursive application of a discrete Jacobian operator that readily and efficiently computes the needed contractions entirely agnostic of the system of partial differential equations (PDEs) being solved.

1. Introduction

In past decades, high-order discrete methods for hyperbolic conservation laws have become one of the central themes in computational fluid dynamics (CFD) due to their potential in achieving highly accurate predictions in balance with trends in current high-performance computing (HPC) architectures. Using high-order methods allows better solution accuracy with a fixed memory profile at the cost of increased floating point operations; in line with next generation HPC systems that are seeing increased computing power with saturation in the memory per compute core.

Practitioners have focused on designing high-arithmetic-intensity data reconstruction and interpolation models in the context of finite volume methods (FVM) and finite difference methods (FDM). Under the dual computational need for accuracy and stability, the CFD community has developed high-order reconstruction and interpolation methods that can produce highly accurate solutions while avoiding unphysical oscillations near the discontinuities. Examples include the early success of the piecewise parabolic method (PPM) by Colella and Woodward [1,2], which has been still actively adopted as a shock-capturing partial differential equation (PDE) solver by many CFD users after about four decades since its introduction. In 1987, Harten et al. [3] proposed an essentially non-oscillatory (ENO) scheme that chooses the appropriate stencil adaptively to prevent spurious oscillations near strong gradients. Liu et al. [4] improved this idea by introducing the weighted

* Corresponding author.
E-mail addresses: ylee109@ucsc.edu (Y. Lee), dlee79@ucsc.edu (D. Lee), adam.reyes@rochester.edu (A. Reyes).

https://doi.org/10.1016/j.jcpx.2021.100098

essentially non-oscillatory (WENO) scheme, which takes a convex combination of all possible stencils, reducing the number of logical calculations needed for the original ENO scheme. The WENO scheme was further improved by Jiang and Shu [5] and became one of the most popular high-order reconstruction and interpolation methods for solving shock-dominant CFD simulation. Several modifications of WENO methods have been proposed over decades, such as WENO-Z [6,7], central-WENO [8,9], Hermite-WENO [10,11] WENO-AO [12], and polynomial-free GP-WENO [13,14], to name a few.

Common to all numerical PDE solvers is the solution lies on the spatio-temporal plane. Numerical errors arise from both the spatial and temporal axis; thereby, a high-order scheme requires a meticulously designed temporal discretization method that ensures the solution's accuracy and stability. However, efforts to achieve a high-order of accuracy in the temporal axis have seen a renewed effort. For decades, the strong stability preserving Runge-Kutta (SSP-RK) time integrator has been considered as the standard temporal integration strategy for an extensive range of high-order numerical schemes for PDE solvers. The key idea of SSP-RK scheme is to maintain the strong stability property (SSP) at high-order accuracy by sequentially applying convex combinations of the first-order forward Euler method as a building-block at each sub-stage of the Runge-Kutta method. In this manner, the total variation diminishing (TVD) property is achieved in each sub-stages; therefore, it ensures the whole scheme's stability.

Despite the high portability and fidelity of SSP-RK methods, the very nature of the SSP-RK method – being a multi-stage approach – increases computational resources in CFD simulations. In SSP-RK methods, the data reconstruction/interpolation and the boundary condition should be applied in each sub-stage, which increases the computational costs and the footprint of data communications in the parallel computational architecture. It makes the simulations using the adaptive mesh refinement (AMR) method less attractive, which progressively refines the grid resolutions and increases data communications around the simulations' interesting features.

To alleviate such issues, many practitioners recently have proposed single-step, high-order time updating strategies based on the so-called Lax–Wendroff (Taylor) method. The core design principle of the Lax-Wendroff method is to convert time derivatives to spatial derivatives through the Lax-Wendroff or Cauchy-Kowalewski procedure (LW/CK hereafter). In this way, the spatial and temporal derivatives are coupled through Jacobians and Hessians; thus, the numerical solutions can be updated in a single step while maintaining the high-order accuracy. In 2001, Toro et al. [15] extended this idea by combining it with the generalized Riemann problems (GRPs) and introduced the Arbitrary high order derivative Riemann problem (ADER) method. Toro and his collaborators applied LW/CK procedures to get the coefficients of the power series expansion of the conservative variables and solved GRPs for each high-order term. ADER methods were further developed in [15–17], and it has grown its popularity over decades, leading to various further modifications. Some examples include ADER-DG [18,19] and ADER-CG [20–22] in the context of discontinuous and continuous Galerkin schemes; other efforts of employing an implicit GRP solver to solve scalar equations with stiff source terms [23], its extensions to second-order schemes for non-linear systems [24] and to general hyperbolic systems [25]. The use of an implicit time Taylor series expansion for GRP was further simplified in the study by Montecinos and Balsara [26]. Along the line of simplifying the standard ADER approach, the Differential Transform Method (DTM) [27] was also adopted to alleviate the cost of the ADER scheme, coined as ADER-DT (or ADER-Taylor) in [28–30].

The Picard integral formulation (PIF) method, proposed by Christlieb et al. [31], is another Lax-Wendroff type time discretization method under the finite difference formulation, which evolves the pointwise conservative variables. Instead of taking temporally pointwise numerical fluxes as in the conventional FDM, the PIF method takes time-averaged numerical fluxes for updating the solutions in a single step. Christlieb and his collaborators demonstrated that LW/CK procedures could successfully be utilized for obtaining high-order terms of the numerical fluxes as in many Lax-Wendroff type works of literature, which results in faster CPU performance than the SSP-RK method with the same order of accuracy. Other studies also have shown that single-step temporal updates are more efficient in terms of CPU time to solution when compared to multi-stage/multi-step methods [21,32,33].

The primary advantage of LW/CK-based methods is the enhanced performance offered by being a single-step method, harnessing the tight coupling of temporal and spatial derivatives through LW/CK procedures to construct high-order time-series Taylor expansions. This becomes hugely attractive in massively parallel computing, minimizing the computational frequency of data transfers between processors each time step, which would need to be repeated for each intermediate RK stage. On the other hand, the dependence of the strong coupling on analytic derivatives of the governing PDEs makes the LW/CK approach less flexible and less broadly applicable to all systems of PDEs.

To meet this end, in [33], we proposed a novel approach to bypass the analytic evaluations of Jacobian and Hessian terms, which are the major implementation hurdles for Lax-Wendroff type methods. This new approach, which we called the system-free (SF) method, adopts the Jacobian-free idea commonly used for iterative methods [34–37]. We implemented this new approach to the original third-order PIF method, called the third-order SF-PIF method (or SF-PIF3). The new SF-PIF3 method shows the same performance results while maintaining the same order of accuracy and stability as the original PIF method. Also, by virtue of the system-independent approach for the *Jacobian-like* tensor contractions, SF-PIF3 can apply to a different hyperbolic system of equations with ease. The core design principle of our SF method is to alleviate the implementation difficulties of the LW/CK-based approaches in the standard PIF method. This idea is similar to a recent study by Montecinos and Balsara [26], where they proposed implicit Taylor series expansion to simplify the LW/CK procedures by considering space and time derivatives of the flux Jacobian in the context of the ADER schemes.

The original PIF method and SF-PIF3 are third-order in time, coupled with the fifth-order spatial reconstruction method (WENO-JS). Assuming that the temporal integration is discretized with a qth order scheme and the spatial with a pth order

scheme (often with $q \leq p$), the overall solution accuracy is determined by the leading errors of the two discretizations, namely, $\mathcal{O}\left(\Delta t^q, \Delta s^p\right)$, where Δt and Δs represent the temporal and spatial length scales. Practically speaking, the spatial errors usually dominate the temporal errors; thus, the high-order methods with $q \leq p$ are justifiable. Nonetheless, we observe that the temporal errors gradually dominate the spatial errors as we increase the grid resolution progressively, which leads us to find a higher than a third-order temporal scheme for maintaining overall solution accuracy even in the finer grid resolutions.

In Lax-Wendroff type time discretization methods, like PIF and SF-PIF3 methods, rank-4 *Jacobian-like* tensors are needed for the fourth-order approximation. Although possible, the need for such analytical handling becomes prohibitive in the Lax-Wendroff type schemes when extending their accuracy beyond third-order due to the drastic growth in complexity.

In this regard, we propose a new fourth-order extension of the SF-PIF3 method. Since SF-PIF3 bypasses all the *Jacobian-like* calculations, our SF approach is further rewarded in a fourth-order extension, which demands a leap in calculation counts for conventional Lax-Wendroff type schemes. Moreover, we present a new improved version of the previous SF approach [33], which applies *a recursive strategy* to obtain the higher-order derivative tensor-vector contractions, promising a more compact code structure and faster performance than the original SF method. With such modification, our fourth-order SF-PIF4 method shows nearly twice faster performance than the optimal fourth-order five-stage SSP-RK method.

We organize the paper as follows. In Section 2 we briefly review the general discretization strategy of the original PIF method. We present the fourth-order extension of the original PIF method in Section 3, and we apply the original SF approach to the fourth-order PIF method in Section 4. The improved recursive SF approach will be introduced in Section 5, which ensures faster performance and a more straightforward code structure. The results of various 2D and 3D benchmark problems are presented in Section 7. We conclude our paper with a summary in Section 8.

2. Picard integral formulation

We are interested in solving the general conservation system of equations in 3D,

$$\partial_t \mathbf{U} + \nabla \cdot \mathcal{F}(\mathbf{U}) = \partial_t \mathbf{U} + \partial_x \mathbf{F}(\mathbf{U}) + \partial_y \mathbf{G}(\mathbf{U}) + \partial_z \mathbf{H}(\mathbf{U}) = 0, \tag{1}$$

where $\mathbf{U}$ is the vector of conservative variables and $\mathbf{F}, \mathbf{G}$, and $\mathbf{H}$ are the flux functions in x-, y- and z-direction, respectively. In the Euler equations, the conservative variables and the flux functions are defined as,

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u\,(E+p) \end{bmatrix}, \quad \mathbf{G}(\mathbf{U}) = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v\,(E+p) \end{bmatrix}, \quad \mathbf{H}(\mathbf{U}) = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w\,(E+p) \end{bmatrix}. \tag{2}$$

Applying the Picard integral formulation (PIF) [38], we take a time average of Eq. (1) within a single time step Δt over an interval $[t^n, t^n + \Delta t] = [t^n, t^{n+1}]$,

$$\mathbf{U}^{n+1} = \mathbf{U}^n - \Delta t \left(\partial_x \mathbf{F}^{avg} + \partial_y \mathbf{G}^{avg} + \partial_z \mathbf{H}^{avg}\right), \tag{3}$$

where $\mathbf{F}^{avg}, \mathbf{G}^{avg}$, and $\mathbf{H}^{avg}$ are the time-averaged fluxes in each direction,

$$\mathbf{F}^{avg}(\mathbf{x}) \equiv \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{U}(\mathbf{x},t))\,dt, \quad \mathbf{G}^{avg}(\mathbf{x}) \equiv \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \mathbf{G}(\mathbf{U}(\mathbf{x},t))\,dt, \quad \mathbf{H}^{avg}(\mathbf{x}) \equiv \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \mathbf{H}(\mathbf{U}(\mathbf{x},t))\,dt, \tag{4}$$

for $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$.

We wish to express the spatial derivatives of the time-averaged fluxes in Eq. (3) using highly approximated numerical fluxes $\hat{\mathbf{f}}, \hat{\mathbf{g}}$, and $\hat{\mathbf{h}}$ at cell interfaces. Taking x-directional flux $\mathbf{F}$, for example, we aim to express

$$\partial_x \mathbf{F}^{avg}\Big|_{\mathbf{x}=\mathbf{x}_{ijk}} = \frac{1}{\Delta x}\left(\hat{\mathbf{f}}_{i+\frac{1}{2},j,k} - \hat{\mathbf{f}}_{i-\frac{1}{2},j,k}\right) + \mathcal{O}(\Delta x^p + \Delta t^q), \qquad \mathbf{x}_{ijk} = (x_i, y_j, z_k). \tag{5}$$

The y- and z-directional derivatives are approximated in a similar fashion.

Finding approximated solutions for the numerical fluxes in the PIF method is nearly identical to the conventional finite difference method (FDM). Following the standard convention in FDM, we treat pointwise x-directional flux $\mathbf{F}(x, y_j, z_k)$ as a 1D cell average of an auxiliary function $\hat{\mathbf{F}}$ in 1D,

$$\mathbf{F}(x, y_j, z_k) = \frac{1}{\Delta x}\int_{x+\frac{\Delta x}{2}}^{x+\frac{\Delta x}{2}} \hat{\mathbf{F}}(\xi, y_j, z_k)\,d\xi\,. \tag{6}$$

Then, by taking an analytical derivative of Eq. (6) with respect to x, we have

$$\partial_x \mathbf{F}\Big|_{\mathbf{x}=\mathbf{x}_{ijk}} = \frac{1}{\Delta x}\left(\hat{\mathbf{F}}(x_{i+\frac{1}{2}}, y_j, z_k) - \hat{\mathbf{F}}(x_{i-\frac{1}{2}}, y_j, z_k)\right). \tag{7}$$

By comparing Eq. (5) and Eq. (7), we can identify the auxiliary function $\hat{\mathbf{F}}$ as a numerical flux $\hat{\mathbf{f}}$. We can follow similar procedures to identify the rest of numerical fluxes $\hat{\mathbf{g}}$ and $\hat{\mathbf{h}}$.

Therefore, determining *spatially* approximated solutions for the numerical fluxes is the inverse problem of Eq. (6), viz. finding pointwise values at cell interfaces, given the volume-averaged quantities at cell centers, which is exactly the same way as the high-order reconstruction procedure used in 1D finite volume method (FVM). Namely, we can use the conventional high-order reconstruction procedures used in FVM for approximating numerical fluxes $\hat{\mathbf{f}}, \hat{\mathbf{g}}$ and $\hat{\mathbf{h}}$, at the desired *spatially* pth-order accuracy by using the cell-centered pointwise analytic fluxes, $\mathbf{F}_{ijk}, \mathbf{G}_{ijk}$, and $\mathbf{H}_{ijk}$. One strategic difference in the PIF method is that we use the *time-averaged fluxes*, $\mathbf{F}^{avg}$, $\mathbf{G}^{avg}$, and $\mathbf{H}^{avg}$, as the inputs of the high-order reconstruction method instead of the pointwise flux values in the conventional FDM in order to assure the anticipated qth-order temporal accuracy for the numerical fluxes $\hat{\mathbf{f}}, \hat{\mathbf{g}}$, and $\hat{\mathbf{h}}$ at cell interfaces.

The time-averaged fluxes are obtained through the Taylor expansion of the pointwise flux around t^n. In the qth-order PIF method, the time-averaged x-directional flux $\mathbf{F}^{avg}$ is approximated as,

$$\begin{aligned}
\mathbf{F}^{avg}(\mathbf{x}) &= \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{x}, t)\, dt \\
&= \mathbf{F}(\mathbf{x}, t^n) + \frac{\Delta t}{2!}\partial_t^{(1)}\mathbf{F}(\mathbf{x}, t)\Big|_{t=t^n} + \frac{\Delta t^2}{3!}\partial_t^{(2)}\mathbf{F}(\mathbf{x}, t)\Big|_{t=t^n} + \frac{\Delta t^3}{4!}\partial_t^{(3)}\mathbf{F}(\mathbf{x}, t)\Big|_{t=t^n} + \cdots \\
&= \sum_{i=0}^{q-1} \frac{\Delta t^i}{(i+1)!}\partial_t^{(i)}\mathbf{F}(\mathbf{x}, t)\Big|_{t=t^n} + \mathcal{O}(\Delta t^q) \\
&= \mathbf{F}^{appx,q}(\mathbf{x}, t^n) + \mathcal{O}(\Delta t^q).
\end{aligned} \tag{8}$$

We will use the *temporally* qth-order approximated fluxes $\mathbf{F}^{appx,q}$ as the inputs of the pth-order reconstruction scheme $\mathcal{R}(\cdot)$ that is combined with a characteristic flux splitting method $\mathcal{FS}(\cdot)$ to apply the pth-order *spatial* approximation to the numerical flux $\hat{\mathbf{f}}$ at cell interfaces,

$$\hat{\mathbf{f}}_{i+\frac{1}{2},j,k} = \mathcal{R}\left(\mathcal{FS}\left(\mathbf{F}^{appx,q}_{i-r,j}, \ldots, \mathbf{F}^{appx,q}_{i+r+1,j}\right)\right) + \mathcal{O}(\Delta x^p), \tag{9}$$

where r represents the stencil radius required for the pth-order reconstruction method, $\mathcal{R}(\cdot)$. This study uses the conventional fifth-order WENO-JS method [5] and the global Lax–Friedrichs flux splitting scheme taking the maximum wave speed over the entire domain and all characteristic fields. The choice of this global Lax-Friedrichs scheme is particularly more diffusive than other possible forms of splitting, although we have found that the added numerical dissipation becomes less significant for designing our fourth-order SF-PIF scheme without sacrificing accuracy.

Recall that the numerical flux in Eq. (9) is a high-order approximated solution *both in time and space* since we used temporally approximated inputs, $\mathbf{F}^{appx,q}$, instead of the temporally pointwise flux values. Therefore, the x-directional numerical flux derived from Eq. (8) and Eq. (9) allows us to update the solution in a single step while maintaining high order accuracy both in space and time. The y- and z-directional numerical fluxes, $\hat{\mathbf{g}}$ and $\hat{\mathbf{h}}$, are obtained in similar fashions. The fully discretized form of the governing equation is given as,

$$\mathbf{U}^{n+1}_{i,j,k} = \mathbf{U}^n_{i,j,k} - \frac{\Delta t}{\Delta x}\left(\hat{\mathbf{f}}_{i+\frac{1}{2},j,k} - \hat{\mathbf{f}}_{i-\frac{1}{2},j,k}\right) - \frac{\Delta t}{\Delta y}\left(\hat{\mathbf{g}}_{i,j+\frac{1}{2},k} - \hat{\mathbf{g}}_{i,j-\frac{1}{2},k}\right) - \frac{\Delta t}{\Delta z}\left(\hat{\mathbf{h}}_{i,j,k+\frac{1}{2}} - \hat{\mathbf{h}}_{i,j,k-\frac{1}{2}}\right). \tag{10}$$

Lastly, it is worth pointing out one practical observation. The approaches in the standard PIF and our SF-PIF reconstruct fluxes directly instead of conserved or primitive variables, where some desirable features such as positivity may be more directly controlled with conservative or primitive variables. One approach to address this issue in flux reconstruction FDM is the so-called "flux limiter" strategy, in which the high-order numerical flux is augmented with some amount of a positivity-preserving flux, such as that of the low-order Lax-Friedrichs method. A positivity-preserving PIF method has been reported in [39] based on the flux limiters and the parametrization studied in [40,41]. Although promising, we did not find it necessary to adopt such a strategy in this paper.

3. The fourth-order extension of the PIF method

The primary goal in this section is to extend the PIF method [38] to fourth-order in time. We consider a fourth-order approximated time-averaged flux in x-direction, $\mathbf{F}^{appx,4}$, from the Taylor expansion of the pointwise flux around t^n. As expressed in Eq. (8) we have,

$$\mathbf{F}^{appx,4}(\mathbf{x}) = \mathbf{F}(\mathbf{x}, t^n) + \frac{\Delta t}{2!}\partial_t^{(1)}\mathbf{F}(\mathbf{x}, t)\Big|_{t=t^n} + \frac{\Delta t^2}{3!}\partial_t^{(2)}\mathbf{F}(\mathbf{x}, t)\Big|_{t=t^n} + \frac{\Delta t^3}{4!}\partial_t^{(3)}\mathbf{F}(\mathbf{x}, t)\Big|_{t=t^n}. \tag{11}$$

The other y- and z-directional approximated fluxes, $\mathbf{G}^{appx,4}$ and $\mathbf{H}^{appx,4}$, are defined in similarly. Our main interest is transforming all the time derivatives in Eq. (11) to the corresponding spatial derivatives; thereby we could express Eq. (11) in a fully explicit form.

For simplicity, we begin to adopt a compact subscript notation of partial derivatives and omit the temporal expression of $t = t^n$. Thus, we rewrite Eq. (1) in a compact form as,

$$\mathbf{U}_t + \nabla \cdot \mathcal{F}(\mathbf{U}) = \mathbf{U}_t + \mathbf{F}_x + \mathbf{G}_y + \mathbf{H}_z = 0. \tag{12}$$

In [33], we introduced the so-called flux equation – an evolution equation of fluxes – by applying a chain rule to Eq. (12). For example, the flux equation of the x-flux is

$$\mathbf{F}_t + \mathbf{F}_\mathbf{U} \cdot \nabla^f = 0, \quad \text{where } \nabla^f = \mathbf{F}_x + \mathbf{G}_y + \mathbf{H}_z. \tag{13}$$

The above flux equation allows us to convert time derivatives of the fluxes to the spatial derivatives via the LW/CK procedure. For instance, the first-order time derivative of $\mathbf{F}$ is easily converted to the spatial derivatives as,

$$\mathbf{F}_t = -\mathbf{F}_\mathbf{U} \cdot \nabla^f. \tag{14}$$

The higher-order time derivatives could be achieved by taking partial derivatives to Eq. (14) recursively. As an example, the second-order term is written as

$$\mathbf{F}_{tt} = \mathbf{F}_{\mathbf{UU}} \cdot \nabla^f \cdot \nabla^f - \mathbf{F}_\mathbf{U} \cdot \nabla_t^f, \tag{15}$$

where

$$\nabla_t^f = -\mathbf{F}_{\mathbf{UU}} \cdot \mathbf{U}_x \cdot \nabla^f - \mathbf{F}_\mathbf{U} \cdot \nabla_x^f - \mathbf{G}_{\mathbf{UU}} \cdot \mathbf{U}_y \cdot \nabla^f - \mathbf{G}_\mathbf{U} \cdot \nabla_y^f - \mathbf{H}_{\mathbf{UU}} \cdot \mathbf{U}_z \cdot \nabla^f - \mathbf{H}_\mathbf{U} \cdot \nabla_z^f. \tag{16}$$

Following the same procedure, we are able to obtain an explicit form of the third-order time derivative of the flux as,

$$\mathbf{F}_{ttt} = -\mathbf{F}_{\mathbf{UUU}} \cdot \nabla^f \cdot \nabla^f \cdot \nabla^f + 3\mathbf{F}_{\mathbf{UU}} \cdot \nabla^f \cdot \nabla_t^f - \mathbf{F}_\mathbf{U} \cdot \nabla_{tt}^f, \tag{17}$$

where

$$\begin{aligned}\nabla_{tt}^f = \;& \mathbf{F}_{\mathbf{UUU}} \cdot \nabla^f \cdot \mathbf{U}_x \cdot \nabla^f + 2\mathbf{F}_{\mathbf{UU}} \cdot \nabla^f \cdot \nabla_x^f - \mathbf{F}_{\mathbf{UU}} \cdot \mathbf{U}_x \cdot \nabla_t^f - \mathbf{F}_\mathbf{U} \cdot \nabla_{tx}^f \\ & + \mathbf{G}_{\mathbf{UUU}} \cdot \nabla^f \cdot \mathbf{U}_y \cdot \nabla^f + 2\mathbf{G}_{\mathbf{UU}} \cdot \nabla^f \cdot \nabla_y^f - \mathbf{G}_{\mathbf{UU}} \cdot \mathbf{U}_y \cdot \nabla_t^f - \mathbf{G}_\mathbf{U} \cdot \nabla_{ty}^f \\ & + \mathbf{H}_{\mathbf{UUU}} \cdot \nabla^f \cdot \mathbf{U}_z \cdot \nabla^f + 2\mathbf{H}_{\mathbf{UU}} \cdot \nabla^f \cdot \nabla_z^f - \mathbf{H}_{\mathbf{UU}} \cdot \mathbf{U}_z \cdot \nabla_t^f - \mathbf{H}_\mathbf{U} \cdot \nabla_{tz}^f,\end{aligned} \tag{18}$$

and

$$\begin{aligned}\nabla_{tx}^f = \;& \mathbf{F}_{\mathbf{UUU}} \cdot \mathbf{U}_x \cdot \nabla^f \cdot \mathbf{U}_x - \mathbf{F}_{\mathbf{UU}} \cdot \mathbf{U}_{xx} \cdot \nabla^f - 2\mathbf{F}_{\mathbf{UU}} \cdot \nabla_x^f \cdot \mathbf{U}_x - \mathbf{F}_\mathbf{U} \cdot \nabla_{xx}^f \\ & - \mathbf{G}_{\mathbf{UUU}} \cdot \mathbf{U}_x \cdot \nabla^f \cdot \mathbf{U}_y - \mathbf{G}_{\mathbf{UU}} \cdot \mathbf{U}_{xy} \cdot \nabla^f - \mathbf{G}_{\mathbf{UU}} \cdot \nabla_x^f \cdot \mathbf{U}_y - \mathbf{G}_{\mathbf{UU}} \cdot \mathbf{U}_x \cdot \nabla_y^f - \mathbf{G}_\mathbf{U} \cdot \nabla_{xy}^f \\ & - \mathbf{H}_{\mathbf{UUU}} \cdot \mathbf{U}_x \cdot \nabla^f \cdot \mathbf{U}_z - \mathbf{H}_{\mathbf{UU}} \cdot \mathbf{U}_{xz} \cdot \nabla^f - \mathbf{H}_{\mathbf{UU}} \cdot \nabla_x^f \cdot \mathbf{U}_z - \mathbf{H}_{\mathbf{UU}} \cdot \mathbf{U}_x \cdot \nabla_z^f - \mathbf{H}_\mathbf{U} \cdot \nabla_{xz}^f,\end{aligned} \tag{19}$$

and similarly for ∇_{ty}^f and ∇_{tz}^f. Collecting Eqs. (14) to (19), we can express the fourth-order approximation of the time-averaged flux $\mathbf{F}^{appx,4}$ explicitly, as the spatial derivatives are readily approximated through the conventional central differencing schemes.

We should note that we adopt the conventional five-points, fourth-order in space central differencing schemes to evaluate the spatial derivative terms, following the original PIF method in [38,31]. Moreover, for reducing the code complexity and improving the code performance, we reuse the divergence of the flux, ∇^f, for calculating high order spatial derivatives, e.g., ∇_x^f, ∇_{xx}^f, and ∇_{xy}^f. This approach requires an additional guard cell layer (resulting in two more guard cells for the five-points derivatives). However, the overall code performance is better than evaluating high-order derivatives in each direction. Also, we have observed that it does not affect the accuracy and the stability of the PIF scheme.

4. The original non-recursive system-free (SF) approach

As firstly proposed in [33], the system-free (SF) method provides a capability to bypass all the analytical derivations of *Jacobian-like* terms ($\mathbf{F}_\mathbf{U}, \mathbf{F}_{\mathbf{UU}}, \cdots$) in the PIF method by considering a central differencing with a small perturbation ε for the input space of the flux function. For example, a dot product between the Jacobian matrix $\mathbf{F}_\mathbf{U}$ and an arbitrary vector $\mathbf{V}$ can be approximated as,

$$\mathbf{F_U}\cdot\mathbf{V}=\frac{1}{2\varepsilon}\Big[\mathbf{F}(\mathbf{U}+\varepsilon\mathbf{V})-\mathbf{F}(\mathbf{U}-\varepsilon\mathbf{V})\Big]+\mathcal{O}(\varepsilon^2). \tag{20}$$

We can extend the above idea to the Hessian tensor contraction with the two same vectors $\mathbf{V}$,

$$\mathbf{F_{UU}}\cdot\mathbf{V}\cdot\mathbf{V}=\frac{1}{\varepsilon^2}\Big[\mathbf{F}(\mathbf{U}+\varepsilon\mathbf{V})-2\mathbf{F}(\mathbf{U})-\mathbf{F}(\mathbf{U}-\varepsilon\mathbf{V})\Big]+\mathcal{O}(\varepsilon^2). \tag{21}$$

The contraction with two different vectors of $\mathbf{V}$ and $\mathbf{W}$ is achieved by a linear combination of Eq. (21) by following the simple vector calculus,

$$\mathbf{F_{UU}}\cdot\mathbf{V}\cdot\mathbf{W}=\frac{1}{2}\Big[\mathbf{F_{UU}}\cdot(\mathbf{V}+\mathbf{W})\cdot(\mathbf{V}+\mathbf{W})-(\mathbf{F_{UU}}\cdot\mathbf{V}\cdot\mathbf{V}+\mathbf{F_{UU}}\cdot\mathbf{W}\cdot\mathbf{W})\Big]. \tag{22}$$

Theoretically speaking, the original system-free procedure in the above can be applied to any arbitrary order of derivatives of the flux function $\mathbf{F}$ with respect to the conservative variable $\mathbf{U}$. For instance, the fourth-order extension of the PIF method requires the third-order derivative of $\mathbf{F}$, i.e., $\mathbf{F_{UUU}}$. Following the same mathematical basis of Eq. (20) and Eq. (21), we obtain

$$\mathbf{F_{UUU}}\cdot\mathbf{V}\cdot\mathbf{V}\cdot\mathbf{V}=\frac{1}{2\varepsilon^3}\Big[-\mathbf{F}(\mathbf{U}-2\varepsilon\mathbf{V})+2\mathbf{F}(\mathbf{U}-\varepsilon\mathbf{V})-2\mathbf{F}(\mathbf{U}+\varepsilon\mathbf{V})+\mathbf{F}(\mathbf{U}+2\varepsilon\mathbf{V})\Big]+\mathcal{O}(\varepsilon^2). \tag{23}$$

We can further extend the procedure to compute the contraction with three different vectors, $\mathbf{V}, \mathbf{W}$, and $\mathbf{X}$,

$$\begin{aligned}\mathbf{F_{UUU}}\cdot\mathbf{V}\cdot\mathbf{W}\cdot\mathbf{X}=\frac{1}{6}\Big[&\mathbf{F_{UUU}}\cdot(\mathbf{V}+\mathbf{W}+\mathbf{X})\cdot(\mathbf{V}+\mathbf{W}+\mathbf{X})\cdot(\mathbf{V}+\mathbf{W}+\mathbf{X})\\&-\mathbf{F_{UUU}}\cdot(\mathbf{V}+\mathbf{W})\cdot(\mathbf{V}+\mathbf{W})\cdot(\mathbf{V}+\mathbf{W})\\&-\mathbf{F_{UUU}}\cdot(\mathbf{V}+\mathbf{X})\cdot(\mathbf{V}+\mathbf{X})\cdot(\mathbf{V}+\mathbf{X})\\&-\mathbf{F_{UUU}}\cdot(\mathbf{W}+\mathbf{X})\cdot(\mathbf{W}+\mathbf{X})\cdot(\mathbf{W}+\mathbf{X})\\&+\mathbf{F_{UUU}}\cdot\mathbf{V}\cdot\mathbf{V}\cdot\mathbf{V}+\mathbf{F_{UUU}}\cdot\mathbf{W}\cdot\mathbf{W}\cdot\mathbf{W}+\mathbf{F_{UUU}}\cdot\mathbf{X}\cdot\mathbf{X}\cdot\mathbf{X}\Big],\end{aligned} \tag{24}$$

and only to see that the number of terms to be computed rapidly increases in high-order tensor contraction terms.

As such, the original SF method in Eq. (24) becomes less attractive for any PIF method higher than third-order accuracy, as it demands increasing complexity in code implementation, which results in a significant loss in the overall performance of the code. For example, it requires 28 times flux function calls for just getting a single tensor contraction, $\mathbf{F_{UUU}}\cdot\mathbf{V}\cdot\mathbf{W}\cdot\mathbf{X}$. We should note that the major bottleneck of the original system-free method stems from Eqs. (22) and (24) that require to perform the *Jacobian-like* approximations multiple times.

To mitigation the computational bottleneck, in the following section, we introduce a newly improved version of the system-free method, which does not require any further modifications, even for the case of the tensor contractions of different vectors.

5. A newly improved recursive system-free (SF) approach

In this section, we will improve the original system-free method [33] to be applied in a recursive manner for higher-order derivatives of $\mathbf{F}$, ensuing much simpler code complexity and faster performance. Primarily, we define a functional $\mathcal{D}_u$ that represents the Jacobian-free method denoted in Eq. (20),

$$\mathbf{F_U}\cdot\mathbf{V}\approx\mathcal{D}_u(\mathbf{F};\,\mathbf{V}):=\frac{1}{2\varepsilon_v}\Big[\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V})-\mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V})\Big], \tag{25}$$

where ε_v is the appropriately calculated ε corresponding to the vector $\mathbf{V}$ by following the original idea of the system-free method in [33],

$$\varepsilon_v=\min\left(\Delta t,\ \bar{\varepsilon}_v\right),\ \text{where}\ \bar{\varepsilon}_v=\frac{\sqrt{\varepsilon^{op}}}{\|\mathbf{V}\|_2}. \tag{26}$$

The study in this paper uses $\varepsilon^{op}=4.8062\times10^{-6}$ that is the optimal ε value for the second-order Jacobian-free approximation in the 64-bit machine. This choice is also justifiable for the recursive scheme considered below, where the functional $\mathcal{D}_u$ itself is defined as the Jacobian-free method fundamentally. More detailed discussions about ε calculations could be found in [33,42] and [36].

We continue to apply $\mathcal{D}_u$ in the following successive fashion to calculate the tensor contractions between higher-order derivatives for the flux function $\mathbf{F}$ and arbitrary vectors. Thus, the Hessian approximation is,

$$\begin{aligned}\mathbf{F_{UU}}\cdot\mathbf{V}\cdot\mathbf{W} &\approx \mathcal{D}_u\Big(\mathcal{D}_u(\mathbf{F};\,\mathbf{V});\,\mathbf{W}\Big)\\ &= \frac{1}{4\varepsilon_v\varepsilon_w}\Big[\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V}+\varepsilon_w\mathbf{W})-\mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V}+\varepsilon_w\mathbf{W})-\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V}-\varepsilon_w\mathbf{W})+\mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V}-\varepsilon_w\mathbf{W})\Big].\end{aligned} \tag{27}$$

Again, following Eq. (26), ε_v and ε_w are the optimal ε values normalized by its corresponding vectors $\mathbf{V}$ and $\mathbf{W}$, respectively.

Note that the improved version of the Hessian-free method in Eq. (27) is applicable regardless the tensor contraction is with two identical vectors (e.g., $\mathbf{F_{UU}}\cdot\mathbf{V}\cdot\mathbf{V}$) or with two distinct vectors (e.g., $\mathbf{F_{UU}}\cdot\mathbf{V}\cdot\mathbf{W}$), hence it does not require separate formulations as in Eqs. (22) and (24).

The simplicity gain from the improved version of the system-free method is further rewarded when we apply it to the higher-order derivatives of $\mathbf{F}$. Following the equivalent strategy, the tensor contraction of the third-order derivative of the flux function, $\mathbf{F_{UUU}}$ with three distinct vectors, $\mathbf{V}, \mathbf{W}$, and $\mathbf{X}$ is written compactly as,

$$\begin{aligned}\mathbf{F_{UUU}}\cdot\mathbf{V}\cdot\mathbf{W}\cdot\mathbf{X} &\approx \mathcal{D}_u\left(\mathcal{D}_u\Big(\mathcal{D}_u(\mathbf{F};\,\mathbf{V});\,\mathbf{W}\Big);\,\mathbf{X}\right)\\ &= \frac{1}{8\varepsilon_v\varepsilon_w\varepsilon_x}\Big[\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V}+\varepsilon_w\mathbf{W}+\varepsilon_x\mathbf{X})-\mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V}+\varepsilon_w\mathbf{W}+\varepsilon_x\mathbf{X})\\ &\quad-\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V}-\varepsilon_w\mathbf{W}+\varepsilon_x\mathbf{X})+\mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V}-\varepsilon_w\mathbf{W}+\varepsilon_x\mathbf{X})\\ &\quad-\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V}+\varepsilon_w\mathbf{W}-\varepsilon_x\mathbf{X})+\mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V}+\varepsilon_w\mathbf{W}-\varepsilon_x\mathbf{X})\\ &\quad+\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V}-\varepsilon_w\mathbf{W}-\varepsilon_x\mathbf{X})-\mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V}-\varepsilon_w\mathbf{W}-\varepsilon_x\mathbf{X})\Big].\end{aligned} \tag{28}$$

Let us consider the number of flux function calls of SF approximations to compare the performance gain from the recursive SF method. Comparing Eq. (24) and (28), for example, the numerical flux function needed to be called 28 times in the original SF method. However, the recursive SF method only requires eight evaluations. This is a huge improvement in both performance and compactness.

With the modified SF method in Eqs. (25) to (28), we can approximate all the tensor contractions needed for calculating temporal flux derivatives in Eqs. (14) to (19) without analytical derivations for *Jacobian-like* terms, giving the system independence of the high-order scheme. We should note that the recursive modifications of the SF method presented in this section do not affect the solution's accuracy and stability compared to the original SF method.

6. Stepwise implementation of recursive SF-PIF method

In this section, we summarize the SF-PIF method proposed in this study in a stepwise fashion. We intend to provide those who want to implement SF-PIF in their codes with a bird-eye view of SF-PIF. Also provided in Appendix A are pseudo-codes describing Eqs. (25), (26), (27), and (28) with extra stepwise details. Below, we omit the discretization indices i, j, k and n for simplicity in representing the conservative variables $\mathbf{U}^n_{ijk}$ and the corresponding fluxes $\mathcal{F}^n_{ijk} = (\mathbf{F}^n_{ijk}, \mathbf{G}^n_{ijk}, \mathbf{H}^n_{ijk})$.

Step 1: Calculate $\nabla^f = \mathbf{F}_x + \mathbf{G}_y + \mathbf{H}_z$ via the standard fourth-order accurate, five-point central differencing scheme on every grid point and save them. These saved flux divergences will be used as inputs for the central differencing formulae in the following steps to get higher-order spatial derivatives.

Step 2: Apply the Jacobian approximation in Eq. (25) in preparation for $\mathbf{F}_t$ as expressed in Eq. (14), and construct the second-order temporally averaged flux $\mathbf{F}^{appx,2} = \mathbf{F} + \Delta t\mathbf{F}_t/2$. Apply the similar procedures to $y-$ and z-directional fluxes to obtain $\mathbf{G}^{appx,2}$ and $\mathbf{H}^{appx,2}$. This finalizes the second-order temporal approximations of pointwise fluxes in all directions.

Step 3: Given the pointwise conservative variables $\mathbf{U}$ and the divergence of fluxes ∇^f from **Step 1**, calculate $\mathbf{U}_x, \mathbf{U}_y, \mathbf{U}_z, \nabla^f_x, \nabla^f_y$, and ∇^f_z via the same five-point central differencing operator in **Step 1**. They will be used as building blocks for constructing $\mathbf{F}_{tt}, \mathbf{G}_{tt}$ and $\mathbf{H}_{tt}$ in the following steps.

Step 4: Apply the Jacobian and Hessian approximations in Eqs. (25) and (27) to the spatially approximated derivative quantities in **Step 3** in order to compute ∇^f_t by following the explicit expression in Eq. (16).

Step 5: Apply the Jacobian approximation in (25) to ∇^f_t from **Step 4** and the Hessian approximation in (27) to ∇^f from **Step 1** in order to get $\mathbf{F}_{tt}$ using Eq. (15); add the computed $\mathbf{F}_{tt}$ to the results of **Step 2** to update the second-order temporal fluxes in **Step 2** to the third-order temporally averaged flux, $\mathbf{F}^{appx,3} = \mathbf{F}^{appx,2} + \Delta t^2\mathbf{F}_{tt}/6$. Perform the similar procedures in $y-$ and z-directions to obtain $\mathbf{G}^{appx,3}$ and $\mathbf{H}^{appx,3}$. This finalizes the third-order temporal approximations of pointwise fluxes in all directions.

Step 6: Using the five-point central differencing, compute the fourth-order accurate approximations of the second derivatives and the mixed-derivatives of the conservative variables and the divergence of fluxes to obtain $\mathbf{U}_{xx}, \mathbf{U}_{xy}, \mathbf{U}_{yy}, \ldots$ etc. and $\nabla^f_{xx}, \nabla^f_{xy}, \nabla^f_{yy}, \ldots$ etc.

Step 7: Apply the tensor contractions of the first, second, and third-order flux derivatives in Eqs. (25), (27), and (28) to the quantities computed and stored from the previous steps in order to calculate ∇^f_{tt} and ∇^f_{tx} by following the explicit relations in Eqs. (18) and (19) respectively. Also calculate ∇^f_{ty} and ∇^f_{tz} similarly.
Step 8: Next, perform the last set of tensor contractions in Eqs. (25), (27), and (28) to construct $\mathbf{F}_{ttt}$ as expressed in Eq. (17). Add the resulting $\mathbf{F}_{ttt}$ to the result of **Step 5** to obtain the fourth-order temporally averaged flux, $\mathbf{F}^{appx,4} = \mathbf{F}^{appx,3} + \Delta t^3 \mathbf{F}_{ttt}/24$. Perform the similar procedures in $y-$ and z-directions to obtain $\mathbf{G}^{appx,4}$ and $\mathbf{H}^{appx,4}$. This finalizes the fourth-order temporal approximations of pointwise fluxes in all directions.
Step 9: Proceed with the conventional FDM procedures for high-order spatial accuracy, viz., apply a high-order reconstruction method with a characteristic flux-splitting strategy in Eq. (9) to the results, $\mathbf{F}^{appx,4}$, $\mathbf{G}^{appx,4}$, and $\mathbf{H}^{appx,4}$, from **Step 8**. For example, taking WENO-JS as a reconstruction method using $\mathbf{F}^{appx,4}$ ensures a temporally fourth-order and spatially fifth-order accurate approximation to the numerical flux, $\hat{\mathbf{f}} = \texttt{WENO-JS}\left(\mathbf{F}^{appx,4}\right) + \mathcal{O}(\Delta x^5, \Delta t^4)$. Perform the similar procedures in $y-$ and z-directions to obtain $\hat{\mathbf{g}}$ and $\hat{\mathbf{h}}$.
Step 10: Lastly, update the solution following Eq. (10).

7. Numerical test results

This section presents numerical results of well-known test problems for benchmarking the modified SF-PIF methods' capabilities. We mainly compare the results from the third-order and the fourth-order temporal methods of SSP-RK and SF-PIF. We used the well-known three-stages, third-order SSP-RK method [43] and the five-stages, fourth-order SSP-RK method [44] (RK3 and RK4 hereafter) for the comparisons of the third-order and fourth-order SF-PIF methods (SF-PIF3 and SF-PIF4 hereafter), respectively.

The main purpose of this section is to demonstrate that the proposed recursive SF-PIF methods produce the same quality of solutions, with less CPU time in the light of the single-stage time update, as compared to RK3 and RK4 methods applied to FDM discretizations. We use the conventional fifth-order WENO-JS [5] spatial reconstruction method for all simulations; thus, we expect fifth-order spatial accuracy $\mathcal{O}(\Delta x^5)$ combined with third $\mathcal{O}(\Delta t^3)$ or fourth-order $\mathcal{O}(\Delta t^4)$ temporal accuracy for all numerical results. The fixed Courant numbers, $C_{\text{cfl}} = 0.4$ and $C_{\text{cfl}} = 0.3$, are used for all 2D and 3D simulations, respectively.

The SF-PIF source code described and used for producing the results in this paper is available at

https://github.com/ylee88/SlugCode2.

The source code and the contents available in GitHub are licensed under the MIT License.

7.1. 2D Euler equations

7.1.1. The Sod's shock tube test (rotated 45°)

We start with the classical shock tube problem of Sod [45], testing a numerical scheme's ability to capture the three wave families of the 1D Riemann problem: a shock, contact discontinuity, and rarefaction fan. Although Sod's problem is a 1D shock tube problem originally, we implement the test in a 2D domain by tilting the shock wave direction by the angle of $\theta = 45°$. We adopt the idea of Kawai [46], where the initial conditions are repeated multiple times along the direction of the wave propagation so that the problem may be executed with periodic boundary conditions. Explicitly, the initial condition is given as,

$$(\rho, u, v, p) = \begin{cases} (1, 0, 0, 1) & \text{for } x_\parallel \le 0.5, \quad 1.5 < x_\parallel \le 2.5, \quad 3.5 < x_\parallel \le 4, \\ (0.125, 0, 0, 0.1) & \text{for } 0.5 < x_\parallel \le 1.5, \quad 2.5 < x_\parallel \le 3.5, \end{cases} \tag{29}$$

where $x_\parallel = x\cos\theta + y\sin\theta$ is the direction parallel to the wave propagation. The simulation domain is a periodic box of $[0, 2/\cos\theta] \times [0, 2/\sin\theta]$. For the final result profiles, we take only the bottom-left quarter of the diagonal axis, $0 \le x_\parallel \le 1$; thus, the number of data points of the result profiles would be a quarter of the grid resolution in x and y, $N_\parallel = N_x/4 = N_y/4$.

The 45°-angled Sod's shock tube test results at $t = 0.2$ are depicted in Fig. 1. We used the grid resolution of $N_x = N_y = 1024$; thus, the figure shows an $N_\parallel = 256$ number of data points along with the diagonal axis, $x_\parallel$. As shown in Fig. 1, SF-PIF methods produce fairly comparable results to those of RK3 and RK4, except for the small oscillation in the shock front of x-velocity. This issue was already discussed in our previous study [33], where the oscillations are originated from the central differencing formulae that we used for getting spatial derivatives. We can minimize the oscillation by applying WENO-*like* differencing introduced in the appendix of [33]. However, we choose to use the conventional central differencing formulae for this study as we did not observe any unphysical oscillations for the rest of the test problems.

7.1.2. The Shu-Osher problem (rotated 45°)

Our next choice of test problem is the Shu-Osher problem [47] that describes the interactions between a Mach 3 shock and a smooth density profile. Initially, a Mach 3 shock wave travels to the right through a sinusoidally perturbed density

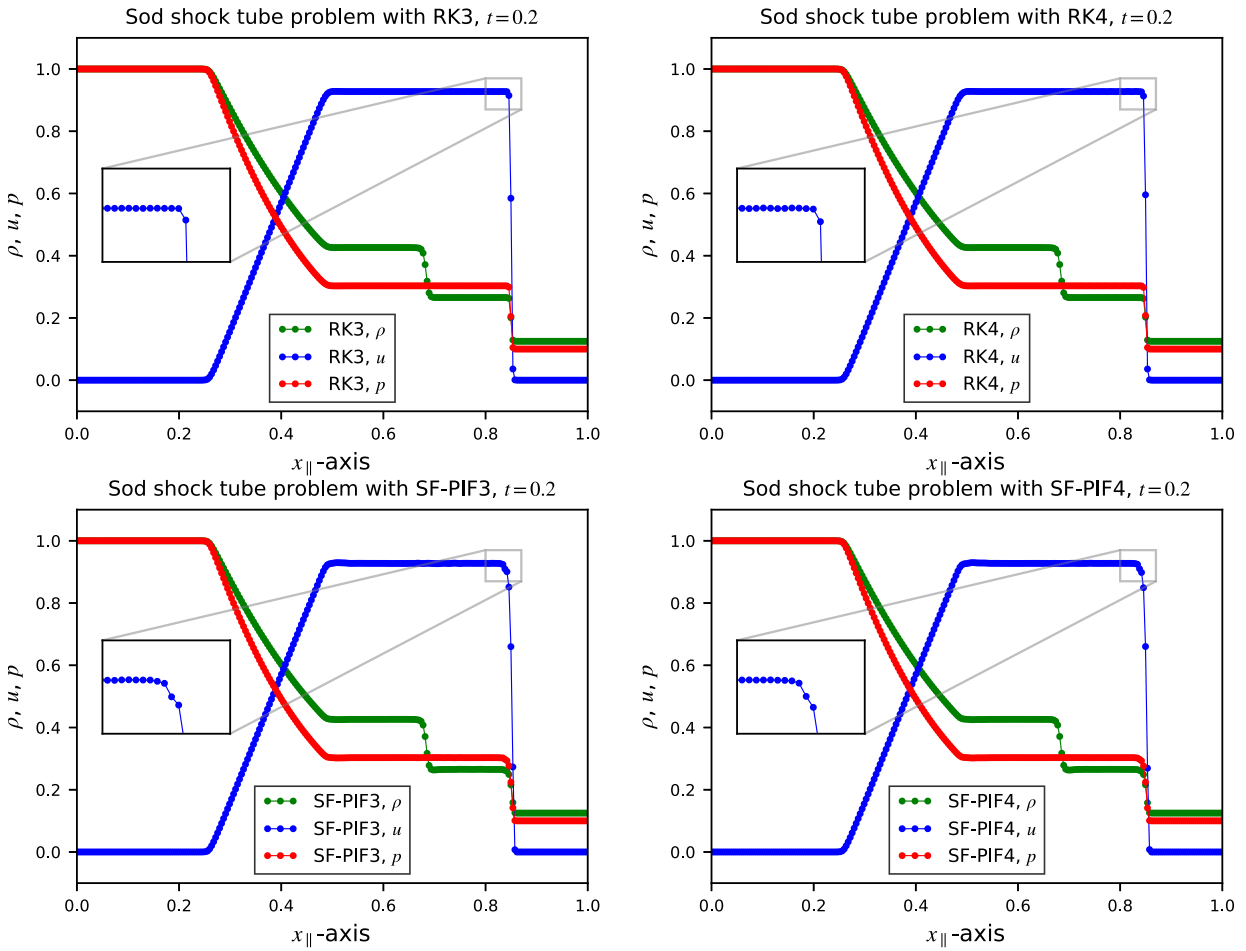

Fig. 1. One-dimensional profiles of the inclined Sod's shock tube problem along the $x_\parallel$ direction, displaying the fluid density ρ, the x-velocity u, and the thermal pressure p, illustrated at $t = 0.2$. All simulations performed in a 2D simulation box of $[1024 \times 1024]$ grid resolution, but the number of data points on each panel is $N_\parallel$ = 256, as we only take the first quarter of the diagonal axis for the shown profiles. (**left column**) The profiles solved by RK3 (top) and SF-PIF3 (bottom). (**right column**) The profiles solved by RK4 (top) and SF-PIF4 (bottom).

profile. As the shock wave propagates along the perturbed region, the profile gets compressed, resulting in a frequency-doubled region behind the shock. As the shock wave moves further to the right, the doubled-frequency region returns to its original frequency, at which point it becomes a sequence of sharp profile instead of smooth sine wave due to the shock-steepening.

We performed the Shu-Osher problem in 2D by inclining the shock wave direction by an angle of $\theta = 45°$, similar to what we did in the Section 7.1.1. We use periodic boundary conditions in both directions of the simulation domain $[0, 20/\cos\theta] \times [0, 20/\sin\theta]$. The grid resolution is $N_x = N_y = 1024$; thus, the effective resolution $N_\parallel$ is 256 as we take only the bottom-left quarter of $x_\parallel$ to report the 1D profiles.

The density profiles along the $x_\parallel$ direction are given in Fig. 2. The four different temporal method choices (RK3, RK4, SF-PIF3, and SF-PIF4) produce reasonably acceptable solution profiles capturing the high-frequency amplitudes fairly well in the frequency-doubled region. We see that the results of RK3 and RK4 are nearly identical, while SF-PIF3 and SF-PIF4 show slight differences near the highest amplitudes. Generally, RK methods capture the amplitudes marginally better, but in the left-most part of the double-frequency region, $x \approx 5.8$, SF-PIF methods capture the highest peak of the amplitude better than the RK methods near the transition between the frequency doubling and the shock steepened perturbations.

7.1.3. Isentropic vortex advection

The isentropic vortex advection problem [48] is one of the most popular test choices to measure the simulation code's accuracy and performance. Although the problem is fully nonlinear, the exact solution is always existent in the form of its initial condition, from which an isentropic vortex is advected through periodic boundaries. We can evaluate the accuracy of a method on nonlinear problems by comparing the final result with its initial condition. Here, we adopt the idea from Spiegel [49], where the simulation domain size is doubled up as $[0, 20] \times [0, 20]$ to prevent vortex-vortex couplings near the periodic boundaries.

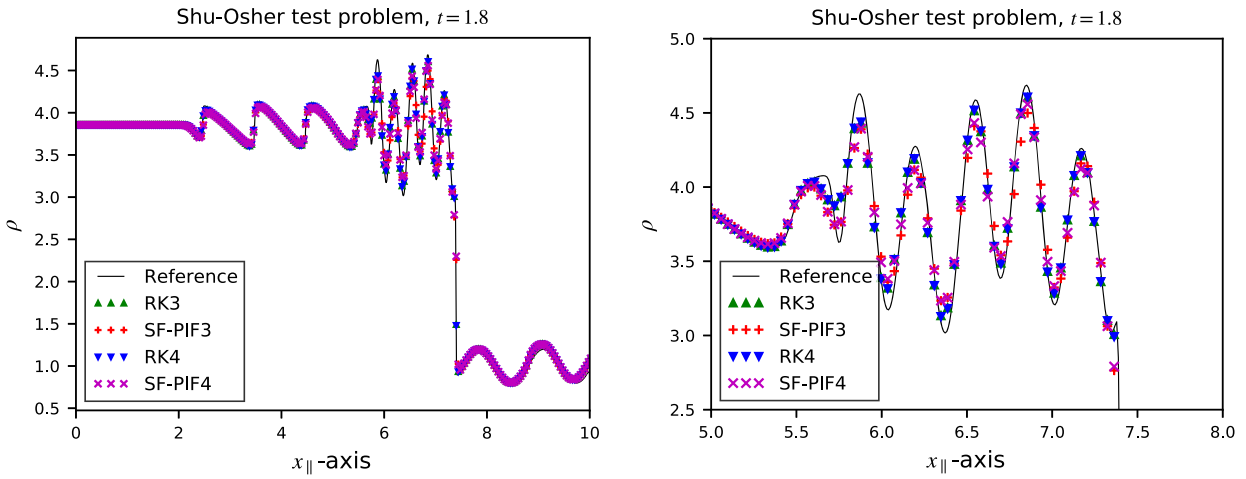

Fig. 2. One dimensional density profiles along the $x_\parallel$ direction of the inclined Shu-Osher problem at $t = 1.8$. The solid line represents the reference solution, solved by RK4 with 1024 data points in the diagonal axis, i.e., $N_\parallel = 1024$. All other solutions, represented by the symbols, are resolved on an $N_\parallel = 256$ grid resolution in the diagonal axis. The detailed view of the high-frequency region is shown on the right panel.

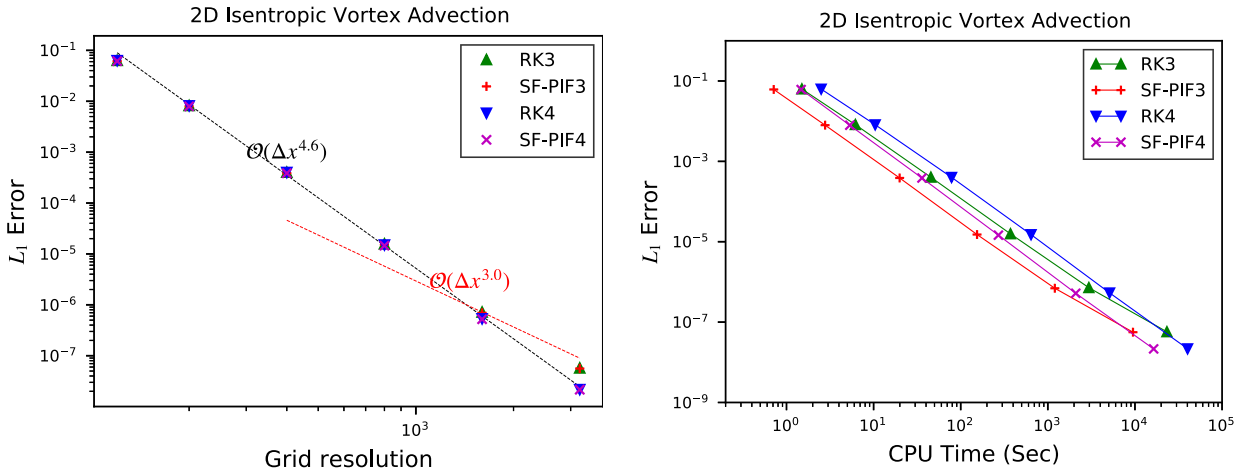

Fig. 3. The L_1 errors of the isentropic vortex advection test problem with respect to the grid resolutions (**left**); with respect to the computation time (**right**).

The results of L_1 errors on six different grid resolutions, $N_x = N_y = 120, 200, 400, 800, 1600$, and 3200, are plotted in the left panel of Fig. 3. As expected, all temporal methods follow the convergence line of order $\sim \mathcal{O}(\Delta x^{4.6})$, which is nearly the same as WENO's fifth-order spatial accuracy. However, at the critical grid resolution, $N_x = N_y = 1600$, the third-order temporal methods of RK3 and SF-PIF3 start to degrade the whole solution accuracy. This behavior can be explained that the errors from the fifth-order spatial WENO solver are dominant on the grid resolutions up to $N_x = N_y = 1600$, after which the truncation errors associated with the third-order temporal integrators become dominant over the error of the fifth-order spatial solver. This result tells us the significance of high-order temporal updates in fine grid resolutions: a high-order spatial method does require a *comparably* high-order temporal method to retain the overall quality of the solutions, particularly when we are motivated to add more grid resolutions to resolve finer scales more accurately. Otherwise, the temporal solver's lower order accuracy can potentially degrade the solution accuracy, contradicting the intended motivation. A direct consequence of this observation applies to CFD simulations on an adaptive mesh refinement (AMR) configuration. For example, consider a mediocre low-order temporal method is integrated with a high-order spatial solver. This combination of such two solvers creates a computational dilemma of not making any further enhancement in solution accuracy as the computational grids are progressively refined to improve the quality of the AMR solutions. Instead, the solution accuracy is to be bounded by the lower temporal accuracy.

To compare the computational performance over the four different temporal integrators, we present L_1 errors as a function of CPU time data on the right panel of Fig. 3. The drop of the convergence rates is again observed in the two third-order solutions of RK3 and SF-PIF3 at the critical resolution $N_x = N_y = 1600$. On the contrary, the two fourth-order solutions of RK4 and SF-PIF4 continue at fourth-order without any change. Up to this resolution, SF-PIF3 is the fastest in reaching any given target L_1 error threshold in CPU time. However, on any grid resolutions finer than the critical resolution, SF-PIF3's L_1

Table 1
The L_1 errors, the rates of convergence, and the computation times for the vortex advection test solved using RK3 and SF-PIF3 methods (**top**); using RK4 and SF-PIF4 methods (**bottom**). All simulation runs are performed on the four 20-cores Cascade Lake Intel Xeon processors, utilized 64 parallel threads. CPU times are measured in seconds, averaged over 10 individual runs.

$N_x = N_y$	RK3				SF-PIF3			
	L_1 error	L_1 order	CPU Time	Speedup	L_1 error	L_1 order	CPU Time	Speedup
120	6.31×10^{-2}	–	1.50 s	1.0	6.16×10^{-2}	–	0.71 s	0.48
200	8.20×10^{-3}	4.00	6.17 s	1.0	7.96×10^{-3}	4.00	2.77 s	0.45
400	4.02×10^{-4}	4.35	45.44 s	1.0	3.86×10^{-4}	4.37	19.89 s	0.44
800	1.57×10^{-5}	4.68	372.47 s	1.0	1.51×10^{-5}	4.68	153.92 s	0.41
1600	7.18×10^{-7}	4.45	2957.26 s	1.0	6.95×10^{-7}	4.44	1203.10 s	0.41
3200	5.72×10^{-8}	3.65	23274.37 s	1.0	5.60×10^{-8}	3.63	9494.65 s	0.41
$N_x = N_y$	RK4				SF-PIF4			
	L_1 error	L_1 order	CPU Time	Speedup	L_1 error	L_1 order	CPU Time	Speedup
120	6.30×10^{-2}	–	2.50 s	1.0	6.14×10^{-2}	–	1.47 s	0.59
200	8.15×10^{-3}	4.00	10.42 s	1.0	7.91×10^{-3}	4.01	5.33 s	0.51
400	4.01×10^{-4}	4.35	78.47 s	1.0	3.85×10^{-4}	4.36	35.89 s	0.46
800	1.51×10^{-5}	4.73	641.50 s	1.0	1.46×10^{-5}	4.72	270.94 s	0.42
1600	5.33×10^{-7}	4.82	5115.47 s	1.0	5.21×10^{-7}	4.81	2091.20 s	0.41
3200	2.17×10^{-8}	4.62	40195.034 s	1.0	2.15×10^{-8}	4.60	16377.73 s	0.41

error drops to the third-order convergence, which ultimately crosses the SF-PIF4's fourth-order straight convergence line at $N_x = N_y = 10^4$, beyond which its error will remain larger than the ones from the fourth-order temporal schemes as long as the convergence rate follows the pattern at the high-resolution tail.

On the other hand, it is distinctively superior to see that SF-PIF4's solution reaches any fixed target error in a *faster* CPU time than the *third-order* RK3's solution while keeping the numerical errors as low as RK4 results at each grid resolution. The detailed simulation data from the vortex advection tests are presented in Table 1. All performance results are measured in second, averaged over 10 simulation runs conducted on two nodes of the UC Santa Cruz's high-performance computer, *lux*. Each node has two 20-core Cascade Lake Intel Xeon processors, and we utilized 64 parallel threads for each simulation run.

7.1.4. 2D Riemann problem: Configuration 3

To test the methods' ability to capture the complex fluid structures in 2D, we performed a two-dimensional Riemann problem called Configuration 3. This specific setup and other types of configurations have been extensively studied in [50–52] and have been adopted as popular benchmark test problems. To set up Configuration 3, we follow the initial condition from [53,32,33]. We conducted numerical experiments on a 1600×1600 grid resolution, which is generally considered as a very high resolution in 2D. This grid resolution choice is made based on our observation in Section 7.1.3 to make sure that the temporal errors are comparable to or dominant over the spatial errors; thereby, we can anticipate temporal error dominant results that allow us to focus on the effects of the four different temporal solvers.

The results at $t = 0.8$ are shown in Fig. 4. The pseudo-colors represent the density map ranging between $[0.1, 1.8]$, and 40 contour lines within the same range are over-plotted as solid black lines. We see that all four different temporal schemes produce well-known, acceptable results, keeping the assumed diagonal symmetry exceptionally well on this high resolution. This problem is highly nonlinear, involving formations of the upward-moving jet, the downward-moving mushroom-jet, secondary Kelvin-Helmholtz instabilities exhibited as the small-scale vortical rollups along the slip lines and along the stems of the two jets. As such, it is a non-trivial task to address if a method under consideration is *better* or *worse* based on the number of such rollups in the simulations (see our discussion in [33]). At best, such quantification can only provide proof of intrinsic information about the amount of numerical dissipation of each method. From this perspective, we conclude that the two SF-PIF solutions produce the equivalent amount of vortical rollups compared with the corresponding RK solutions, confirming the SF-PIF methods' validity.

7.1.5. Double Mach reflection

Our final 2D test case is the double Mach reflection test that launches a strong Mach 10 shock with 60° of an incident angle between the shock plane and the bottom reflecting wedge. The initial condition is the same as the original setup introduced by Woodward and Colella [2], except that we doubled the y-domain size following [54] to prevent numerical artifacts from the top boundary interaction with the secondary shock wave and the slip line.

The density results are presented in Fig. 5. The density color map ranges between $[1.3, 22.6]$, and the solid black lines represent 40 levels of the density contour lines with the same range. The figures are zoomed-in near the vicinity of the jet and the primary triple point, which is widely considered the main area of interest in the Double Mach Reflection test. All simulation runs are performed on a 4096×2048 grid resolution.

The results from the third- and fourth-order SF-PIF methods produce well-acceptable results compared to the corresponding RK methods. Except that there are minor differences in the shape of Kelvin-Helmholtz instabilities along the

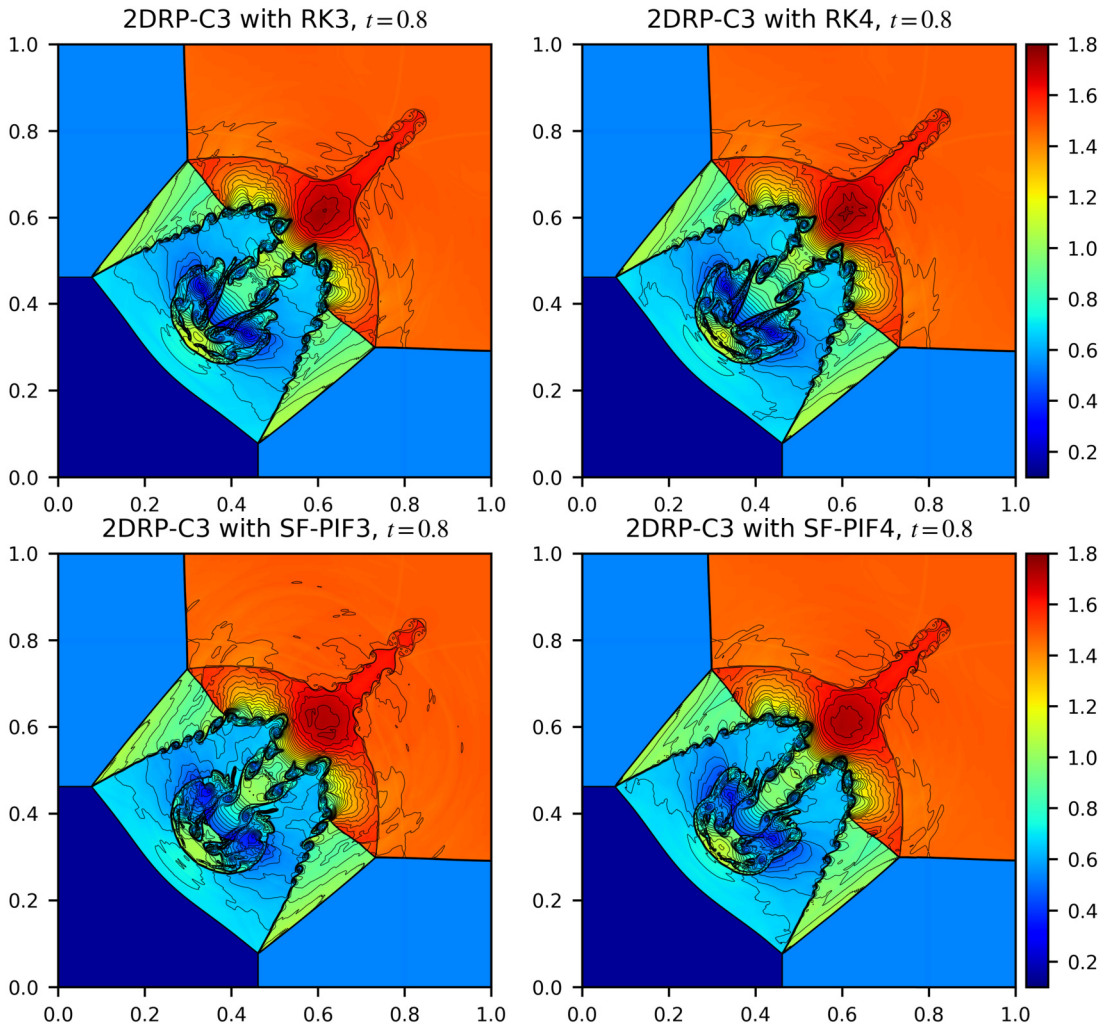

Fig. 4. The density maps of Configuration 3 at $t = 0.8$. (**left column**) The solutions using RK3 (top) and SF-PIF3 (bottom). (**right column**) The solutions using RK4 (top) and SF-PIF4 (bottom). Forty levels of black contour lines are over-plotted in each figure with the same range of the color map. All simulations are performed on a 1600×1600 grid resolution.

primary slip line and the bottom jet, the overall dynamics of the two SF-PIF solutions match well with the RK solutions, validating the fidelity of the proposed SF-PIF methods in the presence of a strong shock.

7.2. 3D Euler equations

7.2.1. 3D Sedov test

To test the code's ability to maintain the spherical symmetry in all spatial directions, we consider the Sedov blast test [55] in 3D. Initially, a point-source of a highly pressurized perturbation is given at the domain center, which leads to a strong spherical shock wave propagating outward from the source. The simulation domain is a 3D square box of $[-1.2, 1.2] \times [-1.2, 1.2] \times [-1.2, 1.2]$ resolved on a $128 \times 128 \times 128$ grid resolution. Outflow boundary conditions are imposed in all directions. The initial conditions are based on the setup found in [56], but we choose the deposited energy, $E_{\text{tot}} = 0.851072$, according to the setup in [57].

Fig. 6 shows the density profiles at $t = 1$ along the diagonal ($x = y = z$) and the x-axis ($y = z = 0$). The results on the left panel are solved with the RK4 method and the right panel with SF-PIF4. The exact self-similar solution is plotted as a reference solution in solid black curves on both figures. As shown, both the RK4 and SF-PIF4 methods show nearly identical results. Despite the visible differences between the diagonal and the x-axis profiles, especially at the highest peak and the shock front location, both fourth-order temporal solvers show well-maintained reflectional symmetry along the normal axis at the origin.

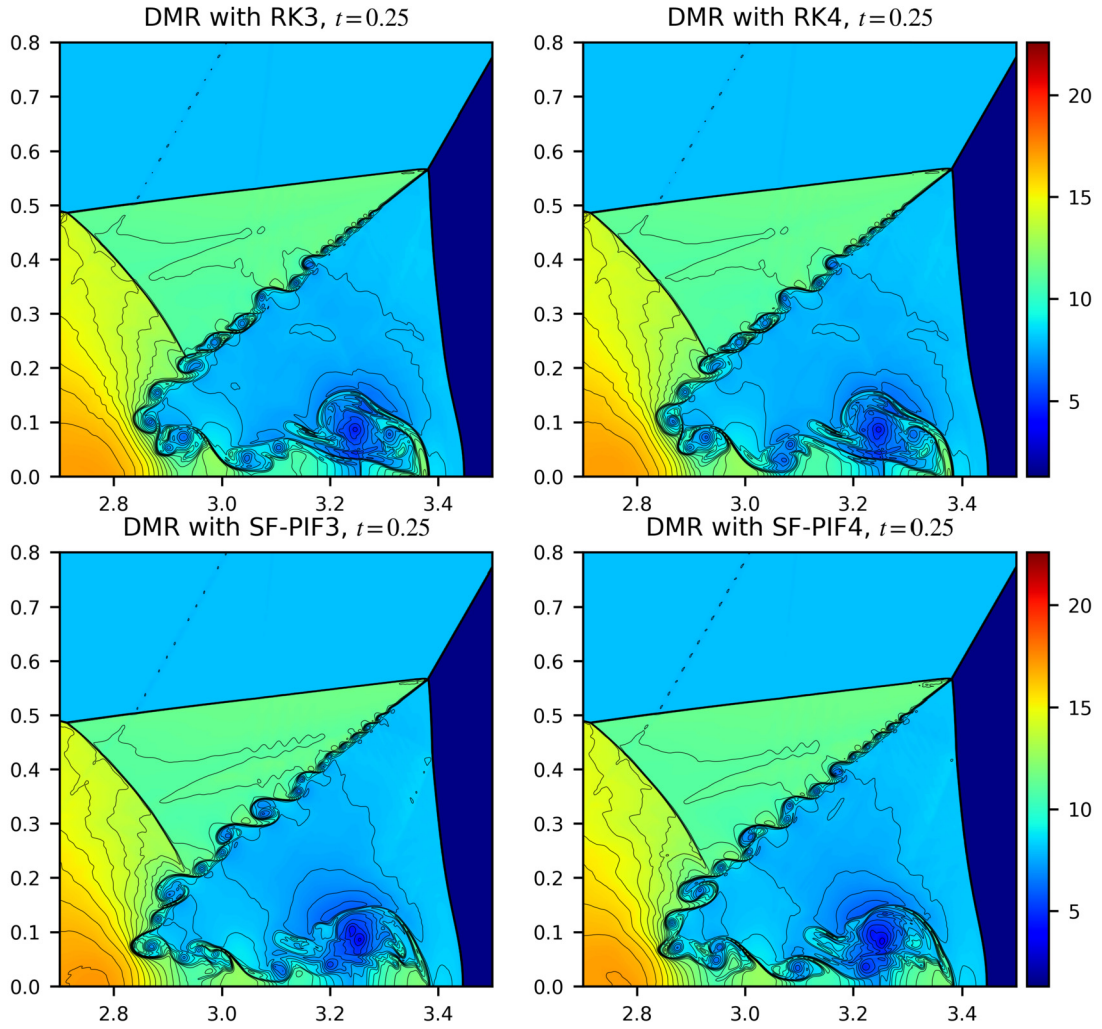

Fig. 5. The density map of the double Mach reflection test at $t = 0.25$ zoomed-in near the jet. Forty levels of contour lines are over-plotted in solid black curves with the same range of the color map. All simulation results are performed on a 4096×2048 grid resolution. (**left column**) The solutions using RK3 (top) and SF-PIF3 (bottom). (**right column**) The solutions using RK4 (top) and SF-PIF4 (bottom).

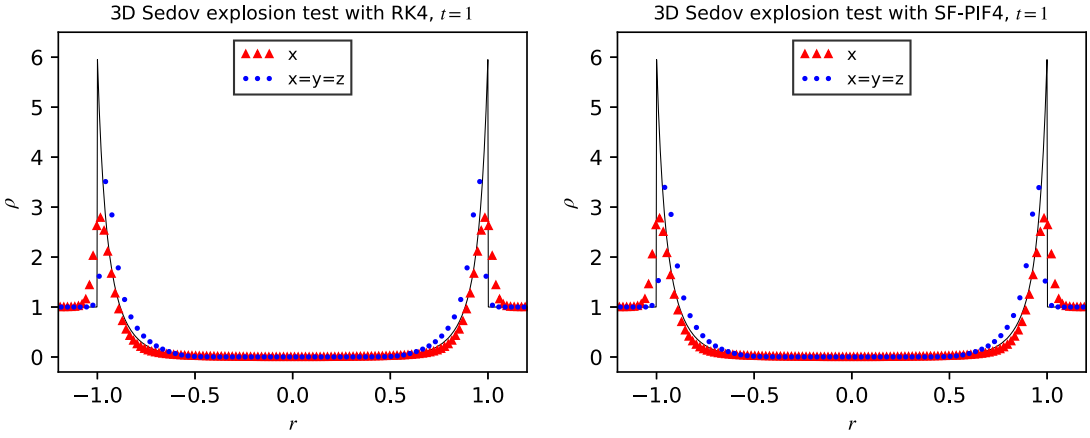

Fig. 6. The density profiles of the 3D Sedov explosion test at $t = 1$. The red triangles represent the density profiles along the x-axis from the origin, while the blue circles represent the density profiles along the diagonal axis. The solid black line is the well-known exact self-similar solution. All simulations are performed on a $128 \times 128 \times 128$ grid resolution, solved with RK4 (**left**) and SF-PIF4 (**right**) temporal solvers.

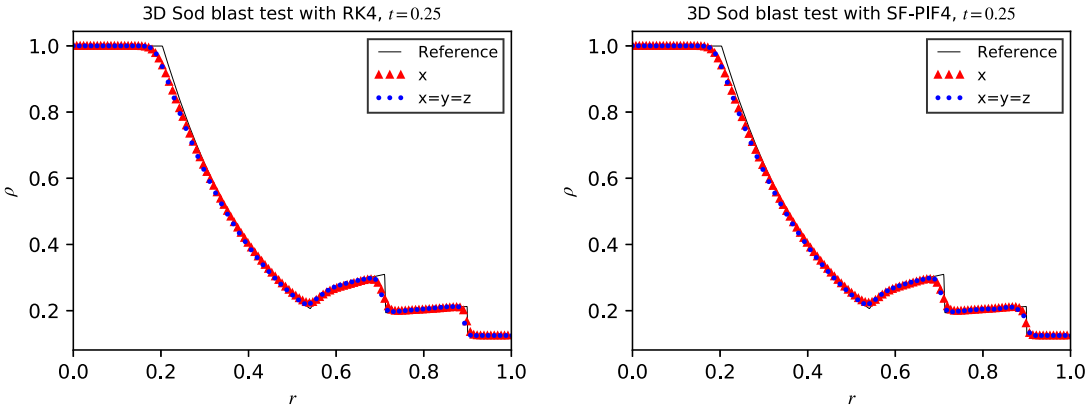

Fig. 7. The density profiles of the 3D Sod's blast test at $t = 0.25$. The red triangles represent the density profiles along the x-axis from the origin, and the blue circles represent the density profiles along the diagonal axis. The solid black curve is the reference solution calculated with the 1D Euler equations with the geometric source term. Simulations are performed on a $256 \times 256 \times 256$ grid resolution, solved with RK4 (**left**) and SF-PIF4 (**right**) temporal solvers.

Table 2
Performance results for the 3DRP test problem. All performance results (measured in seconds) are averaged over five simulation runs conducted on 16 nodes of *lux* cluster. Each node has 2 × 20-core Intel Xeon Gold 6248 (Cascade Lake) CPUs, and we utilized 512 parallel threads for each run.

Grid Resolution	RK3		SF-PIF3		RK4		SF-PIF4	
	CPU Time	Speedup	CPU Time	Speedup	CPU Time	Speedup	CPU Time	Speedup
64×64	1.79 s	1.0	1.09 s	0.61	2.95 s	1.64	2.67 s	1.49
128×128	15.62 s	1.0	7.63 s	0.49	25.88 s	1.66	18.97 s	1.21
256×256	191.00 s	1.0	82.02 s	0.43	321.34 s	1.68	201.40 s	1.05
512×512	2679.85 s	1.0	1173.54 s	0.44	4507.88 s	1.68	2817.78 s	1.05

7.2.2. 3D Sod's blast explosion test

Next, we consider the 3D explosion test problem found in [57]. This test is a three-dimensional extension of the 1D Sod's shock tube problem [45], which we already solved in a rotated 2D shock-tube configuration in Section 7.1.1. The calculations are performed on a $[-1, 1] \times [-1, 1] \times [-1, 1]$ domain with outflow boundary conditions using a $256 \times 256 \times 256$ grid resolution.

The results of the density profiles along the diagonal ($x = y = z$) and x-axis ($y = z = 0$) at $t = 0.25$ are presented in Fig. 7. The solid black curve represents the reference solution using the 1D Euler equations with the appropriate geometric source term according to [57]. The results show that the SF-PIF4 solver captures the shock profile and the spherical symmetry exceedingly well compared to the RK4 result and the reference 1D solution. There is no noticeable difference between the two fourth-order temporal solvers.

7.2.3. 3D Riemann problem

Finally, we performed the 3D Riemann problem presented in [58]. We follow the same initial conditions, consisting of eight constant initial conditions in each octant of the computational domain, $[-1, 1] \times [-1, 1] \times [-1, 1]$, resolved on a $256 \times 256 \times 256$ grid resolution. The setup imposes outflow boundary conditions at all boundaries. The initial condition will carry out 2D Riemann problems at each octant interface, including the diagonal plane of the 3D computational cubic.

The resulting density profiles at $t = 0.53$ are given in Fig. 8. The pseudo-color map ranges between $[0.5, 2.65]$, and we over-plot 40 levels of contour lines using the same range. We observe three different 2D Riemann problems on the left, top, and the diagonal planes in the left panel. The diagonal planes are separately shown in the right panel. SF-PIF4 method is able to capture all the important features as much as the RK4 result, confirming the validity of the SF-PIF4 method in comparison.

Table 2 shows the performance results for the 3D Riemann problem test on four grid resolutions. As shown in the table, the SF-PIF4 method demonstrates nearly the same performance as the *third-order* RK method, especially in high-resolution cases. We should note that the performance gains from the SF-PIF methods are more compensated on the high-resolution grids, which are indispensable for high fidelity physical simulation studies.

8. Conclusion

In this study, we have extended the original third-order (non-recursive) SF-PIF method [33] to a fourth-order temporal scheme with the improved recursive version of the system-free (SF) approach. The newly proposed recursive SF approach

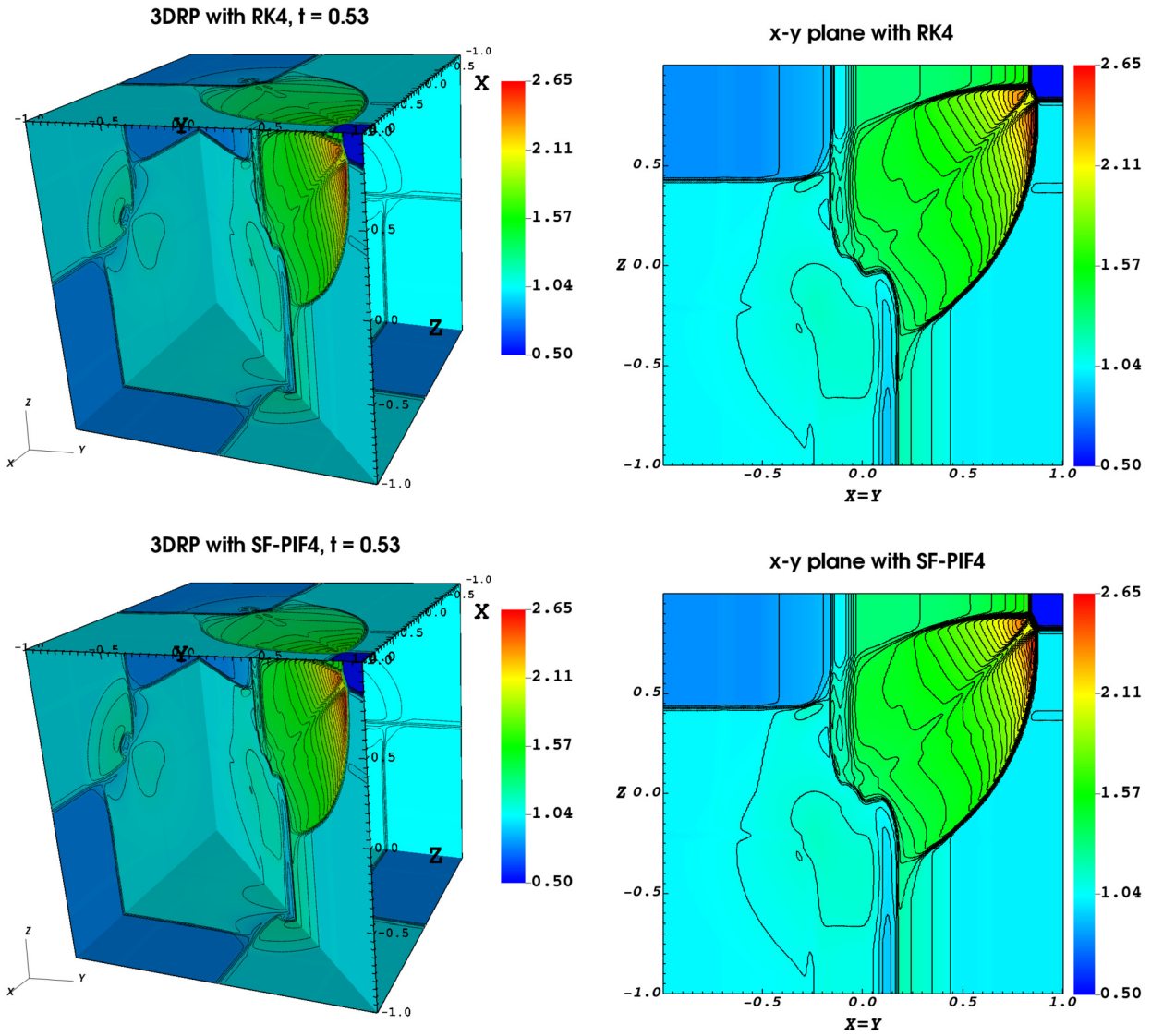

Fig. 8. The density maps of the 3D Riemann problem test at $t = 0.53$. Forty contour lines are over-plotted. The left panels show each face's geometrical views, while the right panels show the detailed picture of the diagonal planes. All simulations are performed on a $256 \times 256 \times 256$ grid resolution, solved with RK4 (**top**) and SF-PIF4 (**bottom**) solvers.

enables the fourth-order extension of the SF-PIF method (SF-PIF4), increasing computational performance gain with a simplistic code structure.

The original SF approach's critical design purpose [33] is to bypass all the analytical derivations of Jacobians, Hessians, and even higher-order derivatives tensor terms. With the SF approach, approximating the tensor contractions of *Jacobian-like* terms in the Lax-Wendroff type time discretization becomes simplified.

In this paper, the SF method's advantage is further enhanced by introducing a new recursive procedure. The recursive SF method requires only a small amount of calculations for approximating the tensor contractions, thereby empowering the fourth-order extension of the SF-PIF method to ease and faster performance.

We have tested our fourth-order and third-order SF-PIF methods with a wide range of test problems in two and three spatial dimensions. The results show the SF-PIF methods have significantly faster computational time performance than the corresponding SSP-RK methods at the same temporal order while maintaining the equivalent solution accuracy. In two-dimensional cases, SF-PIF methods show more than two times faster performance results than the SSP-RK counterparts. Moreover, we demonstrate that the fourth-order SF-PIF's performance results are nearly the same as the *third-order* SSP-RK method. This enhanced performance gain in our SF-PIF solvers can provide a big leap in large-scale parallel computing to save computational costs and reach highly accurate numerical predictions in practice.

We believe the recursive SF method has a broad potential to be relevant in various fields of study. It is neither designed particularly for the PIF method nor any specific numerical methods in computational fluid dynamics. Instead, it is solely intended for approximating the *Jacobian-like* tensor contractions. We presume that our recursive SF schemes are applicable in other numerical algorithms to enhance the calculation speed and ease the code implementations wherever the Jacobians, Hessians, or higher derivative tensors are required.

A further extension is to design the SF-PIF method in arbitrary order. As reported in Section 7.1.3, the temporal errors dominate the spatial errors in high-resolution simulations; thus, it is noteworthy to use the same accuracy in both the spatial and temporal solvers. We realize that a naive extension of the SF-PIF method to the fifth or higher-order could potentially be less attractive due to the drastic increase of complexity in Taylor expansion. We aim to reduce such complexity and make the SF-PIF method an arbitrary order of accuracy, which will be further investigated in our future studies.

CRediT authorship contribution statement

Youngjun Lee: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Dongwook Lee:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Writing – original draft, Writing – review & editing. **Adam Reyes:** Conceptualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We acknowledge the use of the lux supercomputer at UC Santa Cruz, funded by NSF MRI grant AST 1828315. We also thank the anonymous referee for constructive comments and suggestions that have improved the manuscript during the review process.

Appendix A. Pseudocodes for recursive system-free approach

We present four pseudocodes of functions, each of which implements Eqs. (25), (26), (27), and (28), respectively. We assume that the `ConsToFlux` function is defined externally, which takes conservative variables as inputs and return the corresponding flux functions, which vary with the system of equations. For instance, in the case with the Euler's equations, `ConsToFlux` returns the flux functions as prescribed in Eq. (2).

Algorithm 1: A pseudocode for Eq. (26).

Function `GetEpsilon(V)`:
  Input : **V** is an arbitrary vector with the same dimension of **U**
  Output: ε_v is an optimal ε value normalized with **V**
  return $\min\left(\Delta t, \dfrac{\sqrt{4.8062\times10^{-6}}}{\|\mathbf{V}\|_2}\right)$
end

Algorithm 2: A pseudocode for Eq. (25).

Function `GetFuV(U, V)`:
  Input : **U** is a vector of conservative variables;
  V is an arbitrary vector with the same dimension of **U**
  Output: $\mathbf{F_U}\cdot\mathbf{V}$
  $\varepsilon_v \leftarrow$ `GetEpsilon(V)`
  $\mathbf{F}_1 \leftarrow$ `ConsToFlux(U` $+\varepsilon_v\mathbf{V}$`)`
  $\mathbf{F}_2 \leftarrow$ `ConsToFlux(U` $-\varepsilon_v\mathbf{V}$`)`
  return $\dfrac{\mathbf{F}_1-\mathbf{F}_2}{2\varepsilon_v}$
end

Algorithm 3: A pseudocode for Eq. (27).

Function `GetFuuVW` (**U**, **V**, **W**) :
  Input : **U** is a vector of conservative variables;
    V and **W** are arbitrary vectors with the same dimension of **U**
  Output: $\mathbf{F}_{\mathbf{UU}} \cdot \mathbf{V} \cdot \mathbf{W}$
  $\varepsilon_v \leftarrow$ `GetEpsilon` (**V**)
  $\varepsilon_w \leftarrow$ `GetEpsilon` (**W**)
  $\mathbf{C}_1 \leftarrow \varepsilon_v \mathbf{V} + \varepsilon_w \mathbf{W}$
  $\mathbf{C}_2 \leftarrow \varepsilon_v \mathbf{V} - \varepsilon_w \mathbf{W}$
  $\mathbf{F}_1 \leftarrow$ `ConsToFlux` ($\mathbf{U} + \mathbf{C}_1$)
  $\mathbf{F}_2 \leftarrow$ `ConsToFlux` ($\mathbf{U} + \mathbf{C}_2$)
  $\mathbf{F}_3 \leftarrow$ `ConsToFlux` ($\mathbf{U} - \mathbf{C}_2$)
  $\mathbf{F}_4 \leftarrow$ `ConsToFlux` ($\mathbf{U} - \mathbf{C}_1$)
  return $\dfrac{\mathbf{F}_1 - \mathbf{F}_2 - \mathbf{F}_3 + \mathbf{F}_4}{4\varepsilon_v \varepsilon_w}$
end

Algorithm 4: A pseudocode for Eq. (28).

Function `GetFuuuVWX` (**U**, **V**, **W**, **X**) :
  Input : **U** is a vector of conservative variables;
    V, **W**, and **X** are arbitrary vectors with the same dimension of **U**
  Output: $\mathbf{F}_{\mathbf{UUU}} \cdot \mathbf{V} \cdot \mathbf{W} \cdot \mathbf{X}$
  $\varepsilon_v \leftarrow$ `GetEpsilon` (**V**)
  $\varepsilon_w \leftarrow$ `GetEpsilon` (**W**)
  $\varepsilon_x \leftarrow$ `GetEpsilon` (**X**)
  $\mathbf{C}_1 \leftarrow \varepsilon_v \mathbf{V} + \varepsilon_w \mathbf{W} + \varepsilon_x \mathbf{X}$
  $\mathbf{C}_2 \leftarrow \varepsilon_v \mathbf{V} - \varepsilon_w \mathbf{W} - \varepsilon_x \mathbf{X}$
  $\mathbf{C}_3 \leftarrow \varepsilon_v \mathbf{V} - \varepsilon_w \mathbf{W} + \varepsilon_x \mathbf{X}$
  $\mathbf{C}_4 \leftarrow \varepsilon_v \mathbf{V} + \varepsilon_w \mathbf{W} - \varepsilon_x \mathbf{X}$
  $\mathbf{F}_1 \leftarrow$ `ConsToFlux` ($\mathbf{U} + \mathbf{C}_1$)
  $\mathbf{F}_2 \leftarrow$ `ConsToFlux` ($\mathbf{U} - \mathbf{C}_2$)
  $\mathbf{F}_3 \leftarrow$ `ConsToFlux` ($\mathbf{U} + \mathbf{C}_3$)
  $\mathbf{F}_4 \leftarrow$ `ConsToFlux` ($\mathbf{U} - \mathbf{C}_4$)
  $\mathbf{F}_5 \leftarrow$ `ConsToFlux` ($\mathbf{U} + \mathbf{C}_4$)
  $\mathbf{F}_6 \leftarrow$ `ConsToFlux` ($\mathbf{U} - \mathbf{C}_3$)
  $\mathbf{F}_7 \leftarrow$ `ConsToFlux` ($\mathbf{U} + \mathbf{C}_2$)
  $\mathbf{F}_8 \leftarrow$ `ConsToFlux` ($\mathbf{U} - \mathbf{C}_1$)
  return $\dfrac{\mathbf{F}_1 - \mathbf{F}_2 - \mathbf{F}_3 + \mathbf{F}_4 - \mathbf{F}_5 + \mathbf{F}_6 + \mathbf{F}_7 - \mathbf{F}_8}{8\varepsilon_v \varepsilon_w \varepsilon_x}$
end

References

[1] Phillip Colella, Paul R. Woodward, The piecewise parabolic method (PPM) for gas-dynamical simulations, J. Comput. Phys. 54 (1) (1984) 174–201.
[2] Paul Woodward, Phillip Colella, The numerical simulation of two-dimensional fluid flow with strong shocks, J. Comput. Phys. 54 (1) (1984) 115–173.
[3] Ami Harten, Bjorn Engquist, Stanley Osher, Sukumar R. Chakravarthy, Uniformly high order accurate essentially non-oscillatory schemes, III, in: Upwind and High-Resolution Schemes, Springer, 1987, pp. 218–290.
[4] Xu-Dong Liu, Stanley Osher, Tony Chan, et al., Weighted essentially non-oscillatory schemes, J. Comput. Phys. 115 (1) (1994) 200–212.
[5] Guang-Shan Jiang, Chi-Wang Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1) (1996) 202–228.
[6] Rafael Borges, Monique Carmona, Bruno Costa, Wai Sun Don, An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws, J. Comput. Phys. 227 (6) (2008) 3191–3211.
[7] Marcos Castro, Bruno Costa, Wai Sun Don, High order weighted essentially non-oscillatory weno-z schemes for hyperbolic conservation laws, J. Comput. Phys. 230 (5) (2011) 1766–1792.
[8] Doron Levy, Gabriella Puppo, Giovanni Russo, Central weno schemes for hyperbolic systems of conservation laws, Modél. Math. Anal. Numér. (Mathematical Modelling and Numerical Analysis) 33 (3) (1999) 547–571.
[9] Jianxian Qiu, Chi-Wang Shu, On the construction, comparison, and local characteristic decomposition for high-order central weno schemes, J. Comput. Phys. 183 (1) (2002) 187–209.
[10] Jianxian Qiu, Chi-Wang Shu, Hermite weno schemes and their application as limiters for Runge–Kutta discontinuous Galerkin method: one-dimensional case, J. Comput. Phys. 193 (1) (2004) 115–135.
[11] Jianxian Qiu, Chi-Wang Shu, Hermite weno schemes and their application as limiters for Runge–Kutta discontinuous Galerkin method ii: two dimensional case, Comput. Fluids 34 (6) (2005) 642–663.
[12] Dinshaw S. Balsara, Sudip Garain, Chi-Wang Shu, An efficient class of weno schemes with adaptive order, J. Comput. Phys. 326 (2016) 780–804.
[13] Adam Reyes, Dongwook Lee, Carlo Graziani, Petros Tzeferacos, A new class of high-order methods for fluid dynamics simulations using Gaussian process modeling: one-dimensional case, J. Sci. Comput. 76 (1) (2018) 443–480.
[14] Adam Reyes, Dongwook Lee, Carlo Graziani, Petros Tzeferacos, A variable high-order shock-capturing finite difference method with GP-WENO, J. Comput. Phys. 381 (2019) 189–217.
[15] E.F. Toro, R.C. Millington, L.A.M. Nejad, Towards very high order Godunov schemes, in: Godunov Methods, Springer, 2001, pp. 907–940.
[16] Vladimir A. Titarev, Eleuterio F. Toro, ADER: arbitrary high order Godunov approach, J. Sci. Comput. 17 (1–4) (2002) 609–618.
[17] Vladimir A. Titarev, Eleuterio F. Toro, ADER schemes for three-dimensional non-linear hyperbolic systems, J. Comput. Phys. 204 (2) (2005) 715–736.

[18] Francesco Fambri, Michael Dumbser, Olindo Zanotti, Space–time adaptive ADER-DG schemes for dissipative flows: compressible Navier-Stokes and resistive MHD equations, Comput. Phys. Commun. 220 (2017) 297–318.
[19] Olindo Zanotti, Michael Dumbser, Efficient conservative ADER schemes based on WENO reconstruction and space–time predictor in primitive variables, Comput. Astrophys. Cosmol. 3 (1) (2016) 1.
[20] Dinshaw S. Balsara, Tobias Rumpf, Michael Dumbser, Claus-Dieter Munz, Efficient, high accuracy ADER-WENO schemes for hydrodynamics and divergence-free magnetohydrodynamics, J. Comput. Phys. 228 (7) (2009) 2480–2516.
[21] Dinshaw S. Balsara, Chad Meyer, Michael Dumbser, Huijing Du, Zhiliang Xu, Efficient implementation of ADER schemes for Euler and magnetohydrodynamical flows on structured meshes—speed comparisons with Runge-Kutta methods, J. Comput. Phys. 235 (2013) 934–969.
[22] Dinshaw S. Balsara, Higher-order accurate space-time schemes for computational astrophysics—part I: finite volume methods, Living Rev. Comput. Astrophys. 3 (1) (2017) 2.
[23] Gino Montecinos, Eleuterio Toro, Solver for the generalized Riemann problem for balance laws with stiff source terms: the scalar case, in: Hyperbolic Problems: Theory, Numerics and Applications, World Scientific, 2012, pp. 576–583 (in 2 Volumes).
[24] Gino I. Montecinos, Eleuterio F. Toro, Reformulations for general advection–diffusion–reaction equations and locally implicit ADER schemes, J. Comput. Phys. 275 (2014) 415–442.
[25] Eleuterio F. Toro, Gino I. Montecinos, Implicit semi-analytical solution of the generalized Riemann problem for stiff hyperbolic balance laws, J. Comput. Phys. 303 (2015) 146–172.
[26] Gino I. Montecinos, Dinshaw S. Balsara, A simplified Cauchy-Kowalewskaya procedure for the local implicit solution of generalized Riemann problems of hyperbolic balance laws, Comput. Fluids 202 (2020) 104490.
[27] Chaó-Kuang Chen, Shing-Huei Ho, Application of differential transformation to eigenvalue problems, Appl. Math. Comput. 79 (2–3) (1996) 173–188.
[28] Matthew R. Norman, Hal Finkel, Multi-moment ADER-Taylor methods for systems of conservation laws with source terms in one dimension, J. Comput. Phys. 231 (20) (2012) 6622–6642.
[29] Matthew R. Norman, Algorithmic improvements for schemes using the ADER time discretization, J. Comput. Phys. 243 (2013) 176–178.
[30] Matthew R. Norman, A WENO-limited, ADER-DT, finite-volume scheme for efficient, robust, and communication-avoiding multi-dimensional transport, J. Comput. Phys. 274 (2014) 1–18.
[31] Andrew J. Christlieb, Yaman Guclu, David C. Seal, The Picard integral formulation of weighted essentially nonoscillatory schemes, SIAM J. Numer. Anal. 53 (4) (2015) 1833–1856.
[32] Dongwook Lee, Hugues Faller, Adam Reyes, The piecewise cubic method (PCM) for computational fluid dynamics, J. Comput. Phys. 341 (2017) 230–257.
[33] Youngjun Lee, Dongwook Lee, A single-step third-order temporal discretization with Jacobian-free and Hessian-free formulations for finite difference methods, J. Comput. Phys. 427 (2021) 110063.
[34] Charles William Gear, Youcef Saad, Iterative solution of linear equations in ODE codes, SIAM J. Sci. Stat. Comput. 4 (4) (1983) 583–601.
[35] Peter N. Brown, Youcef Saad, Hybrid Krylov methods for nonlinear systems of equations, SIAM J. Sci. Stat. Comput. 11 (3) (1990) 450–481.
[36] Dana A. Knoll, David E. Keyes, Jacobian-free Newton-Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2) (2004) 357–397.
[37] Dana A. Knoll, H. Park, Kord Smith, Application of the Jacobian-free Newton-Krylov method to nonlinear acceleration of transport source iteration in slab geometry, Nucl. Sci. Eng. 167 (2) (2011) 122–132.
[38] David C. Seal, Yaman Güçlü, Andrew J. Christlieb, High-order multiderivative time integrators for hyperbolic conservation laws, J. Sci. Comput. 60 (1) (2014) 101–140.
[39] David C. Seal, Qi Tang, Zhengfu Xu, Andrew J. Christlieb, An explicit high-order single-stage single-step positivity-preserving finite difference WENO method for the compressible Euler equations, J. Sci. Comput. 68 (1) (2016) 171–190.
[40] Andrew J. Christlieb, Yuan Liu, Qi Tang, Zhengfu Xu, High order parametrized maximum-principle-preserving and positivity-preserving weno schemes on unstructured meshes, J. Comput. Phys. 281 (2015) 334–351.
[41] Tao Xiong, Jing-Mei Qiu, Zhengfu Xu, Parametrized positivity preserving flux limiters for the high order finite difference weno scheme solving compressible Euler equations, J. Sci. Comput. 67 (3) (2016) 1066–1088.
[42] Heng-Bin An, Ju Wen, Tao Feng, On finite difference approximation of a matrix-vector product in the Jacobian-free Newton-Krylov method, J. Comput. Appl. Math. 236 (6) (2011) 1399–1409.
[43] Sigal Gottlieb, Chi-Wang Shu, Total variation diminishing Runge-Kutta schemes, Math. Comput. 67 (221) (1998) 73–85.
[44] Raymond J. Spiteri, Steven J. Ruuth, A new class of optimal high-order strong-stability-preserving time discretization methods, SIAM J. Numer. Anal. 40 (2) (2002) 469–491.
[45] Gary A. Sod, A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws, J. Comput. Phys. 27 (1) (1978) 1–31.
[46] Soshi Kawai, Divergence-free-preserving high-order schemes for magnetohydrodynamics: an artificial magnetic resistivity method, J. Comput. Phys. 251 (2013) 292–318.
[47] Chi-Wang Shu, Stanley Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, II, in: Upwind and High-Resolution Schemes, Springer, 1989, pp. 328–374.
[48] Chi-Wang Shu, Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws, in: Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, Springer, 1998, pp. 325–432.
[49] Seth C. Spiegel, H.T. Huynh, James R. DeBonis, A survey of the isentropic Euler vortex problem using high-order methods, in: 22nd AIAA Computational Fluid Dynamics Conference, 2015, p. 2444.
[50] Tong Zhang, Yu Xi Zheng, Conjecture on the structure of solutions of the Riemann problem for two-dimensional gas dynamics systems, SIAM J. Math. Anal. 21 (3) (1990) 593–630.
[51] Carsten W. Schulz-Rinne, Classification of the Riemann problem for two-dimensional gas dynamics, SIAM J. Math. Anal. 24 (1) (1993) 76–88.
[52] Carsten W. Schulz-Rinne, James P. Collins, Harland M. Glaz, Numerical solution of the Riemann problem for two-dimensional gas dynamics, SIAM J. Sci. Comput. 14 (6) (1993) 1394–1414.
[53] Wai-Sun Don, Zhen Gao, Peng Li, Xiao Wen, Hybrid compact-WENO finite difference scheme with conjugate Fourier shock detection algorithm for hyperbolic conservation laws, SIAM J. Sci. Comput. 38 (2) (2016) A691–A711.
[54] Friedemann Kemm, On the proper setup of the double Mach reflection as a test case for the resolution of gas dynamics codes, Comput. Fluids 132 (2016) 72–75.
[55] Leonid Ivanovich Sedov, Similarity and Dimensional Methods in Mechanics, CRC Press, 1993.
[56] Bruce Fryxell, Kevin Olson, Paul Ricker, F.X. Timmes, Michael Zingale, D.Q. Lamb, Peter MacNeice, Robert Rosner, J.W. Truran, H. Tufo, FLASH: an adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes, Astrophys. J. Suppl. Ser. 131 (1) (2000) 273.
[57] Walter Boscheri, Michael Dumbser, A direct arbitrary-Lagrangian–Eulerian ADER-WENO finite volume scheme on unstructured tetrahedral meshes for conservative and non-conservative hyperbolic systems in 3d, J. Comput. Phys. 275 (2014) 484–523.
[58] Dinshaw S. Balsara, Three dimensional HLL Riemann solver for conservation laws on structured meshes; application to Euler and magnetohydrodynamic flows, J. Comput. Phys. 295 (2015) 1–23.