Experimental PDE Solver in Julia – Comparison of Flux Limiting Schemes

Amir Farzin [a,*], Zahir Barahmand [b], Bernt Lie [a]

[a] *Department of Electrical Engineering, IT and Cybernetics, University of South-Eastern Norway,*

[b] *Department of Process, Energy and Environmental Technology, University of South-Eastern Norway.*
{amir.farzin, zahir.barahmand, Bernt.Lie}@usn.com

Abstract

Finite Volume Methods (FVM) are high quality methods for solving conservative/hyperbolic partial differential equations (PDEs). A popular class of high-resolution methods utilize a nonlinear combination of low order methods and high order methods via flux limiting functions. Another class of high-resolution methods is the class of weighted essentially non-oscillatory (WENO) schemes. Here, the focus is on flux limiting schemes. An experimental finite volume (FV) semi-discrete solver for systems of hyperbolic PDEs has been implemented in Julia, utilizing Julia's DifferentialEquations.jl package for handling the time marching. A first order upwind formulation is used for the low order method, and a central second order formulation is used for the high order method. The PDE can be provided either in flux form, or in quasi-linear form. In the former case, automatic differentiation (AD) package ForwardDiff.jl is used to compute the Jacobians of the flux vector. Package LinearAlgebra.jl is used to compute the eigenspace of the Jacobians. The implementation allows for up to 3 internal/external coordinates. More than a dozen flux limiting functions are given, with the possibility of the users to write their own flux limiters. The implementation allows for user provided spatial discretization points, and source terms in the PDE. In this paper, we will compare various flux limiting schemes for PDEs with analytic solutions, and will also compare flux limiting schemes for a simple granulation model (layering). Possible extensions of the experimental implementation include: (i) higher order methods, (ii) more extensive support for boundary conditions, (iii) improved support for source terms.

1. Introduction

A partial differential equation (PDE) is a mathematical equation having two or more independent variables, an unknown function (depending on those variables), and partial derivatives of the unknown function with respect to the independent variables [1]. Solving a PDE leads to a function that solves the equation or, in other words, converts it into an identity when it is replaced into the equation. Although some variants of PDEs have analytical solutions, in general numerical methods are used to solve PDEs.

There are different types of numerical methods for solving PDEs such as finite elements, finite volumes, and finite difference. In this paper, the finite volumes method (FVM) is considered as the basic for converting PDE problem into a set of ordinary differential equations (ODEs). Then, the ODEs are solved using ODE solver. There are several standard ODE solvers available in almost every programming language. Usually, this approach is called semi-discretization.

FVMs are high quality methods for solving conservative/hyperbolic PDEs. To achieve high-order accurate numerical approximation of PDEs, especially in presence of shocks or discontinuities, a group of FVM related schemes called high-resolution schemes are vastly used. Among the methods, flux limiter and WENO[1] schemes are shown effective in solving difficult-to-solve PDE problems [2, 3].

Flux limiters (or slope limiters) schemes utilize a nonlinear combination of low order methods and high order methods via flux limiting functions. One simple but effective combination of methods is to use upwind as the low (first) order and centered difference as the high (second) order schemes (called MUSCL[2] scheme) [2]. MUSCL scheme is the core approach in this paper and more than a dozen of flux limiter functions are utilized to solve the PDEs.

This paper focus on the introduction of an open-source solver for Julia programming language. This

[1] Weighted Essentially Non-Oscillatory

[2] Monotonic Upstream-centered Scheme

solver package is available for the interested reader via the following GitHub link:
https://github.com/amirfarzin/FVM_PDEsolver.jl
In addition, to evaluate the performance of the developed package, several PDE problems are solved using different flux limiter functions. Therefore, as the result, a fair comparison of the functions is also presented in this paper.
The rest of this paper is structured as follows: the second section starts with an introduction to the class of PDEs the solver is design to deal with. Then, the MUSCL scheme equations and flux limiter functions are reviewed. The third section contains a brief documentation for the package. The PDE problems used for evaluation and their solution are presented in section four. In addition, the flux limiter functions are compared based on their performance in solving those PDEs. Finally, the paper ends with discussion in section five.

2. PDE problem definitions

As mentioned, PDEs are equations involving at least two independent variables, one or more dependent variables and their derivatives which are functions the independent variables. The first assumption here, is that time is always an independent variable in all PDEs. Therefore, we denote the variables as follows:

- $z_1, z_2, \dots, z_{n_z}, t$ are independent variables where $n_z \geq 1$. With this notation, the PDE involves $n_z + 1$ independent variables. In vector form, $\boldsymbol{z} = \left[z_1, z_2, \dots, z_{n_z}\right]^T \in \mathbb{R}^{n_z}$ is called the vector of spatial (i.e., internal/external coordination) independent variables. Commonly, the time variable t is in the range $[0, \infty)$ (i.e., $t \in \mathbb{R}^+$). And the spatial variables $z_1, z_2, \dots, z_{n_z}$ are defined in a domain denoted by $\boldsymbol{\Omega} \subset \mathbb{R}^{n_z}$ (i.e., $\boldsymbol{z} \in \boldsymbol{\Omega}$).
- $q_1, q_2, \dots, q_{n_q}$ are dependent variables where $q_i = q_i\left(z_1, z_2, \dots, z_{n_z}, t\right) = q_i(\boldsymbol{z}, t)$ and $n_q \geq 1$. In vector form, $\boldsymbol{q} = \boldsymbol{q}(\boldsymbol{z}, t) = \left[q_1, q_2, \dots, q_{n_q}\right]^T : \mathbb{R}^{n_z+1} \to \mathbb{R}^{n_q}$ is the vector of unknown functions or dependent variables.

With above definitions, the general form of a PDE problem is as follows:

$$\boldsymbol{F}(\boldsymbol{z}, t, \boldsymbol{q}, D\boldsymbol{q}, D^2\boldsymbol{q}, \dots, D^m\boldsymbol{q}) = \boldsymbol{0} \qquad (1)$$

where $D^\alpha \boldsymbol{q}$ denotes the tensor of all partial derivatives of order $0 \leq \alpha \leq m$, and $m \geq 1$ is the highest order of partial derivatives in the system. System of Eq. **Error! Reference source not found.**, in general perspective, consists of n_f equations. And the number of unknown variables is n_q.

- if $n_q = n_f$ then the PDE problem is called determined,
- if $n_q < n_f$ or $n_q > n_f$ then the PDE problem is called over-determined or under-determined, respectively.

Without losing generality, it can be assumed that the system of PDEs is determined (i.e., $n_q = n_f$).
The classification of PDE problems, helps easier referring to what type of PDE someone is dealing with. In addition, it is essential to know the class of PDE while using the textbooks and papers, and selecting the solution methodologies. The PDE problems can be classified in four ways:

- **Order of PDE:** The highest order of partial derivatives exists in any of the system equations (i.e., m).
- **Number of equations:** if the number of equations is one (i.e., $n_q = 1$), then the PDE is called scalar. Otherwise, the equations represent a system of PDEs.
- **Linearity:** A PDE system expressed by Eq. (1) is non-linear unless it fits into one of the following groups: (i) quasi-linear[3], (ii) semi-linear[4], or (iii) linear[5].
- **Homogeneity:** A PDE system represented by Eq. (1) is called homogenous if $\boldsymbol{v} \in \mathbb{R}^{n_q}$ is a solution to the system then $a\boldsymbol{v}$ is also a solution for any scalars a.

The package is developed to solve conservation law equation which appears in several physical phenomena such as electromagnetism, fluid dynamics, heat transfer, etc. However, only semi-linear first order system of PDEs with up to three spatial dimensions is considered. For increasing the readability of the notations, let us replace z_1, z_2, and z_3 with x, y, and z, respectively. Note that, x, y, and z are not necessarily 3-D Cartesian axes, but can represent internal coordination as well.
The differential form of the conservation law is as follows:

$$\partial_t \boldsymbol{q} + \vec{\nabla}.\vec{\boldsymbol{\mathcal{F}}}(\boldsymbol{q}) = \boldsymbol{\sigma}(\boldsymbol{q}) \qquad (2)$$

where $\vec{\nabla} = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right]$ is the divergence vector and $\vec{\boldsymbol{\mathcal{F}}} = [\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{h}]$ is the field vector. Equivalently, the Eq. (2) can be written as:

[3] A PDE system of order m is called **quasi-linear** if it is linear in terms of the highest order (i.e., m) partial derivatives.
[4] A quasi-linear PDE system of order m is called **semi-linear** if the coefficients of the highest order (i.e., m) partial derivatives are only functions of the independent variables.
[5] A PDE is called **linear** if the coefficients of all dependent variables and their derivatives are only functions of the independent variables.

$$\partial_t \boldsymbol{q} + \partial_x \boldsymbol{f}(\boldsymbol{q}) + \partial_y \boldsymbol{g}(\boldsymbol{q}) + \partial_z \boldsymbol{h}(\boldsymbol{q}) = \boldsymbol{\sigma}(\boldsymbol{q}) \quad (3)$$

In above equations, $\boldsymbol{\sigma}(\boldsymbol{q})$ is called the source term. If $\boldsymbol{\sigma}(\boldsymbol{q}) = \boldsymbol{0}$, Eq. (3) represent a homogenous system of PDEs. In above equations, $\boldsymbol{f}$, $\boldsymbol{g}$, and $\boldsymbol{h}$ are arbitrary functions for field vectors in x, y, and z directions respectively.

If the flux functions are differentiable with respect to conserved variable $\boldsymbol{q}$, using the chain rule, the Eq. (3) can be written as follows:

$$\partial_t \boldsymbol{q} + \boldsymbol{J_f} \partial_x \boldsymbol{q} + \boldsymbol{J_g} \partial_y \boldsymbol{q} + \boldsymbol{J_h} \partial_z \boldsymbol{q} = \boldsymbol{\sigma} \quad (4)$$

The matrices $\boldsymbol{J_f} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}$, $\boldsymbol{J_g} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{q}}$, and $\boldsymbol{J_h} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}$ are the Jacobians of $\boldsymbol{f}$, $\boldsymbol{g}$, and $\boldsymbol{h}$. If all the eigenvalues of the Jacobian matrices are real with linearly independent eigenvectors, the system of PDEs is called hyperbolic [4]. Note that this paper and the solver package only consider hyperbolic PDEs. The solver is design to handle the PDEs expressed either by Eq. (3) or Eq. (4).

3. Methodology

In this section, for the simplicity of equations, the methodology is discussed for 1-D homogenous conservation law equation as follows:

$$\partial_t \boldsymbol{q} + \partial_x \boldsymbol{f}(\boldsymbol{q}) = \boldsymbol{0} \quad (5)$$

At the end this section, the method is extended for solving PDE problems of Eq. (3) and (4).

3.1. Grid

To solve Eq. (5), the general idea is to convert it into a set of ODE problems where we need the values of $\partial_t \boldsymbol{q}$ at each time step. As $\boldsymbol{q}$ is a function of time and space, the values of $\partial_t \boldsymbol{q}$ at each time step should be found for every point in the space. Here, the space domain is simply bounded to min/max values (i.e., $x \in [x^{\min}, x^{\max}]$).

Numerically, working with a continuous space domain is impossible. Therefore, it is necessary to discretize the space first. The developed package uses a non-uniform rectangular grid up to three dimensions for space discretization. It is also possible to extend it for more dimensions or improve it with triangular cells. A small part of the x axis is shown in Fig 1.

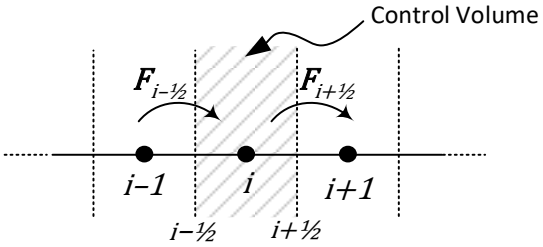

Figure 1: Grid-point cluster for one-dimensional problem

The advantage of non-uniform grid is the variable cell sizes. This feature allows larger cells in non-challenging and smaller cells in critical areas.

Regarding the grid, the following notation is used in the rest of this paper:

- Integer subscripts such as $i-1$, i, and $i+1$ are used for quantities at cell centers. For example, x_i and $\boldsymbol{q}_i$ are the coordination and $\boldsymbol{q}$ vector at the center point of ith cell.
- Subscripts such as $i-\frac{1}{2}$ and $i+\frac{1}{2}$ are used for referring to the quantities at the cell boundaries. For example, as shown in Fig. 1, $\boldsymbol{F}_{i-\frac{1}{2}}$ is the flux vector from $(i-1)$th cell into ith cell at the boundary between them.
- As mentioned, the grid is assumed to be non-uniform which means the lengths of cells can be different. Therefore, the length of the ith cell is denoted by $\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$. In addition, $\Delta x_{i-\frac{1}{2}} = x_i - x_{i-1}$ is the distance between center points of $(i-1)$th and ith cells.

3.2. Finite volume method (FVM)

The FVM is a popular and efficient numerical approach for solving PDE problem. In the FVM, the problem domain (i.e., only space in semi-discretization strategy) is broken into grid cells. Then, the total integral of $\boldsymbol{q}$ over each grid cell is approximated [5]. If the problem presents a conservation equation (e.g., Eq. (5)), the value of the integral is equal to the net flow into the control volume:

$$\partial_t \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \boldsymbol{q} dx = \boldsymbol{F}_{i-\frac{1}{2}} - \boldsymbol{F}_{i+\frac{1}{2}} \quad (6)$$

where $\boldsymbol{F}_{i-\frac{1}{2}}$ is some approximation to the flux along the left boundary of the ith cell:

$$\boldsymbol{F}_{i-\frac{1}{2}} \approx \boldsymbol{f}(\boldsymbol{q})|_{x=x_{i-\frac{1}{2}}} \quad (7)$$

In addition, the cells (called control volumes) are small enough to assume that $\boldsymbol{q}$ is constant in each cell. Let us say the average value of $\boldsymbol{q}$ in ith cell is equal to its value at the center of that control volume (i.e., $\boldsymbol{q}_i$). Replacing $\boldsymbol{q}$ with $\boldsymbol{q}_i$ in Eq. (6) gives:

$$\partial_t \boldsymbol{q}_i \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} dx = \boldsymbol{F}_{i-\frac{1}{2}} - \boldsymbol{F}_{i+\frac{1}{2}}$$

$$\xrightarrow{yields} \quad \partial_t \boldsymbol{q}_i = \frac{\boldsymbol{F}_{i-\frac{1}{2}} - \boldsymbol{F}_{i+\frac{1}{2}}}{\Delta x_i} \quad (8)$$

Hence, we only need to find an accurate approximation of fluxes at the boundaries to solve the PDE problem using the FVM. The difference

between various schemes is how they estimate $\boldsymbol{F}_{i-\frac{1}{2}}$ at each time step.

3.3. High-resolution flux limiter schemes

One drawback of linear methods like upwind is predicted by Godunov's theorem which states that no linear convection scheme of second-order accuracy or higher can be monotonic [6]. This limit would be significant while working with shock waves with sharp gradients. To tackle the problem, non-linear discretization methods are used.

One category of such non-linear approaches is flux limiters (FL) which uses a limiter function for the flux in the boundaries. A unified formulation for flux limiters is presented in [7] which is used here with some manipulations.

As mentioned in previous section, to solve Eq. (5), we need to find an approximation for fluxes at the boundaries (i.e., $F_{i-\frac{1}{2}}$ and $F_{i+\frac{1}{2}}$). A flux limiter scheme gives the following formulae for calculating the boundary fluxes [8]:

$$\begin{cases} \boldsymbol{F}^{FL}_{i+\frac{1}{2}} = \boldsymbol{F}^{low}_{i+\frac{1}{2}} - \Psi(\boldsymbol{r}_i)\left(\boldsymbol{F}^{low}_{i+\frac{1}{2}} - \boldsymbol{F}^{high}_{i+\frac{1}{2}}\right) \\ \boldsymbol{F}^{FL}_{i-\frac{1}{2}} = \boldsymbol{F}^{low}_{i-\frac{1}{2}} - \Psi(\boldsymbol{r}_{i-1})\left(\boldsymbol{F}^{low}_{i-\frac{1}{2}} - \boldsymbol{F}^{high}_{i-\frac{1}{2}}\right) \end{cases} \quad (9)$$

Here, $\Psi(\boldsymbol{r})$ is called the flux limiter function, $\boldsymbol{F}^{low}$ is the low-resolution flux (i.e., the flux calculated based on a low order scheme) and $\boldsymbol{F}^{high}$ is the high-resolution flux (i.e., the flux calculated based on a high order scheme). The value of $\boldsymbol{r}$ at point i is given by:

$$\boldsymbol{r}_i = \left(\frac{\partial \boldsymbol{q}}{\partial x}\right)_{i-\frac{1}{2}} \Big/ \left(\frac{\partial \boldsymbol{q}}{\partial x}\right)_{i+\frac{1}{2}} \quad (10)$$

If the grid is uniform, Eq. (10) reduces to:

$$\boldsymbol{r}_i = \frac{\boldsymbol{q}_i - \boldsymbol{q}_{i-1}}{\boldsymbol{q}_{i+1} - \boldsymbol{q}_i} \quad (11)$$

The package use Eq. (10) which provides the capability of non-uniform grids.

Several methods can be used as the low/high-resolution schemes. Our implementation uses the first order upwind (UW) and central difference (CD) as low-resolution and high-resolution schemes, respectively:

$$\begin{cases} \boldsymbol{F}^{low}_{i-\frac{1}{2}} = \boldsymbol{F}^{UW}\left(\boldsymbol{q}_{i-\frac{1}{2}}\right) \\ \boldsymbol{F}^{high}_{i-\frac{1}{2}} = \boldsymbol{F}^{CD}\left(\boldsymbol{q}_{i-\frac{1}{2}}\right) \end{cases} \quad (12)$$

One promising issue about the UW-CD combination is that it allows a generalization method called κ-scheme to model linear schemes as flux limiters [9].

The UW scheme considers the flow direction when determining the flux value at a boundary [10]: the flow at a boundary is calculated based on the value of conserved variable in the upstream cell (i.e., the cell that corresponds the flow:

$$\begin{cases} \boldsymbol{F}^{UW+}\left(\boldsymbol{q}_{i-\frac{1}{2}}\right) = \boldsymbol{f}(\boldsymbol{q}_{i-1}) \\ \boldsymbol{F}^{UW+}\left(\boldsymbol{q}_{i+\frac{1}{2}}\right) = \boldsymbol{f}(\boldsymbol{q}_i) \end{cases} \quad (13)$$

$$\begin{cases} \boldsymbol{F}^{UW-}\left(\boldsymbol{q}_{i-\frac{1}{2}}\right) = \boldsymbol{f}(\boldsymbol{q}_i) \\ \boldsymbol{F}^{UW-}\left(\boldsymbol{q}_{i+\frac{1}{2}}\right) = \boldsymbol{f}(\boldsymbol{q}_{i+1}) \end{cases} \quad (14)$$

The superscripts + and – are used for distinguishing between positive and negative flux direction.

In contrast, the CD does not consider the flux direction and simply uses the average value of adjacent cells to calculate the flux at a boundary:

$$\begin{cases} \boldsymbol{F}^{CD}\left(\boldsymbol{q}_{i-\frac{1}{2}}\right) = \boldsymbol{f}\left(\frac{\boldsymbol{q}_{i-1} + \boldsymbol{q}_i}{2}\right) \\ \boldsymbol{F}^{CD}\left(\boldsymbol{q}_{i+\frac{1}{2}}\right) = \boldsymbol{f}\left(\frac{\boldsymbol{q}_i + \boldsymbol{q}_{i+1}}{2}\right) \end{cases} \quad (15)$$

For hyperbolic problems the information propagates with finite speed as determined by the eigenvalues of the flux Jacobian [5]. Therefore, the positive eigenvalues of $\boldsymbol{A}$ corresponds to positive fluxes (i.e., left to right) and wise versa [11]. This suggests to separate the positive and negative fluxes using the concept of eigenvalues. Decomposition of the Jacobian matrix (i.e., $\boldsymbol{A}$) in terms of eigenvalue matrix ($\boldsymbol{\Lambda} = \mathrm{diag}\left(\lambda_1, \lambda_2, \dots, \lambda_{n_q}\right)$) and right eigenvector matrix ($\boldsymbol{V}$) yields:

$$\boldsymbol{AV} = \boldsymbol{V\Lambda} \quad (16)$$

As the PDE problem is assumed hyperbolic, the eigenvector matrix $\boldsymbol{V}$ is nonsingular, so that:

$$\boldsymbol{A} = \boldsymbol{V\Lambda V^{-1}} \quad (17)$$

Let us define the positive and negative eigenvalues as follows:

$$\lambda_k^+ \triangleq \max(\lambda_k, 0) = \frac{1}{2}(\lambda_k + |\lambda_k|) \quad (18)$$

$$\lambda_k^- \triangleq \min(\lambda_k, 0) = \frac{1}{2}(\lambda_k - |\lambda_k|) \quad (19)$$

for $1 \le k \le n_q$. Now, using above values, positive and negative eigenvectors are constructed as follows:

$$\boldsymbol{\Lambda}^+ \triangleq \begin{bmatrix} \lambda_1^+ & 0 & \dots & 0 \\ 0 & \lambda_2^+ & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_{n_q}^+ \end{bmatrix} \quad (20)$$

$$\boldsymbol{\Lambda}^{-} \triangleq \begin{bmatrix} \lambda_1^- & 0 & \dots & 0 \\ 0 & \lambda_2^- & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_{n_q}^- \end{bmatrix} \tag{21}$$

Finally, the positive and negative Jacobians are defined as:

$$\boldsymbol{A}^{+} \triangleq \boldsymbol{V}\boldsymbol{\Lambda}^{+}\boldsymbol{V}^{-1} \quad , \quad \boldsymbol{A}^{-} \triangleq \boldsymbol{V}\boldsymbol{\Lambda}^{-}\boldsymbol{V}^{-1} \tag{22}$$

It can easily be shown that:

$$\boldsymbol{A} = \boldsymbol{A}^{+} + \boldsymbol{A}^{-} \tag{23}$$

Now, to put all above equation together, we need to change the right-hand side of Eq. (8) as follows:

$$\frac{\boldsymbol{F}_{i-\frac{1}{2}} - \boldsymbol{F}_{i+\frac{1}{2}}}{\Delta x_i} = -\left(\frac{\partial \boldsymbol{f}(\boldsymbol{q})}{\partial x}\right)_i = -(\boldsymbol{A}\partial_x \boldsymbol{q})_i$$

$$\xrightarrow{yields} \quad \partial_t \boldsymbol{q}_i = -(\boldsymbol{A}\partial_x \boldsymbol{q})_i \tag{24}$$

Replacing Jacobian from Eq. (23) in Eq. (24) yields:

$$\partial_t \boldsymbol{q}_i = -\boldsymbol{A}_i^{+}\partial_x \boldsymbol{q}_i^{+} - \boldsymbol{A}_i^{-}\partial_x \boldsymbol{q}_i^{-} \tag{25}$$

where $\partial_x \boldsymbol{q}^{+}$ and $\partial_x \boldsymbol{q}^{-}$ are defined as:

$$\partial_x \boldsymbol{q}_i^{+} = \frac{\boldsymbol{q}_{i+\frac{1}{2}}^{+} - \boldsymbol{q}_{i-\frac{1}{2}}^{+}}{\Delta x_i} \tag{26}$$

$$\partial_x \boldsymbol{q}_i^{-} = \frac{\boldsymbol{q}_{i+\frac{1}{2}}^{-} - \boldsymbol{q}_{i-\frac{1}{2}}^{-}}{\Delta x_i} \tag{27}$$

Using UW-CD combination the values of $\boldsymbol{q}_{i\mp\frac{1}{2}}^{-/+}$ are as follows:

$$\boldsymbol{q}_{i+\frac{1}{2}}^{-} = \boldsymbol{q}_i + \Psi(\boldsymbol{r}_i)\left(\frac{\boldsymbol{q}_{i+1} - \boldsymbol{q}_i}{2}\right) \tag{28}$$

$$\boldsymbol{q}_{i-\frac{1}{2}}^{-} = \boldsymbol{q}_{i-1} + \Psi(\boldsymbol{r}_{i-1})\left(\frac{\boldsymbol{q}_i - \boldsymbol{q}_{i-1}}{2}\right) \tag{29}$$

$$\boldsymbol{q}_{i+\frac{1}{2}}^{+} = \boldsymbol{q}_{i+1} - \Psi(\boldsymbol{r}_i)\left(\frac{\boldsymbol{q}_{i+1} - \boldsymbol{q}_i}{2}\right) \tag{30}$$

$$\boldsymbol{q}_{i-\frac{1}{2}}^{+} = \boldsymbol{q}_i - \Psi(\boldsymbol{r}_{i-1})\left(\frac{\boldsymbol{q}_i - \boldsymbol{q}_{i-1}}{2}\right) \tag{31}$$

3.4. Generalization

In previous subsection, the methodology derived for solving Eq. (5). However, solving PDE problem of Eq. (3) is straightforward. As we need the values of $\partial_t \boldsymbol{q}$ at each time step, we can apply the same procedure explained for $\partial_x \boldsymbol{f}(\boldsymbol{q})$, on $\partial_y \boldsymbol{g}(\boldsymbol{q})$ and $\partial_z \boldsymbol{h}(\boldsymbol{q})$. In addition, in semi-linear problems, the source term depends only on the values of $\boldsymbol{q}$ and x. Therefore, at each time step, the value of the source term can easily be calculated. This means, the methodology is applicable for solving Eq. (5) without any further manipulation. The complete semi-discretization formula (i.e., generalization of Eq. (25)) is as follows:

$$\begin{aligned} \partial_t \boldsymbol{q}_{ijk} = \boldsymbol{\sigma}(\boldsymbol{q}_{ijk}) &- \boldsymbol{A}_{ijk}^{+}\partial_x \boldsymbol{q}_{ijk}^{+} - \boldsymbol{A}_{ijk}^{-}\partial_x \boldsymbol{q}_{ijk}^{-} \\ &- \boldsymbol{B}_{ijk}^{+}\partial_y \boldsymbol{q}_{ijk}^{+} - \boldsymbol{B}_{ijk}^{-}\partial_y \boldsymbol{q}_{ijk}^{-} \\ &- \boldsymbol{C}_{ijk}^{+}\partial_z \boldsymbol{q}_{ijk}^{+} - \boldsymbol{C}_{ijk}^{-}\partial_z \boldsymbol{q}_{ijk}^{-} \end{aligned} \tag{32}$$

where $\boldsymbol{B}$ and $\boldsymbol{C}$ are Jacobians of flux functions $\boldsymbol{B}$ and $\boldsymbol{C}$, respectively. In addition, in case of problems given as Eq. (4), the Jacobians are already given which reduce a discretization step for computing them numerically.

3.5. Flux limiter functions

Several limiter functions can be found in books and scientific papers. The interested reader may refer to [7], which reviews a wide range of limiter functions in a unified approach. A list of the limiter functions implemented in the package given in Table 1. For each limiter function, its syntax in the package and the formula for $\Psi(r)$ is provided. The first six schemes are linear; while the rest of them are non-linear schemes.

Note that, the developed package has the capability to accept any user-defined limiter function.

Table 1: List of limiter functions

Syntax	Name	Flux limiter formula
uw1	First-order upwind	$\Psi(r) = 0$
uw2	Second order upwind	$\Psi(r) = 1$
uw3	Cubic upwind	$\Psi(r) = \frac{2}{3}r + \frac{1}{3}$
uw4	Quadratic upwind	$\Psi(r) = \frac{3}{4}r + \frac{1}{4}$
scd	Second-order central difference	$\Psi(r) = r$
fr	Fromm	$\Psi(r) = \frac{1}{2}r + \frac{1}{2}$
kn	Koren	$\Psi(r) = \max\left[0, min\left(2r, \min\left(\frac{1+2r}{3}, 2\right)\right)\right]$
sb	Superbee	$\Psi(r) = \max[0, min(2r, 1), min(r, 2)]$
mm	Minmod	$\Psi(r) = \max[0, min(r, 1)]$
mu	MUSCL	$\Psi(r) = \max\left[0, min\left(2r, \frac{r+1}{2}, 2\right)\right]$
ha	Harmonic	$\Psi(r) = \frac{r + \lvert r\rvert}{r+1}$
va1	van Albada 1	$\Psi(r) = \frac{r(r+1)}{r^2+1}$
va2	van Albada 2	$\Psi(r) = \frac{2r}{r^2+1}$
vl	van Leer	$\Psi(r) = \frac{r + \lvert r\rvert}{1 + \lvert r\rvert}$
op	OSPRE	$\Psi(r) = \frac{3r(r+1)}{2(r^2+r+1)}$
hc	HCUS	$\Psi(r) = \frac{1.5(r + \lvert r\rvert)}{r+2}$
hq	HQUICK	$\Psi(r) = \frac{2(r + \lvert r\rvert)}{r+3}$
cm	CHARM	$\Psi(r) = \max\left[0, \frac{r(3r+1)}{(r+1)^2}\right]$
mc	Monotonized central	$\Psi(r) = \max[0, min(2r, 0.5(r+1), 2)]$
sm	Smart	$\Psi(r) = \max[0, min(2r, (0.75r + 0.25), 4)]$
um	UMIST	$\Psi(r) = \max[0, min(2r, (0.75r + 0.25), (0.25r + 0.75), 2)]$

4. Developed package

The package "FVM_PDEsolver.jl" is developed for Julia programming language to solve first-order semi-linear hyperbolic PDE systems using high-resolution flux limiters. It is capable of handling PDEs given as Eq. (3) and Eq. (4). The package uses the semi-discretization method and calculates the right-hand side of Eq. (32). Then, it uses ODE solvers from "DifferentialEquations.jl" to find the final solution of the PDE system. If the equation is in the form of Eq. (4), then the Jacobians of flux functions are calculated numerically using "ForwardDiff.jl" pachage. To calculate the eigenvalues and eigenvectors, "LinearAlgebra.jl" package is used.

4.1. Algorithm

The algorithm is exactly what was discussed as the flux limiter scheme in the previous section. Here, a detailed description of the algorithm is presented:

Step 0: Initialize the gird values for $t = 0$: the current values of $\boldsymbol{q}$ vector is set to the initial values provided by the user.

Step 1: Calculate $\frac{\partial \boldsymbol{q}}{\partial t}$ for each cell in current time step based on the current values of $\boldsymbol{q}$ are as follows (expressed for 1-D case[6]):

1 Calculate $\left(\frac{\partial q}{\partial x}\right)_{i+\frac{1}{2}} = \frac{q_{i+1}-q_i}{x_{i+1}-x_i}$ at all cells boundaries

2 Calculate $\boldsymbol{r}_i = \left(\frac{\partial \boldsymbol{q}}{\partial x}\right)_{i-\frac{1}{2}} \Big/ \left(\frac{\partial \boldsymbol{q}}{\partial x}\right)_{i+\frac{1}{2}}$ for each cell.

3 Calculate $\boldsymbol{q}_{i\mp\frac{1}{2}}^{+/-}$ using Eq. 28 to 31 at all cell boundaries.

4 Calculate $\partial_x \boldsymbol{q}_i^+ = \frac{\boldsymbol{q}_{i+\frac{1}{2}}^+ - \boldsymbol{q}_{i-\frac{1}{2}}^+}{\Delta x}$ and $\partial_x \boldsymbol{q}_i^- = \frac{\boldsymbol{q}_{i+\frac{1}{2}}^- - \boldsymbol{q}_{i-\frac{1}{2}}^-}{\Delta x}$ for each cell.

5 Calculate the Jacobian matrixes for each cell:

5-1 If the equation is in form **Error! Reference source not found.** then by the given function (i.e., $\boldsymbol{A}_i = \boldsymbol{J}_f(\boldsymbol{q}_i)$).

5-2 If the equation is in form **Error! Reference source not found.** then by using numerical method (i.e., $\boldsymbol{A}_i = \frac{\partial f(q)}{\partial \boldsymbol{q}}\Big|_{\boldsymbol{q}=\boldsymbol{q}_i}$).

6 Calculate the eigenvalues and eigenvectors matrices numerically for each cell (i.e., $\boldsymbol{\Lambda}_i$ and $\boldsymbol{V}_i$).

7 Calculate positive/negative eigenvalue matrices for each cell (i.e., $\boldsymbol{\Lambda}_i^+ = \frac{1}{2}(\boldsymbol{\Lambda}_i + |\boldsymbol{\Lambda}_i|)$ and $\boldsymbol{\Lambda}_i^- = \frac{1}{2}(\boldsymbol{\Lambda}_i - |\boldsymbol{\Lambda}_i|)$ where absolute sign denotes the elementwise absolute value).

8 Calculate positive and negative Jacobian matrices for each cell (e.g., $\boldsymbol{A}_i^+ = \boldsymbol{V}_i \boldsymbol{\Lambda}_i^+ \boldsymbol{V}_i^{-1}$ and $\boldsymbol{A}_i^- = \boldsymbol{V}_i \boldsymbol{\Lambda}_i^- \boldsymbol{V}_i^{-1}$).

9 Calculate $\boldsymbol{S}_i = \boldsymbol{S}(\boldsymbol{q}_i)$ for each cell.

10 Calculate $\frac{\partial \boldsymbol{q}_i}{\partial t} = -(\boldsymbol{A}_i^+ \partial_x \boldsymbol{q}_i^+ + \boldsymbol{A}_i^- \partial_x \boldsymbol{q}_i^-) + \boldsymbol{S}_i$

Step 2: Use the ODE solver to solve $\frac{\partial \boldsymbol{q}}{\partial t} = \cdots$ find $\boldsymbol{q}$ for the next time step.

Step 3: Set the current values of $\boldsymbol{q}$ to the values found in step 1.

Step 4: Re-do step 1 to step 3 until reaching t_{end}.

5. Evaluation of package performance

In this section, some example PDE problems are defined and solved using the package.

5.1. Moving wave with constant velocity

A simple step like wave which propagate with constant speed in a 1-D space ($x \in [0,1]$) is defined by the following PDE:

$$\frac{\partial q}{\partial t} + a\frac{\partial q}{\partial x} = 0 \qquad (33)$$

with initial condition:

$$q(t=0,x) = \begin{cases} 1.0 & x \le 0.5 \\ 0.0 & x > 0.5 \end{cases}$$

where $a = 0.5$ is the propagation speed in x direction. As $a > 0$, the exact solution at each time step is defined by shifting the wave position by at in x direction. Assume we want to find the solution at $t = 0.2$. To solve the problem a uniform grid with 0.01 intervals is used. Then, the problem is solved by the package and the results using uw1, uw2, scd, kn, and sb schemes are shown in Figure 2.

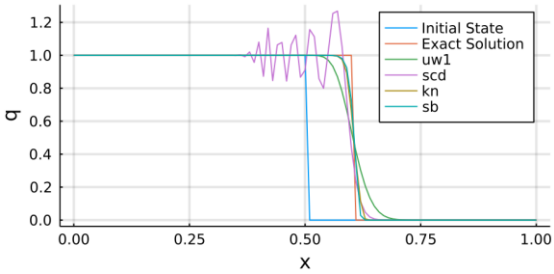

Figure 2: Solution of moving wave with constant velocity problem using different schemes

As expected, the worst scheme is the scd; while the best is sb in this example. The same problem is also solved using all nonlinear schemes available in the package. For better illustration, the values in range [0.5, 0.7] are plotted in Figure 3.

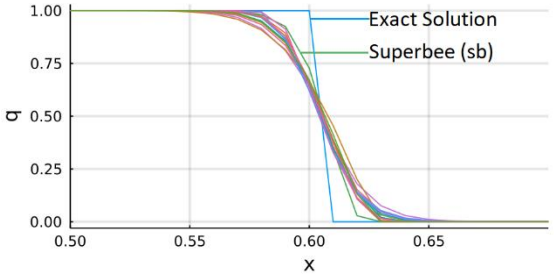

Figure 3: Solution of moving wave with constant velocity problem using all available nonlinear flux limiters

Although the plot in Figure 3 is a little nasty, it shows all nonlinear flux limiter functions have similar performances on this particular problem. However, the sb method shows a slightly better performance than others.

5.2. Euler problem: rarefaction-shock case

[6] For 3-D case, the index i should be changed to ijk and the steps 1-8 should be done for y and z axes as well. Step 10 also calculates Eq. (32).

The Euler equations are a collection of nonlinear hyperbolic conservation rules that regulate the dynamics of compressible fluids while ignoring the effects of body forces and viscous stress [12]. The 1-D Euler equation is expressed by Eq. (5); where $\boldsymbol{q}$ and $\boldsymbol{f}(\boldsymbol{q})$ are given as follows:

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \rho \\ \rho u \\ \rho e \end{bmatrix} \quad , \quad \boldsymbol{f}(\boldsymbol{q}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u h \end{bmatrix} \tag{34}$$

where: $h = e + \frac{p}{\rho}$, $p = \rho(\gamma - 1)\left(e - \frac{1}{2}u^2\right)$

In above equations, ρ is the density, p is the pressure, u is the velocity in x direction, e is the internal energy, h is the static enthalpy, and γ is the ratio of specific heats which is a constant scalar. Although $\boldsymbol{f}(\boldsymbol{q})$ is not expressed explicitly in terms of $\boldsymbol{q}$, it is possible to reformulate it doing some maths:

$$\boldsymbol{f}(\boldsymbol{q}) = \begin{bmatrix} q_2 \\ \frac{q_2^2}{q_1} + q_1(\gamma - 1)\left(\frac{q_3}{q_1} - \frac{1}{2}\left(\frac{q_2}{q_1}\right)^2\right) \\ q_2\left(\frac{q_3}{q_1} + (\gamma - 1)\left(\frac{q_3}{q_1} - \frac{1}{2}\left(\frac{q_2}{q_1}\right)^2\right)\right) \end{bmatrix} \tag{35}$$

Assume the domain $x \in [0,1]$ and following initial conditions:

$$\rho(x, t = 0) = \begin{cases} \rho_L & 0.0 \le x < x_0 \\ \rho_R & x_0 < x \le 1.0 \end{cases}$$

$$u(x, t = 0) = \begin{cases} u_L & 0.0 \le x < x_0 \\ u_R & x_0 < x \le 1.0 \end{cases}$$

$$p(x, t = 0) = \begin{cases} p_L & 0.0 \le x < x_0 \\ p_R & x_0 < x \le 1.0 \end{cases}$$

Using the following parameters, the PDE problem represents a rarefaction-shock case:

$$x_0 = 0.5, \ \Gamma = 1.4,$$
$$\rho_L = 1.0, \ \rho_R = 0.125,$$
$$u_L = 0.0, \ u_R = 0.0,$$
$$p_L = 1.0, \ p_R = 0.1$$

It is desired to find the solution at $t = 0.2$. The exact analytical solution is given in [13]. To solve the problem a uniform grid with 0.01 intervals is used. Figure 4 and Figure 5 shows the solution using uw1 and kn schemes, respectively.

5.3. Shallow water equation: supercritical case

The 1-D shallow water equation is as follows [14]:

$$h_t + (hu)_x = 0$$
$$(hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x = -ghB_x \tag{36}$$

Here, g is the gravitational constant, h is the water depth, u is the mean velocity in x direction, and $B(x)$ is the waterway bottom topography. Although the source term in Eq. (36) has a differential term, as the topography is not changing, the values of B_x are constant and can be pre-calculated.

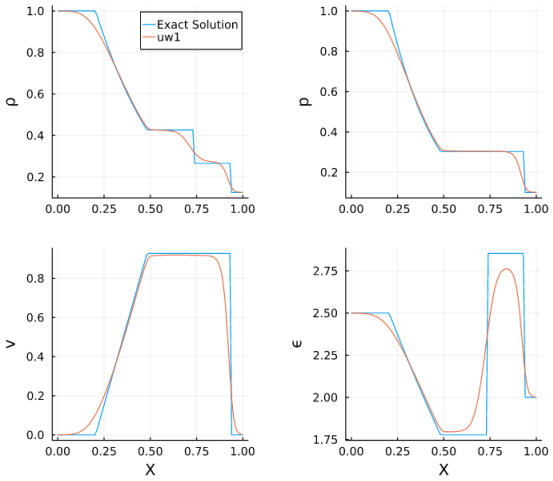

Figure 4: Solution of Euler problems: rarefaction-shock case using uw1 scheme

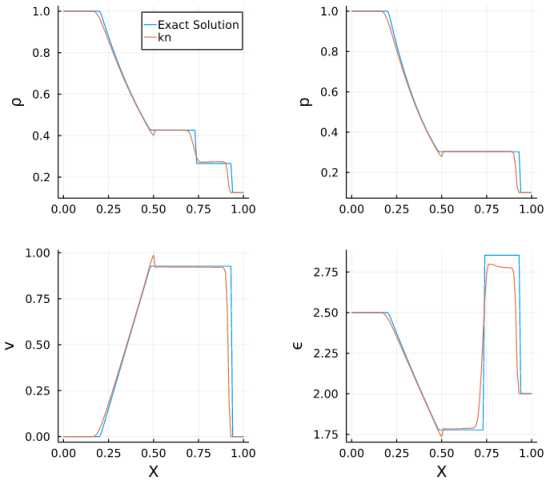

Figure 5: Solution of Euler problems: rarefaction-shock case using kn scheme

To match Eq. (36) with Eq. (3), the following equations should be considered:

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} h \\ hu \end{bmatrix}$$

$$\boldsymbol{f}(\boldsymbol{q}) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix} = \begin{bmatrix} q_2 \\ \frac{q_2^2}{q_1} + \frac{1}{2}gq_1^2 \end{bmatrix} \tag{37}$$

$$\boldsymbol{\sigma}(\boldsymbol{q}) = \begin{bmatrix} 0 \\ -ghB_x \end{bmatrix} = \begin{bmatrix} 0 \\ -gq_1B_x \end{bmatrix}$$

In the definition of $\boldsymbol{\sigma}(\boldsymbol{q})$, g is a constant scalar and B_x is a known function of x. So, the Eq. (36) is semi-linear and the package is capable of solving it. Note that, in some contexts the Eq. (36) is referred to as the dam-break wave equation.

Assume the problem domain $x \in [-10,10]$ and the following initial conditions:

$$h(x,t=0)=\begin{cases}h_L & -10\le x<0\\ h_R & 0<x\le 10\end{cases}$$

$$u(x,t=0)=\begin{cases}u_L & -10\le x<0\\ u_R & 0<x\le 10\end{cases}$$

$$B(x)=\begin{cases}B_L & -10\le x<0\\ B_R & 0<x\le -10\end{cases}$$

The following parameter setting makes the super critical case:

$$g=9.8,\ \ h_L=4.0,\ \ h_R=1.0,$$

$$u_L=-10.0,\ \ u_R=-6.0,\ \ B_L=0.0,\ \ B_R=1.0$$

It is desired to find the solution at $t=0.2$. Although the exact solution is not available, a solution to this problem can be found in [14]. To solve the problem a uniform grid with 0.05 intervals is used. Figure 6 shows the solution using kn schemes.

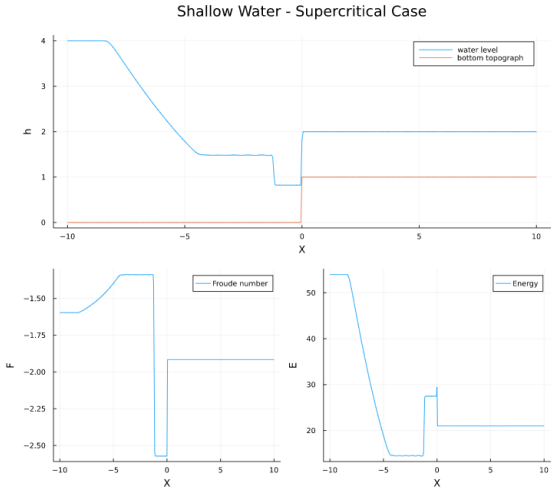

Figure 6: Solution of Shallow water equation: supercritical case using kn scheme

Comparing the result with those presented in [14] approve the performance of the package.

5.4. Granulation process: growth by layering

The initiative of this work was the granulation process modeling and finding the numerical solution for the growth by layering equation. The equation according to [15] is as follows:

$$\frac{\partial}{\partial t}n_b(V_p,t)+G(t)\frac{\partial}{\partial V_p}n_b(V_p,t)=0 \qquad (38)$$

where, V_p is the internal coordination of particles volumes, $G(t)$ is the growth rate, and n_b is the particle size distribution function. Eq. (38) represents a square wave moving with speed $G(t)$. Therefore, the for $G(t)=cte$. the exact solution is available.

As the Jacobian is given outside of the derivative term, this equation is in the form of Eq. (4). Assume the problem domain $V_p\in[0,400]$ and the following initial conditions:

$$n_b(V_p,t=0)=\begin{cases}10\times10^4 & 15\le x<50\\ 0 & \text{otherwise}\end{cases}$$

Given $G(t)=1$, the solution at $t=150$ is desired. To solve the problem a uniform grid with $\Delta V_p=5$ is used. Figure 7 shows the solution using uw1, scd, and kn schemes.

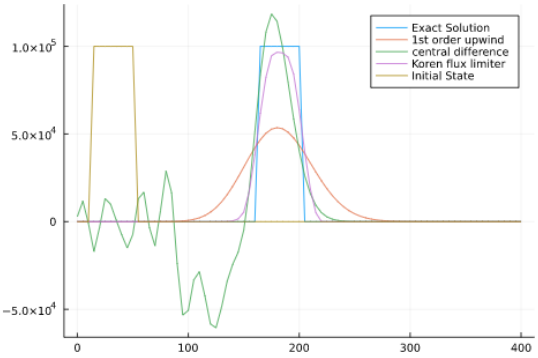

Figure 7: Solution of granulation process: growth by layering problem using different schemes

Comparing the results with the same schemes used in [15], approve the performance of the package. One difference between our results and those given in [15] is that they limited the values of n_b to be always non-negative. This is not done in our solution which caused the scd scheme find negative values for n_b. This issue will be resolved in further improvements of the package.

6. Summary and Discussions

The PDE problems are challenging to solve. High-resolution schemes provide high-order accurate numerical approximation of PDEs, especially in presence of shocks or discontinuities. In this paper, flux limiter schemes, which are among the high-resolution methods, was reviewed in detail. The derived equations made the basis for development of a package named "FVM_PDEsolver.jl" in Julia programming language. The algorithm of solving mechanism is explained and its performance is shown in several examples. The package is an open-source program available in GitHub. The authors aim to improve the package in many aspects in future. Some ideas for improvement are: triangular grids, limiting the quantities' values, adding more schemes, etc. One interesting observation in this paper was the high accuracy of Superbee flux limiter which was the best among all available limiters in the package.

References

[1] A. D. Polyanin, W. E. Schiesser, and A. I. Zhurov, "Partial differential equation," *Scholarpedia*, vol. 3, no. 10, pp. 4605, 2008.

[2] B. Van Leer, "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method," *Journal of*

computational Physics, vol. 32, no. 1, pp. 101-136, 1979.
[3] C.-W. Shu, "Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws," *Advanced numerical approximation of nonlinear hyperbolic equations*, pp. 325-432, 1998.
[4] C. Hirsch, *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*, pp. 105-140: Elsevier, 2007.
[5] R. J. LeVeque, *Finite volume methods for hyperbolic problems*: Cambridge university press, 2002.
[6] S. Godunov, and I. Bohachevsky, "Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics," *Matematičeskij sbornik,* vol. 47, no. 3, pp. 271-306, 1959.
[7] N. P. Waterson, and H. Deconinck, "Design principles for bounded higher-order convection schemes–a unified approach," *Journal of Computational Physics,* vol. 224, no. 1, pp. 182-207, 2007.
[8] J. Pietrzak, "The Use of TVD Limiters for Forward-in-Time Upstream-Biased Advection Schemes in Ocean Modeling," *Monthly Weather Review,* vol. 126, no. 3, pp. 812-830, 1998.
[9] P. L. Roe, "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of computational physics,* vol. 43, no. 2, pp. 357-372, 1981.
[10] H. K. Versteeg, and W. Malalasekera, *An introduction to computational fluid dynamics: the finite volume method*, 2nd ed., pp. 134-178: Pearson education, 2007.
[11] C. Hirsch, *Numerical computation of internal and external flows: Computational Methods for lnviscid and Viscous Flows*, pp. 408-492: Wiley, 1990.
[12] E. F. Toro, *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*: Springer Science & Business Media, 2013.
[13] F. Lora-Clavijo, J. Cruz-Pérez, F. Siddhartha Guzmán, and J. Gonzalez, "Exact solution of the 1D riemann problem in Newtonian and relativistic hydrodynamics," *Revista mexicana de física E,* vol. 59, no. 1, pp. 28-50, 2013.
[14] S. Jin, and X. Wen, "An efficient method for computing hyperbolic systems with geometrical source terms having concentrations," *Journal of Computational Mathematics*, pp. 230-249, 2004.
[15] L. Vesjolaja, B. Glemmestad, and B. Lie, "Solving the population balance equation for granulation processes: particle layering and agglomeration," 2021.