UNIVERSITY OF CALIFORNIA
SANTA CRUZ

A FAST AND PORTABLE HIGH-ORDER IN TEMPORAL METHOD FOR COMPUTATIONAL FLUID DYNAMICS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

APPLIED MATHEMATICS AND STATISTICS

by

Youngjun Lee

August 2021

The Dissertation of Youngjun Lee
is approved:

Dongwook Lee, Chair

Nicholas Brummell

Hongyun Wang

Peter F. Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

A Fast and Portable High-Order in Temporal Method for Computational Fluid Dynamics

by

Youngjun Lee

A clear, concise abstract explaining the why, what, and how of your work.

A loving dedication.

Acknowledgments

The text of this dissertation includes reprints of the following previously published material:

- Youngjun Lee and Dongwook Lee. A single-step third-order temporal discretization with jacobian-free and hessian-free formulations for finite difference methods. *Journal of Computational Physics*, 427:110063, 2021.

- Youngjun Lee, Dongwook Lee, and Adam Reyes. A recursive system-free single-step temporal discretization method for finite difference methods. *Journal of Computational Physics: X*, 12:100098, 2021.

The primary co-author Dongwook Lee listed in these publications directed and supervised the research which forms the basis for this dissertation.

Chapter 1

Introduction

In past decades, the rapid evolution of the high-performance computing (HPC) systems offers growing computational capacity for the numerical simulations of various scientific fields, where conducting a direct experiment is extremely expensive or notoriously complicated. As the computing power of HPC systems gradually increases, scientists can compute more complex and computationally intensive physical models such as visualizing black holes [35, 1], simulating nuclear fusions [33, 27], studying laser-plasma interactions [45, 67], to name a few.

In order to simulate these physical phenomena, it demands meticulously designed numerical algorithms for solving nonlinear, multidimensional, and multiphysics equations judiciously. Generally speaking, numerical algorithms require more computational power for better solution accuracy, *i.e.*, using high-resolution grid configuration.

However, the recent hardware development trend – the progression of the memory capacity per compute core has become gradually saturated [3] – is compelling the HPC community to find more efficient ways that can best exercise computing resources in pursuing computer simulations. As reported in 2014 [22], decreasing memory density per compute core will be the primary limiting factor to the

scalability of scientific simulation codes.

To meet this end, modern practitioners have relentlessly delved into advancing high-arithmetic-intensity models that can increase numerical accuracy per degree of freedom while operating with reduced memory requirements and data transfers in HPC architectures. For example, in the computational fluid dynamics (CFD) community, one such computing paradigm is to promote high-order methods in which high arithmetic intensity is achieved by using an increasing number of higher-order terms. Due to its high availability in increasing the quality of numerical solutions with fewer grid points, high-order discrete methods for hyperbolic conservation laws have become primary themes in the CFD community.

Under the dual computational need for accuracy and stability, the CFD community has developed high-order reconstruction and interpolation strategies that can achieve spatially high-order approximation. [69, 18, 43, 36, 11, 14, 46, 5] However, efforts to achieve a high-order accuracy in the temporal axis have seen a renewed effort. For decades, multi-stage time integrators [31, 32, 30] have been considered as the standard temporal integration strategy for an extensive range of high-order numerical schemes for partial differential equation (PDE) solvers.

This dissertation develops a single-stage, high-order time integration scheme for hyperbolic PDEs. The core design concept is to achieve high-order accuracy within a single step to reduce computational costs for the overall numerical schemes. Another important objective for designing a high-order time integrator in this dissertation is to increase its portability. By designing a time integrator independent of the system of equations, one can provide increased flexibility and ease of code implementation.

Consequently, a newly developed time integrator showed more than two times faster performance gain than the conventional multi-stage methods. Also, it can

readily replace only the temporal part of any existing simulation code.

Chapter 2

Discretization Methods

This dissertation interests in solving the general conservation laws of hyperbolic PDEs, predicting numerical solutions with high-order accuracy. This chapter introduces two general ways to discretize the Euler equations, which will be of particular interest in this dissertation.

2.1 Euler Equations

In three dimensions, the conservation laws may be written as,

$$\partial_t \mathbf{U} + \nabla \cdot \mathcal{F}(\mathbf{U}) = \partial_t \mathbf{U} + \partial_x \mathbf{F}(\mathbf{U}) + \partial_y \mathbf{G}(\mathbf{U}) + \partial_z \mathbf{H}(\mathbf{U}) = 0, \tag{2.1}$$

where $\mathbf{U}$ is a vector of conserved variables and $\mathcal{F} = (\mathbf{F}(\mathbf{U}), \mathbf{G}(\mathbf{U}), \mathbf{H}(\mathbf{U}))^T$ is the flux function in the x, y, and z directions. The conservation law is considered hyperbolic if the flux Jacobian has only real eigenvalues and is diagonalizable. Thus,

$$\mathbf{A} = \partial_{\mathbf{U}} \mathbf{F} = \mathbf{R}\mathbf{\Lambda}\mathbf{L}, \tag{2.2}$$

where $\mathbf{A}$ is the flux Jacobian in x-direction, $\mathbf{\Lambda}$ is a diagonal matrix with real eigenvalues, and $\mathbf{L}$ and $\mathbf{R}$ are corresponding left and right eigenvectors.

This dissertation focuses on solving Euler equations, which govern compressible, adiabatic inviscid flow. In the Euler equations, the conserved variables and the flux functions are defined as,

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u\,(E+p) \end{bmatrix}, \quad \mathbf{G}(\mathbf{U}) = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v\,(E+p) \end{bmatrix}, \quad \mathbf{H}(\mathbf{U}) = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w\,(E+p) \end{bmatrix}. \tag{2.3}$$

In the above equations, ρ is the density, $\mathbf{u} = (u, v, w)^T$ is the velocity, E is the total energy, and p is the pressure of the fluid. E, the total energy of the fluid represents the sum of internal and kinetic energy,

$$E = \epsilon + \frac{1}{2}\rho \mathbf{u}^2, \tag{2.4}$$

where the internal energy of the fluid, ϵ, obeys the equation of state. (EOS) This dissertation uses the ideal gas law:

$$\epsilon = \frac{p}{\gamma - 1}, \tag{2.5}$$

where γ is the specific heat ratio.

In the following, this dissertation uses the Euler equation as an example of the conserved hyperbolic system. However, numerical methods presented in this dissertation are valid for any system of the form of Eq. (2.1). Magnetohydrodynamics (MHD) equations, for example, the general numerical strategies will

be nearly identical to the Euler equations except for the solenoidal constraint of the magnetic field. ($\nabla \cdot \mathbf{B} = 0$) Thus, the high-order methods presented in this dissertation can be applied to the MHD simulation code easily.

2.2 Finite Volume Method

The popular way to consider discretized variables for the conserved system is to cast volume integrals to the governing equations, called the finite volume method. (FVM) Consider Eq. (2.1) discretized on a uniform grid containing cells with equal spacing $(\Delta x, \Delta y, \Delta z)$ in the three spatial dimensions. Then, each cell's center can be indexed by (i, j, k) at (x_i, y_j, z_k) and the cell's face centers at each interface by $(i \pm \frac{1}{2}, j, k), (i, j \pm \frac{1}{2}, k), (i, j, k \pm \frac{1}{2})$. Taking the volume average of each computational cell ($\frac{1}{\mathcal{V}_{ijk}} \int_{\mathcal{V}_{ijk}} \cdot \, d\mathcal{V}$) to Eq. (2.1) and applying the divergence theorem, we have,

$$\frac{1}{\mathcal{V}_{ijk}} \int_{\mathcal{V}_{ijk}} \partial_t \mathbf{U} \, d\mathcal{V} + \frac{1}{\mathcal{V}_{ijk}} \oint_{\mathcal{S}_{ijk}} \mathcal{F}(\mathbf{U}) \cdot \mathbf{n} \, d\mathcal{S} = 0, \tag{2.6}$$

where $\mathcal{V}_{ijk}$ is the volume of the cell at i, j, k, and $\mathcal{S}_{ijk}$ is the surrounding surfaces of the cell at i, j, k.

The semi-discretized form of FVM representation of the conserved system is obtained by substituting the dimensionally split flux functions $(\mathbf{F}(\mathbf{U}), \mathbf{G}(\mathbf{U}), \mathbf{H}(\mathbf{U}))$:

$$\begin{aligned} \partial_t \overline{\mathbf{U}}_{i,j,k} = &- \frac{1}{\Delta x} \left(\widetilde{\mathbf{F}}_{i+\frac{1}{2},j,k} - \widetilde{\mathbf{F}}_{i-\frac{1}{2},j,k} \right) \\ &- \frac{1}{\Delta y} \left(\widetilde{\mathbf{G}}_{i,j+\frac{1}{2},k} - \widetilde{\mathbf{G}}_{i,j-\frac{1}{2},k} \right) \\ &- \frac{1}{\Delta z} \left(\widetilde{\mathbf{H}}_{i,j,k+\frac{1}{2}} - \widetilde{\mathbf{H}}_{i,j,k-\frac{1}{2}} \right). \end{aligned} \tag{2.7}$$

In the above equation, the overline indicates a volume-averaged quantity, while

the tilde indicates a surface average at half-indexed cell face. Note that the above semi-discretized form of FVM scheme is a purely analytical result without any numerical approximation. The numerical methods are used to estimate surface-averaged fluxes at each cell's interfaces and update the volume-averaged conserved variables to the next time step.

The most common way to approximate interfacial fluxes for FVM solver for Euler equations is to solve the Riemann problem at cell interfaces following the Godunov method [29]. The Riemann problem is composed of a conservation equation with a single discontinuity in its initial condition. As firstly introduced by Godunov in [29], the Riemann solver ($\mathcal{RS}$) gives a numerical flux across the discontinuity in the Riemann problem. For example, the numerical flux across the discontinuity at $x_{i+\frac{1}{2},j,k}$ can be calculated as,

$$\hat{\mathbf{f}}_{i+\frac{1}{2},j,k} = \mathcal{RS}(\mathbf{U}^{\mathrm{L}}_{i+\frac{1}{2},j,k},\ \mathbf{U}^{\mathrm{R}}_{i+\frac{1}{2},j,k}). \tag{2.8}$$

The precedent task for solving the Riemann problem is to determining Riemann states at interfaces. Note that the inputs of the Riemann solver are regarded as *pointwise* values, ($\mathbf{U}^{\mathrm{L}}_{i+\frac{1}{2},j,k},\ \mathbf{U}^{\mathrm{R}}_{i+\frac{1}{2},j,k}$) while the fundamental data type of FVM is the *volume-averaged* values. ($\overline{\mathbf{U}}_{i,j,k}$) To specify the pointwise Riemann states at interfaces with given volume-averaged conserved variables, they must be reconstructed from the neighboring volume-averaged quantities. For example, a one-dimensional stencil with radius=r, the left Riemann states at $i+\frac{1}{2}$ can be reconstructed from cell-centered volume-averaged conserved variables ($\overline{\mathbf{U}}_{i-r,j,k},\ldots,\overline{\mathbf{U}}_{i,j,k},\ldots,\overline{\mathbf{U}}_{i+r,j,k}$), using p-th order accurate reconstruction operator $\mathcal{R}(\cdot)$:

$$\mathbf{U}^{\mathrm{L}}_{i+\frac{1}{2},j,k} = \mathcal{R}(\overline{\mathbf{U}}_{i-r,j,k},\ldots,\overline{\mathbf{U}}_{i+r,j,k}) + \mathcal{O}(\Delta x^p). \tag{2.9}$$

The spatial order of accuracy, p, of the FVM solver is thereby determined by choice of the reconstruction operator, $\mathcal{R}(\cdot)$ which will be discussed in **find my ref!**.

It is important to note that the numerical flux resulting from the Riemann solver is also a pointwise representation, while the surface-averaged fluxes ($\widetilde{\mathbf{F}}_{i\pm\frac{1}{2},j,k}, \widetilde{\mathbf{G}}_{i,j\pm\frac{1}{2},k}, \widetilde{\mathbf{H}}_{i,j,k\pm\frac{1}{2}}$) are needed for FVM formulation as presented in Eq. (2.7). This should be addressed thoroughly, as a naive approximation of the pointwise flux to the surface-averaged flux is bounded by second-order accuracy no matter the accuracy of the Riemann states:

$$\begin{aligned}\widetilde{\mathbf{F}}_{i+\frac{1}{2},j,k} &= \frac{1}{\Delta y \Delta z}\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}\int_{z_{k-\frac{1}{2}}}^{z_{k+\frac{1}{2}}} \mathbf{F}(x_{i+\frac{1}{2}}, y, z)\, dz\, dy \\ &= \mathbf{F}_{i+\frac{1}{2},j,k} + \mathcal{O}(\Delta y^2, \Delta z^2).\end{aligned} \tag{2.10}$$

The conventional way to achieve higher than second-order accuracy in FVM solver is to solve the Riemann problem at multiple quadrature points on each face. [63, 44, 71] More recent studies proposed ways to avoid multiple calls of Riemann solver, reconstructing surface-averaged fluxes from pointwise Riemann fluxes, [13, 26] using linear combinations of Riemann fluxes, [24, 23] to name a few.

2.3 Finite Difference method

As it firstly proposed in [60], the finite difference method (FDM) seeks a discretization of the spatial derivatives of the fluxes in *pointwise* representation.

Assuming that there exist numerical fluxes satisfy the conservative form as,

$$\begin{aligned}\partial_t \overline{\mathbf{U}}_{i,j,k} = &- \frac{1}{\Delta x}\left(\hat{\mathbf{f}}_{i+\frac{1}{2},j,k} - \hat{\mathbf{f}}_{i-\frac{1}{2},j,k}\right) \\ &- \frac{1}{\Delta y}\left(\hat{\mathbf{g}}_{i,j+\frac{1}{2},k} - \hat{\mathbf{g}}_{i,j-\frac{1}{2},k}\right) \\ &- \frac{1}{\Delta z}\left(\hat{\mathbf{h}}_{i,j,k+\frac{1}{2}} - \hat{\mathbf{h}}_{i,j,k-\frac{1}{2}}\right),\end{aligned} \tag{2.11}$$

where $\hat{\mathbf{f}}_{i\pm\frac{1}{2},j,k}, \hat{\mathbf{g}}_{i,j\pm\frac{1}{2},k}, \hat{\mathbf{h}}_{i,j,k\pm\frac{1}{2}}$ are the *pointwise* numerical fluxes in each direction at half-indexed cell-face centers. The remaining task is to identify the numerical fluxes in desired order of accuracy, p, which satisfy,

$$\partial_x \mathbf{F}|_{\mathbf{x}=\mathbf{x}_{ijk}} = \frac{1}{\Delta x}\left(\hat{\mathbf{f}}_{i+\frac{1}{2},j,k} - \hat{\mathbf{f}}_{i-\frac{1}{2},j,k}\right) + \mathcal{O}(\Delta x^p), \quad \mathbf{x}_{ijk} = (x_i, y_j, z_k), \tag{2.12}$$

and similarly in y and z fluxes.

In order to specify the numerical fluxes for FDM, consider the pointwise x-flux $\mathbf{F}(x, y_j, z_k)$ as a one-dimensional cell average of an auxiliary function $\hat{\mathbf{F}}$,

$$\mathbf{F}(x, y_j, z_k) = \frac{1}{\Delta x}\int_{x-\frac{\Delta x}{2}}^{x+\frac{\Delta x}{2}} \hat{\mathbf{F}}(\xi, y_j, z_k)\, d\xi\,. \tag{2.13}$$

Then the analytic derivative of Eq. (2.13) at $x = x_i$ in x-direction becomes

$$\partial_x \mathbf{F}|_{x=x_i} = \frac{1}{\Delta x}\left(\hat{\mathbf{F}}(x_{i+\frac{1}{2}}, y_j, z_k) - \hat{\mathbf{F}}(x_{i-\frac{1}{2}}, y_j, z_k)\right). \tag{2.14}$$

Comparing Eq. (2.12) and Eq. (2.14), the numerical fluxes in FDM are obtained with desired order of accuracy, p, if they can be defined with the following relationship with $\hat{\mathbf{F}}$,

$$\hat{\mathbf{f}}_{i+\frac{1}{2},j,k} = \hat{\mathbf{F}}(x_{i+\frac{1}{2}}, y_j, z_k) + \mathcal{O}(\Delta x^p). \tag{2.15}$$

Mathematically speaking, the inverse problem of Eq. (2.15) is exactly the same

as the conventional 1D reconstruction problem in FVM, the operation of which is specifically designed to find the primitive function value $\hat{\mathbf{F}}$ at a certain location (mostly, $x_{i\pm\frac{1}{2}}$) in the i-th cell, given the integral-averaged (or volume-averaged) values $\mathbf{F}$ at nearby stencil points as input. Namely, this can be written as

$$\hat{\mathbf{F}}(\xi, y_j, z_k) = \mathcal{R}\left(\mathbf{F}_{i-r,j,k}, \ldots, \mathbf{F}_{i+r,j,k}\right) + \mathcal{O}(\Delta x^p), \quad \xi \in [x_{i-\frac{1}{2}}, x_{i-\frac{1}{2}}], \tag{2.16}$$

where $\mathcal{R}(\cdot)$ is a p-th order accurate reconstruction operator that used for FVM formulation in Section 2.2, and will be discussed in **find my ref!**.

Contrary to FVM, the conservative FDM uses only the pointwise values for constructing numerical strategies, not requiring the data conversion between pointwise and volume-averaged quantities. In addition, the high-order reconstruction schemes used for constructing Riemann states in FVM can be used for constructing numerical fluxes in FDM without intense changes in the simulation code – only a simple change in the input variables for the reconstruction operator. This simplicity in numerical strategy attracts researchers in the CFD community, leading various adoptions in high-order solvers. [36, 58, 46, 17, 55]

Compared to FVM, the major difference of FDM is obtaining high-order numerical fluxes directly from the reconstruction operator. Although it simplifies the numerical schemes, the direct formulation of the numerical fluxes hinders its further modifications while keeping a high-order convergence rate.

For example, the adaptive mesh refinement (AMR) grid configuration [9, 10] requires numerical fluxes splitting between the coarse grid to the fine grid in a conservative manner. Conventionally, this is ensured by an additional flux correction step in FVM formulation. [10] However, in FDM, a different approach should be taken because modifying the high-order numerical fluxes may spoil the order of accuracy. One possible way to maintain conservation across coarse to the fine

grid points is to apply nonlinear interpolation on the conserved variables, imply them as boundary conditions, and distribute the calculated errors among coarse grid points. [56]

Another limitation on the direct formulation of numerical fluxes in FDM is the lack of the option to include substructure in the wave model, which Riemann solver typically does in FVM to resolve certain features better. Del Zanna proposed one alternative way of FDM [20, 21], which views numerical fluxes as the Riemann fluxes with the series of high-order correction terms. This approach can achieve a high-order convergence rate with Riemann fluxes like in FVM, without the additional data type conversions required of conventional FVM. [54]

2.4 Conclusion

Two major discretization strategies for conservative systems have been presented. Both the finite difference and finite volume methods are able to achieve high-order spatial accuracy by reconstructing volume-averaged quantities to the pointwise values.

The finite difference method

- evolves the pointwise conserved variables,
- provides a straightforward framework without data type conversions,
- requires high-order numerical fluxes constructed from the pointwise, cell-centered physical fluxes directly from the high-order reconstruction methods and
- may be delicate with the additional modifications on the numerical fluxes.

The finite volume method

- evolves the volume-averaged conserved variables,
- requires rigorous data type conversions to maintain high-order accuracy,
- requires high-order reconstruction of the Riemann states from the cell-centered conserved variables and
- guarantees the conservation laws over the whole spatiotemporal domain by design.

Chapter 3

High-Order Methods for FDM

As mentioned in Chapter 1, the high-order numerical schemes are ideal for maximizing the solution accuracy on a given grid resolution; thus, they can produce more accurate solutions on a limited memory capacity.

The discretization strategies for conservative PDEs presented in Chapter 2 can achieve high-order solution accuracy both in spatial and temporal dimensions. Typically, the spatial accuracy is determined by the accuracy of the reconstruction scheme used for estimating Riemann states (FVM) or numerical fluxes (FDM) at cells interfaces. On the other hand, the numerical time integration strategy for the semi-discretized form of the conservative PDEs (Eqs. (2.7) and (2.11)) will settle the temporal accuracy. Since the solution lies on the spatiotemporal plane, both the temporal and spatial errors should be contemplated in a controlled manner to analyze the numerical solutions properly.

The fundamental mission of the numerical scheme is to estimate the values on unknown points with given available data in an accurate fashion. One straightforward way is to interpolate/reconstruct the data to form piecewise polynomials and estimate the desired points using the polynomials. In this way, the order of accuracy will be determined by the degree of polynomials; the number of data

points used to interpolate/reconstruct the data in each stencil.

However, nonlinear systems like Euler equations, the main interest in this dissertation, require more sophisticated care in designing high-order numerical methods because the system can form and evolve discontinuity profiles even with smooth initial conditions. Generally speaking, an artless high-degree piecewise polynomial function can not handle the discontinuous profiles and introduces spurious oscillations caused by over- and under-shooted profiles of the underlying function. Therefore, the stability of the numerical method must be taken into account for solving hyperbolic PDEs in high-order accuracy.

One way to ensure the stability of the numerical scheme is to limit the total variation (TV) of the solution, called the total variation diminishing (TVD) property. The total variation of the solution in one-dimensional discrete solution u^n at time $t = t^n$ is defined as,

$$TV(u^n) \coloneqq \sum_i \left| u^n_{i+1} - u^n_i \right|, \tag{3.1}$$

and it should be bounded by the total variation of the initial conditions to ensure stability as,

$$TV(u^{n+1}) \leq TV(u^n). \tag{3.2}$$

Van Leer introduced TVD flux limiters to provide TVD property on the piecewise linear functions [68] by limiting the slope of the second-degree polynomial. A simple choice of the TVD slope limiter for the second-degree profile is the minmod slope limiter,

$$\text{minmod}(a, b) = \begin{cases} a, & \text{if } |a| < |b| \text{ and } ab > 0 \\ b, & \text{if } |a| > |b| \text{ and } ab > 0 \\ 0, & \text{if } ab < 0, \end{cases} \tag{3.3}$$

where $a = \frac{u_{i+1}-u_i}{\Delta x}$ and $b = \frac{u_i-u_{i-1}}{\Delta x}$ are the forward and backward slope at the cell I_i centered at $x = x_i$. The TVD slope limiters are also used for the popular third-order scheme, the piecewise parabolic method (PPM) by Woodward and Colella [18], for enforcing the TVD property on the quadratic polynomial.

The TVD property is also crucial in designing high-order temporal integration schemes. In the *method of lines* approximations, the semi-discretized form of PDEs (Eqs. (2.7) and (2.11)) can be viewed as an ordinary differential equation (ODE) system in the temporal axis, which an ODE solver can discretize. Thus, the well-known ODE solver, Runge-Kutta (RK) method, can be used as the time integration method in the method of lines schemes; however, the TVD property must be enforced to preserve the numerical stability. Shu and Osher designed TVD limited Runge-Kutta methods [59, 57] for solving hyperbolic conservation laws. Subsequently, Gottlieb and her collaborators [31, 32, 30] further developed this idea to the strong stability preserving Runge-Kutta (SSP-RK) method, which is by far the most popular high-order time integration method used in the CFD community.

This chapter will introduce general schemes for achieving high-order accuracy both in spatial and temporal dimensions in the finite difference method, which is the main interest in this dissertation. The high-order reconstruction schemes in Section 3.1 predict FDM numerical fluxes at cell interfaces, and they are identical to the reconstruction methods used to predict interfacial Riemann states in FVM. The Runge-Kutta methods and the Lax-Wendroff type methods will be introduced in Section 3.2 for achieving high-order in time accuracy for FDM formulation.

3.1 High-Order Reconstruction Schemes

The high-order reconstruction schemes play a pivotal role in reducing numerical errors on the *spatial axis* for solving discrete PDEs under FVM and FDM formulations. The reconstruction schemes furnish a pointwise representation of the given volume-averaged data; thus, they are suitable for estimating pointwise values with given volume-averaged conserved variables (FVM) or volume-averaged auxiliary functions (FDM). Modern practitioners in the CFD community have focused on designing high-order reconstruction schemes that can generate highly accurate solutions while maintaining numerical stability at discontinuous regions. Examples include the early success of the piecewise parabolic method (PPM) by Colella and Woodward [18], 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 the finite difference method, the numerical fluxes can be estimated through reconstruction schemes by inputting the pointwise physical fluxes at the cell centers. Assuming that a p-th order reconstruction scheme, $\mathcal{R}(\cdot)$, taking the $2r$ length of stencil centered at cell I_i, and generating estimated pointwise value at $x = x_{i+\frac{1}{2}}$, then the FDM numerical flux $\hat{\mathbf{f}}_{i+\frac{1}{2}}$ at $x = x_{i+\frac{1}{2}}$ can be found by,

$$\hat{\mathbf{f}}_{i+\frac{1}{2}} = \mathcal{R}(\mathbf{F}_{i-r}, \ldots, \mathbf{F}_{i+r}) + \mathcal{O}(\Delta x^p). \tag{3.4}$$

However, one additional numerical step should be needed for the robustness of the solution. It is well-known that the upwind numerical fluxes can provide more robust solutions in the CFD community. In FVM, this is achieved by solving the Riemann problem, but for FDM, one separated routine should be considered to provide upwind property in the numerical flux. The flux splitting method

is the general way to yield upwinding numerical flux in FDM by splitting the pointwise fluxes into two components moving towards and away from the interface of interest. For example, one can split the flux function into two parts:

$$\mathbf{F}_i = \mathbf{F}_i^+ + \mathbf{F}_i^-, \qquad \mathbf{F}_i^\pm = \frac{1}{2}\left(\mathbf{F}_i \pm \alpha^k \mathbf{U}_i\right), \tag{3.5}$$

where α^k is the global maximum characteristic speed of k-th characteristic field, i.e., the maximum absolute value of k-th eigenvalue of flux Jacobian $\partial_{\mathbf{U}}\mathbf{F}$ over the whole domain. This procedure is called the global Lax–Friedrichs flux splitting [36] since we take global maximum for each α^k. The reconstruction method is applied to the positive and negative parts of the fluxes $\mathbf{F}_i^\pm$ to construct numerical fluxes, and then they are collected at each interface:

$$\hat{\mathbf{f}}_{i+\frac{1}{2}} = \hat{\mathbf{f}}^+_{i+\frac{1}{2}} + \hat{\mathbf{f}}^-_{i+\frac{1}{2}}, \qquad \hat{\mathbf{f}}^+_{i+\frac{1}{2}} = \mathcal{R}(\mathbf{F}^+_s), \;\; \hat{\mathbf{f}}^-_{i+\frac{1}{2}} = \mathcal{R}(\mathbf{F}^-_{s'}), \tag{3.6}$$

where the sub-index s represents the stencil ranging from $i-r, \ldots, i+r$, while at the same time, $s' = 2i - s + 1$.

Although the global Lax–Friedrichs flux splitting provides improved stability and robustness of the numerical scheme, it also introduces numerical dissipation into the solution. One possible way to minimize the numerical dissipation of the flux splitting is to project the fluxes into the characteristic field, so-called Rusanov Lax–Friedrichs flux splitting. [19, 46] The Rusanov Lax-Friedrichs splitting projects pointwise physical fluxes to the left- and the right-going parts according to the characteristic decomposition of the Jacobian matrix,

$$\partial_{\mathbf{U}}\mathbf{F}|_{\mathbf{U}_{i+\frac{1}{2}}} = \mathbf{R}_{i+\frac{1}{2}}\mathbf{\Lambda}_{i+\frac{1}{2}}\mathbf{L}_{i+\frac{1}{2}}, \quad \mathbf{U}_{i+\frac{1}{2}} = \frac{\mathbf{U}_i + \mathbf{U}_{i+1}}{2}, \tag{3.7}$$

where $\mathbf{R}$ and $\mathbf{L}$ are the matrices of right and left eigenvectors, and $\mathbf{\Lambda}$ is the diagonal matrix whose diagonal entries are corresponding eigenvalues. The projection proceeds to construct s different left-going $(-)$ and right-going $(+)$ characteristic states of the pointwise fluxes, denoted as $\mathbf{V}^{k,\pm}_{(i+\frac{1}{2}):s}$ to the cell interface $i+\frac{1}{2}$ as

$$\begin{aligned}\mathbf{V}^{k,+}_{(i+\frac{1}{2}):s} &= \frac{1}{2}\mathbf{L}^{k}_{i+\frac{1}{2}} \cdot \left(\mathbf{F}_{s} + \alpha^{k}\mathbf{U}_{s}\right), \\ \mathbf{V}^{k,-}_{(i+\frac{1}{2}):s} &= \frac{1}{2}\mathbf{L}^{k}_{i+\frac{1}{2}} \cdot \left(\mathbf{F}_{s'} - \alpha^{k}\mathbf{U}_{s'}\right).\end{aligned} \tag{3.8}$$

Again, s and s' are representing the stencil $s = i-r, \ldots, i+r,\ s' = 2i-s+1$, and the superscript k represents each characteristic field. The coefficient α^{k} is chosen to be the maximum absolute value of the k-th characteristic speed over the entire computational domain, resulting in the so-called global Lax-Friedrichs flux splitting. The projected fluxes will be taken into the reconstruction scheme and projected back to the numerical fluxes:

$$\hat{\mathbf{f}}_{i+\frac{1}{2}} = \sum_{k}\left(\hat{\mathbf{V}}^{k,+}_{i+\frac{1}{2}} + \hat{\mathbf{V}}^{k,-}_{i+\frac{1}{2}}\right)\mathbf{R}^{k}_{i+\frac{1}{2}}, \quad \hat{\mathbf{V}}^{k,\pm}_{i+\frac{1}{2}} = \mathcal{R}\left(\mathbf{V}^{k,\pm}_{(i+\frac{1}{2}):s}\right). \tag{3.9}$$

3.1.1 Weighted Essentially Non-Oscillatory Methods

Generally speaking, the high-order reconstruction schemes based on the polynomial approach assuming that there exists a unique polynomial $\phi(x)$ satisfies volume-averaging conditions:

$$\frac{1}{\Delta x}\int_{I_k}\phi(x)\,dx = \overline{q}_{k}, \quad I_k \in S \tag{3.10}$$

where k is the index of cell within a stencil S and $\overline{q}_{k}$ is a volume-averaged quantity at k. However, when S contains a strong gradient of the volume-averaged data $\overline{q}_{k}$, then it suffers from spurious oscillations since the polynomial $\phi(x)$ is assumed to

be smooth over the stencil. Handling discontinuous profiles is essential in solving Euler equations, as it can form discontinuous profiles even with smooth initial conditions.

In 1987, Harten et al. [34] proposed an essentially non-oscillatory (ENO) scheme that chooses the appropriate stencil adaptively to avoid containing any discontinuities in the stencil. In ENO methods, the non-oscillatory stencil is chosen by considering all possible stencils of the required size and measure the smoothness on each of them. Liu et al. [43] improved this idea by introducing the weighted 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 [36] and became one of the most popular high-order reconstruction and interpolation methods for solving shock-dominant CFD simulation.

Conventionally, WENO method with m sub-stencils uses $2m-1$ data points and ensures $2m-1$ order of accuracy in a smooth region. As an example, the popular five-points, fifth-order WENO-JS [36] method is presented below.

Consider three sub-stencils,

$$S_m = \{I_{i-3+m}, I_{i-2+m}, I_{i-1+m}\}, \quad m = 1, 2, 3, \tag{3.11}$$

then the second degree polynomials $p_m(x)$ can be constructed on each S_m as,

$$\int_{I_k} p_m(x)\, dx = \overline{q}_k, \quad I_k \in S_m. \tag{3.12}$$

The reconstructed profiles are depicted in Fig. 3.1. With three sub-polynomials, the reconstructed pointwise values at the cell interfaces $q_{i\pm\frac{1}{2}}$ can be represented

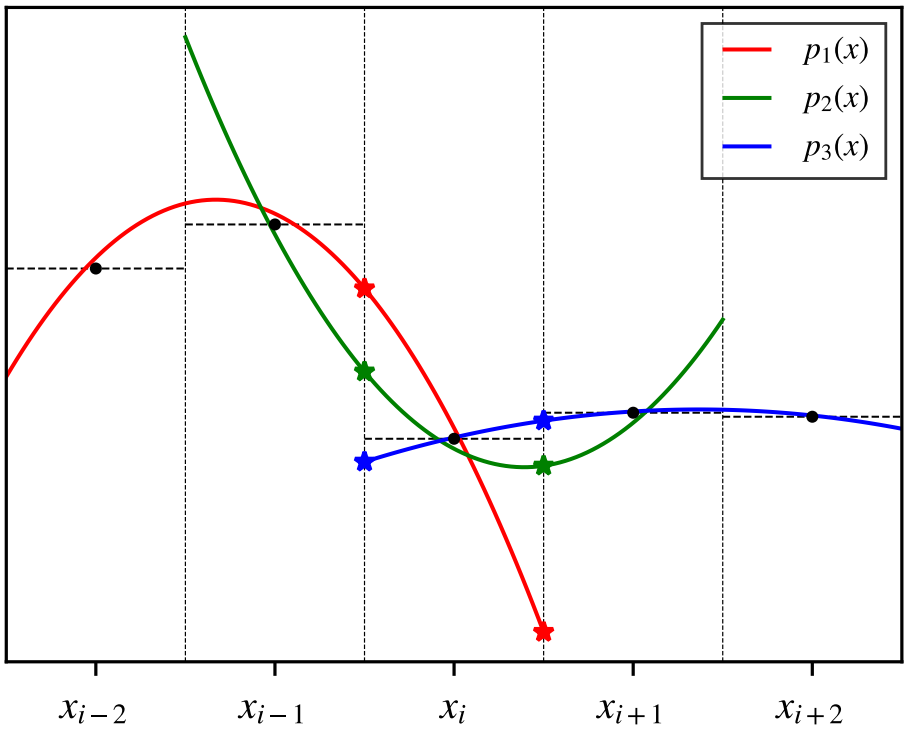

Figure 3.1: The reconstructed profiles in three sub-stencils of the fifth-order WENO-JS scheme. The black dots and horizontal dotted lines represent the volume averaged quantities. Note that there is a sharp discontinuity at $x = x_{i-\frac{1}{2}}$, resulting three different reconstructed pointwise values at $x = x_{i\pm\frac{1}{2}}$ of each polynomial, marked as stars. In ENO perspective, $p_3(x)$ is an appropriate choice, as $S_3 = \{I_i, I_{i+1}, I_{I+2}\}$ does not include the discontinuous point.

as a convex combination with nonlinear weights ω_m:

$$q_{i\pm\frac{1}{2}} = \sum_{m=1}^{3} \omega_m p_m(x_{i\pm\frac{1}{2}}). \tag{3.13}$$

The core design principle of WENO is to construct nonlinear weights ω_m that adaptively select smooth stencils and converges into the linear reconstruction scheme when all stencils are smooth. Mathematically speaking, the nonlinear

weights should be converged into the *linear weights* $\gamma_m,\ m = 1, 2, 3$,

$$\phi(x) = \sum_{m=1}^{3} \gamma_m p_m(x), \tag{3.14}$$

where $\phi(x)$ is the reconstructed polynomial within the whole stencil $S = \bigcup_{m=1}^{3} S_m$,

$$\frac{1}{\Delta x} \int_{I_k} \phi(x)\, dx = \overline{q}_k, \quad I_k \in \bigcup_{m=1}^{3} S_m. \tag{3.15}$$

As the $\phi(x)$ is the quartic polynomial, it ensures fifth-order convergence rate of the estimations for pointwise values,

$$q_{i\pm\frac{1}{2}} = \phi(x_{i\pm\frac{1}{2}}) + \mathcal{O}(\Delta x^5). \tag{3.16}$$

Jiang and Shu [36] proposed a functional form of the nonlinear weights by,

$$\omega_m = \frac{\widetilde{\omega}_m}{\sum_s \widetilde{\omega}_s}, \quad \widetilde{\omega}_m = \frac{\gamma_m}{(\epsilon + \beta_m)^p}, \tag{3.17}$$

where ϵ is the small number (e.g., 1.0×10^{-36}) to prevent division by zero, β_m is the smoothness indicator which measures the smoothness of the data in the given stencil S_m. The parameter p is an amplification factor for the difference of scales when a discontinuity is present on one of the candidate stencils.

The remaining step for WENO-JS is to construct the smoothness indicator, which has large values when the stencil data is not smooth and becomes arbitrary small in a smooth stencil; thus, it converges to the linear weight. Jiang and Shu proposed a way to measure the smoothness of the profile based on its second derivatives. In the fifth-order WENO-JS method, the smoothness indicators β_m

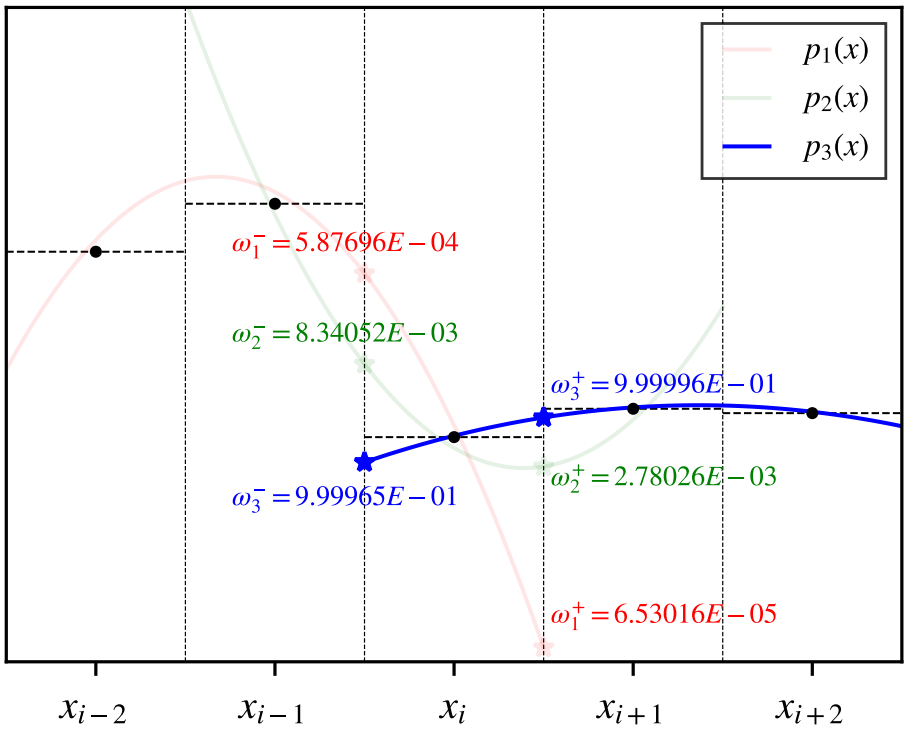

Figure 3.2: The reconstructed fifth-order WENO profiles of the same data stencil in Fig. 3.1, combined with WENO-JS nonlinear weights. (Eq. (3.17)) The opacity of each line measures the nonlinear weights on that sub-stencil, and calculated nonlinear weights are noted in the figure. Note that the nonlinear weights of S_3 are dominant over other stencils, S_1 and S_2, resulting in ENO-style stencil selection.

are given by,

$$\beta_m = \sum_{n=1}^{2} \left(\Delta x^{2n-1} \int_{I_i} \left[\frac{d^n p_m(x)}{dx^n} \right]^2 dx \right). \tag{3.18}$$

Fig. 3.2 shows the effects of the nonlinear weights on the WENO profiles. The nonlinear weights are calculated through Eq. (3.17), with $p = 2$, $\epsilon = 1.0 \times 10^{-36}$. As illustrated in the figure, the nonlinear weights successfully detect the sharp gradient at $x = x_{i-\frac{1}{2}}$ and weighting on $p_3(x)$ dominantly to avoid reconstruction on the discontinuity.

3.1.2 Gaussian Process Reconstruction

Over decades, the high-order data reconstruction/interpolation methods for solving hyperbolic PDEs are based on the polynomial approach. Like the WENO method discussed in the previous section, the idea starts by assuming a unique polynomial represents the stencil data. The polynomial-based approaches are the most successful and popular reconstruction/interpolation methods in the CFD community [68, 18, 36, 40] because of their mathematical simplicity.

However, the polynomial-based reconstruction/interpolation schemes have some downsides. Firstly, since the polynomials are not able to represent the discontinuous data, it is notoriously prone to lead numerical oscillations. There are several ways to avoid the oscillations, like the WENO method in 3.1.1, but it usually brings complexity and computational expenses. Another issue for the polynomial-based approach is that the method must be carried out on a fixed size data stencil, which means changing in stencil size – thus it changes the order of accuracy – requires a complete redesign of the code.

Recently, practitioners have designed *non-polynomial* reconstruction/interpolation methods. Reyes et al. [53, 54] proposed a novel way to use the Gaussian process (GP) to estimate the data at any arbitrary point in high-order accuracy. Based on the stochastic process, GP reconstruction/interpolation methods are able to predict the data on desired points (usually at cell interfaces) without considering any polynomials; therefore, the code is readily extended to a higher order by simply changing the size of the stencil. In the language of GP, this process can be interpreted in the way that a probability distribution for the unknown function values $f(x_*)$ (pointwise values at arbitrary points x_*) can be trained by the known data $\overline{q}_i$ (volume-averaged quantities at cell centers), with posterior mean and uncertainty that are compatible with the known observations.

A GP is fully defined by two functions:

- a mean function $\mu_f(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ over $\mathbb{R}^N$, and
- a covariance kernel function, which is symmetric and positive-definite integral kernel $K(\mathbf{x}, \mathbf{y}) = \mathbb{E}[(f(\mathbf{x}) - \mu_f(\mathbf{x}))(f(\mathbf{y}) - \mu_f(\mathbf{y}))]$ over $\mathbb{R}^N \times \mathbb{R}^N$,

The function f is then said to belong to the GP with mean and covariance function, written as $f \sim GP(\mu_f(\mathbf{x}), K(\mathbf{x}, \mathbf{y}))$.

Suppose there are N sample points for the function f, namely, $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]$ that are known, the likelihood $\mathcal{L}$, the probability of $\mathbf{f}$ given the prior GP model, of the input data $\mathbf{f}$, is given by,

$$\mathcal{L} = P(\mathbf{f}) \equiv (2\pi)^{-\frac{N}{2}} \det|\mathbf{K}|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\mathbf{f} - \boldsymbol{\mu}_\mathbf{f})^T \mathbf{K} (\mathbf{f} - \boldsymbol{\mu}_\mathbf{f})\right], \tag{3.19}$$

where $\mathbf{K} = [K_{ij}]_{i,j=1,\ldots,N}$ with $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.

The goal of GP is to make a probabilistic statement about the value of $f_* = f(\mathbf{x}_*)$ of unknown function $f \sim GP(\mu_f, K)$ with given function samples. By utilizing the conditioning property of GP from the theory of Bayesian inference, the updated posterior mean function can be obtained as,

$$\tilde{f}_* \equiv \mu_f(\mathbf{x}_*) + \mathbf{k}_*^T \mathbf{K}^{-1}(\mathbf{f} - \boldsymbol{\mu}_\mathbf{f}), \tag{3.20}$$

where $\mathbf{k}_* = [k_{*,i}]_{i=1,\ldots,N}$ with $k_{*,i} = K(\mathbf{x}_*, \mathbf{x}_i)$. The detailed derivations of Eq. (3.20) can be found in the Appendix of [53]. It is common practice to take the zero mean everywhere, then Eq. (3.20) becomes

$$\tilde{f}_* = \mathbf{k}_*^T \mathbf{K}^{-1} \mathbf{f}, \tag{3.21}$$

reveals GP prediction with known function samples $\mathbf{f}$, which is the *pointwise*

representation.

However, in the reconstruction scheme, the known function samples should be given as *volume averaged* quantities, not as pointwise values. Therefore, the GP prediction has to be modified to adapt to the data type changes.

Favorably, the volume averaging operator constitutes a linear operation (see Eq. (3.10)). The linear operations on Gaussian random variables result in new Gaussian random variables with linearly transformed mean and covariance; thus, the GP *interpolation* scheme (Eq. (3.21)) can be transformed into the GP *reconstruction* scheme with proper linear functionals that represent volume-averaging.

Consider a measure $dg_k(\mathbf{x})$ on the function $f(\mathbf{x})$ over the cell $I_k = \prod_{d=x,y,z} I_k^{(d)}$ with 1D cells $I_k^{(d)} = \left[x_k^{(d)} - \frac{\Delta^{(d)}}{2},\ x_k^{(d)} + \frac{\Delta^{(d)}}{2}\right]$, where $d = x, y, z$ represent the direction of the spatial dimension. This defines the linear functionals

$$G_k \equiv \int f(\mathbf{x})\, dg_k(\mathbf{x}), \tag{3.22}$$

which represent the volume integral operations on $f(\mathbf{x})$. The measure $dg_k(\mathbf{x})$ are taken to be the cell volume-average measures as,

$$dg_k(\mathbf{x}) = \begin{cases} d^3\mathbf{x} \cdot \prod\limits_{d=x,y,z} \frac{1}{\Delta^{(d)}} & \text{if } \mathbf{x} \in I_k \\ 0 & \text{otherwise,} \end{cases} \tag{3.23}$$

where $\Delta^{(d)}$ is the grid spacing in the d-direction. Then, the vector $\mathbf{G}\,[G_1, \ldots, G_N]^T$ is normally distributed with mean $\mathbb{E}(\mathbf{G}) = \boldsymbol{\mu}_{\mathbf{G}} = [\mu_{G_1}, \ldots, \mu_{G_N}]^T$ and covariance matrix $\mathbf{C} = [C_{kh}]_{k,h=1,\ldots,N}$, where

$$\mu_{G_k} = \mathbb{E}[G_k] = \int \mathbb{E}\left[f(\mathbf{x})\right] dg_k(\mathbf{x}) = \int \mu_f(\mathbf{x})\, dg_k(\mathbf{x}), \tag{3.24}$$

and

$$\begin{aligned} C_{kh} &= \mathbb{E}\left[\left(G_k - \mu_{G_k}\right)\left(G_h - \mu_{G_h}\right)\right] \\ &= \int\int \mathbb{E}\left[\left(f(\mathbf{x}) - \mu_f(\mathbf{x})\right)\left(f(\mathbf{y}) - \mu_f(\mathbf{y})\right)\right] dg_k(\mathbf{x})\, dg_h(\mathbf{y}) \\ &= \int\int K(\mathbf{x},\mathbf{y})\, dg_k(\mathbf{x})\, dg_h(\mathbf{y})\,. \end{aligned} \tag{3.25}$$

Thus, the GP distribution on the function $f \sim GP(\mu, K)$ conducts a multivariate Gaussian distribution on N-dimensional vector $\mathbf{G}$ of linear functionals of f.

In order to generalize Eq. (3.21) for reconstruction, the remaining task is to define the prediction vector $\mathbf{T}_* = [\mathbf{T}_{*,k}]_{k=1,\ldots,N}$ at any arbitrary point of interest $\mathbf{x}_*$ as,

$$\begin{aligned} T_{*,k} &= \mathbb{E}\left[\left(f(\mathbf{x}_*) - \mu_f(\mathbf{x}_*)\right)\left(G_k - \mu_{G_k}\right)\right] \\ &= \int K(\mathbf{x}_*,\mathbf{x})\, dg_k(\mathbf{x})\,. \end{aligned} \tag{3.26}$$

Finally, the pointwise estimation of $f(\mathbf{x}_*)$ at the point of $\mathbf{x}_*$, reconstructed from the volume-averaged data $\mathbf{G}$ is given by,

$$\tilde{f}_* = \mathbf{T}_*^T \mathbf{C}^{-1} \mathbf{G}, \tag{3.27}$$

with zero mean values. Eq. (3.27) shows the explicit form of GP reconstruction with known volume-averaged data points, $\mathbf{G}$. The terms $\mathbf{C}$ and $\mathbf{T}$ are determined by the choice of the covariance kernel function, $K(\mathbf{x},\mathbf{y})$, which measures the relationship between pairs of data. In this dissertation, the "Squared Exponential" (SE) kernel used for GP reconstruction.

$$K_{\mathrm{SE}}(\mathbf{x},\mathbf{y}) = \Sigma^2 \exp\left[-\frac{(\mathbf{x}-\mathbf{y})^2}{2\ell^2}\right]. \tag{3.28}$$

The SE kernel has two hyperparameters Σ and ℓ, but the hyperparameter Σ has no effect on the posterior mean function, so this dissertation set $\Sigma = 1$ for simplicity. On the other hand, the hyperparameter ℓ expresses the correlation length scale of the model, so it should be chosen meticulously corresponding to the physical length scale of the grid configuration.

For 1D reconstructions, $T_{*,k}$ and C_{kh} for SE kernel becomes,

$$T_{*,k} = \sqrt{\frac{\pi}{2}} \frac{\ell}{\Delta} \left\{ \operatorname{erf}\left[\frac{\Delta_{k*} + 1/2}{\sqrt{2}\ell/\Delta} \right] - \operatorname{erf}\left[\frac{\Delta_{k*} - 1/2}{\sqrt{2}\ell/\Delta} \right] \right\}, \tag{3.29}$$

and

$$\begin{aligned} C_{kh} = \sqrt{\pi} \left(\frac{\ell}{\Delta} \right)^2 \Bigg\{ & \left(\frac{\Delta_{kh}+1}{\sqrt{2}\ell/\Delta} \operatorname{erf}\left[\frac{\Delta_{kh}+1}{\sqrt{2}\ell/\Delta} \right] + \frac{\Delta_{kh}-1}{\sqrt{2}\ell/\Delta} \operatorname{erf}\left[\frac{\Delta_{kh}-1}{\sqrt{2}\ell/\Delta} \right] \right) \\ & + \frac{1}{\sqrt{\pi}} \left(\exp\left[-\frac{(\Delta_{kh}+1)^2}{2(\ell/\Delta)^2} \right] + \exp\left[-\frac{(\Delta_{kh}-1)^2}{2(\ell/\Delta)^2} \right] \right) \\ & -2 \left(\frac{\Delta_{kh}}{\sqrt{2}\ell/\Delta} \operatorname{erf}\left[\frac{\Delta_{kh}}{\sqrt{2}\ell/\Delta} \right] + \frac{1}{\sqrt{\pi}} \exp\left[-\frac{\Delta_{kh}^2}{2(\ell/\Delta)^2} \right] \right) \Bigg\}, \end{aligned} \tag{3.30}$$

where $\Delta_{kh} = (x_k - x_h)/\Delta$ and Δ is the grid spacing along the 1D direction. Note that the analytic derivations of $T_{*,k}$ and C_{kh} above only depend on the grid spacing and the length between the prediction point $\mathbf{x}_*$ and the locations of known training data. With uniform grid configuration, those values are established at the initial step. Since the prediction points are the cell interfaces $x_{i\pm\frac{1}{2}}$ for the conventional FDM constructions, one can save,

$$\mathbf{z}_{i\pm\frac{1}{2}} \coloneqq \mathbf{T}_{i\pm\frac{1}{2}}^T \mathbf{C}^{-1}, \tag{3.31}$$

as the weighting factor for the reconstruction can be expressed as,

$$q_{i\pm\frac{1}{2}} = \mathbf{z}_{i\pm\frac{1}{2}}^T \mathbf{G}, \tag{3.32}$$

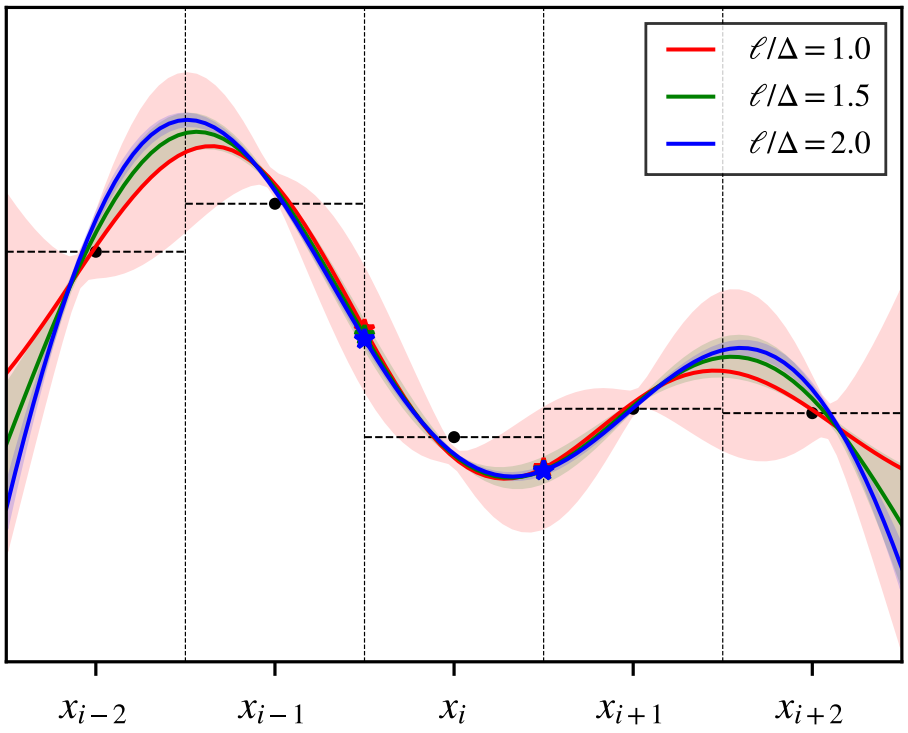

Figure 3.3: GP reconstructed profiles with the same data as Figs. 3.1 and 3.2 with different hyperparameters $\ell = 1.0, 1.5, 3.0$. Note that all the reconstructed profiles produce a new local minimum near $x = x_{i+\frac{1}{2}}$, which violates the monotonic-preserving condition and leads to numerical oscillations. The shaded areas represent 95% confidence regions from the posterior variance.

for computational efficiency.

However, the GP reconstruction also needs special handling for the discontinuous profiles, like the nonlinear weightings in the WENO method in Section 3.1.1. Fig. 3.3 shows the GP reconstructions with the same data as Figs. 3.1 and 3.2, with $\ell = 1, 1.5, 3$. The initial profile has a strong gradient at $x = x_{i-\frac{1}{2}}$; hence, the GP reconstructed profiles have undershot values at $x = x_{i+\frac{1}{2}}$.

Reyes et al. [54] proposed a WENO-*like* approach to the GP reconstruction/interpolations by considering the GP marginal likelihood of the local stencil data for measuring the smoothness of the stencil, and use them to construct nonlinear weights as like in the standard WENO method.

For five points, fifth-order GP reconstruction, suppose three sub-stencils as like fifth-order WENO method. (Eq. (3.11)) Then there are three reconstructed pointwise values for each of the candidate stencils S_m,

$$q_{i\pm\frac{1}{2},m} = \mathbf{z}^T_{i\pm\frac{1}{2},m}\mathbf{G}_m. \tag{3.33}$$

The final reconstructed value is taken as the combinations of three candidate GP approximations with nonlinear weights,

$$q_{i\pm\frac{1}{2}} = \sum_{m=1}^{3} \omega_m q_{i\pm\frac{1}{2},m}. \tag{3.34}$$

As in the conventional WENO method, the nonlinear weights ω_m must be reduced to the optimal (linear) weights γ_m in a smooth region, so the weighted combination Eq. (3.34) converges to the GP approximation over the whole stencil $S = \bigcup_m S_m$. The optimal weights γ_m must satisfy,

$$\mathbf{z}^T_{i\pm\frac{1}{2}}\mathbf{G} = q_{i\pm\frac{1}{2}} = \sum_{m=1}^{3} \gamma_m q_{i\pm\frac{1}{2},m} = \sum_{m=1}^{3} \gamma_m \mathbf{z}^T_{i\pm\frac{1}{2},m}\mathbf{G}_m, \tag{3.35}$$

or explicitly,

$$\gamma_1 \left.\begin{bmatrix} \mathrm{z}^{\pm}_{1,1} \\ \mathrm{z}^{\pm}_{2,1} \\ \mathrm{z}^{\pm}_{3,1} \\ 0 \\ 0 \end{bmatrix}\right\} \mathbf{z}_{i\pm\frac{1}{2},1} + \gamma_2 \left.\begin{bmatrix} 0 \\ \mathrm{z}^{\pm}_{1,2} \\ \mathrm{z}^{\pm}_{2,2} \\ \mathrm{z}^{\pm}_{3,2} \\ 0 \end{bmatrix}\right\} \mathbf{z}_{i\pm\frac{1}{2},2} + \gamma_3 \left.\begin{bmatrix} 0 \\ 0 \\ \mathrm{z}^{\pm}_{1,3} \\ \mathrm{z}^{\pm}_{2,3} \\ \mathrm{z}^{\pm}_{3,3} \end{bmatrix}\right\} \mathbf{z}_{i\pm\frac{1}{2},3} = \left.\begin{bmatrix} \mathrm{z}^{\pm}_{1} \\ \mathrm{z}^{\pm}_{2} \\ \mathrm{z}^{\pm}_{3} \\ \mathrm{z}^{\pm}_{4} \\ \mathrm{z}^{\pm}_{5} \end{bmatrix}\right\} \mathbf{z}_{i\pm\frac{1}{2}} \ . \tag{3.36}$$

The Eq. (3.36) can be rewritten in the matrix form of overdetermined system as,

$$\begin{bmatrix} \mathrm{z}_{1,1}^{\pm} & 0 & 0 \\ \mathrm{z}_{2,1}^{\pm} & \mathrm{z}_{1,2}^{\pm} & 0 \\ \mathrm{z}_{3,1}^{\pm} & \mathrm{z}_{2,2}^{\pm} & \mathrm{z}_{1,3}^{\pm} \\ 0 & \mathrm{z}_{3,2}^{\pm} & \mathrm{z}_{2,3}^{\pm} \\ 0 & 0 & \mathrm{z}_{3,3}^{\pm} \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix} = \begin{bmatrix} \mathrm{z}_{1}^{\pm} \\ \mathrm{z}_{2}^{\pm} \\ \mathrm{z}_{3}^{\pm} \\ \mathrm{z}_{4}^{\pm} \\ \mathrm{z}_{5}^{\pm} \end{bmatrix}. \tag{3.37}$$

Thus, the optimal weights $\gamma_m, m = 1, 2, 3$ are obtained by solving the overdetermined system, Eq. (3.37). It should be noted that the optimal weights are entirely determined by the choice of kernel function and the stencil size, so they are computed and stored before the simulation begins.

For constructing WENO nonlinear weights Eq. (3.17), the only remaining task is to determine the smoothness indicator, β_m, which measures the degree of smoothness of a given stencil of data. Unlike the conventional WENO method, which measures the smoothness of the data by calculating L_2 norms of all the derivatives of the reconstructed polynomials, the GP reconstruction method should take a different approach to specify the smoothness indicator because there is no polynomial defined in GP. One successful practice is to use the marginal likelihood of the data. The likelihood function is well-furnished to gauge the deviations from smoothness given a sufficiently smooth covariance kernel function, SE kernel, for example. Thus, the GP predictions have smaller likelihoods to non-smooth function by its design.

Suppose the negative log of the GP marginal likelihood as

$$-\log \mathcal{L} = \frac{N}{2} \log [2\pi] + \frac{1}{2} \log |\det \mathbf{K}_m| + \frac{1}{2} \left(\mathbf{f}_m - \boldsymbol{\mu}_{\mathbf{f}}\right)^T \mathbf{K}_m^{-1} \left(\mathbf{f}_m - \boldsymbol{\mu}_{\mathbf{f}}\right), \tag{3.38}$$

then the three terms on the right-hand side of Eq. (3.38) are revealed as a normal-

ization, a complexity penalty, and a data fit term, respectively. The normalization term and the complexity penalty have no effects on defining smoothness indicators in a uniform grid configuration; only the data fit term along with the choice of zero mean is used for constructing smoothness indicators in GP,

$$\beta_m = \mathbf{f}_m^T \left(\mathbf{K}_m^{-1} \right) \mathbf{f}_m. \tag{3.39}$$

To handle discontinuity, a second hyperparameter, σ, should be used in Eq. (3.39), to discriminate discontinuities from smooth regions. σ should be on the order of the grid spacing.

Like GP reconstruction weights Eq. (3.32), the calculations of GP smoothness indicators β_m can be expressed in a more computationally efficient form. Considering the eigensystem $\mathbf{K}_m^{-1} = \sum_i \mathbf{v}_i^m \left(\mathbf{v}_i^m\right)^T / \lambda_i^m$,

$$\beta_m = \sum_{i=1}^{3} \mathbf{f}_m^T \left(\frac{\mathbf{v}_i^m \left(\mathbf{v}_i^m\right)^T}{\lambda_i} \right) \mathbf{f}_m, \tag{3.40}$$

in the five-point, three-stencil GP-WENO method. Again, the observed (training) data is in the volume-averaged form; thus, the data-type conversion should be examined:

$$\mathbf{f}_m = \mathbf{Z}_m^T \mathbf{G}_m, \tag{3.41}$$

where each column vector of $\mathbf{Z}_m$ is given by the GP reconstructing weight, $\mathbf{z}$, (see Eq. (3.31)) for each elements of $\mathbf{f}_m$.

Lastly, the calculation of the smoothness indicator beta can be expressed in

the compact form as,

$$\begin{aligned}\beta_m &= \sum_{i=1}^{3} \left(\frac{\left(\mathbf{v}_i^m\right)^T \mathbf{Z}_m^T \mathbf{G}_m}{\sqrt{\lambda_i^m}} \right)^2 \\ &= \sum_{i=1}^{3} \left(\mathbf{P}_i^m \mathbf{G}_m\right)^2,\end{aligned} \tag{3.42}$$

where $\mathbf{P}_i^m \coloneqq \frac{\left(\mathbf{v}_i^m\right)^T \mathbf{Z}_m^T}{\sqrt{\lambda_i^m}}$, which can be established before the simulation start, in uniform grid configuration.

Now, the nonlinear weights for GP-WENO reconstruction are fully determined with Eq. (3.17). In the stepwise representation, the GP-WENO reconstruction scheme for FDM formulation in the unifrom grid can be summarized as below:

1. Before the simulation starts, calculate the following values and store them for later reconstructions:
   (a) Reconstruction weights, $\mathbf{z}_m$ (Eq. (3.33)) for each candidate stencil, m.
   (b) Linear weights, γ_m, by the solving overdetermined system, Eq. (3.37), using the least square method.
   (c) $\mathbf{P}_i^m$ (Eq. (3.42)) for calculating smoothness indicator β_m in later.
2. During the simulation, at each reconstruction step of cell I_i:
   (a) Calculate nonlinear weights, ω_m, following the conventional WENO method. (Eq. (3.17))
   (b) Compute m-number of reconstructed candidate data, $q_{i\pm\frac{1}{2},m}$. (Eq. (3.33))
   (c) Taking weighted combinations of $q_{i\pm\frac{1}{2},m}$, with nonlinear weights, ω_m, and finalize the reconstruction step at $x_i = x_{i\pm\frac{1}{2}}$. (Eq. (3.34))

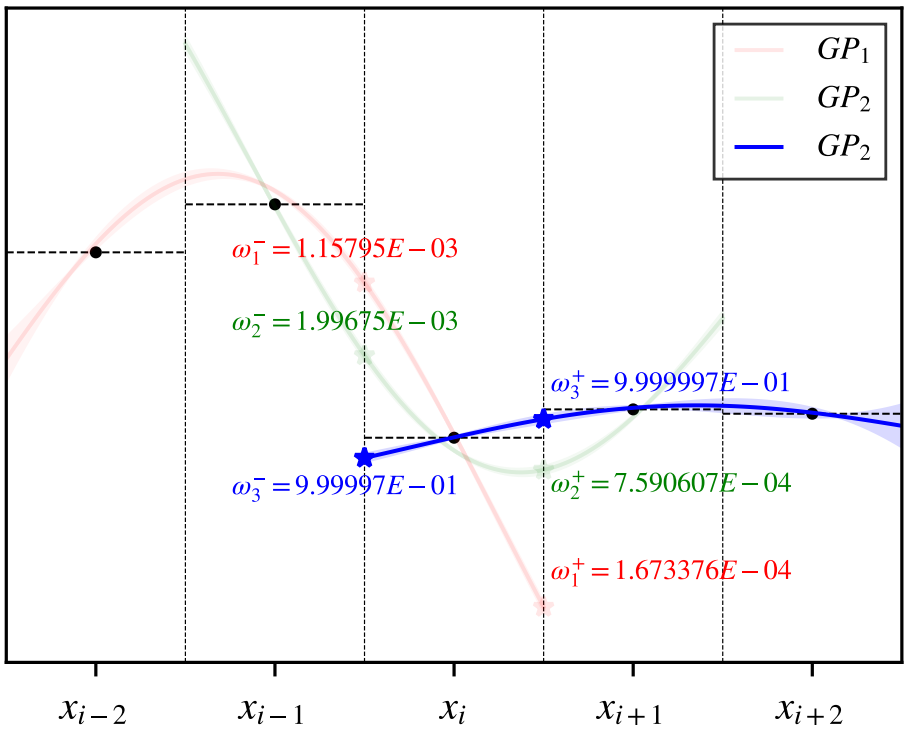

Figure 3.4: GP-WENO Reconstructed profiles with nonlinear weights with the same data as Fig. 3.3. Hyperparameters $\ell = 2\Delta$ and $\sigma = 2\Delta$ are used. 95% confidence regions are shaded with the correspoinding colors, and the opacity of each prediction measures the nonlinear weights on that sub-stencil.

3.2 High-Order Time Integration Schemes

A high-order temporal discretization scheme ought to be considered alongside the high-order spatial data interpolation/reconstruction to acquire highly accurate numerical solutions in FDM formulation. The numerical errors arise from both spatial and temporal discretizations since the solution of the conservative PDEs lies on the spatio-temporal plane. The overall order of solution accuracy will be determined by the highest order term of the truncation errors both from the spatial and temporal discretization, i.e., $\mathcal{O}(\Delta s^p, \Delta t^q)$. For example, if the leading error term from the temporal discretization is significantly larger than the leading error term from the spatial discretizations, $\mathcal{O}(\Delta t^q) > \mathcal{O}(\Delta s^p)$, then the solution accuracy will be degraded by order of temporal accuracy, q, no matter the order of spatial accuracy is used. Therefore, a high-order FDM scheme requires a meticulously designed temporal discretization method that ensures the solution's accuracy and stability.

3.2.1 Strong Stability Preserving Runge-Kutta Methods

The Runge-Kutta (RK) method is the most popular way to integrate the semi-discretized form of FDM (Eq. (2.11)) over the time axis. The key idea is to treat the Eq. (2.11) as an ordinary differential equation (ODE) at each lattice point on the computational mesh as,

$$\partial_t \mathbf{U}_i = \mathcal{L}_i(\mathbf{U}), \tag{3.43}$$

where the right-hand side operator $\mathcal{L}_i$ is given by the spatial discretization method at cell I_i. The RK approach can be viewed as a strategy to integrate PDEs by solving several ODE problems at each discretized time domain. In general, m-stage RK method which integrate Eq. (3.43) from $t = t^n$ to $t = t^n + \Delta t = t^{n+1}$ is

given by,

$$\begin{aligned}
\mathbf{U}^{(0)} &= \mathbf{U}^n, \\
\mathbf{U}^{(l)} &= \sum_{k=0}^{l-1} \left(\alpha_{l,k} \mathbf{U}^{(k)} + \Delta t \beta_{l,k} \mathcal{L}(\mathbf{U}^{(k)}) \right), \quad \alpha_{l,k} \geq 0, \quad l = 1, \ldots, m \\
\mathbf{U}^{n+1} &= \mathbf{U}^{(m)},
\end{aligned} \tag{3.44}$$

where the spatial discretization index, i, is omitted for simplicity. Therefore, the coefficients $\alpha_{l,k}$ and $\beta_{l,k}$ fully determine the numerical accuracy and stability of the RK scheme.

Like the spatial discretization method, it is crucial to consider the TVD property of the temporal discretization scheme. Gottlieb and her collaborators [31, 32, 30] developed the so-called strong stability preserving Runge-Kutta (SSP-RK) method, which ensures TVD property by sequentially applying convex combinations of the first-order forward Euler method as a building-block at each sub-stage. In this way, the desired TVD property is achieved if each of the sub-stage is TVD.

SSP-RK method uses that the forward Euler method for building each sub-stages. Since the forward Euler method is strongly stable under the Courant–Friedrichs–Lewy (CFL) condition; thus, if all sub-stages of the RK method can be described as a form of the forward Euler method, then the TVD property is fulfilled by virtue of the forward Euler method. It is easy to see in Eq. (3.44) that the sub-stages of the RK method, $\mathbf{U}^{(l)}$ can be described as the forward Euler method if all the $\beta_{l,k}$ are nonnegative $\beta_{l,k} \geq 0$ by replacing Δt by $\frac{\beta_{l,k}}{\alpha_{l,k}} \Delta t$. For example, in [31], Gottlieb

found the optimal third-order SSP-RK method given by,

$$\begin{aligned}
\mathbf{U}^{(1)} &= \mathbf{U}^n + \Delta t \mathcal{L}(\mathbf{U}^n), \\
\mathbf{U}^{(2)} &= \frac{3}{4}\mathbf{U}^n + \frac{1}{4}\mathbf{U}^{(1)} + \frac{1}{4}\Delta t \mathcal{L}(\mathbf{U}^{(1)}), \\
\mathbf{U}^{n+1} &= \frac{1}{3}\mathbf{U}^n + \frac{2}{3}\mathbf{U}^{(2)} + \frac{2}{3}\Delta t \mathcal{L}(\mathbf{U}^{(2)}),
\end{aligned} \tag{3.45}$$

which requires three sub-stages for integrating from $t = t^n$ to $t = t^{n+1}$.

The above third-order, three-stages SSP-RK method Eq. (3.45) is by far the most famous high-order time integrator used in most high-order method researches in the CFD community. In practice, spatial accuracy is often considered to carry more weight than temporal accuracy in designing higher accurate spatial models [8, 46], so combining the high-order spatial method (generally, fifth-order or higher) with a relatively low-order temporal scheme (third-order SSP-RK) could be justifiable. However, as it is shown in [42], the order of convergence rate of the solution can be degraded by the order of temporal method (e.g., third-order) in high-resolution computational grids, so using a higher than third-order temporal scheme is required for maintaining desired spatial accuracy in a high-resolution grid.

In contrast to the third-order SSP-RK3 method, devising a fourth-order SSP-RK4 is more involved to meet the favorable SSP property, which is ensured by positive coefficients. Several theoretical studies have shown that a fourth-order SSP-RK4 cannot be formulated with just four sub-stages and positive coefficients [31], meaning that the classical four-stage, fourth-order RK is not SSP. For example,

by far the most optimal fourth-order, fourth-stage SSP-RK method is:

$$
\begin{aligned}
\mathbf{U}^{(1)} &= \mathbf{U}^n + \frac{1}{2}\Delta t\mathcal{L}(\mathbf{U}^n), \\
\mathbf{U}^{(2)} &= \frac{649}{1600}\mathbf{U}^{(0)} - \frac{10890423}{25193600}\Delta t\tilde{\mathcal{L}}(\mathbf{U}^n) + \frac{951}{1600}\mathbf{U}^{(1)} + \frac{5000}{7873}\Delta t\mathcal{L}(\mathbf{U}^{(1)}), \\
\mathbf{U}^{(3)} &= \frac{53989}{2500000}\mathbf{U}^n - \frac{102261}{5000000}\Delta t\tilde{\mathcal{L}}(\mathbf{U}^n) + \frac{4806213}{20000000}\mathbf{U}^{(1)} \\
&\quad - \frac{5121}{20000}\Delta t\tilde{\mathcal{L}}(\mathbf{U}^{(1)}) + \frac{23619}{32000}\mathbf{U}^{(2)} + \frac{7873}{10000}\Delta t\mathcal{L}(\mathbf{U}^{(2)}), \\
\mathbf{U}^{(4)} &= \frac{1}{5}\mathbf{U}^n + \frac{1}{10}\Delta t\mathcal{L}(\mathbf{U}^n) + \frac{6127}{30000}\mathbf{U}^{(1)} + \frac{1}{6}\Delta t\mathcal{L}(\mathbf{U}^{(1)}) + \frac{7873}{30000}\mathbf{U}^{(2)} \\
&\quad + \frac{1}{3}\mathbf{U}^{(3)} + \frac{1}{6}\Delta t\mathcal{L}(\mathbf{U}^{(3)}).
\end{aligned}
\tag{3.46}
$$

Note that the above four-stage, fourth-order SSP-RK method uses the adjoint spatial operator, $\tilde{\mathcal{L}}$, to bear the negative coefficients, e.g., $-\frac{10890423}{25193600}$ and $-\frac{102261}{5000000}$. Numerically speaking, the only difference between $\mathcal{L}$ and $\tilde{\mathcal{L}}$ is the direction of the upwind limiting. Although the computational cost of calculating $\mathcal{L}(\mathbf{U})$ and $\tilde{\mathcal{L}}(\mathbf{U})$ are identical, but it demands separated codes, and leads implementation complexity.

Spiteri and Ruuth [61] proposed a five-stage, fourth-order SSP-RK4 method that does not require to use adjoint operator:

$$
\begin{aligned}
\mathbf{U}^{(1)} &= \mathbf{U}^n + 0.391752226571890\Delta t\mathcal{L}(\mathbf{U}^n), \\
\mathbf{U}^{(2)} &= 0.444370493651235\mathbf{U}^n + 0.555629506348765\mathbf{U}^{(1)} + 0.368410593050371\Delta t\mathcal{L}(\mathbf{U}^{(1)}), \\
\mathbf{U}^{(3)} &= 0.620101851488403\mathbf{U}^n + 0.379898148511597\mathbf{U}^{(2)} + 0.251891774271694\Delta t\mathcal{L}(\mathbf{U}^{(2)}), \\
\mathbf{U}^{(4)} &= 0.178079954393132\mathbf{U}^n + 0.821920045606868\mathbf{U}^{(3)} + 0.544974750228521\Delta t\mathcal{L}(\mathbf{U}^{(3)}), \\
\mathbf{U}^{n+1} &= 0.517231671970585\mathbf{U}^{(2)} + 0.096059710526147\mathbf{U}^{(3)} + 0.063692468666290\Delta t\mathcal{L}(\mathbf{U}^{(3)}) \\
&\quad + 0.386708617503268\mathbf{U}^{(4)} + 0.226007483236906\Delta t\mathcal{L}(\mathbf{U}^{(4)}).
\end{aligned}
\tag{3.47}
$$

In this dissertation, the fourth-order SSP-RK4 method refers the Spiteri and Ruuth method, Eq. (3.47).

In many studies about high-order CFD solvers [31, 32, 30, 46, 20, 21, 53, 54], the SSP-RK schemes have proven high fidelity and portability, guaranteeing high-order accuracy and numerical stability with TVD property. However, the very nature of the SSP-RK method – being a multi-stage approach – increases computational costs in CFD simulations. In SSP-RK methods, the data reconstruction/interpolation (i.e., $\mathcal{L}(\cdot)$) and the boundary condition should be applied in each sub-stage, which increases the computational resources 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.

3.2.2 Lax–Wendroff Type Methods

The Lax-Wendroff method [39] rely on the Taylor expansion in time to achieve high-order in time accuracy:

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t\, \partial_t \mathbf{U}\big|^n + \frac{\Delta t^2}{2!}\, \partial_t^2 \mathbf{U}\Big|^n + \mathcal{O}(\Delta t^3). \tag{3.48}$$

The temporal derivatives can be transformed into spatial derivatives by applying the Lax-Wendroff or Cauchy-Kowalewski procedure (LW/CK hereafter). In one-

dimensional conservative PDEs, for example,

$$\begin{aligned}
\partial_t \mathbf{U} &= -\partial_x \mathbf{F}, \\
\partial_t^2 \mathbf{U} &= \partial_t \left(-\partial_x \mathbf{F} \right) \\
&= -\partial_x \left(\partial_{\mathbf{U}} \mathbf{F} \cdot \partial_t \mathbf{U} \right) \\
&= \partial_x \left(\partial_{\mathbf{U}} \mathbf{F} \cdot \partial_x \mathbf{F} \right),
\end{aligned} \tag{3.49}$$

where $\partial_{\mathbf{U}} \mathbf{F}$ is a flux Jacobian matrix. In the original Lax-Wendroff method [39] used an approximation of $\partial_x \left(\partial_{\mathbf{U}} \mathbf{F} \cdot \partial_x \mathbf{F} \right) \approx \frac{1}{\Delta x} \left(\partial_{\mathbf{U}} \mathbf{F}|_{i+\frac{1}{2}} \cdot \partial_x \mathbf{F} - \partial_{\mathbf{U}} \mathbf{F}|_{i-\frac{1}{2}} \cdot \partial_x \mathbf{F} \right)$, but it is possible to get an explicit form as,

$$\partial_t^2 \mathbf{U} = \partial_x \left(\partial_{\mathbf{U}} \mathbf{F} \cdot \partial_x \mathbf{F} \right) = \partial_{\mathbf{U}}^2 \mathbf{F} \cdot \partial_x \mathbf{F} + \partial_{\mathbf{U}} \mathbf{F} \cdot \partial_x^2 \mathbf{F}, \tag{3.50}$$

with flux Hessian tensor, $\partial_{\mathbf{U}}^2 \mathbf{F}$.

The primary advantage of the Lax-Wendroff method is that it can achieve a high order in time accuracy within a single step of the calculation. The idea to construct the time-Taylor series by harnessing the tight coupling of temporal and spatial derivatives through LW/CK procedures inspires many practitioners to develop single-step, high-order methods based on the Lax-Wendroff method.

In 2001, Toro et al. [65] 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 constructed Riemann problems for each spatial derivative at cell interface, $x_{i+\frac{1}{2}}$:

$$\begin{aligned}
&\partial_t \mathbf{U}_x^{(k)} + \partial_{\mathbf{U}} \mathbf{F}|_{i+\frac{1}{2}} \, \partial_x \mathbf{U}_x^{(k)} = 0, \quad \text{where } \partial_{\mathbf{U}} \mathbf{F}|_{i+\frac{1}{2}} = \partial_{\mathbf{U}} \mathbf{F}(\mathbf{U}_{i+\frac{1}{2}}, 0^+), \\
&\mathbf{U}_x^{(k)}(x,0) = \begin{cases} \frac{\partial^k}{\partial x^k} \mathbf{U}_{i+\frac{1}{2},L}, & x < x_{i+\frac{1}{2}}, \\ \frac{\partial^k}{\partial x^k} \mathbf{U}_{i+\frac{1}{2},R}, & x > x_{i+\frac{1}{2}}, \end{cases}
\end{aligned} \tag{3.51}$$

where $\mathbf{U}_x^{(k)} = \frac{\partial^k \mathbf{U}}{\partial x^k}$, and $\mathbf{U}_{i+\frac{1}{2},LR}$ represent left and right Riemann states at cell interface, $x_{i+\frac{1}{2}}$. Reconstructed profiles can find the Riemann states of the spatial derivatives. The resulting solutions of the above Riemann problems are then applied for LW/CK procedures to get the temporal derivatives, $\frac{\partial^k \mathbf{U}}{\partial t^k}\Big|_{i+\frac{1}{2}}$, and they are used to construct the time-Taylor series of the conservative variables at cell interfaces:

$$\mathbf{U}(x_{i+\frac{1}{2}}, \tau) = \mathbf{U}(x_{i+\frac{1}{2}}, 0^+) + \sum_{k=1}^{r-1} \left[\frac{\partial^k}{\partial t^k} \mathbf{U}(x,t)(x_{i+\frac{1}{2}}, 0^+) \right] \frac{\tau^k}{k!}. \tag{3.52}$$

ADER methods were further developed in [65, 62, 64], and it has grown its popularity over decades, leading to various further modifications. ADER-DG [25, 70] and ADER-CG [7, 6, 4] 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 [47], its extensions to second-order schemes for nonlinear systems [49] and to general hyperbolic systems [66]. The use of an implicit time Taylor series expansion for GRP was further simplified in the study by Montecinos and Balsara [48]. Along the line of simplifying the standard ADER approach, the Differential Transform Method (DTM) [15] was also adopted to alleviate the cost of the ADER scheme, coined as ADER-DT (or ADER-Taylor) in [52, 50, 51].

In general, Lax-Wendroff type methods are able to update the solution in single-step with high-order temporal accuracy. The fundamental advantage of being a single-stage method is the enhanced performance. 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.

Chapter 4

System-Free Picard Integral Formulation

Unlike the broad usage of SSP-RK in various discrete PDE solvers, the developments of ADER mentioned above have been exclusively applied to FV and DG methods, but FD methods. This is mainly because the fundamental principle of obtaining high-order accuracy in the original ADER scheme relies on solving generalized (or high-order) Riemann problems, which are the characteristic building blocks of FV and DG methods.

Recently, Christlieb et al. introduced a new high-order temporal scheme for FDM, the so-called Picard integral formulation (PIF) method. [17, 55] The PIF discretization is based on the constructions of high-order approximation to the time-averaged fluxes over $[t^n, t^{n+1}]$, allowing high-order temporal accuracy in a single-step update. Firstly introduced in [17], the PIF method demonstrated third-order temporally accurate numerical fluxes by computing the coefficients of the time-Taylor expansion of the averaged fluxes via LW/CK procedure which converts the high-order temporal derivatives terms into the spatial derivatives.

However, like many Lax-Wendroff type methods, the PIF method requires

finding analytic derivations for flux Jacobians and Hessians. Although the Jacobian and the Hessian calculations can be easily obtained with the aid of symbolic manipulators such as `SymPy`, `Mathematica`, or `Maple`, it still demands complicated coding/debugging efforts and ample memory consumption. Furthermore, as the Jacobian/Hessian calculations highly depend on the type of the governing system under consideration, it is required to re-derive the Jacobian/Hessian terms analytically every time we need to solve a new system, e.g., shallow water equations or magnetohydrodynamics (MHD) equations, to name a few. In addition, the calculation complexities of the Jacobian-like terms are drastically increasing with the number of spatial dimensions and the order of accuracy.

4.1 Picard Integral Formulation

Applying the Picard integral formulation (PIF), the governing equations Eq. (2.1) can be discretized by taking a time average within a single time step Δt over an interval $[t^n, t^{n+1}]$,

$$\mathbf{U}^{n+1} = \mathbf{U}^n - \Delta t \left(\partial_x \mathbf{F}^{avg} + \partial_y \mathbf{G}^{avg} + \partial_z \mathbf{H}^{avg} \right), \tag{4.1}$$

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

$$\begin{aligned} \mathbf{F}^{avg}(\mathbf{x}) &= \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{U}(\mathbf{x}, t))\, dt, \\ \mathbf{G}^{avg}(\mathbf{x}) &= \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} \mathbf{G}(\mathbf{U}(\mathbf{x}, t))\, dt, \\ \mathbf{H}^{avg}(\mathbf{x}) &= \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} \mathbf{H}(\mathbf{U}(\mathbf{x}, t))\, dt, \end{aligned} \tag{4.2}$$

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

The goal is to express the spatial derivatives of the time-averaged fluxes

in Eq. (4.1) using highly approximated numerical fluxes $\hat{\mathbf{f}}, \hat{\mathbf{g}}$, and $\hat{\mathbf{h}}$ at cell interfaces:

$$\begin{aligned}
\partial_x \mathbf{F}^{avg}|_{\mathbf{x}=\mathbf{x}_{ijk}} &= \frac{1}{\Delta x}\left(\hat{\mathbf{f}}_{i+\frac{1}{2},j,k} - \hat{\mathbf{f}}_{i-\frac{1}{2},j,k}\right) + \mathcal{O}(\Delta x^p + \Delta t^q), \\
\partial_y \mathbf{G}^{avg}|_{\mathbf{x}=\mathbf{x}_{ijk}} &= \frac{1}{\Delta y}\left(\hat{\mathbf{g}}_{i,j+\frac{1}{2},k} - \hat{\mathbf{g}}_{i,j-\frac{1}{2},k}\right) + \mathcal{O}(\Delta y^p + \Delta t^q), \\
\partial_z \mathbf{H}^{avg}|_{\mathbf{x}=\mathbf{x}_{ijk}} &= \frac{1}{\Delta z}\left(\hat{\mathbf{h}}_{i,j,k+\frac{1}{2}} - \hat{\mathbf{h}}_{i,j,k-\frac{1}{2}}\right) + \mathcal{O}(\Delta z^p + \Delta t^q),
\end{aligned} \tag{4.3}$$

where $\mathbf{x}_{ijk} = (x_i, y_j, z_k)$ is the discretization indices.

The above equation is almost analog to Eq. (2.12), which is the conventional way to construct numerical fluxes for FDM through the high-order reconstruction schemes (Eq. (2.16)). The only difference is to take time-averaged fluxes, $\mathbf{F}^{avg}, \mathbf{G}^{avg}$, and $\mathbf{H}^{avg}$ as an input of the reconstruction scheme rather than taking pointwise fluxes. Thus Eq. (4.3) states that with highly approximated time-averaged fluxes, the resulting numerical fluxes from the conventional reconstruction schemes will be high-order in time and space.

With PIF-numerical fluxes, the governing equation can be expressed in a fully discretized form as,

$$\begin{aligned}
\mathbf{U}^{n+1}_{i,j,k} = \mathbf{U}^{n}_{i,j,k} &- \frac{\Delta t}{\Delta x}\left(\hat{\mathbf{f}}_{i+\frac{1}{2},j,k} - \hat{\mathbf{f}}_{i-\frac{1}{2},j,k}\right) \\
&- \frac{\Delta t}{\Delta y}\left(\hat{\mathbf{g}}_{i,j+\frac{1}{2},k} - \hat{\mathbf{g}}_{i,j-\frac{1}{2},k}\right) \\
&- \frac{\Delta t}{\Delta z}\left(\hat{\mathbf{h}}_{i,j,k+\frac{1}{2}} - \hat{\mathbf{h}}_{i,j,k-\frac{1}{2}}\right),
\end{aligned} \tag{4.4}$$

which requires only a single update while attaining high-order accuracy both in time and space.

It is worth remarking that the derived governing form in Eq. (4.4) for PIF is something in between those of FVM and FDM. It is different from that of FVM in that it does not carry any spatial average but the temporal average. It is also

different from that of FDM in that it does involve the temporal average in the fluxes in Eq. (4.2), to which the numerical fluxes $\hat{\mathbf{f}}, \hat{\mathbf{g}}$, and $\hat{\mathbf{h}}$ approximate.

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,

$$\begin{aligned}
\mathbf{F}^{avg}(\mathbf{x}) &= \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{x},t)\,dt \\
&= \mathbf{F}(\mathbf{x},t^n) + \frac{\Delta t}{2!}\partial_t^{(1)}\mathbf{F}(\mathbf{x},t)\bigg|_{t=t^n} + \frac{\Delta t^2}{3!}\partial_t^{(2)}\mathbf{F}(\mathbf{x},t)\bigg|_{t=t^n} + \frac{\Delta t^3}{4!}\partial_t^{(3)}\mathbf{F}(\mathbf{x},t)\bigg|_{t=t^n} + \cdots \\
&= \sum_{i=0}^{q-1}\frac{\Delta t^i}{(i+1)!}\partial_t^{(i)}\mathbf{F}(\mathbf{x},t)\bigg|_{t=t^n} + \mathcal{O}(\Delta t^q) \\
&= \mathbf{F}^{appx,q}(\mathbf{x},t^n) + \mathcal{O}(\Delta t^q).
\end{aligned} \tag{4.5}$$

The *temporally* qth-order approximated fluxes $\mathbf{F}^{appx,q}$ will be used 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,

$$\hat{\mathbf{f}}_{i+\frac{1}{2},j,k} = \mathcal{R}\left(\mathcal{FS}\left(\mathbf{F}^{appx,q}_{i-r,j},\ldots,\mathbf{F}^{appx,q}_{i+r+1,j}\right)\right) + \mathcal{O}(\Delta x^p), \tag{4.6}$$

where r represents the stencil radius required for the pth-order reconstruction method, $\mathcal{R}(\cdot)$. The details of the high-order reconstruction methods, $\mathcal{R}(\cdot)$, and the flux-splitting methods, $\mathcal{FS}(\cdot)$ are described in Chapter 3.

Therefore, the primary objective of the PIF method is to approximate the time-averaged fluxes in the desired order q, i.e., obtaining $\mathbf{F}^{appx,q}$. For instance, the fourth-order PIF method is characterized by the fourth-order approximated time-averaged flux in the x-direction, $\mathbf{F}^{appx,4}$, from the Taylor expansion of the

pointwise flux around t^n. As expressed in Eq. (4.5), the fourth-order PIF method requires

$$\mathbf{F}^{appx,4}(\mathbf{x}) = \mathbf{F}(\mathbf{x}, t^n) + \frac{\Delta t}{2!}\partial_t^{(1)}\mathbf{F}(\mathbf{x}, t)\bigg|_{t=t^n} + \frac{\Delta t^2}{3!}\partial_t^{(2)}\mathbf{F}(\mathbf{x}, t)\bigg|_{t=t^n} + \frac{\Delta t^3}{4!}\partial_t^{(3)}\mathbf{F}(\mathbf{x}, t)\bigg|_{t=t^n}. \tag{4.7}$$

The other y- and z-directional approximated fluxes, $\mathbf{G}^{appx,4}$ and $\mathbf{H}^{appx,4}$, are defined in similarly. The only remaining task for the fouth-order PIF method (PIF4) is transforming all the time derivatives in Eq. (4.7) to the corresponding spatial derivatives via LW/CK procedures; thereby we could express Eq. (4.7) in a fully explicit form.

For simplicity, the compact subscript notation of partial derivatives is adopted in the following discussions. The subscripts represent the partial derivatives, and the temporal expression of $t = t^n$ is omitted. In the compact notation, Eq. (2.1) can be rewritten as,

$$\mathbf{U}_t + \nabla \cdot \mathcal{F}(\mathbf{U}) = \mathbf{U}_t + \mathbf{F}_x + \mathbf{G}_y + \mathbf{H}_z = 0. \tag{4.8}$$

By applying the chain rule to $\mathbf{F}_t$, the evolution equation of the x-flux, $\mathbf{F}$ can be obtained as,

$$\mathbf{F}_t = \mathbf{F}_\mathbf{U}\mathbf{U}_t, \tag{4.9}$$

where $\mathbf{F}_\mathbf{U}$ is the x-directional flux Jacobian matrix. The above equation can be combined with Eq. (4.8), resulting the explicit expression for $\mathbf{F}_t$ as,

$$\mathbf{F}_t = -\mathbf{F}_\mathbf{U}\nabla^f, \quad \text{where } \nabla^f = \mathbf{F}_x + \mathbf{G}_y + \mathbf{H}_z \tag{4.10}$$

The higher-order time derivatives could be achieved by taking partial derivatives

to Eq. (4.10) recursively. As an example, the second-order term is written as

$$\mathbf{F}_{tt} = \mathbf{F}_{\mathbf{UU}} \cdot \nabla^f \cdot \nabla^f - \mathbf{F}_{\mathbf{U}} \cdot \nabla^f_t, \tag{4.11}$$

where

$$\begin{aligned} \nabla^f_t = & - \mathbf{F}_{\mathbf{UU}} \cdot \mathbf{U}_x \cdot \nabla^f - \mathbf{F}_{\mathbf{U}} \cdot \nabla^f_x \\ & - \mathbf{G}_{\mathbf{UU}} \cdot \mathbf{U}_y \cdot \nabla^f - \mathbf{G}_{\mathbf{U}} \cdot \nabla^f_y \\ & - \mathbf{H}_{\mathbf{UU}} \cdot \mathbf{U}_z \cdot \nabla^f - \mathbf{H}_{\mathbf{U}} \cdot \nabla^f_z, \end{aligned} \tag{4.12}$$

and $\mathbf{F}_{\mathbf{UU}}$ is the x-directional flux Hessian tensor. In Euler equations, the flux Hessians, $\mathbf{F}_{\mathbf{UU}}, \mathbf{G}_{\mathbf{UU}}$, and $\mathbf{H}_{\mathbf{UU}}$ are the symmetric, rank-3 tensors, so a dot product between the Hessian tensor and a vector is to be understood as a tensor contraction. Thus a double dot product between the Hessian tensor and two vectors, i.e., $\mathbf{F}_{\mathbf{UU}} \cdot (\,) \cdot (\,)$ yields a vector of the same dimension with $\mathbf{U}$.

Following the same procedure, an explicit form of the third-order time derivative of the flux can be obtained as,

$$\mathbf{F}_{ttt} = -\mathbf{F}_{\mathbf{UUU}} \cdot \nabla^f \cdot \nabla^f \cdot \nabla^f + 3\mathbf{F}_{\mathbf{UU}} \cdot \nabla^f \cdot \nabla^f_t - \mathbf{F}_{\mathbf{U}} \cdot \nabla^f_{tt}, \tag{4.13}$$

where

$$\begin{aligned} \nabla^f_{tt} = \quad & \mathbf{F}_{\mathbf{UUU}} \cdot \nabla^f \cdot \mathbf{U}_x \cdot \nabla^f + 2\mathbf{F}_{\mathbf{UU}} \cdot \nabla^f \cdot \nabla^f_x - \mathbf{F}_{\mathbf{UU}} \cdot \mathbf{U}_x \cdot \nabla^f_t - \mathbf{F}_{\mathbf{U}} \cdot \nabla^f_{tx} \\ & + \mathbf{G}_{\mathbf{UUU}} \cdot \nabla^f \cdot \mathbf{U}_y \cdot \nabla^f + 2\mathbf{G}_{\mathbf{UU}} \cdot \nabla^f \cdot \nabla^f_y - \mathbf{G}_{\mathbf{UU}} \cdot \mathbf{U}_y \cdot \nabla^f_t - \mathbf{G}_{\mathbf{U}} \cdot \nabla^f_{ty} \\ & + \mathbf{H}_{\mathbf{UUU}} \cdot \nabla^f \cdot \mathbf{U}_z \cdot \nabla^f + 2\mathbf{H}_{\mathbf{UU}} \cdot \nabla^f \cdot \nabla^f_z - \mathbf{H}_{\mathbf{UU}} \cdot \mathbf{U}_z \cdot \nabla^f_t - \mathbf{H}_{\mathbf{U}} \cdot \nabla^f_{tz}, \end{aligned} \tag{4.14}$$

and

$$\begin{aligned}\nabla^f_{tx} = \quad & \mathbf{F}_{\mathbf{UUU}} \cdot \mathbf{U}_x \cdot \nabla^f \cdot \mathbf{U}_x - \mathbf{F}_{\mathbf{UU}} \cdot \mathbf{U}_{xx} \cdot \nabla^f - 2\mathbf{F}_{\mathbf{UU}} \cdot \nabla^f_x \cdot \mathbf{U}_x - \mathbf{F}_{\mathbf{U}} \cdot \nabla^f_{xx} \\ & -\mathbf{G}_{\mathbf{UUU}} \cdot \mathbf{U}_x \cdot \nabla^f \cdot \mathbf{U}_y - \mathbf{G}_{\mathbf{UU}} \cdot \mathbf{U}_{xy} \cdot \nabla^f - \mathbf{G}_{\mathbf{UU}} \cdot \nabla^f_x \cdot \mathbf{U}_y \\ & -\mathbf{G}_{\mathbf{UU}} \cdot \mathbf{U}_x \cdot \nabla^f_y - \mathbf{G}_{\mathbf{U}} \cdot \nabla^f_{xy} \\ & -\mathbf{H}_{\mathbf{UUU}} \cdot \mathbf{U}_x \cdot \nabla^f \cdot \mathbf{U}_z - \mathbf{H}_{\mathbf{UU}} \cdot \mathbf{U}_{xz} \cdot \nabla^f - \mathbf{H}_{\mathbf{UU}} \cdot \nabla^f_x \cdot \mathbf{U}_z \\ & -\mathbf{H}_{\mathbf{UU}} \cdot \mathbf{U}_x \cdot \nabla^f_z - \mathbf{H}_{\mathbf{U}} \cdot \nabla^f_{xz},\end{aligned} \tag{4.15}$$

and similarly for ∇^f_{ty} and ∇^f_{tz}.

Collecting Eqs. (4.10) to (4.15) the fourth-order approximation of the time-averaged x-flux, $\mathbf{F}^{appx,4}$ can be expressed in the explicit form, as the spatial derivatives are readily approximated through the conventional central differencing schemes. In this dissertation, the conventional five-point central differencing formulae are used:

$$\mathbf{F}_x\big|_{\mathbf{x}=\mathbf{x}_{ijk}} = \frac{\mathbf{F}_{i-2} - 8\mathbf{F}_{i-1} + 8\mathbf{F}_{i+1} - \mathbf{F}_{i+2}}{12\Delta x} + \mathcal{O}(\Delta x^4), \tag{4.16}$$

$$\mathbf{F}_{xx}\big|_{\mathbf{x}=\mathbf{x}_{ijk}} = \frac{-\mathbf{F}_{i-2} + 16\mathbf{F}_{i-1} - 30\mathbf{F}_i + 16\mathbf{F}_{i+1} - \mathbf{F}_{i+2}}{12\Delta x^2} + \mathcal{O}(\Delta x^4). \tag{4.17}$$

For the cross derivatives,

$$\mathbf{F}_{xy}\big|_{\mathbf{x}=\mathbf{x}_{ijk}} = \frac{\mathbf{F}_{i+1,j+1} - \mathbf{F}_{i-1,j+1} - \mathbf{F}_{i+1,j-1} + \mathbf{F}_{i-1j-1}}{4\Delta x \Delta y} + \mathcal{O}(\Delta x^2, \Delta y^2). \tag{4.18}$$

WENO-weighted finite difference operators should be introduced here.

In practical code implementation, reusing the flux divergence ∇^f for calculating high-order spatial derivatives is more efficient than calculating them directly.

For example, ∇_x^f can be calculated as $\texttt{dx}(\nabla^f)$, with the numerical spatial derivative function $\texttt{dx}(\cdot)$, rather than calculating as $\nabla_x^f = \mathbf{F}_{xx} + \mathbf{G}_{yx} + \mathbf{H}_{zx}$. 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 without affecting the accuracy of the scheme.

The PIF method is a very efficient numerical strategy to update the solution in FDM formulation. Once the high-order time-averaged fluxes are determined, the solution can be updated through a single step by following the exact same process for the conventional FDM spatial reconstruction. Using the conventional spatial strategy of the FDM formulation, the PIF method can be "swapped" readily with the SSP-RK scheme in the existing simulation code for improving the code performance.

However, the direct analytic derivations for flux Jacobians, Hessians (and more) remain as the implementation hurdle for the PIF method. Unlike SSP-RK methods, the PIF method requires different code implementation for a different system of equations only because of the *Jacobian-like* terms. (e.g., $\mathbf{F}_{\mathbf{U}}, \mathbf{F}_{\mathbf{UU}}, \mathbf{F}_{\mathbf{UUU}}, \ldots$) This dissertation aims to tackle this problem, making a different strategy to use the LW/CK procedure, which does not require analytical derivations of *Jacobian-like* terms.

4.2 System-Free Approach

This section aims to provide a new alternate formulation of computing the multiplications of Jacobian-vector and Hessian-vector-vector terms in Eqs. (4.10) to (4.15). The new approach will replace the necessity for analytical derivations of the Jacobian-like terms in the original PIF method that is system-dependent,

with a new system-independent formulation, based on the so-called "Jacobian-free" method, which is widely used for Newton-Krylov-type iterative schemes [28, 12, 37, 38].

Suppose the Taylor expansion for the flux vector $\mathbf{F}$ at a small displacement from $\mathbf{U}$,

$$\mathbf{F}(\mathbf{U}+\varepsilon\mathbf{V}) = \mathbf{F}(\mathbf{U}) + \varepsilon\mathbf{F}_{\mathbf{U}}\cdot\mathbf{V} + \frac{1}{2}\varepsilon^2\mathbf{F}_{\mathbf{UU}}\cdot\mathbf{V}\cdot\mathbf{V} + \mathcal{O}(\varepsilon^3), \tag{4.19a}$$

$$\mathbf{F}(\mathbf{U}-\varepsilon\mathbf{V}) = \mathbf{F}(\mathbf{U}) - \varepsilon\mathbf{F}_{\mathbf{U}}\cdot\mathbf{V} + \frac{1}{2}\varepsilon^2\mathbf{F}_{\mathbf{UU}}\cdot\mathbf{V}\cdot\mathbf{V} + \mathcal{O}(\varepsilon^3), \tag{4.19b}$$

where $\mathbf{V}$ is an arbitrary vector that has the same number of components as $\mathbf{U}$, and ε is a small scalar perturbation. By subtracting Eq. (4.19b) from Eq. (4.19a), we get an expression of a central differencing that is of second-order in ε,

$$\mathbf{F}_{\mathbf{U}}\cdot\mathbf{V} = \frac{1}{2\varepsilon}\Big[\mathbf{F}(\mathbf{U}+\varepsilon\mathbf{V}) - \mathbf{F}(\mathbf{U}-\varepsilon\mathbf{V})\Big] + \mathcal{O}(\varepsilon^2). \tag{4.20}$$

Alternatively, the first-order forward differencing or the backward differencing can be used here. However, the above second-order central differencing is used for this dissertation, so that the order of accuracy of the entire system-free approach consistently scales with $\mathcal{O}(\varepsilon^2)$, given that the Hessian approximation described in the following is to be bounded by $\mathcal{O}(\varepsilon^2)$. With the system-free approximation of Jacobian, all the Jacobian-vector products in Eqs. (4.10) to (4.15) are to be replaced with the central differencing in Eq. (4.20).

For the approximation for Hessians, it is imperative to classify the types of the Hessian tensor contraction. The first type is the Hessian tensor contracts with the same vector twice, e.g., $\mathbf{F}_{\mathbf{UU}}\cdot\mathbf{V}\cdot\mathbf{V}$, and the second type is the tensor contracts with two different vectors, e.g., $\mathbf{F}_{\mathbf{UU}}\cdot\mathbf{V}\cdot\mathbf{W}$.

For the first type, we use a Taylor expansion analogous to Eq. (4.19) to ap-

proximate the Hessian-vector-vector product with a central differencing of order $\mathcal{O}(\varepsilon^2)$,

$$\mathbf{F_{UU}} \cdot \mathbf{V} \cdot \mathbf{V} = \frac{1}{\varepsilon^2}\Big[\mathbf{F}(\mathbf{U}+\varepsilon\mathbf{V}) - 2\mathbf{F}(\mathbf{U}) - \mathbf{F}(\mathbf{U}-\varepsilon\mathbf{V})\Big] + \mathcal{O}(\varepsilon^2). \tag{4.21}$$

Using a simple vector calculus, the second type can be derived from the first type in Eq. (4.21) by exploring a symmetric property of the Hessians,

$$\mathbf{F_{UU}} \cdot \mathbf{V} \cdot \mathbf{W} = \frac{1}{2}\Big[\mathbf{F_{UU}} \cdot (\mathbf{V}+\mathbf{W}) \cdot (\mathbf{V}+\mathbf{W}) - (\mathbf{F_{UU}} \cdot \mathbf{V} \cdot \mathbf{V} + \mathbf{F_{UU}} \cdot \mathbf{W} \cdot \mathbf{W})\Big]. \tag{4.22}$$

The Hessian approximations derived here are now ready to be substituted in Eqs. (4.11) to (4.15).

Theoretically speaking, the 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 PIF method Eq. (4.7) requires the third-order derivative of $\mathbf{F}$, i.e., $\mathbf{F_{UUU}}$. Following the same mathematical basis of Eqs. (4.20) and (4.21), the tensor contractions with the same vectors can be approximated as,

$$\begin{aligned}\mathbf{F_{UUU}} \cdot \mathbf{V} \cdot \mathbf{V} \cdot \mathbf{V} = \frac{1}{2\varepsilon^3}\Big[&- \mathbf{F}(\mathbf{U}-2\varepsilon\mathbf{V}) + 2\mathbf{F}(\mathbf{U}-\varepsilon\mathbf{V}) \\ &- 2\mathbf{F}(\mathbf{U}+\varepsilon\mathbf{V}) + \mathbf{F}(\mathbf{U}+2\varepsilon\mathbf{V})\Big] + \mathcal{O}(\varepsilon^2).\end{aligned} \tag{4.23}$$

We can further extend the procedure to compute the contraction with three dif-

ferent vectors, $\mathbf{V}, \mathbf{W}$, and $\mathbf{X}$,

$$\begin{aligned}\mathbf{F_{UUU}} \cdot \mathbf{V} \cdot \mathbf{W} \cdot \mathbf{X} = \frac{1}{6}\Big[&\mathbf{F_{UUU}} \cdot (\mathbf{V}+\mathbf{W}+\mathbf{X}) \cdot (\mathbf{V}+\mathbf{W}+\mathbf{X}) \cdot (\mathbf{V}+\mathbf{W}+\mathbf{X}) \\ &- \mathbf{F_{UUU}} \cdot (\mathbf{V}+\mathbf{W}) \cdot (\mathbf{V}+\mathbf{W}) \cdot (\mathbf{V}+\mathbf{W}) \\ &- \mathbf{F_{UUU}} \cdot (\mathbf{V}+\mathbf{X}) \cdot (\mathbf{V}+\mathbf{X}) \cdot (\mathbf{V}+\mathbf{X}) \\ &- \mathbf{F_{UUU}} \cdot (\mathbf{W}+\mathbf{X}) \cdot (\mathbf{W}+\mathbf{X}) \cdot (\mathbf{W}+\mathbf{X}) \\ &+ \mathbf{F_{UUU}} \cdot \mathbf{V} \cdot \mathbf{V} \cdot \mathbf{V} \\ &+ \mathbf{F_{UUU}} \cdot \mathbf{W} \cdot \mathbf{W} \cdot \mathbf{W} \\ &+ \mathbf{F_{UUU}} \cdot \mathbf{X} \cdot \mathbf{X} \cdot \mathbf{X}\Big],\end{aligned} \tag{4.24}$$

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

4.2.1 The proper choices of ε

In the above system-free approximations, the choice of ε has to be considered carefully as it affects the solution accuracy and stability. On one hand, ε is needed to be minimized to improve the approximated solution accuracy, the quality of which will scale as the truncation error of $\mathcal{O}(\varepsilon^2)$. On the other hand, if it is too small the solution would be contaminated by the floating-point roundoff error which is bounded by the machine accuracy $\varepsilon_{\text{mach}}$ [37]. Therefore, ε is to be determined judiciously to provide a good balance between the two types of error.

A recent study by An et al. [2] presents an effective analysis of choosing ε in the context of the Jacobian-free Newton-Krylov iterative framework. The authors have shown how to compute an ideal value of ε which minimizes the error of the central differencing in the Jacobian-vector approximation.

The main idea in [2] is to find a good balance between the truncation error $\mathcal{O}(\varepsilon^2)$ of each Jacobian-free approximation in Eq. (4.20) and Hessian-free approximation in Eq. (4.21), and the intrinsic floating-point roundoff error $\delta\mathbf{F}(\mathbf{U})$ when calculating the target exact function value $\mathbf{F}(\mathbf{U})$ with an approximate value $\mathbf{F}(\mathbf{U}) + \delta\mathbf{F}(\mathbf{U})$. The perturbation $\delta\mathbf{F}(\mathbf{U})$ may include any errors characterized in computer arithmetic such as roundoff errors, and is assumed to be bounded by the machine accuracy.

Let $\mathbf{F}(\mathbf{U})$ be an exact function value of $\mathbf{F}$ at $\mathbf{U}$, and let $\mathbf{F}^*(\mathbf{U}) = \mathbf{F}(\mathbf{U}) + \delta\mathbf{F}(\mathbf{U})$ be an approximation to $\mathbf{F}(\mathbf{U})$, where $\delta\mathbf{F}(\mathbf{U})$ is a perturbation of $\mathbf{F}(\mathbf{U})$ that is potentially due from roundoff errors and truncation errors and is assumed to be bounded by the machine accuracy, i.e., $||\delta\mathbf{F}(\mathbf{U})|| \leq \varepsilon_{\text{mach}}$. The main idea is to choose an optimal ε value for the Jacobian-free approximation Eq. (4.20), $\varepsilon_{\text{jac}}^{op}$, in such a way that the error is minimized when approximating $\mathbf{F}_{\mathbf{U}} \cdot \mathbf{V}$ using the central differencing approximation of $\mathbf{F}^*(\mathbf{U})$ in Eq. (4.20), i.e.,

$$\begin{aligned}\mathbf{F}_{\mathbf{U}} \cdot \mathbf{V} &\approx \frac{1}{2\sigma}\Big[\mathbf{F}^*(\mathbf{U} + \sigma\mathbf{V}) - \mathbf{F}^*(\mathbf{U} - \sigma\mathbf{V})\Big] \\ &= \frac{1}{2\sigma}\Big[\mathbf{F}(\mathbf{U} + \sigma\mathbf{V}) + \delta\mathbf{F}(\mathbf{U} + \sigma\mathbf{V}) - \mathbf{F}(\mathbf{U} - \sigma\mathbf{V}) - \delta\mathbf{F}(\mathbf{U} - \sigma\mathbf{V})\Big].\end{aligned} \tag{4.25}$$

For the sake of this analysis, we assume that the function $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ is defined to be continuously differentiable sufficiently everywhere, $\mathbf{F} \in C^k(\mathbb{R}^n)$. We now define the error E by the difference between the central differencing approximation

in Eq. (4.25) and $\mathbf{F}_{\mathbf{U}} \cdot \mathbf{V}$,

$$
\begin{aligned}
E &= \frac{1}{2\sigma}\Big[\mathbf{F}^*(\mathbf{U}+\sigma\mathbf{V}) - \mathbf{F}^*(\mathbf{U}-\sigma\mathbf{V})\Big] - \mathbf{F}_{\mathbf{U}} \cdot \mathbf{V} \\
&= \frac{1}{2\sigma}\Big[\mathbf{F}(\mathbf{U}+\sigma\mathbf{V}) - \mathbf{F}(\mathbf{U}-\sigma\mathbf{V})\Big] + \frac{1}{2\sigma}\Big[\delta\mathbf{F}(\mathbf{U}+\sigma\mathbf{V}) - \delta\mathbf{F}(\mathbf{U}-\sigma\mathbf{V})\Big] - \mathbf{F}_{\mathbf{U}} \cdot \mathbf{V} \\
&= \frac{1}{2\sigma}\left[2\sigma\mathbf{F}_{\mathbf{U}} \cdot \mathbf{V} + \sigma^3 \int_0^1 (1-t)^2\, \mathbf{F}^{(3)}(\mathbf{U}+t\sigma\mathbf{V}) \cdot \mathbf{V}^3\, dt\right] \\
&\qquad + \frac{1}{2\sigma}\Big[\delta\mathbf{F}(\mathbf{U}+\sigma\mathbf{V}) - \delta\mathbf{F}(\mathbf{U}-\sigma\mathbf{V})\Big] - \mathbf{F}_{\mathbf{U}} \cdot \mathbf{V} \\
&= \mathcal{O}\left(\frac{\sigma^2}{2} + \frac{\varepsilon_{\text{mach}}}{2\sigma}\right),
\end{aligned}
\tag{4.26}
$$

where the Taylor series expansion around $\mathbf{U}$ is used for each term in which the remainders after the third power are given as the integral form as below,

$$
\begin{aligned}
\mathbf{F}(\mathbf{U}+\sigma\mathbf{V}) &= \mathbf{F}(\mathbf{U}) + \sigma\mathbf{F}_{\mathbf{U}} \cdot \mathbf{V} + \frac{\sigma^2}{2}\partial^2_{\mathbf{U}}\mathbf{F} \cdot \mathbf{V} \cdot \mathbf{V} \\
&\qquad + \frac{\sigma^3}{2}\int_0^1 (1-t)^2\, \mathbf{F}^{(3)}(\mathbf{U}+t\sigma\mathbf{V}) \cdot \mathbf{V}^3\, dt, \\
\mathbf{F}(\mathbf{U}-\sigma\mathbf{V}) &= \mathbf{F}(\mathbf{U}) - \sigma\mathbf{F}_{\mathbf{U}} \cdot \mathbf{V} + \frac{\sigma^2}{2}\partial^2_{\mathbf{U}}\mathbf{F} \cdot \mathbf{V} \cdot \mathbf{V} \\
&\qquad - \frac{\sigma^3}{2}\int_0^1 (1-t)^2\, \mathbf{F}^{(3)}(\mathbf{U}+t\sigma\mathbf{V}) \cdot \mathbf{V}^3\, dt\,.
\end{aligned}
\tag{4.27}
$$

The optimal choice of $\varepsilon^{op}_{\text{jac}}$ is to be obtained by considering the minimization problem of the leading error term in the last line in Eq. (4.26),

$$
\varepsilon^{op}_{\text{jac}} = \arg\min_{\sigma>0}\left(\frac{\sigma^2}{2} + \frac{\varepsilon_{\text{mach}}}{2\sigma}\right) = \left(\frac{\varepsilon_{\text{mach}}}{2}\right)^{\frac{1}{3}} \approx 4.8062 \times 10^{-6}, \tag{4.28}
$$

where $\varepsilon_{\text{mach}} \sim 2.2204 \times 10^{-16}$ is used assuming a double-precision in a typical 64-bit machine.

Following the similar procedures, the optimal epsilon value for the Hessian-free

approximation Eq. (4.21), $\varepsilon_{\text{hes}}^{op}$ can be found as,

$$\varepsilon_{\text{hes}}^{op} = \arg\min_{\sigma>0} \left(\frac{\sigma^2}{3} + \frac{\varepsilon_{\text{mach}}}{\sigma^2} \right) = (3\varepsilon_{\text{mach}})^{\frac{1}{4}} \approx 1.6065 \times 10^{-4}. \tag{4.29}$$

However, direct use of ε^{op} as the displacement step size in the central differencing schemes in Eq. (4.20) and Eq. (4.21) is not a good idea for stability reasons. Usually, the vector $\mathbf{V}$ in Eq. (4.20) and Eq. (4.21) could have an enormous value in a strong shock region, so it is safer to use a smaller step size to preserve the needed stability. To meet this, the ideal value, ε^{op} should be normalized by the magnitude of the vector $\mathbf{V}$. There are several prescriptions available in the Jacobian-free Newton–Krylov literatures [37, 12] to help finalize the decision of choosing a proper value of ε as a function of ε^{op}. Nonetheless, as reported in [41], a simple approach of taking a square root of ε^{op} with a simple normalization is sufficient to attain the desired accuracy and stability, which is given as,

$$\overline{\varepsilon} = \frac{\sqrt{\varepsilon^{op}}}{\|\mathbf{V}\|_2}. \tag{4.30}$$

Lastly, the ε estimation can be finalized by taking the minimum value between $\overline{\varepsilon}$ and Δt,

$$\varepsilon = \min\left(\overline{\varepsilon},\ \Delta t\right), \tag{4.31}$$

to prevent the division by zero case.

4.3 Recursive System-Free Approach

The original system-free (SF) approach presented in the previous section provides good approximations of tensor contractions between *Jacobian-like* terms and vectors. However, the original SF method 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, Eq. (4.24) requires 28 times flux function calls for just getting a single tensor contraction, $\mathbf{F}_{\mathbf{UUU}} \cdot \mathbf{V} \cdot \mathbf{W} \cdot \mathbf{X}$. It is worth noting that the major bottleneck of the original SF method stems from Eq. (4.22) and Eq. (4.24) that require to perform the *Jacobian-like* approximations multiple times.

To avoid the additional modifications for the case of the tensor contractions with different vectors, the improved version of the SF method was proposed in [42]. This new, improved SF method, apply the Jacobian-free method recursively to construct the high-order Jacobian-like terms.

The recursive SF method starts from defining a functional $\mathcal{D}_u$ that represents the Jacobian-free method denoted in Eq. (4.20):

$$\mathbf{F}_{\mathbf{U}} \cdot \mathbf{V} \approx \mathcal{D}_u(\mathbf{F}\,;\,\mathbf{V}) \coloneqq \frac{1}{2\varepsilon_v}\Big[\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V}) - \mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V})\Big], \tag{4.32}$$

where ε_v is the appropriately calculated ε corresponding to the vector $\mathbf{V}$ by following the original system-free method Eq. (4.30),

$$\varepsilon_v = \min\left(\bar{\varepsilon}_v,\ \Delta t\right), \quad \text{where } \bar{\varepsilon}_v = \frac{\sqrt{\varepsilon^{op}}}{\|\mathbf{V}\|_2}. \tag{4.33}$$

The recursive SF method uses $\varepsilon^{op} = 4.8062 \times 10^{-6}$ that is the optimal ε value for the second-order Jacobian-free approximation in the 64-bit machine as it shown in Eq. (4.28). 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.

By applying $\mathcal{D}_u$ in the following successive fashion, the tensor contractions between higher order derivatives for the flux function $\mathbf{F}$ and arbitrary vectors.

For instance, the Hessian approximation becomes,

$$\begin{aligned}\mathbf{F}_{\mathbf{UU}} \cdot \mathbf{V} \cdot \mathbf{W} &\approx \mathcal{D}_u\Big(\mathcal{D}_u(\mathbf{F}\,;\,\mathbf{V})\,;\,\mathbf{W}\Big) \\ &= \frac{1}{4\varepsilon_v\varepsilon_w}\Big[\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V}+\varepsilon_w\mathbf{W}) - \mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V}+\varepsilon_w\mathbf{W}) \\ &\qquad -\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V}-\varepsilon_w\mathbf{W}) + \mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V}-\varepsilon_w\mathbf{W})\Big].\end{aligned} \tag{4.34}$$

Again, following Eq. (4.33), 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. (4.34) is applicable regardless the tensor contraction is with two identical vectors (e.g., $\mathbf{F}_{\mathbf{UU}} \cdot \mathbf{V} \cdot \mathbf{V}$) or with two distinct vectors (e.g., $\mathbf{F}_{\mathbf{UU}} \cdot \mathbf{V} \cdot \mathbf{W}$), hence it does not require separate formulations as in Eqs. (4.22) and (4.24).

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

$$\begin{aligned}\mathbf{F}_{\mathbf{UUU}} \cdot \mathbf{V} \cdot \mathbf{W} \cdot \mathbf{X} &\approx \mathcal{D}_u\left(\mathcal{D}_u\Big(\mathcal{D}_u(\mathbf{F}\,;\,\mathbf{V})\,;\,\mathbf{W}\Big)\,;\,\mathbf{X}\right) \\ &= \frac{1}{8\varepsilon_v\varepsilon_w\varepsilon_x}\Big[\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V}+\varepsilon_w\mathbf{W}+\varepsilon_x\mathbf{X}) - \mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V}+\varepsilon_w\mathbf{W}+\varepsilon_x\mathbf{X}) \\ &\qquad -\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V}-\varepsilon_w\mathbf{W}+\varepsilon_x\mathbf{X}) + \mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V}-\varepsilon_w\mathbf{W}+\varepsilon_x\mathbf{X}) \\ &\qquad -\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V}+\varepsilon_w\mathbf{W}-\varepsilon_x\mathbf{X}) + \mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V}+\varepsilon_w\mathbf{W}-\varepsilon_x\mathbf{X}) \\ &\qquad +\mathbf{F}(\mathbf{U}+\varepsilon_v\mathbf{V}-\varepsilon_w\mathbf{W}-\varepsilon_x\mathbf{X}) - \mathbf{F}(\mathbf{U}-\varepsilon_v\mathbf{V}-\varepsilon_w\mathbf{W}-\varepsilon_x\mathbf{X})\Big].\end{aligned} \tag{4.35}$$

Here, the performance between the recursive SF method and the original SF method can be compared by the number of flux function calls. For instance, considering the case of approximating $\mathbf{F_{UUU}} \cdot \mathbf{V} \cdot \mathbf{W} \cdot \mathbf{X}$ term, the original SF method requires 28 function calls. On the other hand, the recursive SF method needs only eight evaluations. This is a huge improvement in both performance and compactness.

By utilizing Eqs. (4.32) to (4.35), all the tensor contractions needed in the fourth-order PIF method in Eqs. (4.10) to (4.15) can be approximated without the analytical calculations of the *Jacobian-like* terms. Combining the recursive SF method, the PIF method can be implemented more efficiently, allowing the system independence of the high-order scheme. It should be noted 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.

The fourth-order SF-PIF4 method can be summarized as the following stepwise fashion. The discretization indices i, j, k and n are omitted 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. (4.32) in preparation for $\mathbf{F}_t$ as expressed in Eq. (4.10), 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. (4.32) and (4.34) to the spatially approximated derivative quantities in **Step 3** in order to compute ∇^f_t by following the explicit expression in Eq. (4.12).

Step 5: Apply the Jacobian approximation in (4.32) to ∇^f_t from **Step 4** and the Hessian approximation in (4.34) to ∇^f from **Step 1** in order to get $\mathbf{F}_{tt}$ using Eq. (4.11); 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. (4.32), (4.34) and (4.35) 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. (4.14) and (4.15) respectively. Also calculate ∇^f_{ty} and ∇^f_{tz} similarly.

Step 8: Next, perform the last set of tensor contractions in Eqs. (4.32), (4.34) and (4.35) to construct $\mathbf{F}_{ttt}$ as expressed in Eq. (4.13). 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. (4.6) to the results, $\mathbf{F}^{appx,4}$, $\mathbf{G}^{appx,4}$, and $\mathbf{H}^{appx,4}$, from **Step 8**. For example, taking WENO-JS (Section 3.1.1) 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. (4.4).

4.4 Conclusion

The Picard integral formulation (PIF) method is one of the Lax-Wendroff class high-order in temporal integration strategies for FDM discretization. By virtue of a single-stage time integrator, the PIF method can perform faster than the traditional multi-stage method. Also, the PIF method does not depend on the spatial reconstruction scheme; thus, it can be combined with any high-order spatial method in general.

However, as like other Lax-Wendroff type schemes, the PIF method highly depends on the system of equations, requiring analytical derivations for *Jacobian-*

like terms. Although the symbolic manipulation tools can aid these calculations, *Jacobian-like* terms remain as a major implementation hurdle due to their perplexing structures.

The (original, non-recursive) system-free (SF) approach provides capability to Lax-Wendroff type scheme to bypass all the analytical derivations of Jacobian-like terms, approximating tensor contractions between Jacobian-like terms and arbitrary vectors. The major advantage of SF method lies in ease of its code implementation for practical use. By combining with PIF method, SF-PIF method can be applied to any system of equations to furnish high-order in temporal accuracy in a single-step. However, the increasing number of calculations needed for higher order derivatives of the flux function $\mathbf{F}$ with respect to the conservative variables $\mathbf{U}$ makes the SF method less attractive for higher than third-order PIF method.

The improved version, recursive SF approach is then introduced to minimize the number of calculation needed for approximating the tensor contractions. The recursive SF method introduced a functional representing the Jacobian-free method, then the higher order derivative terms can be obtained by applying the functional recursively. This feature allows to extend SF-PIF method to the fourth-order accuracy efficiently.

It is important to note that the SF method is neither designed particularly for the PIF method nor any specific numerical methods in CFD. Instead, it is solely intended for approximating the Jacobian-like tensor contractions, so the SF approach is applicable in otehr numerical algorithms to enhance the calculation speed and implementation efficiency.

Chapter 5

Results

This chapter will provide various numerical test results of SF-PIF methods. In order to examine the numerical capabilities of the SF method, several well-known numerical benchmark problems are conducted, and the traditional SSP-RK methods' results will be provided with the same initial conditions as counterparts of SF-PIF methods for comparisons. SF-PIF3 and SF-PIF4 will refer to the *recursive* SF method in Section 4.3 with third-order and fourth-order PIF methods, respectively, and RK3 and RK4 will refer to the three-stage, third-order SSP-RK method Eq. (3.45), five-stage, fourth-order SSP-RK method Eq. (3.47), respectively. The original SF approach (Section 4.2) with the PIF method is denoted by oSF-PIF.

5.1 Performance of SF-PIF method

The main advantage of the PIF method is the performance gain compared to the SSP-RK methods. This section will compare the performance of PIF methods (with or without the SF approach) and the SSP-RK method. The main purpose of this section is to check if the SF-PIF methods provide improved performance while

maintaining the same accuracy as SSP-RK methods. Theoretically speaking, the SF and oSF approach should not affect the solution's accuracy and stability, so the original PIF method's results are presented for comparisons.

All test results in this section use the standard fifth-order WENO-JS method (Section 3.1.1) for the spatially high-order reconstruction scheme. Therefore, the expected truncation error is $\mathcal{O}(\Delta s^5, \Delta t^q)$, where q is the order of the temporal scheme.

5.2 SF-PIF method with GP-WENO

5.2.1 Hyperparameters

5.3 SF-PIF method with WENO-JS

Chapter 6

Conclusion

Bibliography

[1] Antxón Alberdi, JL Gómez Fernández, Event Horizon Telescope Collaboration, et al. First m87 event horizon telescope results. i. the shadow of the supermassive black hole. 2019.

[2] Heng-Bin An, Ju Wen, and Tao Feng. On finite difference approximation of a matrix-vector product in the Jacobian-free Newton-Krylov method. *Journal of Computational and Applied Mathematics*, 236(6):1399–1409, 2011.

[3] Norbert Attig, Paul Gibbon, and Th Lippert. Trends in supercomputing: The european path to exascale. *Computer physics communications*, 182(9):2041–2046, 2011.

[4] Dinshaw S Balsara. Higher-order accurate space-time schemes for computational astrophysics—part I: finite volume methods. *Living reviews in computational astrophysics*, 3(1):2, 2017.

[5] Dinshaw S Balsara, Sudip Garain, and Chi-Wang Shu. An efficient class of weno schemes with adaptive order. *Journal of Computational Physics*, 326:780–804, 2016.

[6] Dinshaw S Balsara, Chad Meyer, Michael Dumbser, Huijing Du, and Zhiliang Xu. Efficient implementation of ADER schemes for Euler and magnetohydrodynamical flows on structured meshes—speed comparisons with Runge-Kutta methods. *Journal of Computational Physics*, 235:934–969, 2013.

[7] Dinshaw S Balsara, Tobias Rumpf, Michael Dumbser, and Claus-Dieter Munz. Efficient, high accuracy ADER-WENO schemes for hydrodynamics and divergence-free magnetohydrodynamics. *Journal of Computational Physics*, 228(7):2480–2516, 2009.

[8] Dinshaw S Balsara and Chi-Wang Shu. Monotonicity preserving weighted essentially non-oscillatory schemes with increasingly high order of accuracy. *Journal of Computational Physics*, 160(2):405–452, 2000.

[9] Marsha J Berger and Phillip Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1):64–84, 1989.

[10] Marsha J Berger and Randall J Leveque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM Journal on Numerical Analysis*, 35(6):2298–2316, 1998.

[11] Rafael Borges, Monique Carmona, Bruno Costa, and Wai Sun Don. An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws. *Journal of Computational Physics*, 227(6):3191–3211, 2008.

[12] Peter N Brown and Youcef Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM Journal on Scientific and Statistical Computing*, 11(3):450–481, 1990.

[13] Pawel Buchmüller and Christiane Helzel. Improved accuracy of high-order weno finite volume methods on cartesian grids. *Journal of Scientific Computing*, 61(2):343–368, 2014.

[14] Marcos Castro, Bruno Costa, and Wai Sun Don. High order weighted essentially non-oscillatory weno-z schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 230(5):1766–1792, 2011.

[15] Chaó-Kuang Chen and Shing-Huei Ho. Application of differential transformation to eigenvalue problems. *Applied mathematics and computation*, 79(2-3):173–188, 1996.

[16] Andrew J Christlieb, Xiao Feng, Yan Jiang, and Qi Tang. A high-order finite difference weno scheme for ideal magnetohydrodynamics on curvilinear meshes. *SIAM Journal on Scientific Computing*, 40(4):A2631–A2666, 2018.

[17] Andrew J Christlieb, Yaman Guclu, and David C Seal. The Picard integral formulation of weighted essentially nonoscillatory schemes. *SIAM Journal on Numerical Analysis*, 53(4):1833–1856, 2015.

[18] Phillip Colella and Paul R Woodward. The piecewise parabolic method (PPM) for gas-dynamical simulations. *Journal of computational physics*, 54(1):174–201, 1984.

[19] Nelida Črnjarić-Žic, S Vuković, and Luka Sopta. On different flux splittings and flux functions in weno schemes for balance laws. *Computers & fluids*, 35(10):1074–1092, 2006.

[20] L Del Zanna, N Bucciantini, and P Londrillo. An efficient shock-capturing central-type scheme for multidimensional relativistic flows-ii. magnetohydrodynamics. *Astronomy & Astrophysics*, 400(2):397–413, 2003.

[21] L Del Zanna, O Zanotti, N Bucciantini, and P Londrillo. Echo: a eulerian conservative high-order scheme for general relativistic magnetohydrodynamics and magnetodynamics. *Astronomy & Astrophysics*, 473(1):11–30, 2007.

[22] Jack Dongarra, Jeffrey Hittinger, John Bell, Luis Chacon, Robert Falgout, Michael Heroux, Paul Hovland, Esmond Ng, Clayton Webster, and Stefan Wild. Applied mathematics research for exascale computing. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2014.

[23] Michael Dumbser, Dinshaw S Balsara, Eleuterio F Toro, and Claus-Dieter Munz. A unified framework for the construction of one-step finite volume and discontinuous galerkin schemes on unstructured meshes. *Journal of Computational Physics*, 227(18):8209–8253, 2008.

[24] Michael Dumbser, Martin Käser, Vladimir A Titarev, and Eleuterio F Toro. Quadrature-free non-oscillatory finite volume schemes on unstructured meshes for nonlinear hyperbolic systems. *Journal of Computational Physics*, 226(1):204–243, 2007.

[25] Francesco Fambri, Michael Dumbser, and Olindo Zanotti. Space–time adaptive ADER-DG schemes for dissipative flows: Compressible Navier-Stokes and resistive MHD equations. *Computer Physics Communications*, 220:297–318, 2017.

[26] Kyle Gerard Felker and James M Stone. A fourth-order accurate finite volume method for ideal mhd via upwind constrained transport. *Journal of Computational Physics*, 375:1365–1400, 2018.

[27] Jim A Gaffney, Scott T Brandon, Kelli D Humbird, Michael KG Kruse, Ryan C Nora, J Luc Peterson, and Brian K Spears. Making inertial confinement fusion models more predictive. *Physics of Plasmas*, 26(8):082704, 2019.

[28] Charles William Gear and Youcef Saad. Iterative solution of linear equations in ODE codes. *SIAM journal on scientific and statistical computing*, 4(4):583–601, 1983.

[29] SK Godunov. A difference scheme for numerical computation of discontinuous solutions of fluid dynamics. *Mat. Sb*, 47:271–306, 1959.

[30] Sigal Gottlieb, David I Ketcheson, and Chi-Wang Shu. *Strong stability preserving Runge-Kutta and multistep time discretizations*. World Scientific, 2011.

[31] Sigal Gottlieb and Chi-Wang Shu. Total variation diminishing Runge-Kutta schemes. *Mathematics of computation of the American Mathematical Society*, 67(221):73–85, 1998.

[32] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM review*, 43(1):89–112, 2001.

[33] Brian M Haines, Gary P Grim, James R Fincke, Rahul C Shah, Chad J Forrest, Kevin Silverstein, Frederic J Marshall, Melissa Boswell, Malcolm M Fowler, Robert A Gore, et al. Detailed high-resolution three-dimensional simulations of omega separated reactants inertial confinement fusion experiments. *Physics of Plasmas*, 23(7):072709, 2016.

[34] Ami Harten, Bjorn Engquist, Stanley Osher, and Sukumar R Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes, iii. In *Upwind and high-resolution schemes*, pages 218–290. Springer, 1987.

[35] Oliver James, Eugénie von Tunzelmann, Paul Franklin, and Kip S Thorne. Gravitational lensing by spinning black holes in astrophysics, and in the movie interstellar. *Classical and Quantum Gravity*, 32(6):065001, 2015.

[36] Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted ENO schemes. *Journal of computational physics*, 126(1):202–228, 1996.

[37] Dana A Knoll and David E Keyes. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[38] Dana A Knoll, H Park, and Kord Smith. Application of the Jacobian-free Newton-Krylov method to nonlinear acceleration of transport source iteration in slab geometry. *Nuclear Science and Engineering*, 167(2):122–132, 2011.

[39] Peter Lax. Systems of conservation laws. Technical report, LOS ALAMOS NATIONAL LAB NM, 1959.

[40] Dongwook Lee, Hugues Faller, and Adam Reyes. The piecewise cubic method (pcm) for computational fluid dynamics. *Journal of Computational Physics*, 341:230–257, 2017.

[41] Youngjun Lee and Dongwook Lee. A single-step third-order temporal discretization with jacobian-free and hessian-free formulations for finite difference methods. *Journal of Computational Physics*, 427:110063, 2021.

[42] Youngjun Lee, Dongwook Lee, and Adam Reyes. A recursive system-free single-step temporal discretization method for finite difference methods. *Journal of Computational Physics: X*, 12:100098, 2021.

[43] Xu-Dong Liu, Stanley Osher, Tony Chan, et al. Weighted essentially non-oscillatory schemes. *Journal of computational physics*, 115(1):200–212, 1994.

[44] Peter McCorquodale and Phillip Colella. A high-order finite-volume method for conservation laws on locally refined grids. *Communications in Applied Mathematics and Computational Science*, 6(1):1–25, 2011.

[45] Jena Meinecke, HW Doyle, Francesco Miniati, Anthony R Bell, R Bingham, Robert Crowston, RP Drake, Milad Fatenejad, Michel Koenig, Yasuhiro Kuramitsu, et al. Turbulent amplification of magnetic fields in laboratory laser-produced shock waves. *Nature Physics*, 10(7):520–524, 2014.

[46] Andrea Mignone, Petros Tzeferacos, and Gianluigi Bodo. High-order conservative finite difference GLM–MHD schemes for cell-centered MHD. *Journal of Computational Physics*, 229(17):5896–5920, 2010.

[47] Gino Montecinos and 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 (In 2 Volumes)*, pages 576–583. World Scientific, 2012.

[48] Gino I Montecinos and Dinshaw S Balsara. A simplified cauchy-kowalewskaya procedure for the local implicit solution of generalized riemann problems of hyperbolic balance laws. *Computers & Fluids*, 202:104490, 2020.

[49] Gino I Montecinos and Eleuterio F Toro. Reformulations for general advection–diffusion–reaction equations and locally implicit ader schemes. *Journal of Computational Physics*, 275:415–442, 2014.

[50] Matthew R Norman. Algorithmic improvements for schemes using the ADER time discretization. *Journal of Computational Physics*, 243:176–178, 2013.

[51] Matthew R Norman. A WENO-limited, ADER-DT, finite-volume scheme for efficient, robust, and communication-avoiding multi-dimensional transport. *Journal of Computational Physics*, 274:1–18, 2014.

[52] Matthew R Norman and Hal Finkel. Multi-moment ADER-Taylor methods for systems of conservation laws with source terms in one dimension. *Journal of Computational Physics*, 231(20):6622–6642, 2012.

[53] Adam Reyes, Dongwook Lee, Carlo Graziani, and Petros Tzeferacos. A new class of high-order methods for fluid dynamics simulations using gaussian process modeling: One-dimensional case. *Journal of Scientific Computing*, 76(1):443–480, 2018.

[54] Adam Reyes, Dongwook Lee, Carlo Graziani, and Petros Tzeferacos. A variable high-order shock-capturing finite difference method with gp-weno. *Journal of Computational Physics*, 381:189–217, 2019.

[55] David C Seal, Qi Tang, Zhengfu Xu, and Andrew J Christlieb. An explicit high-order single-stage single-step positivity-preserving finite difference WENO method for the compressible Euler equations. *Journal of Scientific Computing*, 68(1):171–190, 2016.

[56] Chaopeng Shen, Jing-Mei Qiu, and Andrew Christlieb. Adaptive mesh refinement based on high order finite difference weno scheme for multi-scale simulations. *Journal of Computational Physics*, 230(10):3780–3802, 2011.

[57] Chi-Wang Shu. Total-variation-diminishing time discretizations. *SIAM Journal on Scientific and Statistical Computing*, 9(6):1073–1084, 1988.

[58] Chi-Wang Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In *Advanced numerical approximation of nonlinear hyperbolic equations*, pages 325–432. Springer, 1998.

[59] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of computational physics*, 77(2):439–471, 1988.

[60] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes, ii. In *Upwind and High-Resolution Schemes*, pages 328–374. Springer, 1989.

[61] Raymond J Spiteri and Steven J Ruuth. A new class of optimal high-order strong-stability-preserving time discretization methods. *SIAM Journal on Numerical Analysis*, 40(2):469–491, 2002.

[62] Vladimir A Titarev and Eleuterio F Toro. ADER: Arbitrary high order Godunov approach. *Journal of Scientific Computing*, 17(1-4):609–618, 2002.

[63] Vladimir A Titarev and Eleuterio F Toro. Finite-volume weno schemes for three-dimensional conservation laws. *Journal of Computational Physics*, 201(1):238–260, 2004.

[64] Vladimir A Titarev and Eleuterio F Toro. ADER schemes for three-dimensional non-linear hyperbolic systems. *Journal of Computational Physics*, 204(2):715–736, 2005.

[65] Eleuterio F Toro, RC Millington, and LAM Nejad. Towards very high order godunov schemes. In *Godunov methods*, pages 907–940. Springer, 2001.

[66] Eleuterio F Toro and Gino I Montecinos. Implicit, semi-analytical solution of the generalized riemann problem for stiff hyperbolic balance laws. *Journal of Computational Physics*, 303:146–172, 2015.

[67] P Tzeferacos, A Rigby, AFA Bott, AR Bell, R Bingham, A Casner, F Cattaneo, EM Churazov, J Emig, F Fiuza, et al. Laboratory evidence of dynamo amplification of magnetic fields in a turbulent plasma. *Nature communications*, 9(1):1–8, 2018.

[68] Bram Van Leer. Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method. *Journal of computational Physics*, 32(1):101–136, 1979.

[69] Paul Woodward and Phillip Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of computational physics*, 54(1):115–173, 1984.

[70] Olindo Zanotti and Michael Dumbser. Efficient conservative ADER schemes based on WENO reconstruction and space–time predictor in primitive variables. *Computational astrophysics and cosmology*, 3(1):1, 2016.

[71] Rui Zhang, Mengping Zhang, and Chi-Wang Shu. On the order of accuracy and numerical performance of two classes of finite volume weno schemes. *Communications in Computational Physics*, 9(3):807–827, 2011.