INL REPORT

INL/EXT-16-39214
Unlimited Release
Printed August 2016

BIGHORN Computational Fluid Dynamics Theory, Methodology, and Code Verification & Validation Benchmark Problems

Yidong Xia
David Andrs
Richard C. Martineau

Revised February 7, 2017

Prepared by
Idaho National Laboratory
Idaho Falls, Idaho 83415

The Idaho National Laboratory is a multiprogram laboratory operated by
Battelle Energy Alliance for the United States Department of Energy
under DOE Idaho Operations Office. Contract DE-AC07-05ID14517.

Idaho National Laboratory

INL/EXT-16-39214
Unlimited Release
Printed August 2016

BIGHORN Computational Fluid Dynamics Theory, Methodology, and Code Verification & Validation Benchmark Problems

Yidong Xia
Department of Materials Science and Engineering
Energy, Environment Science & Technology Directorate
Idaho National Laboratory
P.O. Box 3560
Idaho Falls, ID 83415-2025

David Andrs
Department of Modeling and Simulation
Nuclear Science & Technology Directorate
Idaho National Laboratory
P.O. Box 3825
Idaho Falls, ID 83415-2025

Richard C. Martineau
Department of Modeling and Simulation
Nuclear Science & Technology Directorate
Idaho National Laboratory
P.O. Box 3840
Idaho Falls, ID 83415-2025

Revised February 7, 2017

Abstract

This document presents the theoretical background for a hybrid finite-element / finite-volume fluid flow solver, namely BIGHORN, based on the Multiphysics Object Oriented Simulation Environment (MOOSE) computational framework developed at the Idaho National Laboratory (INL). An overview of the numerical methods used in BIGHORN are discussed and followed by a presentation of the formulation details. The document begins with the governing equations for the compressible fluid flow, with an outline of the requisite constitutive relations. A second-order finite volume method used for solving the compressible fluid flow problems is presented next. A Pressure-Corrected Implicit Continuous-fluid Eulerian (PCICE) formulation for time integration is also presented. The multi-fluid formulation is being developed. Although multi-fluid is not fully-developed, BIGHORN has been designed to handle multi-fluid problems. Due to the flexibility in the underlying MOOSE framework, BIGHORN is quite extensible, and can accommodate both multi-species and multi-phase formulations. This document also presents a suite of verification & validation benchmark test problems for BIGHORN. The intent for this suite of problems is to provide baseline comparison data that demonstrates the performance of the BIGHORN solution methods on problems that vary in complexity from laminar to turbulent flows. Wherever possible, some form of solution verification has been attempted to identify sensitivities in the solution methods, and suggest best practices when using BIGHORN.

Acknowledgment

The work described in this document has been authored by Battelle Energy Alliance, LLC under the Idaho National Laboratory's Laboratory Directed Research and Development (LDRD) project entitled "Development of a Multiphysics Algorithm for Analyzing the Integrity of Nuclear Reactor Containment Vessels Subjected to Extreme Thermal and Overpressure Loading Conditions" (Track No. 14-104) with the U.S. Department of Energy. Dr. Richard C. Martineau served as the Principal Investigator of this project, whose support is greatly appreciated by the lead author.

Contents

Appendix

Figures

Tables

1 Introduction

1.1 Code Overview

The work described in this report documents the development of a Computational Fluid Dynamics (CFD) simulation code, namely BIGHORN, for modeling and predicting the multi-component, multi-phase fluid dynamics in multi-dimensions. The code is developed on a parallel Multiphysics Object Oriented Simulation Environment (MOOSE) computational framework developed at Idaho National Laboratory (INL) for providing finite element / discontinuous Galerkin solutions of coupled system of nonlinear partial differential equations. MOOSE was originally developed for modeling multiphysics problems often encountered in nuclear reactor and fuel performance analysis. The main purpose of this report is to present the theoretical background for the BIGHORN hybrid finite-element / finite-volume fluid flow solver. In the sections that follow, the governing equations are presented with an outline of the requisite constitutive relations. An overview of the numerical methods used in BIGHORN are discussed and followed by a presentation of the formulation details. A second-order finite volume method used for solving the compressible fluid flow problems is presented next. A Pressure-Corrected Implicit Continuous-fluid Eulerian (PCICE) formulation for time integration is also presented. The multi-fluid formulation is being developed. Although multi-fluid is not fully-developed, BIGHORN has been designed to handle multi-fluid problems. Due to the flexibility in the underlying MOOSE framework, BIGHORN is quite extensible, and can accommodate both multi-species and multi-phase formulations. This document also presents a suite of verification & validation benchmark test problems for BIGHORN. The intent for this suite of problems is to provide baseline comparison data that demonstrates the performance of the BIGHORN solution methods on problems that vary in complexity from laminar to turbulent flows. Wherever possible, some form of solution verification has been attempted to identify sensitivities in the solution methods, and suggest best practices when using BIGHORN.

1.2 Architecture and Design

BIGHORN has been designed for the simulation of single-/multi-species, single-/multi-phase fluid flows. The architecture of BIGHORN has a plug-and-play modular design structure by representing each piece of the residual term in a weak form of the governing PDEs as a “Kernel”. Kernels may be coupled together to achieve different application goals. All kernels are required to supply a residual, which usually involves summing products of finite element shape functions. The basic architecture of the code allows convenient coupling of different processes and incorporation of new physics.

Fig. 1 shows the basic architecture of the BIGHORN code, with the Kernels at the uppermost level, directly underlain by the numerical framework and solver libraries used to couple the Kernels and perform the simulations. Currently primary Kernels (primary variables) have been written to describe the following physics of fluid:

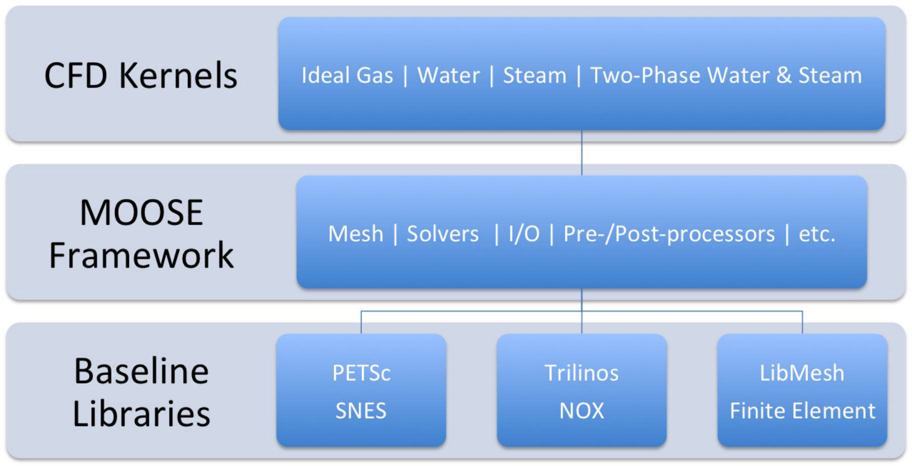

CFD Kernels
Ideal Gas | Water | Steam | Two-Phase Water & Steam
MOOSE Framework
Mesh | Solvers | I/O | Pre-/Post-processors | etc.
Baseline Libraries
PETSc SNES
Trilinos NOX
LibMesh Finite Element

Figure 1. The object-oriented architecture used to develop the BIGHORN code.

- Single-phase flow of ideal gas, or liquid water, or steam;
- Presumptive two-phase flow of water and steam.

An auxiliary variable system has been built into BIGHORN to handle solving most all of the derived quantities and variables that are dependent on the primary kernels mentioned above. The number of auxiliary kernels needed for a given simulation depends on the choice of primary variables and whether they are formulated in terms of single-phase or two-phase. In general, a simulation run with the two-phase formulation requires the most auxiliary kernels and has the highest computational burden. The auxiliary kernels consist of

- Equation-of-State (EOS) calculations;
  - Ideal gas (for dry air)
  - Stiffened gas (for single-phase water or steam)
  - IAPWS-1997 formulation for two-phase water and steam
- Pressure
- Temperature

- Velocity components
- Speed of sound
- Mach number
- Enthalpy
- Entropy (if possible)

In addition to the primary and auxiliary physics kernels, other kernels are required for the mesh, material properties (and some additional supporting calculations), boundary conditions, code execution / solver parameters, and data output.

2 Governing Equations of Fluid Dynamics

2.1 Navier–Stokes Equations

The Navier-Stokes equations governing unsteady compressible viscous flows can be expressed as

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \boldsymbol{F}(\mathbf{U}) = \nabla \cdot \boldsymbol{G}(\mathbf{U}) + \mathbf{S}(\mathbf{U}) \tag{1}$$

where $\boldsymbol{F}$ is the advective (inviscid) flux tensor, $\boldsymbol{G}$ is the diffusive (viscous) flux tensor, $\boldsymbol{S}$ is the vector of source term, and ∇ is the divergence operator.

The conservative variable vector $\mathbf{U}$ is defined by

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix} \tag{2}$$

where ρ, p, and E denote the density, pressure, and specific total energy of the fluid, respectively, and u, v, and w are the velocity components of the flow in the coordinate direction x, y and z. The pressure can be computed from the equation of state

$$p = (\gamma - 1)\rho \left(E - \frac{1}{2}(u^2 + v^2 + w^2) \right) \tag{3}$$

which is valid for perfect gas, and the ratio of the specific heats γ is assumed to be constant and equal to 1.4. Furthermore, the specific total enthalpy H is defined as

$$H = E + \frac{p}{\rho} \tag{4}$$

In Eq. (1), the advective (inviscid) flux tensor $\boldsymbol{F} = (\mathbf{F}_x, \mathbf{F}_y, \mathbf{F}_z)$ is defined by

$$\mathbf{F}_x = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(\rho E + p) \end{bmatrix} \quad \mathbf{F}_y = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \\ v(\rho E + p) \end{bmatrix} \quad \mathbf{F}_z = \begin{bmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + p \\ w(\rho E + p) \end{bmatrix} \tag{5}$$

and the viscous flux tensor $\boldsymbol{G}$ is defined by

$$\mathbf{G}_x = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + q_x \end{bmatrix}$$
$$\mathbf{G}_y = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + q_y \end{bmatrix} \tag{6}$$
$$\mathbf{G}_z = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + q_z \end{bmatrix}$$

where the viscous stress tensor $\boldsymbol{\tau}$ is expressed as

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{bmatrix} \tag{7}$$

The Newtonian fluid with the Stokes hypothesis is valid under the current framework, since only air is considered. Thus $\boldsymbol{\tau}$ is symmetric and the tensor is a linear function of the velocity gradients

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij} \tag{8}$$

where δ_{ij} is the Kronecker delta function, and μ represents the molecular viscosity coefficient (often referred to as dynamic viscosity coefficient as well), which can be determined through Sutherland's law

$$\frac{\mu}{\mu_0} = \left(\frac{T}{T_0}\right)^{\frac{3}{2}} \frac{T_0 + S}{T + S} \tag{9}$$

where μ_0 denotes the viscosity coefficient at the reference temperature T_0, and S is a constant which is assumed the value $S = 110K$. The temperature of the fluid T is determined by

$$T = \frac{P}{\rho R} \tag{10}$$

where R denotes the universal gas constant for perfect gas.

The heat flux vector q_j, which is formulated according to Fourier's law, is given by

$$q_j = -\lambda\frac{\partial T}{\partial x_j} \tag{11}$$

where λ is the thermal conductivity coefficient and expressed as

$$\lambda = \frac{\mu c_p}{Pr} \tag{12}$$

where c_p is the specific heat capacity at constant pressure and Pr is the nondimensional laminar Prandtl number, which is taken as 0.7 for air.

2.2 Euler Equations

If the effect of viscosity and thermal conduction as well as the source term are neglected in Eq. (1), then we arrived at the Euler equations expressed as below, which govern unsteady compressible inviscid flows

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \boldsymbol{F}(\mathbf{U}) = 0 \tag{13}$$

2.3 Nondimensionalization

The governing equations are often put into the nondimensional form. The advantage in doing so is that the characteristic parameters such as Mach number, Reynolds number, and Prandtl number can be varied independently. Also, by nondimensionalizing the governing equations, the flow variables are "normalized", so that their values fall between certain prescribed limits such as 0 and 1. Many different nondimensionalizing procedures are possible. In this work, we use the following four reference variables: length, density, velocity and temperature. The choice of each reference variable is summarized in Table 1.

Table 1. Reference variables for nondimensionalization of the governing equations

Variable	Reference
Length L_{ref}	Problem dependent (cylinder diameter, plate length, etc) d, l
Density ρ_{ref}	Freestream density ρ_∞
Velocity V_{ref},	Freestream speed of sound a_∞
Temperature T_{ref}	Freestream temperature T_∞

The nondimensional variables are denoted by an overbar

$$\begin{aligned}
\bar{L} &= \frac{L}{L_{\text{ref}}}, \\
\bar{\rho} &= \frac{\rho}{\rho_\infty}, \\
\bar{u} &= \frac{u}{a_\infty}, \\
\bar{v} &= \frac{v}{a_\infty}, \\
\bar{w} &= \frac{w}{a_\infty}, \\
\bar{T} &= \frac{T}{T_\infty},
\end{aligned}$$

and accordingly, the derived normalized variables are expressed in the following manner

$$\begin{aligned}
\bar{p} &= \frac{p}{\rho_\infty a_\infty^2}, \\
\bar{h} &= \frac{h}{a_\infty^2}, \\
\bar{c}_p &= \frac{c_p}{a_\infty^2/T_\infty}, \\
\bar{\mu} &= \frac{\mu}{\rho_\infty a_\infty}.
\end{aligned}$$

It is also trivial to derive the nondimensional equation of state as

$$\bar{p} = \frac{1}{\gamma}\bar{\rho}\bar{T}$$

The freestream Mach number M_∞ is defined as

$$M_\infty = \frac{V_\infty}{a_\infty}$$

The freestream Reynolds number Re_∞ is determined as

$$Re_\infty = \frac{\rho_\infty a_\infty L_{\text{ref}}}{\mu_\infty}$$

The Prandtl number is written as

$$Pr = \frac{\mu_\infty c_p}{\lambda}$$

In the normalized governing equations, the nondimensional viscous stress tensor is

$$\bar{\tau}_{ij} = \bar{\mu}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} - \frac{2}{3}\frac{\partial \bar{u}_k}{\partial x_k}\delta_{ij}\right)$$

and the nondimensional heat flux $\bar{q}_j$ vector is

$$\bar{q}_j = -\bar{\mu}\bar{c}_p \frac{1}{Pr}\frac{\partial \bar{T}}{\partial x_j}$$

The nondimensional molecular viscosity coefficient $\bar{\mu}$ is computed with the dimensionless Sutherland's law

$$\bar{\mu} = \frac{M_\infty}{Re_\infty}\bar{T}^{\frac{3}{2}}\frac{1+S/T_\infty}{\bar{T}+S/T_\infty}$$

For the installation of a specific flow problem, the nondimensional input parameters include two fixed-value quantities $\bar{\rho}_\infty = 1$ and $a_\infty = 1$, and five user-adjustable quantities: M_∞, angle of attack α, yaw angle β, Re_∞ and Pr. With these inputs, a uniform flow field is prescribed for a steady-state problem at the initialization stage and the corresponding conservative variables are

$$\begin{aligned}
\bar{\rho}_\infty &= 1, \\
\bar{\rho u}_\infty &= M_\infty \cos\alpha \cos\beta, \\
\bar{\rho v}_\infty &= M_\infty \cos\alpha \sin\beta, \\
\bar{\rho w}_\infty &= M_\infty \sin\alpha, \\
\bar{\rho E}_\infty &= \frac{1}{\gamma(\gamma-1)} + \frac{1}{2}M_\infty^2.
\end{aligned}$$

The other derived dimensionless variables are

$$\begin{aligned}
\bar{p}_\infty &= \frac{1}{\gamma}, \\
\bar{\mu}_\infty &= \frac{M_\infty}{Re_\infty}, \\
\bar{c}_p &= \frac{1}{\gamma-1}, \\
\bar{\lambda} &= \bar{\mu}\frac{1}{Pr}\frac{1}{\gamma-1}.
\end{aligned}$$

From now on, all variables and equations that appear in the text are assumed to be in the nondimensional system and therefore the overbar sign will be dropped for simplicity.

3 Finite Volume Spatial Discretization

3.1 Overview

CFD has become an indispensable tool for a variety of applications in science and engineering. General speaking, numerical methods used in CFD can be classified by the mesh they use to discretize a computational domain as structured grid methods, unstructured grid methods, and Cartesian grid methods. The structured grid methods alone are not practical for engineering applications, as they have a disadvantage for gridding complex geometries. More often, they are used in the context of Chimera or overlapping approaches [1,2] to simplify the grid generation process for a complex configuration. The difficulty of generating a structured grid for complex geometries and the desire in the engineering community to simulate numerically flows past increasingly complex geometries have fueled interest in the development of unstructured grid methods [3–5]. Unstructured grids provide great flexibility in dealing with the complex geometries encountered in practice and offer a natural framework for solution-adaptive mesh refinement. However, the computational costs and memory requirements for unstructured triangular/tetrahedral grids are generally higher than for structured grids. Although the solution accuracy may not be strongly affected by element type even in the boundary layers, computational efficiency can benefit substantially through the use of prismatic elements in the boundary layers and Cartesian cells in the inviscid regions. This is due to a simple fact that approximately, five to six times more tetrahedra than hexahedra are required to fill a given region with a fixed number of nodes. Although the boundary layers regions occupy only a small portion of the computational domain, it is not uncommon for more than half of the mesh resolution to be packed into this small region, and thus the quadrilateral elements in 2D and prismatic elements in 3D can lead to a significant saving in both memory requirements and computational costs. From the computational efficiency point of view, unstructured triangular / tetrahedral elements should be kept minimal. The advantages of the Cartesian grid approaches [6–10] include ease of grid generation, lower computational storage requirements, and significantly less operational count per cell. However, the main challenge in using Cartesian methods is how to deal with arbitrary boundaries, as the grids are not body-aligned. The cells of a Cartesian mesh near the body can extend through surfaces of boundaries. Accurate means of representing boundary conditions in cells that intersect surfaces are essential for successful Cartesian methods. Since each grid type has its own advantages and disadvantages, the best grid approach is clearly a hybrid one that combines the advantages and strengths of all these three grid types.

The finite volume (FV) methods are probably the most successful class of spatial discretization techniques in computational fluid dynamics, due to their relative simplicity and great flexibility. In recent years, significant progress has been made in developing FV methods to solve the compressible Navier-Stokes equations on hybrid grids [11–18]. Nowadays, the FV methods are widely and routinely used for solving flow problems of scientific and industrial interest [19]. There exist two major classes of the finite volume methods: cell-centered and vertex-centered finite volume methods. Although the debate that which one has an edge over the other will probably never be settled, a cell-centered finite volume method is preferred in the context of arbitrary grids, which may contain hanging nodes. The existence of several cell types in a hybrid grid poses a great challenge to numerical methods. Although it is unavoidable to treat different cell types differently during the pre- and post-processing stages, it is undesirable that an algorithm depends on a mesh topology during the flow solution stage. The required conditional statements not only lead to an untidy code but also adversely affect speed of programs. Therefore, it is desirable to design an algorithm that treats different cell types in the same way. Such an algorithm is termed grid-transparent in the literature, which does not require any information on the local cell topology. A grid-transparent scheme has a number of advantages: first of all it can significantly reduce the discretization stencils compared to a non-grid-transparent scheme; secondly it can increase the speed of programs; and last it can facilitate the implementation of implicit schemes and parallelization.

A first order cell-centered finite volume method is in general stable on arbitrary grids. However, the second-order finite volume methods based on a piecewise linear reconstruction suffer from the so-called linear instability on unstructured tetrahedral grids, when the reconstruction stencil only involves adjacent face-neighboring cells [20]. One way to overcome this problem is to use extended stencils, which unfortunately will sacrifice the compactness of the underlying data structure. Furthermore, these linear reconstruction based finite volume methods suffer from non-physical oscillations in the vicinity of strong discontinuities for convection-dominant flows. Alternatively, total variation diminishing (TVD) or essentially non-oscillatory (ENO) / weighted ENO (WENO) reconstruction methods can be used to obtain a linear polynomial solution. The TVD and ENO/WENO high order methods are designed to suppress the spurious oscillations in the vicinity of discontinuities, and enhance the order of accuracy of the underlying first-order finite volume methods, thus achieving both linear and non-linear stability. Slope limiters are widely used in the finite volume methods to modify the piecewise linear reconstruction and thus to satisfy TVD condition. Unfortunately, the use of limiters will reduce the order of accuracy to first order in the presence of local extrema. Indeed, the limiters to enhance TVD/MUSCL conditions are less robust than the strategies of essential ENO/WENO reconstruction. The ENO schemes were initially introduced by Harten et al., [21] in which oscillations up to the order of the truncation error are allowed to overcome the drawbacks and limitations of limiter-based schemes. ENO schemes avoid interpolation across high-gradient regions through biasing of the reconstruction. This biasing is achieved by reconstructing the solution on several stencils at each location, and selecting the reconstruction, which is in some sense the smoothest. This allows ENO schemes to retain higher-order accuracy near high-gradient regions. However, the selection process can lead to convergence problems and loss of accuracy in regions with smooth solution variations. To counter these problems, the so-called weighted ENO scheme introduced by Liu et al. [22] is designed to present better convergence rate for steady state problems, better smoothing for the flux vectors, and better accuracy using the same stencils than the ENO scheme. The WENO scheme uses a suitably weighted combination of all reconstructions rather than just the one that is judged to be the smoothest. The weighting is designed to favor the smooth reconstruction in the sense that its weight is small, if the oscillation of a reconstructed polynomial is high and its weight is order of one, if a reconstructed polynomial has low oscillation. The development of ENO/WENO-based finite volume methods has been and still remains one of the active research topics in CFD as witnessed by abundance of literatures [23–28].

3.2 Mesh and Finite Volumes

Let us define a domain, Ω, in **finite space**, and divide Ω into a set of disjoint finite volume, and apply the conservation laws on each finite volume. The division of Ω gives rise to the **mesh** or **grid**, as follows,

$$\begin{aligned} &\Omega_i, \qquad i = 1, 2, \ldots, N_\Omega \\ &\Omega = \sum_{i=1}^{N_\Omega} \Omega_i \end{aligned} \quad .$$

Ω_i can be any convex polygonal cell in 2D/3D, e.g., triangle and quadrilateral in 2D, and tetrahedron, pyramid, prism, and hexahedron in 3D (see Fig. 2). Notice that the mesh can consist of

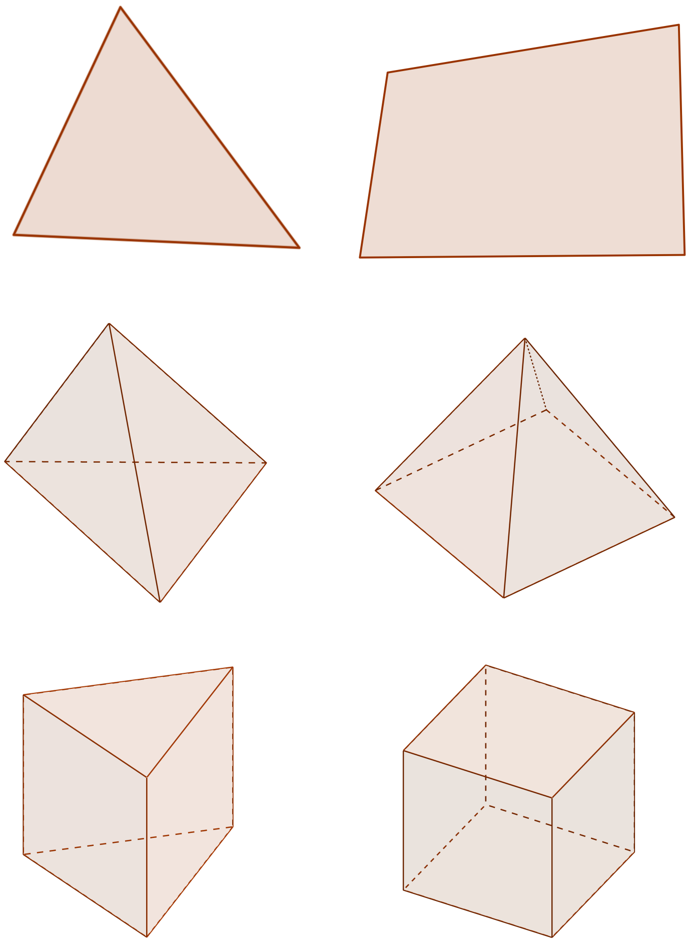


Figure 2. Examples of finite volume cells in 2D and 3D.

different types of those cells, for example, triangular and quadrilateral cells in a 2D mesh. Such a mesh is called **hybrid mesh**. Usually the mesh is taken to be **conforming** in the sense that there are no hanging nodes. But it is possible to use meshes with hanging nodes also, which can be advantageous when doing mesh adaptation.

Once a mesh has been formed, we have to create finite volumes on which the conservation law will be applied. This can be done in two ways, depending on where the solution is stored. If the solution is stored at the center of Ω_i, then Ω_i itself is the finite volume, i.e., $C_i = \Omega_i$. This gives rise to the **cell-centered** finite volume method. Alternatively, the solution can be stored at the vertices of the mesh. Then around each vertex, i, we have to construct a cell, C_i. This gives rise to the **vertex-centered** finite volume method. In either case we can obtain a collection of disjoint finite volumes $C_i, i = 1, 2, ..., N_C$ such that $\Omega = \sum_{i=1}^{N_C} C_i$. However, this work adopts the cell-centered finite volume method, since it is a subset of the discontinuous Galerkin methods. The solution of cell-centered finite volume method in each cell can be regarded as a piecewise constant polynomial in terms of discontinuous Galerkin methods.

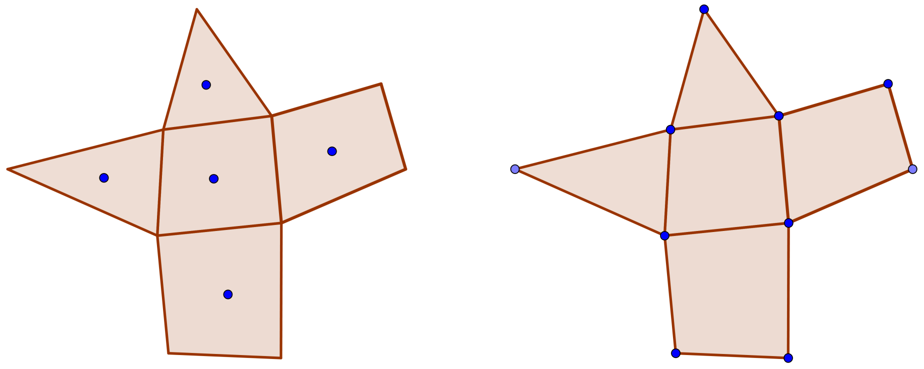


Figure 3. Schematics of cell-centered and vertex-centered finite volume methods.

Some notations of geometric information are defined for convenience. An internal face, $S_{ij} = \partial C_i \cap \partial C_j$, is the common face between C_i and C_j. A boundary face, $S_{ib} = \partial C_i \cap \partial \Omega$, is a face of C_i on the boundary of Ω. For each cell C_i, N_i represents a set of neighboring cells, C_j, having a common face, S_{ij}, with C_i. Fig. 4 shows two example cells and their face neighboring cells in 2D.

3.3 FVM for the Compressible Navier–Stokes Equations

Recall the compressible Navier–Stokes equations in differential form:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \boldsymbol{F} = \nabla \cdot \boldsymbol{G} + \mathbf{S}, \tag{14}$$

which can be integrated over each cell, C_i, as follows,

$$\int_{C_i} \frac{\partial \mathbf{U}}{\partial t}\, \mathrm{d}V + \int_{C_i} \nabla \cdot \boldsymbol{F}\, \mathrm{d}V - \int_{C_i} \nabla \cdot \boldsymbol{G}\, \mathrm{d}V - \int_{C_i} \mathbf{S}\, \mathrm{d}V = 0. \tag{15}$$

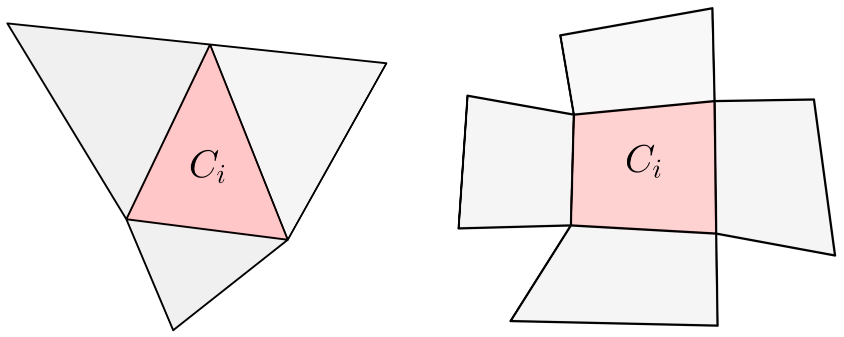


Figure 4. Examples of finite volume cells and their face neighboring cells in 2D.

Define **cell-average value**

$$\mathbf{U}_i(t) = \frac{1}{|C_i|}\int_{C_i} \mathbf{U}(\mathbf{x},t)\ \mathrm{d}V \tag{16}$$

Using the divergence theorem, the following equations can then be derived from Eq. (15):

$$|C_i|\frac{\mathrm{d}\mathbf{U}_i}{\mathrm{d}t} + \int_{\partial C_i} \boldsymbol{F}\cdot\mathbf{n}\ \mathrm{d}S - \int_{\partial C_i} \boldsymbol{G}\cdot\mathbf{n}\ \mathrm{d}S - \int_{C_i} \mathbf{S}\ \mathrm{d}V = 0. \tag{17}$$

$$\begin{aligned} |C_i|\frac{\mathrm{d}\mathbf{U}_i}{\mathrm{d}t} + \sum_{j\in N_i}\int_{S_{ij}} \boldsymbol{F}\cdot\mathbf{n}\ \mathrm{d}S + \sum_{S_{ib}\in\partial\Omega}\int_{S_{ib}} \boldsymbol{F}\cdot\mathbf{n}\ \mathrm{d}S \\ - \sum_{j\in N_i}\int_{S_{ij}} \boldsymbol{G}\cdot\mathbf{n}\ \mathrm{d}S - \sum_{S_{ib}\in\partial\Omega}\int_{S_{ib}} \boldsymbol{G}\cdot\mathbf{n}\ \mathrm{d}S - \int_{C_i} \mathbf{S}\ \mathrm{d}V = 0, \end{aligned} \tag{18}$$

where $\mathbf{n}$ is the unit normal vector pointing from C_i to C_j, as shown in Fig. 5.

The flux integral can be approximately calculated by numerical quadrature. For the first- and second-order accurate schemes, it is enough to use mid-point rule of integration as follows,

$$\int_{S_{ij}} \boldsymbol{F}\cdot\mathbf{n}\ \mathrm{d}S - \int_{S_{ij}} \boldsymbol{G}\cdot\mathbf{n}\ \mathrm{d}S \approx (\boldsymbol{F}\cdot\mathbf{n})_{ij}|S_{ij}| - (\boldsymbol{G}\cdot\mathbf{n})_{ij}|S_{ij}|. \tag{19}$$

The approach to compute the flux is to be discussed. We have two states, $\mathbf{U}_{ij}$ and $\mathbf{U}_{ji}$ coming from cells C_i and C_j, respectively. A numerical flux function of Godunov-type or flux vector splitting can be applied for the inviscid part, etc

$$(\boldsymbol{F}\cdot\mathbf{n})_{ij} \approx \mathcal{H}^{\mathrm{inv}}(\mathbf{U}_{ij},\mathbf{U}_{ji},\mathbf{n}_{ij}). \tag{20}$$

On boundary faces S_{ib}, the flux should be determined using appropriate boundary conditions,

$$(\boldsymbol{F}\cdot\mathbf{n})_{ib} \approx \mathcal{H}_b^{\mathrm{inv}}(\mathbf{U}_{ib},\mathbf{U}_{b},\mathbf{n}_{ib}). \tag{21}$$

The viscous fluxes,

$$(\boldsymbol{G}\cdot\mathbf{n})_{ij} \approx \mathcal{H}^{\mathrm{vis}}(\mathbf{U}_{ij},\mathbf{U}_{ji},\mathbf{n}_{ij}), \tag{22}$$

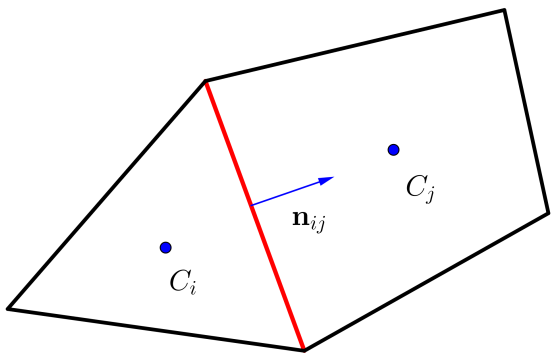


Figure 5. Example of a common face between cells C_i and C_j in 2D.

and

$$(\boldsymbol{G}\cdot\mathbf{n})_{ib} \approx \mathcal{H}_b^{\text{vis}}(\mathbf{U}_{ib},\mathbf{U}_b,\mathbf{n}_{ib}) \tag{23}$$

can be computed using central difference type approximations, which will be discussed later.

Finally the semi-discrete form can be arrived as follows,

$$\begin{aligned} |C_i|\frac{\mathrm{d}\mathbf{U}_i}{\mathrm{d}t} + \sum_{j\in N_i}\mathcal{H}^{\text{inv}}(\mathbf{U}_{ij},\mathbf{U}_{ji},\mathbf{n}_{ij})|S_{ij}| + \sum_{S_{ib}\in\partial\Omega}\mathcal{H}_b^{\text{inv}}(\mathbf{U}_{ib},\mathbf{U}_b,\mathbf{n}_{ib})|S_{ib}| \\ - \sum_{j\in N_i}\mathcal{H}^{\text{vis}}(\mathbf{U}_{ij},\mathbf{U}_{ji},\mathbf{n}_{ij}) - \sum_{S_{ib}\in\partial\Omega}\mathcal{H}_b^{\text{vis}}(\mathbf{U}_{ib},\mathbf{U}_b,\mathbf{n}_{ib}) - \int_{C_i}\mathbf{S}\,\mathrm{d}V = 0. \end{aligned} \tag{24}$$

Eq. (24) is a system of ODE, which can be integrated in time using various schemes like explicit Runge–Kutta or implicit schemes.

3.4 First-Order FVM

In the case of first-order finite volume methods, the solution in each cell is assumed to be a constant in space. Then on any interior face S_{ij}, the two states are simply

$$\mathbf{U}_{ij} = \mathbf{U}_i, \qquad \mathbf{U}_{ji} = \mathbf{U}_j. \tag{25}$$

The inviscid flux is then approximated as

$$\mathcal{H}^{\text{inv}}(\mathbf{U}_{ij},\mathbf{U}_{ji},\mathbf{n}_{ij}) = \mathcal{H}^{\text{inv}}(\mathbf{U}_i,\mathbf{U}_j,\mathbf{n}_{ij}), \tag{26}$$

which leads to a first-order accurate scheme. If the numerical flux $\mathcal{H}^{\text{inv}}$ is well designed, then these schemes can be very stable, robust, and have desired properties like monotonicity and entropy condition. However, the first-order finite volume schemes usually introduce too much error and lead to poor resolution of shocks, contact waves, and vorticity.

4 High-Resolution Methods

4.1 Reconstruction

To achieve spatial accuracy higher than the first order, the solution in each cell needs to be reconstructed. The simplest approach is to perform piecewise linear reconstruction. The reconstruction can be performed on

- Conserved variables
- Primitive variables, e.g., (ρ, u, v, w, p) or (T, u, v, w, p)
- Characteristic variables

Notice that though the conserved variables satisfy the conservation of reconstructed solution easily, they are not as robust as the primitive variables as to ensure the positivity of density and pressure. Moreover, characteristic variables lead to more accurate schemes at a slightly more computational cost. Considering these factors, this work adopts the primitive variables for reconstruction.

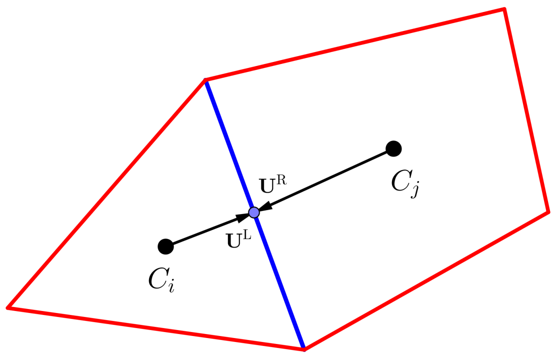


Figure 6. Schematics of reconstructed solution on a common face in 2D.

From now on, let us denote $\bar{\mathbf{U}}_i$ as the underlying cell-average solution in each cell, and $\mathbf{U}_i$ the linear polynomial solution to be reconstructed. Using the reconstruction process, two reconstructed states $\mathbf{U}_{ij}$ and $\mathbf{U}_{ji}$ at the common face S_{ij} can be obtained to compute the inviscid flux

$$\mathbf{H}^{\mathrm{inv}}(\mathbf{U}_{ij}, \mathbf{U}_{ji}, \mathbf{n}_{ij}).$$

By assuming the gradient of $\mathbf{U}_i$ at the center of C_i, the reconstructed solution for C_i is

$$\mathbf{U}(\mathbf{r}) = \bar{\mathbf{U}}_i + (\mathbf{r} - \mathbf{r}_i) \cdot \nabla \mathbf{U}_i$$

where $\mathbf{r} = (\mathbf{x}, \mathbf{y}, \mathbf{z})$, and $\mathbf{r}_i = (x_i, y_i, z_i)$.

As shown in Fig. 6, two states can be obtained at the center of face S_{ij} (i.e., $\mathbf{r} = \mathbf{r}_{ij}$), ,

$$\mathbf{U}_{ij} = \bar{\mathbf{U}}_i + (\mathbf{r}_{ij} - \mathbf{r}_i) \cdot \nabla \mathbf{U}_i,$$
$$\mathbf{U}_{ji} = \bar{\mathbf{U}}_j + (\mathbf{r}_{ij} - \mathbf{r}_j) \cdot \nabla \mathbf{U}_j.$$

Notice that in order to ensure monotone solutions, a limiter function has to be calculated in each cell. The limited reconstructed values are

$$\mathbf{U}_{ij} = \bar{\mathbf{U}}_i + \phi_i (\mathbf{r}_{ij} - \mathbf{r}_i) \cdot \nabla \mathbf{U}_i,$$
$$\mathbf{U}_{ji} = \bar{\mathbf{U}}_j + \phi_j (\mathbf{r}_{ij} - \mathbf{r}_j) \cdot \nabla \mathbf{U}_j.$$

Since $\mathbf{U}$ has several components, the limiter function is computed for each component. Popular limiters include the min-max limiter of Barth–Jespersen [29], and Venkatakrishnan limiter which will be introduced later.

4.1.1 Least-Squares Gradient Reconstruction

One of the most commonly used and simplest reconstruction schemes is the least-squares reconstruction, where the computation of gradients is performed in the form of a minimization problem. The complete details of this reconstruction procedure can be found in [30]. However, the procedure is summarized here for completeness. Consider a cell C_i and assume that the solution varies linearly in the union of the cell C_i, and its face-neighboring cell j. Then, the change in cell-centered values of the solution can be computed by

$$\nabla \cdot (\mathbf{r}_j - \mathbf{r}_i) = \mathbf{U}_j - \mathbf{U}_i \tag{27}$$

where $\mathbf{r}_i$ and $\mathbf{r}_j$ are the position vector for the center of cells i and j, respectively. Similar equations could be written for all adjacent cells that share a face with the cell C_i subject to an arbitrary weighting factor w_i. This yields the following non-square matrix, for example, in 3D,

$$\begin{bmatrix} w_1(x_1 - x_i) & w_1(y_1 - y_i) & w_1(z_1 - z_i) \\ \vdots & \vdots & \vdots \\ w_n(x_n - x_i) & w_n(y_n - y_i) & w_n(z_n - z_i) \end{bmatrix} \begin{bmatrix} \mathbf{U}_x \\ \mathbf{U}_y \\ \mathbf{U}_z \end{bmatrix} = \begin{bmatrix} w_1(\mathbf{U}_1 - \mathbf{U}_i) \\ \vdots \\ w_n(\mathbf{U}_n - \mathbf{U}_i) \end{bmatrix}, \tag{28}$$

and in 2D,

$$\begin{bmatrix} w_1(x_1 - x_i) & w_1(y_1 - y_i) \\ \vdots & \vdots \\ w_n(x_n - x_i) & w_n(y_n - y_i) \end{bmatrix} \begin{bmatrix} \mathbf{U}_x \\ \mathbf{U}_y \end{bmatrix} = \begin{bmatrix} w_1(\mathbf{U}_1 - \mathbf{U}_i) \\ \vdots \\ w_n(\mathbf{U}_n - \mathbf{U}_i) \end{bmatrix} \tag{29}$$

where n is the number of the face-neighboring cells for the cell C_i, numbered from 1 to n. In the case of 1D mesh, the stencil involved to form the non-square matrix for an interior cell is its left- and right-neighboring cells,

$$\begin{bmatrix} w_1(x_{i-1} - x_i) \\ w_2(x_{i+1} - x_i) \end{bmatrix} \begin{bmatrix} \mathbf{U}_x \end{bmatrix} = \begin{bmatrix} w_1(\mathbf{U}_{i-1} - \mathbf{U}_i) \\ w_2(\mathbf{U}_{i+1} - \mathbf{U}_i) \end{bmatrix} \tag{30}$$

Eqs. (28) – (30) can be solved using the least-squares method. The algorithm can be implemented using the face-based data structure. This formulation provides a freedom in the choice of weighting coefficients w_i. These weighting coefficients can be selected as a function of the geometry and/or solution. Classical approximations in one dimension can be recovered by choosing geometrical weights of the form $w_i = 1/|\mathbf{r}_j - \mathbf{r}_i|^t$ for values of $t = 0, 1, 2$. The numerical experiments in this work were performed using $t = 1$. Take an interior quadrilateral cell for example (see Fig. 7), the number of face-neighboring cells is four. Consequently, the size of the resulting non-square matrix is 4×2. If there are boundary faces involved, proper boundary conditions have to be used to calculate the values of the ghost cells first. In the present work, this over-determined linear system of 4 equations for 2 unknowns is solved in the least-squares sense using either normal equation approach decomposition to obtain the first derivatives of the reconstructed linear polynomial solution. Note that we need at least two neighboring cells to apply the least-squares method in 2D. One can easily verify that this least-squares reconstruction satisfies the so-called 1-exactness, i.e., it can reconstruct a linear polynomial function exactly.

This idea can be easily extended to 3D, e.g., in the following non-square matrix which can be solved the same way as introduced above. Note that we need at least three neighboring cells to apply the least-squares method in 3D. The least squares method gives accurate gradient estimates. But on highly stretched grids, it can lead to unstable schemes. The use of distance based weight alleviates the problem to some extent.

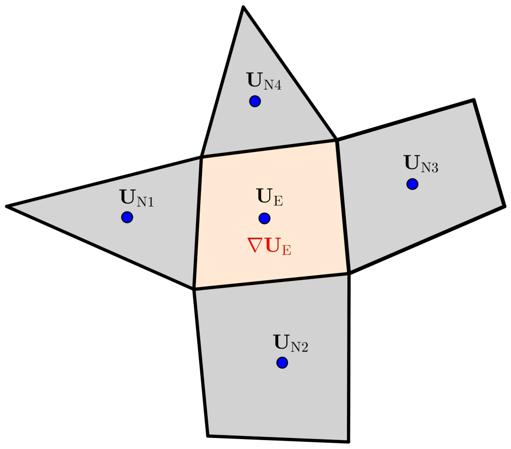

$\mathbf{U}_{\mathrm{N4}}$
$\mathbf{U}_{\mathrm{N3}}$
$\mathbf{U}_{\mathrm{N1}}$
$\mathbf{U}_{\mathrm{E}}$
$\nabla\mathbf{U}_{\mathrm{E}}$
$\mathbf{U}_{\mathrm{N2}}$

Figure 7. Example of the stencil required for reconstruction of an interior cell in 2D.

4.1.2 Green–Gauss Gradient Reconstruction

By applying the Green theorem to cell C_i, we can get the following equation

$$\int_{C_i} \nabla \mathbf{U}_i \, \mathrm{d}V = \int_{\partial C_i} \mathbf{U}_i \cdot \mathbf{n} \, \mathrm{d}S, \tag{31}$$

in which the integral can be computed using the mid-point rule of quadrature. Finally, the variable gradients can be obtained with the equation as below

$$\nabla \mathbf{U}_i = \frac{1}{|C_i|} \sum_{j \in N_i} \left(\phi \mathbf{U}_i + (1-\phi) \mathbf{U}_j \right) \mathbf{n} |S_{ij}| \tag{32}$$

where the weighted distance factor

$$\phi = |\mathbf{r}_{ij} - \mathbf{r}_i| / (|\mathbf{r}_{ij} - \mathbf{r}_i| + |\mathbf{r}_{ij} - \mathbf{r}_j|).$$

4.1.3 Min-Max Gradient Limiter

The basic idea of min-max gradient limiter is that the reconstructed states $U_i j$ must remain between the minimum and maximum values in the stencil of cell C_i.

Define

$$\begin{aligned} U_i^m &= \min_{j \in N_i} (U_j, U_i), \\ U_i^M &= \max_{j \in N_i} (U_j, U_i). \end{aligned} \tag{33}$$

Note that U_i is a component of the solution vector $\mathbf{U}_i$ in cell C_i, for example, ρ, u, v, w, and p, if we choose primitive variables for the reconstruction process. Then we want to choose the larger value of $0 \leq \phi_i \leq 1$ so that

$$U_i^m \leq U_i + \phi_i (\mathbf{r}_{ij} - \mathbf{r}_i) \cdot \nabla U_i \leq U_i^M, \qquad \forall C_j \in N_i \tag{34}$$

Define

$$\phi_{ij} = \begin{cases} \min(1, \frac{U_i^M - U_i}{\Delta_{ij}}) & \text{if} \Delta_{ij} > 0 \\ \min(1, \frac{U_i^m - U_i}{\Delta_{ij}}) & \text{if} \Delta_{ij} < 0 \\ 1 & \text{otherwise} \end{cases} \tag{35}$$

where $\Delta_{ij} = (\mathbf{r}_{ij} - \mathbf{r}_i) \cdot \nabla U_i$. Finally,

$$\phi_i = \min_{j \in N_i} \phi_{ij}. \tag{36}$$

For scalar conservation laws, one can show that this scheme with a monotone flux satisfies local maximum principle and hence is stable in maximum norm. For Euler equations, this leads to a very robust scheme but it is not very accurate since it can clip smooth extrema also. Moreover, this limiter is not a smooth function due to use of min and max functions. This leads to slow convergence to steady state solutions and in fact we can not obtain convergence to machine zero in most cases.

4.1.4 WENO Gradient Limiter

This least-squares reconstructed finite volume method can be successfully used to solve the 2D compressible Euler equations for smooth flows on arbitrary grids and is able to achieve the designed second order of accuracy and significantly improve the accuracy of the underlying first-order finite volume method. However, when extended to solve the 3D compressible Euler equations on tetrahedral grids, this cell-centered finite volume method suffers from the so-called linear instability, which occurs even for the linear hyperbolic equation [20]. This linear instability is attributed to the fact that the reconstruction stencils only involve von Neumann neighborhood, i.e., adjacent face-neighboring cells [20]. The linear stability can be achieved using extended stencils, which will unfortunately sacrifice the compactness of the underlying FV methods. Furthermore, such a linear reconstruction-based finite volume method cannot maintain the non-linear instability, leading to non-physical oscillations in the vicinity of strong discontinuities. Alternatively, ENO/WENO can be used to reconstruct a linear polynomial solution, which can not only enhance the order of accuracy of the underlying finite volume method but also achieve both linear and nonlinear stability.

Specifically, the WENO scheme introduced by Dumber et al. [24, 25] is adopted in this work, where an entire linear polynomial solution on cell C_i is obtained using a nonlinear WENO reconstruction as a convex combination of the least-squares reconstructed first derivatives at the cell itself ($k = 0$) and its face-neighboring cells ($k = 1, \ldots, N_{\text{face}}$),

$$\nabla U_i^{\text{WENO}} = \sum_{k=0}^{N_{\text{face}}} w_k \nabla U_k^{\text{LS}} \tag{37}$$

where the superscript "WENO" denotes the WENO limited gradient of the variable, "LS" the least-squares reconstructed gradient of the variable, N_{face} is the number of the face-neighboring cells for cell C_i, and w_k the normalized nonlinear weights. Note that U_i is a component of the solution vector $\mathbf{U}_i$ in cell C_i, for example, ρ, u, v, w, and p, if we choose primitive variables for the reconstruction process.

The w_k is computed as

$$w_k = \frac{\tilde{w}_k}{\sum_{k=0}^{N_{\text{face}}} \tilde{w}_k} \tag{38}$$

The non-normalized nonlinear weights, $\tilde{w}_k$, are functions of the linear weights, λ_k, and the so-called oscillation indicator o_k:

$$\tilde{w}_k = \frac{\lambda_k}{(\varepsilon + o_i)^{\gamma}} \tag{39}$$

where ε is a small positive number used to avoid division by zero, and γ an integer parameter to control how fast the non-linear weights decay for non-smooth stencils. The oscillation indicator is simply defined as

$$o_k = \sqrt{\nabla U_k^{\text{LS}} \cdot \nabla U_k^{\text{LS}}} \tag{40}$$

The linear weights λ_k can be chosen to balance the accuracy and the non-oscillatory property of the finite volume method.

To summarize, the least-squares reconstructed polynomial at the cell itself serves as the central stencil, and the least-squares reconstructed polynomials on its face-neighboring cells act as biased stencils in this WENO reconstruction. Notice that this WENO gradient reconstruction is not compact anymore as neighbor's neighbors are used in the solution update. However, the stencil used in the reconstruction is compact, involving only von Neumann neighbors. Consequently, this WENO method can be implemented in a compact manner.

4.2 TVD Slope Limiters

All of the the slope reconstruction and limiting methods described so far can be readily applied on 1D, 2D, and 3D unstructured meshes. Moreover, a number of classic Total Variation Diminishing (TVD) slope limiters have been implemented in BIGHORN exclusively for 1D unstructured meshes. Those TVD-type slope limiters, for example, minmod, superbee, and MC, were originally designed for structured meshes. Nevertheless, it is straightforward to implement those TVD-type slope limiters in the case of 1D for an unstructured code structure, since it is trivial to locate and obtain the required information of the left- and right-neighboring cells for each 1D cell , (i.e. the information in the $(i-1)$-th and $(i+1)$-th cells for the i-th cell). Those TVD-type slope limiters are usually more robust and perform better than those dimension-agnostic reconstruction and limiting methods to obtain high-resolution monotonic solutions on 1D meshes, and therefore are preferred for 1D problems.

For a complete description of TVD and slope limiter, see [31].

4.2.1 Minmod Slope Limiter

One choice of slope that gives second-order accuracy for smooth solutions while still satisfying the TVD property is the *minmod slope*

$$\phi_i = \text{minmod}\left(\frac{Q_i - Q_{i-1}}{\Delta x}, \frac{Q_{i+1} - Q_i}{\Delta x}\right) \tag{41}$$

where the minmod function of two arguments is defined by

$$\text{minmod}(a,b) = \begin{cases} a & \text{if } |a| < |b| \text{ and } ab > 0, \\ b & \text{if } |b| < |a| \text{ and } ab > 0, \\ 0 & \text{if } ab \leq 0. \end{cases} \tag{42}$$

If a and b have the same sign, then this selects the one that is smaller in modulus, else it returns zero.

Rather than defining the slope on the ith cell by always using the downwind difference (which would give the LaxWendroff method), or by always using the upwind difference (which would give the BeamWarming method), the minmod method compares the two slopes and chooses the one that is smaller in magnitude. If the two slopes have different sign, then the value Q_i must be a local maximum or minimum, and it is easy to check in this case that we must set $\phi_i = 0$ in order to satisfy the TVD condition. The minmod method does a fairly good job of maintaining good accuracy in the smooth hump and also sharp discontinuities in the square wave, with no oscillations. Sharper resolution of discontinuities can be achieved with other limiters that do not reduce the slope as severely as minmod near a discontinuity.

4.2.2 Superbee Slope Limiter

One choice of limiter that gives the sharper reconstruction, while still giving second order accuracy for smooth solutions, is the so-called *superbee* limiter introduced by Roe [32]:

$$\phi_i = \text{maxmod}(\phi^{(1)_i}, \phi^{(2)_i}) \tag{43}$$

where

$$\phi_i^{(1)} = \text{minmod}\left(\frac{Q_{i+1}-Q_i}{\Delta x}, 2\frac{Q_i - Q_{i-1}}{\Delta x}\right),$$
$$\phi_i^{(2)} = \text{minmod}\left(2\frac{Q_{i+1}-Q_i}{\Delta x}, \frac{Q_i - Q_{i-1}}{\Delta x}\right).$$

Each one-sided slope is compared with twice the opposite one-sided slope. Then the maxmod function in Eq. (43) selects the argument with larger modulus. In regions where the solution is smooth this will tend to return the larger of the two one-sided slopes, but will still be giving an approximation, and hence we expect second-order accuracy. The superbee limiter is also TVD in general.

With the superbee method, the discontinuity stays considerably sharper than with the minmod method. On the other hand, there is a tendency of the smooth hump to become steeper and squared off. This is sometimes a problem with superbee — by choosing the larger of the neighboring slopes it tends to steepen smooth transitions near inflection points.

4.2.3 MC Slope Limiter

Another popular choice is the *monotonized central-difference limiter* (MC limiter), which was proposed by van Leer [33]:

$$\phi_i = \text{minmod}\left(\left(\frac{Q_{i+1}-Q_{i-1}}{2\Delta x}\right), 2\left(\frac{Q_i - Q_{i-1}}{\Delta x}\right), 2\left(\frac{Q_{i+1}-Q_i}{\Delta x}\right)\right). \tag{44}$$

This compares the central difference of Fromm's method with twice the one-sided slope to either side. In smooth regions this reduces to the centered slope of Fromm's method and hence does not tend to artificially steepen smooth slopes to the extent that superbee does. The MC limiter appears to be a good default choice for a wide class of problems.

5 Temporal Discretization and Integration Methods

5.1 Explicit Time Integration

The finite volume spatial discretization of the governing equations leads to a system of ordinary differential equations (ODEs) in time, see Eq. (24). By moving all the non-time derivative terms of Eq. (24) from the left to the right side, an elemental semi-discrete form can be derived as

$$|C_i|\frac{\mathrm{d}\mathbf{U}_i}{\mathrm{d}t} = \mathbf{R}_i(\mathbf{U}_i) \tag{45}$$

where $\mathbf{R}_i$ is the elemental residual vector for the ith element, and approaches zero for a steady-state solution. $\mathbf{U}_i$ is the elemental solution vector of Neqn degrees of freedom to be evolved in time. Each $\mathbf{U}_i$ represents solution of Neqn degrees of freedom for the ith cell (for the compressible Navier–Stokes / Euler equations, Neqn $= 3$ in 1D, 4 in 2D, and 5 in 3D).

5.1.1 One-Step Explicit Euler Scheme

The one-step explicit Euler scheme can be simply expressed as follows

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \frac{\Delta t}{|C_i|}\mathbf{R}_i(\mathbf{U}_i^n) \tag{46}$$

where the superscript "n" denotes the current time level, and "$n+1$" denotes the next time level. This temporal discretization scheme is first-order accurate in time.

5.1.2 Two-Step Explicit Mid-Point Runge-Kutta Scheme

The two-step explicit mid-point Runge-Kutta scheme can be described in the following two stages

$$\begin{aligned} &\text{Stage (1)} \quad \mathbf{U}_i^{(1)} = \mathbf{U}_i^n + \frac{1}{2}\frac{\Delta t}{|C_i|}\mathbf{R}_i(\mathbf{U}_i^n) \\ &\text{Stage (2)} \quad \mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \frac{\Delta t}{|C_i|}\mathbf{R}_i(\mathbf{U}_i^{(1)}) \end{aligned} \tag{47}$$

This temporal discretization scheme is second-order accurate in time.

5.1.3 Two-Step Explicit TVD Runge-Kutta Scheme

The two-step explicit TVD Runge-Kutta scheme can be expressed in the following two stages

$$\begin{aligned} &\text{Stage (1)} \quad \mathbf{U}_i^{(1)} = \mathbf{U}_i^n + \frac{1}{2}\frac{\Delta t}{|C_i|}\mathbf{R}_i(\mathbf{U}_i^n) \\ &\text{Stage (2)} \quad \mathbf{U}_i^{n+1} = \frac{1}{2}\mathbf{U}_i^n + \frac{1}{2}\left(\mathbf{U}_i^{(1)} + \frac{\Delta t}{|C_i|}\mathbf{R}_i(\mathbf{U}_i^{(1)})\right) \end{aligned} \tag{48}$$

This temporal discretization scheme is second-order accurate in time, and used in many transient benchmark test cases in this work. The validation of its temporal convergence order of accuracy is shown in Fig. 8 in an independent test case.

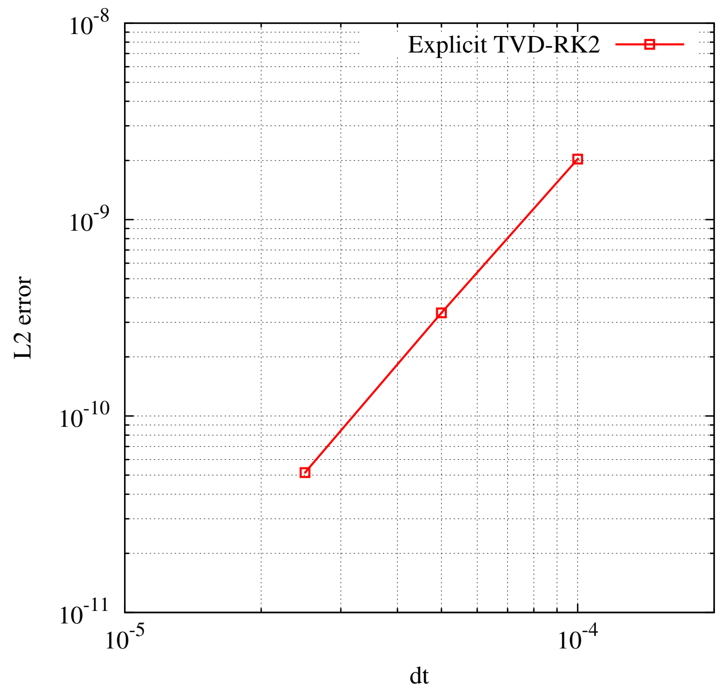


Figure 8. Verification of temporal accuracy for the two-step TVD Runge-Kutta scheme.

5.1.4 Three-Step Explicit TVD Runge-Kutta Scheme

The following explicit three-stage TVD Runge-Kutta scheme is also widely used to advance the solution in time [34–36]:

$$
\begin{aligned}
&\text{Stage (1)} \quad \mathbf{U}^{(1)} = \mathbf{U}^n + \frac{\Delta t}{|C_i|}\mathbf{R}(\mathbf{U}^n) \\
&\text{Stage (2)} \quad \mathbf{U}^{(2)} = \frac{3}{4}\mathbf{U}^n + \frac{3}{4}\left(\mathbf{U}^{(1)} + \frac{\Delta t}{|C_i|}\mathbf{R}(\mathbf{U}^{(1)})\right) \\
&\text{Stage (3)} \quad \mathbf{U}^{n+1} = \frac{1}{3}\mathbf{U}^n + \frac{2}{3}\left(\mathbf{U}^{(2)} + \frac{\Delta t}{|C_i|}\mathbf{R}(\mathbf{U}^{(2)})\right)
\end{aligned}
\tag{49}
$$

This temporal discretization scheme is third-order accurate in time, and linearly stable for a CFL number less than or equal to $1/(2p+1)$.

5.2 Implicit Time Integration

Several implicit temporal discretization schemes for implicit time integration in the BIGHORN code are available to choose from the underlying MOOSE framework. In order to achieve the best solution efficiency, appropriate methods should be chosen for each specific problem. For example, the first-order backward Euler scheme is usually used for steady-state smooth flow problems; the second-order backward differentiation formula (BDF2) or second-order Crank–Nicholson (CN2) scheme is often used for time-accurate unsteady flow simulations. To best retain details of vortices or features of turbulent flows in long time evolution, the third- or fourth-order diagonally implicit Runge–Kutta (DIRK) schemes are preferred. Reference literatures include [37–41]. Those are among the most popular methods for the implicit solution of fluid flow problems. Other methods, for example, the class of Rosenbrock schemes [42, 43], can also be potentially implemented and used for long time evolution problems. For low-Mach flow problems, a Pressure-Corrected Implicit Continuous-fluid Eulerian (PCICE) formulation [44, 45] can be used. The details of the PCICE scheme are described in Section 8.

6 Numerical Flux Schemes

6.1 Overview

If the inviscid fluxes $\mathbf{H}^{\text{inv}}$ at the interface in Eq. (26) are simply evaluated as an arithmetical average of the fluxes at cell C_i, and cell C_j or computed using an arithmetical average of the flow variables at cell C_i, and cell C_j, the resulting finite volume method, equivalent to a central difference scheme, allows for the appearance of checker boarding modes, and thus suffers from linear instability, unless some type of numerical dissipation in the form of artificial viscosity is introduced. To construct a stable scheme and thus ensure linear stability of a finite volume method for the compressible Euler equations, any of the Riemann solvers can be formulated by adopting different forms for the numerical fluxes at the interface. A number of numerical schemes, including Roe's flux difference splitting [46], the family of AUSM schemes [47–50], HLLC [51, 52], and Edwards' low-diffusion flux-splitting scheme (LDFSS) [53, 54] can be used to compute the inviscid fluxes at the interfaces. If the cell-averaged variables are used to compute the numerical fluxes, the resulting upwind finite volume method is only first order accurate in space. A higher order of accuracy can be achieved using a reconstruction scheme, which consists of finding a polynomial representation to the solution in each control volume, giving the cell-averaged solutions in each control volume. For the linear reconstruction where a linear polynomial solution on each cell is reconstructed using cell-averaged values of the flow variables in the neighboring cells, the computation of the gradients of the flow variables in the control volume is simply required.

6.2 HLLC Schemes

The explicit form of the HLLC flux function [51] evaluated at the center of face S_{ij} is defined by

$$\mathbf{H}_{\text{HLLC}}(\mathbf{U}_l, \mathbf{U}_r, \mathbf{n}_{ij}) = \begin{cases} \mathbf{H}_l(\mathbf{U}_l) & \text{if } S_L > 0 \\ \mathbf{H}(\mathbf{U}_l^*) & \text{if } S_L \le 0 < S_M \\ \mathbf{H}(\mathbf{U}_r^*) & \text{if } S_M \le 0 \le S_R \\ \mathbf{H}_r(\mathbf{U}_r) & \text{if } S_R < 0 \end{cases} \tag{50}$$

where the subscript "l" and "r" denote the state vectors $\mathbf{U}_i$ from cell C_i and $\mathbf{U}_j$ from cell C_j. The symbols with superscript "$*$" are defined by

$$\mathbf{U}_l^* = \begin{bmatrix} \rho_l^* \\ (\rho u)_l^* \\ (\rho v)_l^* \\ (\rho w)_l^* \\ (\rho E)_l^* \end{bmatrix} = \Omega_l \begin{bmatrix} \rho_l(S_L - q_l) \\ (S_L - q_l)(\rho u)_l + (p^* - p_l)n_x \\ (S_L - q_l)(\rho v)_l + (p^* - p_l)n_y \\ (S_L - q_l)(\rho w)_l + (p^* - p_l)n_z \\ (S_L - q_l)(\rho E)_l - p_l q_l + p^* S_M \end{bmatrix}, \tag{51}$$

$$\mathbf{U}_r^* = \begin{bmatrix} \rho_r^* \\ (\rho u)_r^* \\ (\rho v)_r^* \\ (\rho w)_r^* \\ (\rho E)_r^* \end{bmatrix} = \Omega_r \begin{bmatrix} \rho_r(S_R - q_r) \\ (S_R - q_r)(\rho u)_r + (p^* - p_r)n_x \\ (S_R - q_r)(\rho v)_r + (p^* - p_r)n_y \\ (S_R - q_r)(\rho w)_r + (p^* - p_r)n_z \\ (S_R - q_r)(\rho E)_r - p_r q_r + p^* S_M \end{bmatrix}, \tag{52}$$

and

$$\mathbf{H}_l^* \equiv \mathbf{H}(\mathbf{U}_l^*) = \begin{bmatrix} \rho_l^* S_M \\ (\rho u)_l^* S_M + p^* n_x \\ (\rho v)_l^* S_M + p^* n_y \\ (\rho w)_l^* S_M + p^* n_z \\ ((\rho E)_l^* + p^*) S_M \end{bmatrix}, \tag{53}$$

$$\mathbf{H}_r^* \equiv \mathbf{H}(\mathbf{U}_r^*) = \begin{bmatrix} \rho_r^* S_M \\ (\rho u)_r^* S_M + p^* n_x \\ (\rho v)_r^* S_M + p^* n_y \\ (\rho w)_r^* S_M + p^* n_z \\ ((\rho E)_r^* + p^*) S_M \end{bmatrix}. \tag{54}$$

The Ω_l and Ω_r are defined as below:

$$\Omega_l \equiv (S_L - S_M)^{-1}, \tag{55}$$

$$\Omega_r \equiv (S_R - S_M)^{-1}. \tag{56}$$

The p^* is defined by

$$p^* = \rho_l(q_l - S_L)(q_l - S_M) + p_l = \rho_r(q_r - S_R)(q_r - S_M) + p_r, \tag{57}$$

where

$$q_l \equiv u_l n_x + v_l n_y + w_l n_z, \tag{58}$$

$$q_r \equiv u_r n_x + v_r n_y + w_r n_z, \tag{59}$$

with $(n_x, n_y, n_z)^{\mathrm{T}}$ being the unit vector normal to face S_{ij}. S_M is taken from Batten et al. [55]:

$$S_M = \frac{\rho_r q_r (S_R - q_r) - \rho_l q_l (S_L - q_l) + p_l - p_r}{\rho_r (S_R - q_r) - \rho_l (S_L - q_l)} \tag{60}$$

and S_L, S_R are taken from Einfeldt et al. [56]:

$$S_L = \min\left[\lambda_1(\mathbf{U}_l), \lambda_1(\mathbf{U}^{\mathrm{Roe}})\right], \tag{61}$$

$$S_R = \max\left[\lambda_m(\mathbf{U}^{\mathrm{Roe}}), \lambda_m(\mathbf{U}_r)\right] \tag{62}$$

where $\lambda_1(\mathbf{U}^{\mathrm{Roe}})$ and $\lambda_m(\mathbf{U}^{\mathrm{Roe}})$ are the smallest and largest eigenvalues of the Roe matrix [46].

For the linearization of inviscid flux functions, we use an implicit HLLC flux scheme with the frozen acoustic wave-speed originally introduced by Batten et.al. [51] for the finite volume methods, which demonstrated an uncompromised speed of convergence and robustness for smooth flows. The implicit form (differentiation) of the HLLC flux functions in Eq. (50) are given by

$$\mathbf{H}^{n+1}_{\mathrm{HLLC}}(\mathbf{U}_l, \mathbf{U}_r, \mathbf{n}_{ij}) = \begin{cases} \mathbf{H}^n_l + \dfrac{\partial \mathbf{H}_l}{\partial \mathbf{U}_l}\Delta\mathbf{U}_l & \text{if } S_L > 0 \\[2ex] (\mathbf{H}^*_l)^n + \dfrac{\partial \mathbf{H}^*_l}{\partial \mathbf{U}_l}\Delta\mathbf{U}_l + \dfrac{\partial \mathbf{H}^*_l}{\partial \mathbf{U}_r}\Delta\mathbf{U}_r & \text{if } S_L \le 0 < S_M \\[2ex] (\mathbf{H}^*_r)^n + \dfrac{\partial \mathbf{H}^*_r}{\partial \mathbf{U}_l}\Delta\mathbf{U}_l + \dfrac{\partial \mathbf{H}^*_r}{\partial \mathbf{U}_r}\Delta\mathbf{U}_r & \text{if } S_M \le 0 \le S_R \\[2ex] \mathbf{H}^n_r + \dfrac{\partial \mathbf{H}_r}{\partial \mathbf{U}_r}\Delta\mathbf{U}_r & \text{if } S_R < 0 \end{cases} \tag{63}$$

where for the supersonic case, $\partial\mathbf{H}_l/\partial\mathbf{U}_l$ and $\partial\mathbf{H}_r/\partial\mathbf{U}_r$ are linearization of the original flux functions that can be computed as below,

$$\frac{\partial \mathbf{H}}{\partial \mathbf{U}} \equiv \frac{\partial(\boldsymbol{F}\cdot\mathbf{n})}{\partial \mathbf{U}} = \begin{bmatrix} 0 & n_x & n_y & n_z & 0 \\ \frac{\gamma-1}{2}V^2 n_x - V_\perp u & (2-\gamma)n_x u + V_\perp & n_y u - (\gamma-1)n_x v & n_z u - (\gamma-1)n_x w & (\gamma-1)n_x \\ \frac{\gamma-1}{2}V^2 n_y - V_\perp v & n_x v - (\gamma-1)n_y u & (2-\gamma)n_y v + V_\perp & n_z v - (\gamma-1)n_y w & (\gamma-1)n_y \\ \frac{\gamma-1}{2}V^2 n_z - V_\perp w & n_x w - (\gamma-1)n_z u & n_y w - (\gamma-1)n_z v & (2-\gamma)w n_z + V_\perp & (\gamma-1)n_z \\ (\frac{\gamma-1}{2}V^2 - h)V_\perp & n_x h - (\gamma-1)V_\perp u & n_y h - (\gamma-1)V_\perp v & n_z h - (\gamma-1)V_\perp w & \gamma V_\perp \end{bmatrix} \tag{64}$$

where $V^2 \equiv \mathbf{V}\cdot\mathbf{V} \equiv u^2 + v^2 + w^2$, $V_\perp \equiv \mathbf{V}\cdot\mathbf{n}$, and $h = (\rho E + p)/\rho$.

For the subsonic case, the HLLC Jacobian matrices are given by

$$\frac{\partial \mathbf{H}_l^*}{\partial \mathbf{U}_l} = \begin{bmatrix} \left(\frac{\partial \rho_l^*}{\partial \mathbf{U}_l}\right)^{\mathrm{T}} S_M + \left(\frac{\partial S_M}{\partial \mathbf{U}_l}\right)^{\mathrm{T}} \rho_l^* \\ \left(\frac{\partial (\rho u)_l^*}{\partial \mathbf{U}_l}\right)^{\mathrm{T}} S_M + \left(\frac{\partial S_M}{\partial \mathbf{U}_l}\right)^{\mathrm{T}} (\rho u)_l^* + \left(\frac{\partial p_l^*}{\partial \mathbf{U}_l}\right)^{\mathrm{T}} n_x \\ \left(\frac{\partial (\rho v)_l^*}{\partial \mathbf{U}_l}\right)^{\mathrm{T}} S_M + \left(\frac{\partial S_M}{\partial \mathbf{U}_l}\right)^{\mathrm{T}} (\rho v)_l^* + \left(\frac{\partial p_l^*}{\partial \mathbf{U}_l}\right)^{\mathrm{T}} n_y \\ \left(\frac{\partial (\rho w)_l^*}{\partial \mathbf{U}_l}\right)^{\mathrm{T}} S_M + \left(\frac{\partial S_M}{\partial \mathbf{U}_l}\right)^{\mathrm{T}} (\rho w)_l^* + \left(\frac{\partial p_l^*}{\partial \mathbf{U}_l}\right)^{\mathrm{T}} n_z \\ \left(\frac{\partial (\rho E)_l^*}{\partial \mathbf{U}_l} + \frac{\partial p_l^*}{\partial \mathbf{U}_l}\right)^{\mathrm{T}} S_M + \left(\frac{\partial S_M}{\partial \mathbf{U}_l}\right)^{\mathrm{T}} ((\rho E)_l^* + p^*) \end{bmatrix},$$
$$\frac{\partial \mathbf{H}_l^*}{\partial \mathbf{U}_r} = \begin{bmatrix} \left(\frac{\partial \rho_l^*}{\partial \mathbf{U}_r}\right)^{\mathrm{T}} S_M + \left(\frac{\partial S_M}{\partial \mathbf{U}_r}\right)^{\mathrm{T}} \rho_l^* \\ \left(\frac{\partial (\rho u)_l^*}{\partial \mathbf{U}_r}\right)^{\mathrm{T}} S_M + \left(\frac{\partial S_M}{\partial \mathbf{U}_r}\right)^{\mathrm{T}} (\rho u)_l^* + \left(\frac{\partial p_l^*}{\partial \mathbf{U}_r}\right)^{\mathrm{T}} n_x \\ \left(\frac{\partial (\rho v)_l^*}{\partial \mathbf{U}_r}\right)^{\mathrm{T}} S_M + \left(\frac{\partial S_M}{\partial \mathbf{U}_r}\right)^{\mathrm{T}} (\rho v)_l^* + \left(\frac{\partial p_l^*}{\partial \mathbf{U}_r}\right)^{\mathrm{T}} n_y \\ \left(\frac{\partial (\rho w)_l^*}{\partial \mathbf{U}_r}\right)^{\mathrm{T}} S_M + \left(\frac{\partial S_M}{\partial \mathbf{U}_r}\right)^{\mathrm{T}} (\rho w)_l^* + \left(\frac{\partial p_l^*}{\partial \mathbf{U}_r}\right)^{\mathrm{T}} n_z \\ \left(\frac{\partial (\rho E)_l^*}{\partial \mathbf{U}_r} + \frac{\partial p_l^*}{\partial \mathbf{U}_r}\right)^{\mathrm{T}} S_M + \left(\frac{\partial S_M}{\partial \mathbf{U}_r}\right)^{\mathrm{T}} ((\rho E)_l^* + p^*) \end{bmatrix} \tag{65}$$

where the vectors $\partial S_M/\partial \mathbf{U}_l$, $\partial S_M/\partial \mathbf{U}_r$, $\partial p^*/\partial \mathbf{U}_l$, and $\partial p^*/\partial \mathbf{U}_r$ can be computed approximately.

The S_M derivatives are computed as follows

$$\frac{\partial S_M}{\partial \mathbf{U}_l} = \tilde{\rho}^{-1} \begin{bmatrix} -q_l^2 + V_l^2(\gamma-1)/2 + S_M S_L \\ n_x(2q_l - S_L - S_M) - (\gamma-1)u_l \\ n_y(2q_l - S_L - S_M) - (\gamma-1)v_l \\ n_z(2q_l - S_L - S_M) - (\gamma-1)w_l \\ \gamma-1 \end{bmatrix},$$
$$\frac{\partial S_M}{\partial \mathbf{U}_r} = \tilde{\rho}^{-1} \begin{bmatrix} -q_r^2 - V_r^2(\gamma-1)/2 - S_M S_R \\ n_x(-2q_r + S_R + S_M) + (\gamma-1)u_r \\ n_y(-2q_r + S_R + S_M) + (\gamma-1)v_r \\ n_z(-2q_r + S_R + S_M) + (\gamma-1)w_r \\ -(\gamma-1) \end{bmatrix} \tag{66}$$

where $\tilde{p} \equiv \rho_r(S_R - q_r) - \rho_l(S_L - q_l)$.

Taking differentiation of Eq. (57) with respect to $\mathbf{U}_l$ and $\mathbf{U}_r$ respectively gives

$$\begin{aligned} \frac{\partial p^*}{\partial \mathbf{U}_l} &= \rho_r(S_R - q_r)\frac{\partial S_M}{\partial \mathbf{U}_l}, \\ \frac{\partial p^*}{\partial \mathbf{U}_r} &= \rho_l(S_L - q_l)\frac{\partial S_M}{\partial \mathbf{U}_r}. \end{aligned} \tag{67}$$

The remaining terms in Eq. (65) are given as follows:

$$\begin{aligned} \frac{\partial \rho_l^*}{\partial \mathbf{U}_l} &= \Omega_l \rho_l^* \frac{\partial S_M}{\partial \mathbf{U}_l} + \Omega_l \begin{bmatrix} S_L \\ -n_x \\ -n_y \\ -n_z \\ 0 \end{bmatrix}, \\ \frac{\partial \rho_l^*}{\partial \mathbf{U}_r} &= \Omega_l \rho_l^* \frac{\partial S_M}{\partial \mathbf{U}_r}, \end{aligned} \tag{68}$$

and

$$\begin{aligned} \frac{\partial (\rho u)_l^*}{\partial \mathbf{U}_l} &= \Omega_l \left(n_x \frac{\partial p^*}{\partial \mathbf{U}_l} + (\rho u)_l^* \frac{\partial S_M}{\partial \mathbf{U}_l} \right) + \Omega_l \begin{bmatrix} q_l u_l - n_x V_l^2(\gamma-1)/2 \\ S_L - q_l + n_x(\gamma-2)u_l \\ -u_l n_y + n_x(\gamma-1)v_l \\ -u_l n_z + n_x(\gamma-1)w_l \\ -(\gamma-1)n_x \end{bmatrix}, \\ \frac{\partial (\rho u)_l^*}{\partial \mathbf{U}_r} &= \Omega_l \left(n_x \frac{\partial p^*}{\partial \mathbf{U}_r} + (\rho u)_l^* \frac{\partial S_M}{\partial \mathbf{U}_r} \right), \end{aligned} \tag{69}$$

and

$$\begin{aligned} \frac{\partial (\rho v)_l^*}{\partial \mathbf{U}_l} &= \Omega_l \left(n_y \frac{\partial p^*}{\partial \mathbf{U}_l} + (\rho v)_l^* \frac{\partial S_M}{\partial \mathbf{U}_l} \right) + \Omega_l \begin{bmatrix} q_l v_l - n_y V_l^2(\gamma-1)/2 \\ -v_l n_x + n_y(\gamma-1)u_l \\ S_L - q_l + n_y(\gamma-2)v_l \\ -v_l n_z + n_y(\gamma-1)w_l \\ -(\gamma-1)n_y \end{bmatrix}, \\ \frac{\partial (\rho v)_l^*}{\partial \mathbf{U}_r} &= \Omega_l \left(n_y \frac{\partial p^*}{\partial \mathbf{U}_r} + (\rho v)_l^* \frac{\partial S_M}{\partial \mathbf{U}_r} \right), \end{aligned} \tag{70}$$

and

$$\begin{aligned} \frac{\partial (\rho w)_l^*}{\partial \mathbf{U}_l} &= \Omega_l \left(n_z \frac{\partial p^*}{\partial \mathbf{U}_l} + (\rho w)_l^* \frac{\partial S_M}{\partial \mathbf{U}_l} \right) + \Omega_l \begin{bmatrix} q_l w_l - n_z V_l^2(\gamma-1)/2 \\ -w_l n_x + n_z(\gamma-1)u_l \\ -w_l n_y + n_z(\gamma-1)v_l \\ S_L - q_l + n_z(\gamma-2)w_l \\ -(\gamma-1)n_z \end{bmatrix}, \\ \frac{\partial (\rho w)_l^*}{\partial \mathbf{U}_r} &= \Omega_l \left(n_z \frac{\partial p^*}{\partial \mathbf{U}_r} + (\rho w)_l^* \frac{\partial S_M}{\partial \mathbf{U}_r} \right), \end{aligned} \tag{71}$$

and

$$
\begin{aligned}
\frac{\partial(\rho E)_l^*}{\partial \mathbf{U}_l} &= \Omega_l \left(\frac{\partial p^*}{\partial \mathbf{U}_l} S_M + (p^* + (\rho E)_l^*) \frac{\partial S_M}{\partial \mathbf{U}_l} \right) + \Omega_l \begin{bmatrix} ((\rho E)_l + p_l) q_l / \rho_l - q_l V_l^2 (\gamma - 1)/2 \\ -n_x ((\rho E)_l + p_l)/\rho_l + (\gamma - 1) u_l q_l \\ -n_y ((\rho E)_l + p_l)/\rho_l + (\gamma - 1) v_l q_l \\ -n_z ((\rho E)_l + p_l)/\rho_l + (\gamma - 1) w_l q_l \\ S_L - q_l \gamma \end{bmatrix}, \\
\frac{\partial(\rho E)_l^*}{\partial \mathbf{U}_r} &= \Omega_l \left(\frac{\partial p^*}{\partial \mathbf{U}_r} S_M + (p^* + (\rho E)_l^*) \frac{\partial S_M}{\partial \mathbf{U}_r} \right).
\end{aligned}
\tag{72}
$$

In the case where $S_M < 0$, the relevant HLLC Jacobian matrices are obtained by simply interchanging subscript $l \leftrightarrow r$ and $L \leftrightarrow R$ in Eq. (68) through Eq. (72). This completes the definition of the frozen acoustic wavespeed version of the implicit HLLC flux. In addition, it was also found by Batten et al. [51] that very little speedup could be further achieved by computing the fully linearized implicit HLLC flux, due to the fact that the extra work required to compute the Jacobians including the differentiations of the acoustic wavespeed does not significantly favor this version over the approximated form.

7 Boundary Conditions

Reference includes [57–59].

7.1 Slip Wall Boundary Condition

For the Euler equations, the boundary condition must prevent the fluid from penetrating the wall. We adopted a *weak Riemann* approach for applying the slip condition on the advection term. The procedures are as follows.

First, if the interior velocity at the boundary is $\mathbf{V}_\mathrm{i} = [u_\mathrm{i}, v_\mathrm{i}, w_\mathrm{i}]^\mathrm{T}$, the velocity at the ghost state can be calculated as:

$$\mathbf{V}_\mathrm{o} = \mathbf{V}_\mathrm{i} - 2(\mathbf{V}_\mathrm{i} \cdot \mathbf{n})\mathbf{n} \tag{73}$$

This is equivalent to imposing the same tangential component in the ghost state as in the interior, $\mathbf{V}_\mathrm{o,t} = \mathbf{V}_\mathrm{i,t}$, and an opposite normal component of the velocity, $\mathbf{V}_\mathrm{o,n} = -\mathbf{V}_\mathrm{i,n}$, where the subscript "t" denotes the tangential direction and the subscript "n" denotes the normal direction.

The density and internal energy are extrapolated from the interior such that the complete ghost state is

$$\mathbf{U}_\mathrm{o} = \begin{bmatrix} \rho_\mathrm{o} \\ (\rho u)_\mathrm{o} \\ (\rho v)_\mathrm{o} \\ (\rho w)_\mathrm{o} \\ (\rho E)_\mathrm{o} \end{bmatrix} = \begin{bmatrix} \rho_\mathrm{i} \\ \rho_\mathrm{i}(u_\mathrm{i} - 2(\mathbf{V}_\mathrm{i} \cdot \mathbf{n})n_x) \\ \rho_\mathrm{i}(v_\mathrm{i} - 2(\mathbf{V}_\mathrm{i} \cdot \mathbf{n})n_y) \\ \rho_\mathrm{i}(w_\mathrm{i} - 2(\mathbf{V}_\mathrm{i} \cdot \mathbf{n})n_z) \\ (\rho E)_\mathrm{i} \end{bmatrix}, \tag{74}$$

and the boundary flux is calculated through a Riemann solver as

$$\mathbf{H}_\mathrm{b}^\mathrm{inv} = \mathbf{H}^\mathrm{inv}(\mathbf{U}_\mathrm{i}, \mathbf{U}_\mathrm{o}, \mathbf{n}_\mathrm{ib}). \tag{75}$$

The normal component of the velocity evaluated by the Riemann solver is zero since the only non-zero contribution to the flux function are those coming from the pressure in the momentum equations.

The flux Jacobian matrix, $\mathrm{d}\mathbf{H}^\mathrm{inv}/\mathrm{d}\mathbf{U}_\mathrm{i}$, can be derived using the chain rule:

$$\frac{\mathrm{d}\mathbf{H}^\mathrm{inv}}{\mathrm{d}\mathbf{U}_\mathrm{i}} = \frac{\partial \mathbf{H}^\mathrm{inv}}{\partial \mathbf{U}_\mathrm{i}} + \frac{\partial \mathbf{H}^\mathrm{inv}}{\partial \mathbf{U}_\mathrm{o}} \frac{\mathrm{d}\mathbf{U}_\mathrm{o}}{\mathrm{d}\mathbf{U}_\mathrm{i}} \tag{76}$$

where the calculation of the matrices $\partial \mathbf{H}^\mathrm{inv}/\partial \mathbf{U}_\mathrm{i}$ and $\partial \mathbf{H}^\mathrm{inv}/\partial \mathbf{U}_\mathrm{o}$ is based on the specific flux function being used. The matrix $\mathrm{d}\mathbf{U}_\mathrm{o}/\mathrm{d}\mathbf{U}_\mathrm{i}$ is easy to calculate and given as below

$$\frac{\mathrm{d}\mathbf{U}_\mathrm{o}}{\mathrm{d}\mathbf{U}_\mathrm{i}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 - 2n_x^2 & -2n_x n_y & -2n_x n_z & 0 \\ 0 & -2n_x n_y & 1 - 2n_y^2 & -2n_y n_z & 0 \\ 0 & -2n_x n_z & -2n_y n_z & 1 - 2n_z^2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{77}$$

7.2 Riemann Invariant Boundary Condition

Type	Specify	Extrapolate	Update
Inflow / outflow	M_∞	Entropy (outflow)	ρ, p

The Riemann invariants correspond to the incoming R^- and outgoing R^+ characteristic waves. The invariants determine the locally normal velocity component and the speed of sound. The entropy, s, and the speed of sound, a, are used to determine the density and pressure on the boundary.

The outgoing Riemann invariant, R^+, uses the condition from the interior of the domain, $\mathbf{Q}_\mathrm{i} = [\rho_\mathrm{i}, u_\mathrm{i}, v_\mathrm{i}, w_\mathrm{i}, p_\mathrm{i}]^\mathrm{T}$:

$$R^+ = V_{\mathrm{i},\perp} + \frac{2a_\mathrm{i}}{\gamma - 1} \tag{78}$$

where

$$V_{\mathrm{i},\perp} = \mathbf{V}_\mathrm{i} \cdot \mathbf{n} = u_\mathrm{i} n_x + v_\mathrm{i} n_y + w_\mathrm{i} n_z \tag{79}$$

$$a_\mathrm{i} = \left(\frac{\gamma p_\mathrm{i}}{\rho_\mathrm{i}} \right)^{\frac{1}{2}} \tag{80}$$

The incoming Riemann invariant, R^-, uses the condition from the exterior of the domain, $\mathbf{Q}_\mathrm{o} = [\rho_\mathrm{o}, u_\mathrm{o}, v_\mathrm{o}, w_\mathrm{o}, p_\mathrm{o}]^\mathrm{T}$:

$$R^- = V_{\mathrm{o},\perp} - \frac{2a_\mathrm{o}}{\gamma - 1} \tag{81}$$

where

$$V_{\mathrm{o},\perp} = \mathbf{V}_\mathrm{o} \cdot \mathbf{n} = u_\mathrm{o} n_x + v_\mathrm{o} n_y + w_\mathrm{o} n_z \tag{82}$$

$$a_\mathrm{o} = \left(\frac{\gamma p_\mathrm{o}}{\rho_\mathrm{o}} \right)^{\frac{1}{2}} \tag{83}$$

If the flow at the boundary is locally supersonic leaving the domain, then no incoming characteristic waves exist; thus, R^- is set equal to

$$R^- = V_{\mathrm{i},\perp} + \frac{2a_\mathrm{i}}{\gamma - 1} \tag{84}$$

If the flow at the boundary is locally supersonic entering the domain, then no outgoing characteristic waves exist, and R^+ is set equal to

$$R^+ = V_{\mathrm{o},\perp} - \frac{2a_\mathrm{o}}{\gamma - 1} \tag{85}$$

A velocity, V_b, and the speed of sound, a_b, at the boundary are the sum and difference of the invariants:

$$V_\mathrm{b} = \frac{1}{2}(R^+ + R^-) \tag{86}$$

$$a_\mathrm{b} = \frac{1}{4}(\gamma - 1)(R^+ - R^-) \tag{87}$$

The velocity to be imposed on the boundary depends on the local direction of flow. A non-positive value of $V_{\mathrm{i},\perp}$ (i.e., $V_{\mathrm{i},\perp} \leq 0$) indicates that the flow is entering the domain, and the entropy of the exterior condition is used. The velocity and entropy on the boundary are calculated from the following equations:

$$u_\mathrm{b} = u_\mathrm{o} + (V_\mathrm{b} - V_{\mathrm{o},\perp})n_x \tag{88}$$

$$v_\mathrm{b} = v_\mathrm{o} + (V_\mathrm{b} - V_{\mathrm{o},\perp})n_y \tag{89}$$

$$w_\mathrm{b} = w_\mathrm{o} + (V_\mathrm{b} - V_{\mathrm{o},\perp})n_z \tag{90}$$

$$s_\mathrm{b} = \frac{a_\mathrm{o}^2}{\gamma \rho_\mathrm{o}^{\gamma-1}} \tag{91}$$

Conversely, if the sign of $V_{\mathrm{i},\perp}$ is positive ($V_{\mathrm{i},\perp} > 0$), then the flow is leaving the domain, and the entropy is extrapolated from the interior condition and is used to update the density at the boundary. The velocity and entropy on the boundary are calculated from the following equations:

$$u_\mathrm{b} = u_\mathrm{i} + (V_\mathrm{b} - V_{\mathrm{i},\perp})n_x \tag{92}$$

$$v_\mathrm{b} = v_\mathrm{i} + (V_\mathrm{b} - V_{\mathrm{i},\perp})n_y \tag{93}$$

$$w_\mathrm{b} = w_\mathrm{i} + (V_\mathrm{b} - V_{\mathrm{i},\perp})n_z \tag{94}$$

$$s_\mathrm{b} = \frac{a_\mathrm{i}^2}{\gamma \rho_\mathrm{i}^{\gamma-1}} \tag{95}$$

Denote the $V_{\mathrm{b},\perp}$ as below:

$$V_{\mathrm{b},\perp} = \mathbf{V}_\mathrm{b} \cdot \mathbf{n} = u_\mathrm{b} n_x + v_\mathrm{b} n_y + w_\mathrm{b} n_z \tag{96}$$

The density and pressure on the boundary are then calculated as follows:

$$\rho_\mathrm{b} = \left(\frac{a_\mathrm{b}^2}{\gamma s_\mathrm{b}} \right)^{\frac{1}{\gamma-1}} \tag{97}$$

$$p_\mathrm{b} = \frac{\rho_\mathrm{b} a_\mathrm{b}^2}{\gamma} \tag{98}$$

So the right-side state variable vector is

$$\mathbf{Q}_\mathrm{b} = \begin{bmatrix} \rho_\mathrm{b}, & u_\mathrm{b}, & v_\mathrm{b}, & w_\mathrm{b}, & p_\mathrm{b} \end{bmatrix}^\mathrm{T} \tag{99}$$

The flux Jacobian matrix, $\partial \mathbf{F} / \partial \mathbf{U}_\mathrm{i}$, can be derived using the chain rule:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{U}_\mathrm{i}} = \frac{\partial \mathbf{F}}{\partial \mathbf{Q}_\mathrm{b}} \frac{\partial \mathbf{Q}_\mathrm{b}}{\partial \mathbf{U}_\mathrm{i}}, \tag{100}$$

where the $\partial\mathbf{F}/\partial\mathbf{Q}_\mathrm{b}$ is given as below:

$$\frac{\partial\mathbf{F}}{\partial\mathbf{Q}_\mathrm{b}} = \begin{bmatrix} V_{\mathrm{b},\perp} & \rho_\mathrm{b} n_x & \rho_\mathrm{b} n_y & \rho_\mathrm{b} n_z & 0 \\ V_{\mathrm{b},\perp} u_\mathrm{b} & 2\rho_\mathrm{b} u_\mathrm{b} n_x & \rho_\mathrm{b} u_\mathrm{b} n_y & \rho_\mathrm{b} u_\mathrm{b} n_z & n_x \\ V_{\mathrm{b},\perp} v_\mathrm{b} & \rho_\mathrm{b} v_\mathrm{b} n_x & 2\rho_\mathrm{b} v_\mathrm{b} n_y & \rho_\mathrm{b} v_\mathrm{b} n_z & n_y \\ V_{\mathrm{b},\perp} w_\mathrm{b} & \rho_\mathrm{b} w_\mathrm{b} n_x & \rho_\mathrm{b} w_\mathrm{b} n_y & 2\rho_\mathrm{b} w_\mathrm{b} n_z & n_z \\ \dfrac{V_{\mathrm{b},\perp} V_\mathrm{b}^2}{2} & H_\mathrm{b} n_x + \rho_\mathrm{b} u_\mathrm{b} V_{\mathrm{b},\perp} & H_\mathrm{b} n_y + \rho_\mathrm{b} v_\mathrm{b} V_{\mathrm{b},\perp} & H_\mathrm{b} n_z + \rho_\mathrm{b} w_\mathrm{b} V_{\mathrm{b},\perp} & \dfrac{\gamma V_{\mathrm{b},\perp}}{\gamma-1} \end{bmatrix}, \tag{101}$$

where $H_\mathrm{b} = \rho E_\mathrm{b} + p_\mathrm{b}$. For the **inflow boundary**, the $\partial\mathbf{Q}_\mathrm{b}/\partial\mathbf{U}_\mathrm{i}$ can be computed as

$$\frac{\partial\mathbf{Q}_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} = \begin{bmatrix} \dfrac{\partial\rho_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} \\ \dfrac{\partial u_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} \\ \dfrac{\partial v_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} \\ \dfrac{\partial w_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} \\ \dfrac{\partial p_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} \end{bmatrix} = \begin{bmatrix} \dfrac{\rho_\mathrm{b}}{\gamma-1}\left(\dfrac{2}{a_\mathrm{b}}\dfrac{\partial a_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}}\right) \\ n_x\left(\dfrac{1}{2}\dfrac{\partial R^+}{\partial\mathbf{U}_\mathrm{i}}\right) \\ n_y\left(\dfrac{1}{2}\dfrac{\partial R^+}{\partial\mathbf{U}_\mathrm{i}}\right) \\ n_z\left(\dfrac{1}{2}\dfrac{\partial R^+}{\partial\mathbf{U}_\mathrm{i}}\right) \\ p_\mathrm{b}\left(\dfrac{1}{\rho_\mathrm{b}}\dfrac{\partial\rho_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} + \dfrac{2}{a_\mathrm{b}}\dfrac{\partial a_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}}\right) \end{bmatrix}. \tag{102}$$

For the **outflow boundary**, the $\partial\mathbf{Q}_\mathrm{b}/\partial\mathbf{U}_\mathrm{i}$ can be computed as

$$\frac{\partial\mathbf{Q}_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} = \begin{bmatrix} \dfrac{\partial\rho_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} \\ \dfrac{\partial u_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} \\ \dfrac{\partial v_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} \\ \dfrac{\partial w_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} \\ \dfrac{\partial p_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} \end{bmatrix} = \begin{bmatrix} \dfrac{\rho_\mathrm{b}}{\gamma-1}\left(\dfrac{2}{a_\mathrm{b}}\dfrac{\partial a_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} - \dfrac{1}{s_\mathrm{b}}\dfrac{\partial s_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}}\right) \\ n_x\left(\dfrac{1}{2}\dfrac{\partial R^+}{\partial\mathbf{U}_\mathrm{i}} - \dfrac{\partial V_{\mathrm{i},\perp}}{\partial\mathbf{U}_\mathrm{i}}\right) + \dfrac{\partial u_\mathrm{i}}{\partial\mathbf{U}_\mathrm{i}} \\ n_y\left(\dfrac{1}{2}\dfrac{\partial R^+}{\partial\mathbf{U}_\mathrm{i}} - \dfrac{\partial V_{\mathrm{i},\perp}}{\partial\mathbf{U}_\mathrm{i}}\right) + \dfrac{\partial v_\mathrm{i}}{\partial\mathbf{U}_\mathrm{i}} \\ n_z\left(\dfrac{1}{2}\dfrac{\partial R^+}{\partial\mathbf{U}_\mathrm{i}} - \dfrac{\partial V_{\mathrm{i},\perp}}{\partial\mathbf{U}_\mathrm{i}}\right) + \dfrac{\partial w_\mathrm{i}}{\partial\mathbf{U}_\mathrm{i}} \\ p_\mathrm{b}\left(\dfrac{1}{\rho_\mathrm{b}}\dfrac{\partial\rho_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} + \dfrac{2}{a_\mathrm{b}}\dfrac{\partial a_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}}\right) \end{bmatrix}, \tag{103}$$

where the $\partial s_\mathrm{b}/\partial\mathbf{U}_\mathrm{i}$ is

$$\frac{\partial s_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} = \frac{a_\mathrm{i}}{\gamma\rho_\mathrm{i}^\gamma}\left(2\rho_\mathrm{i}\frac{\partial a_\mathrm{i}}{\partial\mathbf{U}_\mathrm{i}} - (\gamma-1)a_\mathrm{i}\frac{\partial\rho_\mathrm{i}}{\partial\mathbf{U}_\mathrm{i}}\right). \tag{104}$$

Other derivatives needed to complete the calculation of Eqs. (102), (103), and (104) include

$$\frac{\partial a_\mathrm{b}}{\partial\mathbf{U}_\mathrm{i}} = \frac{\gamma-1}{4}\frac{\partial R^+}{\partial\mathbf{U}_\mathrm{i}}, \tag{105}$$

$$\frac{\partial R^{+}}{\partial \mathbf{U}_{\mathrm{i}}} = \frac{\partial V_{\mathrm{i},\perp}}{\partial \mathbf{U}_{\mathrm{i}}} + \frac{2}{\gamma-1}\frac{\partial a_{\mathrm{i}}}{\partial \mathbf{U}_{\mathrm{i}}}, \tag{106}$$

$$\frac{\partial a_{\mathrm{i}}}{\partial \mathbf{U}_{\mathrm{i}}} = \frac{a_{\mathrm{i}}}{2p_{\mathrm{i}}}\left(\frac{\partial p_{\mathrm{i}}}{\partial \mathbf{U}_{\mathrm{i}}} - \frac{p_{\mathrm{i}}}{\rho_{\mathrm{i}}}\frac{\partial \rho_{\mathrm{i}}}{\partial \mathbf{U}_{\mathrm{i}}}\right), \tag{107}$$

$$\frac{\partial V_{\mathrm{i},\perp}}{\partial \mathbf{U}_{\mathrm{i}}} = \left[-\frac{V_{\mathrm{i},\perp}}{\rho_{\mathrm{i}}}, \frac{n_x}{\rho_{\mathrm{i}}}, \frac{n_y}{\rho_{\mathrm{i}}}, \frac{n_z}{\rho_{\mathrm{i}}}, 0\right], \tag{108}$$

and

$$\frac{\partial \mathbf{Q}_{\mathrm{i}}}{\partial \mathbf{U}_{\mathrm{i}}} = \begin{bmatrix} \frac{\partial \rho_{\mathrm{i}}}{\partial \mathbf{U}_{\mathrm{i}}} \\ \frac{\partial u_{\mathrm{i}}}{\partial \mathbf{U}_{\mathrm{i}}} \\ \frac{\partial v_{\mathrm{i}}}{\partial \mathbf{U}_{\mathrm{i}}} \\ \frac{\partial w_{\mathrm{i}}}{\partial \mathbf{U}_{\mathrm{i}}} \\ \frac{\partial p_{\mathrm{i}}}{\partial \mathbf{U}_{\mathrm{i}}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{u_{\mathrm{i}}}{\rho_{\mathrm{i}}} & \frac{1}{\rho_{\mathrm{i}}} & 0 & 0 & 0 \\ -\frac{v_{\mathrm{i}}}{\rho_{\mathrm{i}}} & 0 & \frac{1}{\rho_{\mathrm{i}}} & 0 & 0 \\ -\frac{w_{\mathrm{i}}}{\rho_{\mathrm{i}}} & 0 & 0 & \frac{1}{\rho_{\mathrm{i}}} & 0 \\ -\frac{V_{\mathrm{i},\perp}}{\rho_{\mathrm{i}}} & \frac{n_x}{\rho_{\mathrm{i}}} & \frac{n_y}{\rho_{\mathrm{i}}} & \frac{n_z}{\rho_{\mathrm{i}}} & 0 \end{bmatrix}. \tag{109}$$

8 An Improved PCICE-FEM Algorithm for All-Speed Flows

8.1 Overview

Reference includes [44, 45].

8.2 Governing Equations

In order to develop an all-speed and all-fluid simulation capability, the conservative form of the Navier–Stokes equations are required. These equations in vector form for single-phase fluids are the conservation of mass,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = \dot{m}, \tag{110}$$

the balance of momentum,

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot ((\rho \mathbf{u}) \otimes \mathbf{u}) = -\nabla \cdot (p\boldsymbol{I}) + \nabla \cdot \boldsymbol{\tau} + \dot{\mathbf{b}}, \tag{111}$$

and the conservation of total energy,

$$\frac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho \mathbf{u} H) = \nabla \cdot (\boldsymbol{\tau} \cdot \mathbf{u}) + \nabla \cdot \mathbf{q} + \mathbf{u} \cdot \mathbf{b} + \dot{e}. \tag{112}$$

In Eqs. (110) – (112), ρ, $\rho\mathbf{u}$, and ρE are the conservative variables — density, momentum, and total energy, respectively; p is the thermodynamic pressure, $\boldsymbol{I}$ is the identity matrix; $\mathbf{u}$ is the velocity field vector; T is the absolute temperature; $\mathbf{q} = k\nabla T$ is the heat flux vector with k being the thermal conductivity; and $\boldsymbol{\tau}$ denotes the viscous stress tensor. The system source terms are the mass source, $\dot{m}$, the body force $\mathbf{b}$, the body force kinetic energy effect, $\mathbf{u} \cdot \mathbf{b}$, and the volumetric heat source, $\dot{e}$. In Eq. (112), H is the total enthalpy, and defined by

$$H = \frac{\rho E + p}{\rho}. \tag{113}$$

For closure, an equation of state (EOS) is required, and assumed to take the general form as below,

$$p = f(\rho, e), \tag{114}$$

where e is the specific internal energy. While the PCICE algorithm is not restricted to any specific EOS, the ideal gas EOS is used throughout the development for it's simplicity and functional dependence on the density and internal energy,

$$p = (\gamma - 1)\rho e = \rho R_c T. \tag{115}$$

where R_c is the gas constant per unit mass of the fluid.

The Navier–Stokes equations can also be written in a compact differential form as

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = \nabla \cdot \mathbf{F}_\mathrm{v} + \mathbf{Q}, \tag{116}$$

where $\mathbf{U}$ is the vector of the conservative variables, $(\rho, \rho\mathbf{u}, \rho E)^\mathrm{T}$. Also, $\mathbf{F}$ and $\mathbf{F}_\mathrm{v}$ are the convective and viscous flux vectors, and $\mathbf{Q}$ is the source vector, given by

$$\mathbf{F} = \begin{pmatrix} \rho\mathbf{u} \\ (\rho\mathbf{u}) \otimes \mathbf{u} + p\boldsymbol{I} \\ (\rho\mathbf{u})H \end{pmatrix}, \mathbf{F}_\mathrm{v} = \begin{pmatrix} 0 \\ \boldsymbol{\tau} \\ \boldsymbol{\tau} \cdot \mathbf{u} + \nabla \cdot \mathbf{q} \end{pmatrix}, \mathbf{Q} = \begin{pmatrix} \dot{m} \\ \mathbf{b} \\ \mathbf{u} \cdot \mathbf{b} + \dot{e} \end{pmatrix}. \tag{117}$$

By setting the right hand side to zero, Eq. (116) represents the compressible Euler equations.

8.3 The Improved PCICE Algorithm

The PCICE algorithm is based on the idea that both the balance of momentum and the conservation of mass equations can be solved simultaneously to provide for a mathematically strong coupling between p, $\rho\mathbf{u}$, and ρ. The PCICE algorithm is basically composed of two phases: 1) an explicit predictor, and 2) a semi-implicit corrector.

8.3.1 Explicit Predictor

The explicit predictor phase of the PCICE algorithm is composed of a basic two-step Runge–Kutta time integration scheme, which is second-order accurate in time,

$$\mathbf{U}^{(1/2)} = \mathbf{U}^n - \frac{\Delta t}{2} \nabla \cdot (\mathbf{F}_\mathrm{proj} - \mathbf{F}_\mathrm{v})^n + \frac{\Delta t}{2} \mathbf{Q}^n, \tag{118}$$

$$\mathbf{U}^{(1)} = \mathbf{U}^n - \Delta t \nabla \cdot (\mathbf{F}_\mathrm{proj} - \mathbf{F}_\mathrm{v})^{(1/2)} + \Delta t \mathbf{Q}^{(1/2)}, \tag{119}$$

where

$$\mathbf{F}_\mathrm{proj} = \begin{pmatrix} \rho\mathbf{u} \\ (\rho\mathbf{u}) \otimes \mathbf{u} \\ (\rho E)\mathbf{u} \end{pmatrix} \tag{120}$$

is the partial convective flux. The pressure gradient term is excluded in the explicit predictor phase of the PCICE algorithm, but it will be included in the next semi-implicit corrector phase. In Eq. (119), $\mathbf{U}^{(1)}$ is the partial solution for the explicit predictor phase of the PCICE algorithm. It is important that the quantities ρ, $\rho\mathbf{u}$, and ρE be advanced with high-order monotonic algorithms such as FCT, TVD, or ENO.

8.3.2 Semi-Implicit Corrector

The PCICE algorithm is based on the idea that both the balance of momentum and conservation of mass equations can be solved simultaneously.

Corrector Temporal Discretization

The PCICE algorithm temporal discretization of the governing equations is based on the second-order Crank–Nicolson scheme (CN2), in order:

1) the balance of momentum,

$$\begin{aligned}(\rho\mathbf{u})^{(i)} - (\rho\mathbf{u})^{n} = & -\frac{\Delta t}{2}\nabla\cdot\left[(\rho\mathbf{u}\otimes\mathbf{u})^{(i-1)} + (\rho\mathbf{u}\otimes\mathbf{u})^{n}\right] - \frac{\Delta t}{2}\nabla\cdot\left[(p^{(i)}+p^{n})\boldsymbol{I}\right] \\ & + \frac{\Delta t}{2}\nabla\cdot\left[(\boldsymbol{\tau}\cdot\mathbf{u})^{(i-1)} + (\boldsymbol{\tau}\cdot\mathbf{u})^{n}\right] + \frac{\Delta t}{2}(\mathbf{b}^{(i-1)}+\mathbf{b}^{n}),\end{aligned} \tag{121}$$

2) the conservation of mass,

$$\rho^{(i)} - \rho^{n} = -\frac{\Delta t}{2}\nabla\cdot\left[(\rho\mathbf{u})^{(i)} + (\rho\mathbf{u})^{n}\right] + \frac{\Delta t}{2}(\dot{m}^{(i-1)}+\dot{m}^{n}), \tag{122}$$

3) the conservation of total energy,

$$\begin{aligned}(\rho E)^{(i)} - (\rho E)^{n} = & -\frac{\Delta t}{2}\nabla\cdot\left[(\rho\mathbf{u})^{(i)}H^{(i)} + (\rho\mathbf{u})^{n}H^{n}\right] + \frac{\Delta t}{2}\nabla\cdot(\mathbf{q}^{(i-1)}+\mathbf{q}^{n}) \\ & + \frac{\Delta t}{2}\nabla\cdot\left[(\boldsymbol{\tau}\cdot\mathbf{u})^{(i-1)} + (\boldsymbol{\tau}\cdot\mathbf{u})^{n}\right] + \frac{\Delta t}{2}(\dot{e}^{(i-1)}+\dot{e}^{n}),\end{aligned} \tag{123}$$

where

$$H^{(i)} = \frac{(\rho E)^{(i-1)} + p^{(i)}}{\rho^{(i)}}. \tag{124}$$

In the above temporal discretization, i is a correction index, and $i = 1$ refers to the advanced time predictor solution, $\mathbf{U}^{(1)}$, given by Eq. (119). The PCICE algorithm requires two semi-implicit corrections to achieve second-order accuracy in time. For the semi-implicit correction in Eqs. (121)-(123), $i = 2$ and 3, respectively.

Intermediate Momentum Solution

Directly substituting Eq. (121) into Eq. (122) will yield the second-order derivatives of the outer product contained in the balance of momentum convective flux terms, and the third-order derivatives for the divergence of the divergence of the viscous stress tensor. In avoid those difficult terms, an intermediate explicit momentum solution, composed of the previous iteration terms in Eq. (121), is employed,

$$\begin{aligned}\mathbf{S}^{(i)} = (\rho\mathbf{u})^{n} & - \frac{\Delta t}{2}\nabla\cdot\left[(\rho\mathbf{u}\otimes\mathbf{u})^{(i-1)} + (\rho\mathbf{u}\otimes\mathbf{u})^{n}\right] \\ & + \frac{\Delta t}{2}\nabla\cdot\left[(\boldsymbol{\tau}\cdot\mathbf{u})^{(i-1)} + (\boldsymbol{\tau}\cdot\mathbf{u})^{n}\right] + \frac{\Delta t}{2}(\mathbf{b}^{(i-1)}+\mathbf{b}^{n}).\end{aligned} \tag{125}$$

Solving this intermediate step allows Eq. (121) to be re-written in terms of the intermediate momentum solution, $\mathbf{S}^{(i)}$, by substituting Eq. (125) into Eq. (121), yielding

$$(\rho\mathbf{u})^{(i)} = \mathbf{S}^{(i)} - \frac{\Delta t}{2}\nabla\cdot\left[(p^{(i)} + p^{n})\boldsymbol{I}\right]. \tag{126}$$

Eq. (126) is the pressure correction equation for the momentum components once $p^{(i)}$ is known. As will be shown below, the explicit values of $\mathbf{S}^{(i)}$ will be incorporated into the pressure Poisson equation, while the integral form of $\mathbf{S}^{(i)}$ will be employed in the momentum component pressure correction for efficiency.

Pressure Poisson Equation

The first point to consider is what pressure variable the pressure Poisson equation (PPE) should solve. There are three obvious choices: the thermodynamic pressure, $p^{(i)}$, the change in pressure across a time step, $p^{(i)} - p^{n}$, and a pressure correction variable, $\delta p^{(i)} = p^{(i)} - p^{(i-1)}$. We have found that a pressure correction variable provides the best performance, and great ease in applying the Dirichlet and Neumann boundary conditions for solving the Krylov subspace method, such as GMRES and BiCSTAB.

Solving the pressure correction variable for $\delta p^{(i)}$ and substituting into Eq. (126) yields

$$(\rho\mathbf{u})^{(i)} = \mathbf{S}^{(i)} - \frac{\Delta t}{2}\nabla\cdot\left[(\delta p^{(i)} + p^{(i-1)} + p^{n})\boldsymbol{I}\right]. \tag{127}$$

Eq. (127) is mathematically identical to Eq. (121), and is now in a form that can be easily substituted into Eq. (122), which eliminates $(\rho\mathbf{u})^{(i)}$ as an unknown, and yields the preliminary form of the PPE,

$$\rho^{(i)} - \rho^{n} = -\frac{\Delta t}{2}\nabla\cdot\left[\mathbf{S}^{(i)} + (\rho\mathbf{u})^{n}\right] + \frac{\Delta t}{2}(\dot{m}^{(i-1)} + \dot{m}^{n}) + \frac{\Delta t^2}{4}\nabla\cdot\nabla(\delta p^{(i)} + p^{(i-1)} + p^{n}). \tag{128}$$

Note that Eq. (128) is still a representation of the change in density composed of the explicit convection and source terms with an implicit pressure correction. For the same efficiency reason that we collect explicit momentum terms into $\mathbf{S}^{(i)}$, we now collect the explicit mass convection, source, and pressure terms of Eq. (128) into a new mass variable,

$$G^{(i)} = -\frac{\Delta t}{2}\nabla\cdot\left[\mathbf{S}^{(i)} + (\rho\mathbf{u})^{n}\right] + \frac{\Delta t}{2}(\dot{m}^{(i-1)} + \dot{m}^{n}) + \frac{\Delta t^2}{4}\nabla\cdot\nabla(p^{(i-1)} + p^{n}). \tag{129}$$

$G^{(i)}$ differs from $\mathbf{S}^{(i)}$ in that it will be used only in an integral form. Eq. (128) can now be represented in terms of $G^{(i)}$, yielding,

$$\rho^{(i)} - \rho^{n} = G^{(i)} + \frac{\Delta t^2}{4}\nabla\cdot\nabla\delta p^{(i)}. \tag{130}$$

Eq. (130) is the pressure correction equation of the PCICE algorithm for density once $\delta p^{(i)}$ is known.

Eq. (130) results from the substitution of the momentum balance equations into the mass conservation equations to eliminate the implicit momentum components as unknowns. This leaves the situation where there is now one equation and two unknowns, i.e., $\rho^{(i)}$ and $\delta p^{(i)}$. To achieve the closure, we employ the general-form EOS in Eq. (114) to express $\rho^{(i)}$ in terms of $\delta p^{(i)}$. Differentiating Eq. (114) with respect to time

$$\rho^{(i)} - \rho^n = \frac{\delta p^{(i)} + p^{(i-1)} - p^n}{\frac{\partial f}{\partial \rho}\big|_e} - \frac{\frac{\partial f}{\partial e}\big|_\rho}{\frac{\partial f}{\partial \rho}\big|_e}(e^{(i-1)} - e^n) \tag{131}$$

and equating with the change in density term of Eq.(130) yields the final form of the PPE of the PCICE algorithm,

$$\frac{1}{\frac{\partial f}{\partial \rho}\big|_e}\delta p^{(i)} - \frac{\Delta t^2}{4}\nabla\cdot\nabla\delta p^{(i)} = \frac{\frac{\partial f}{\partial e}\big|_\rho}{\frac{\partial f}{\partial \rho}\big|_e}(e^{(i-1)} - e^n) - \frac{1}{\frac{\partial f}{\partial \rho}\big|_e}(p^{(i-1)} - p^n) + G^{(i)}. \tag{132}$$

Pressure Correction Equations

At this point, all the pressure correction equations have appeared in the PCICE algorithm. With the solution of Eq. (132) for $\delta p^{(i)}$, Eqs. (126), (130), and (123) can be solved, in order:

1) momentum:

$$(\rho\mathbf{u})^{(i)} = \mathbf{S}^{(i)} - \frac{\Delta t}{2}\nabla\cdot\left[(p^{(i)} + p^n)\boldsymbol{I}\right]. \tag{133}$$

2) mass:

$$\rho^{(i)} = \rho^n + G^{(i)} + \frac{\Delta t^2}{4}\nabla\cdot\nabla\delta p^{(i)}. \tag{134}$$

3) total energy:

$$\begin{aligned}(\rho E)^{(i)} = (\rho E)^n &- \frac{\Delta t}{2}\nabla\cdot\left[(\rho\mathbf{u})^{(i)}H^{(i)} + (\rho\mathbf{u})^n H^n\right] + \frac{\Delta t}{2}\nabla\cdot(\mathbf{q}^{(i-1)} + \mathbf{q}^n)\\ &+ \frac{\Delta t}{2}\nabla\cdot\left[(\boldsymbol{\tau}\cdot\mathbf{u})^{(i-1)} + (\boldsymbol{\tau}\cdot\mathbf{u})^n\right] + \frac{\Delta t}{2}(\dot{e}^{(i-1)} + \dot{e}^n),\end{aligned} \tag{135}$$

The PCICE Algorithmic Steps

The algorithmic steps for the improved PCICE algorithm are as follows:

Algorithm 1 PCICE

```
1:  procedure EXPLICIT PREDICTOR
2:      Solve Eq. (118) for $\mathbf{U}^{(1/2)}$.
3:      Solve Eq. (119) for $\mathbf{U}^{(1)}$.
4:  end procedure
5:  procedure SEMI-IMPLICIT CORRECTOR
6:      Stage 1 of CN2:
7:          Solve Eq. (132) for $\delta p^{(2)}$, and subsequently $p^{(2)}$.
8:          Solve Eq. (125) for $\mathbf{S}^{(2)}$.
9:          Solve Eq. (133) for $(\rho\mathbf{u})^{(2)}$, Eq. (134) for $\rho^{(2)}$, and Eq. (135) for $(\rho E)^{(2)}$.
10:     Stage 2 of CN2:
11:         Solve Eq. (132) for $\delta p^{(3)}$, and subsequently $p^{(3)}$.
12:         Solve Eq. (125) for $\mathbf{S}^{(3)}$.
13:         Solve Eq. (133) for $(\rho\mathbf{u})^{(3)}$, Eq. (134) for $\rho^{(3)}$, and Eq. (135) for $(\rho E)^{(3)}$.
14: end procedure
```

9 Code Verification and Validation Test Cases

9.1 Overview

Verification testing is part of our software quality control process and ensures that BIGHORN is solving problems of interest to the Idaho National Laboratory's Laboratory Directed Research and Development (LDRD) project entitled "Development of a Multiphysics Algorithm for Analyzing the Integrity of Nuclear Reactor Containment Vessels Subjected to Extreme Thermal and Overpressure Loading Conditions", while meeting the necessary design requirements. It is one component of a larger testing infrastructure. This section identifies verification & validation problems of interest to the project, and summarizes the BIGHORN solutions to these problems. We anticipate that this section, like the code, will change over time. This section will have a wide audience of readers from the nuclear reactor safety and analysis areas, and is designed to be a concise report of both test setup and results. Contributors are encouraged to provide additional problems when appropriate, and include references to more in-depth discussions when needed. The tests are organized by methods and physics to enable a quick survey of code capabilities. Each test has a subsection in this section with subsubsections describing why the test is included (§Problem Description), and the setup and computational results of the test (§Problem Setup). All the files required to reproduce the test results are located in the BIGHORN repository under the `test/cnsfv` directory. Reference includes [60–63].

9.2 Sod Shock Tube

9.2.1 Problem Description

The Sod shock tube is a Riemann problem used as a standard test problem in computational hydrodynamics. The initial conditions are simple: a contact discontinuity separating gasses with different pressure and density and zero velocity everywhere. The governing equations are the 1D Euler equations with a constant ratio of specific heats of 1.4. The computational domain is bounded between $x = 0$ and $x = 1$, and 400 elements are uniformly distributed in the domain. The reflectory condition is imposed at the left and right boundaries. The initial conditions are described as below:

$$\begin{cases} \rho = 1, v_x = 0, p = 1 & \text{for } x = [0, 0.5] \\ \rho = 0.125, v_x = 0, p = 0.1 & \text{for } x = (0.5, 0.1] \end{cases}$$

9.2.2 Problem Setup

The numerical methods used to setup the simulations are summarized in the following table. The simulation was started at $t = 0$, and terminated at $t = 0.2$. The computed density, Mach number, and pressure profiles are compared with the analytical solution data, and plotted in Figs. 9, 10, 11, respectively.

Item	Specifics
Governing equations	1D Euler
Fluid properties	Ideal gas
Spatial discretization	Discontinuous Galerkin
Solution polynomial	Piecewise constant (cell-average)
Reconstruction	None
Stabilization scheme	Minmod slope limiter
Numerical flux scheme	HLLC
Execution	Transient
Temporal discretization	Explicit two-step TVD Runge-Kutta
Linear solver	None

9.2.3 Input Files

The input file for this test case can be found at `tests/cnsfv/1d_sod_shock_tube.i` under the BIGHORN repository.

9.2.4 Mesh Files

The framework generated mesh is used for this test case. No mesh file is required.

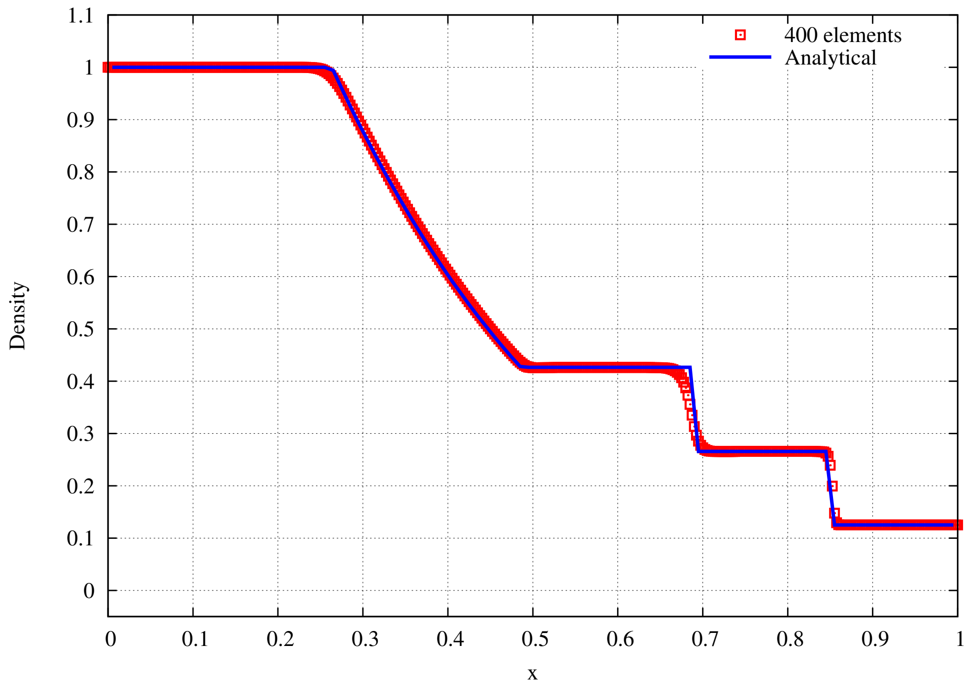


Figure 9. Density profile at $t = 0.2$ for the Sod shock tube.

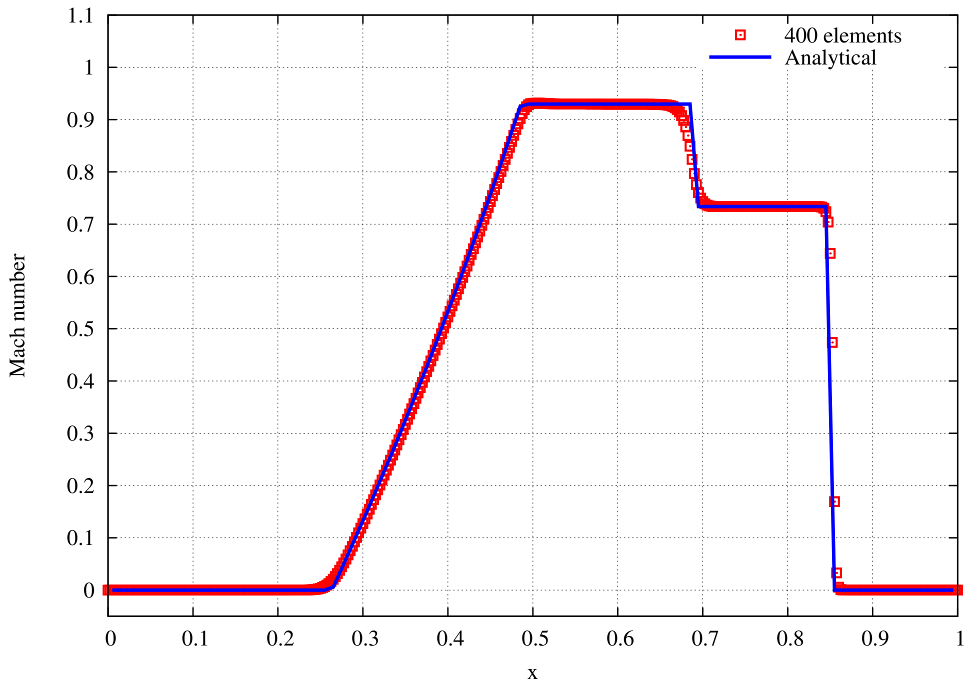


Figure 10. Mach number profile at $t = 0.2$ for the Sod shock tube.

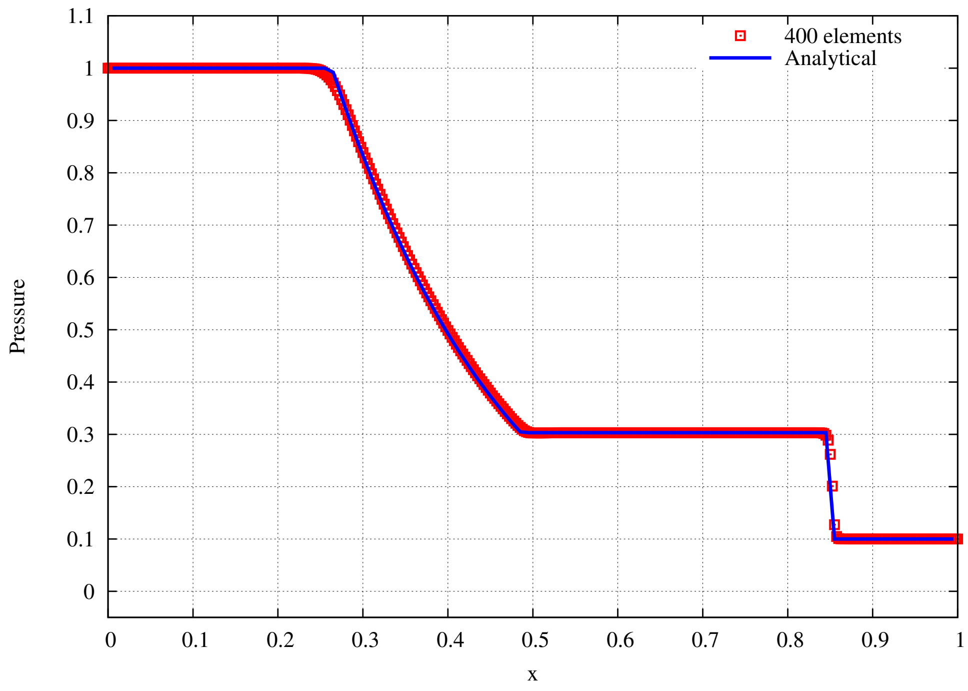


Figure 11. Pressure profile at $t = 0.2$ for the Sod shock tube.

9.3 Lax Shock Tube

9.3.1 Problem Description

The Lax shock tube is a Riemann problem used as a standard test problem in computational hydrodynamics. It is more challenging than the Sod shock tube problem, as the shock wave involved is much stronger in this problem. The governing equations are the 1D Euler equations with a constant ratio of specific heats of 1.4. The computational domain is bounded between $x = 0$ and $x = 1$, and 400 elements are uniformly distributed in the domain. The reflectory condition is imposed at the left and right boundaries. The initial conditions are described as below:

$$\begin{cases} \rho = 0.445, v_x = 0.698, p = 3.528 & \text{for } x = [0, 0.5] \\ \rho = 0.5, v_x = 0, p = 0.571 & \text{for } x = (0.5, 0.1] \end{cases}$$

9.3.2 Problem Setup

The numerical methods used to setup the simulations are summarized in the following table. The simulation was started at $t = 0$, and terminated at $t = 0.15$. The computed density, Mach number, and pressure profiles are plotted in Figs. 12, 13, 14, respectively.

Item	Specifics
Governing equations	1D Euler
Fluid properties	Ideal gas
Spatial discretization	Discontinuous Galerkin
Solution polynomial	Piecewise constant (cell-average)
Reconstruction	None
Stabilization scheme	Minmod slope limiter
Numerical flux scheme	HLLC
Execution	Transient
Temporal discretization	Explicit two-step TVD Runge-Kutta
Linear solver	None

9.3.3 Input Files

The input file for this test case can be found at `tests/cnsfv/1d_lax_shock_tube.i` under the BIGHORN repository.

9.3.4 Mesh Files

The framework generated mesh is used for this test case. No mesh file is required.

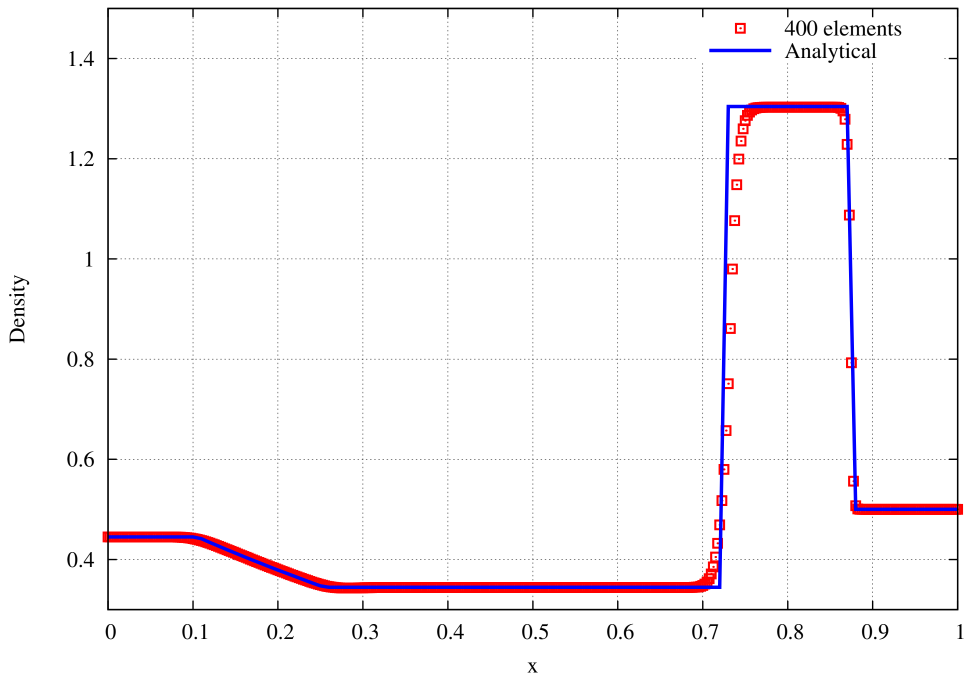


Figure 12. Density profile at $t = 0.15$ for the Lax shock tube.

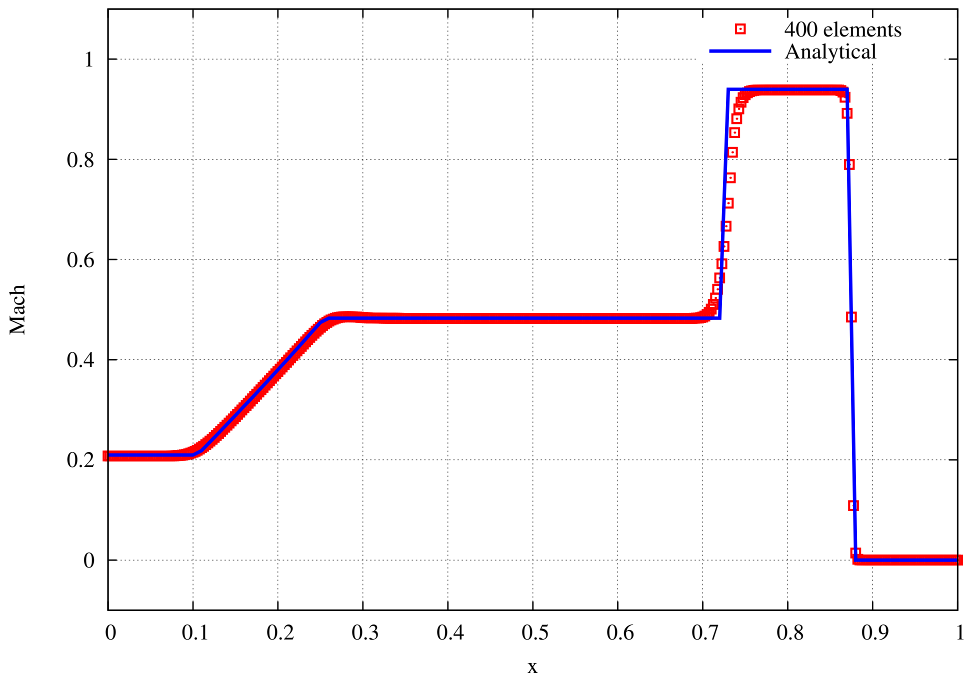


Figure 13. Mach number profile at $t = 0.15$ for the Lax shock tube.

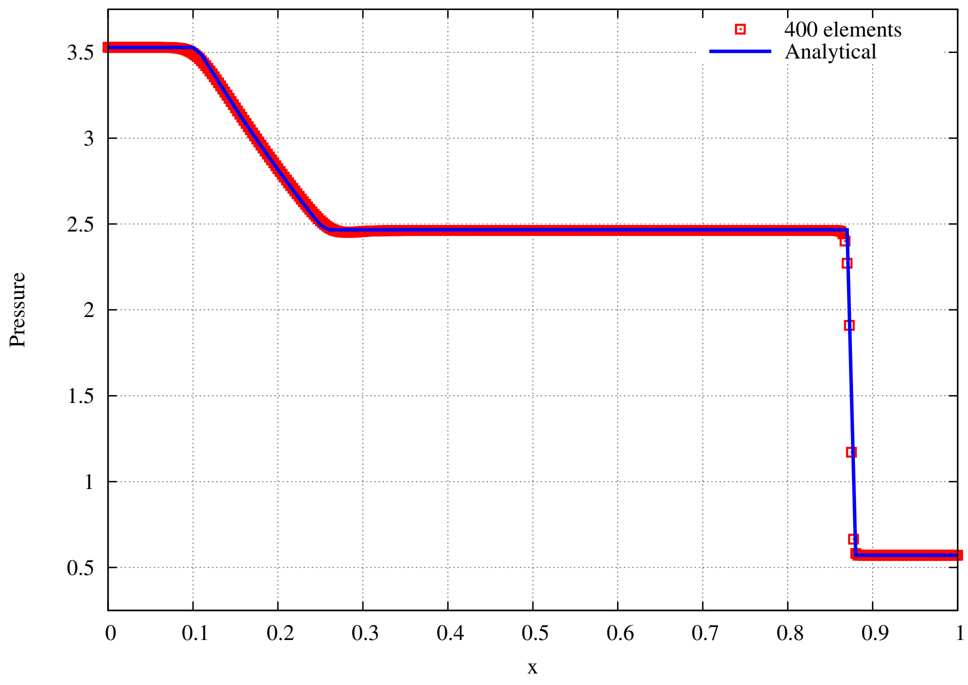


Figure 14. Pressure profile at $t = 0.15$ for the Lax shock tube.

9.4 Woodward–Collela Blast Wave

9.4.1 Problem Description

This problem was first used as a test problem by Woodward & Colella [64]. The two discontinuities in the initial data each have the form of a shock-tube problem, and yield strong shock waves and contact discontinuities going inwards and rarefaction waves going outwards. This is a challenging test problem because of the strength of the shocks involved and the interaction of the different waves. The governing equations are the 1D Euler equations with a constant ratio of specific heats of 1.4. The computational domain is bounded between $x = 0$ and $x = 1$. Two meshes that consist of 500 and 4000 equi-distant elements respectively, were used in computation. The reflectory condition is imposed at the left and right boundaries. The initial conditions can be described as below:

$$\begin{cases} \rho = 1, v_x = 0, p = 1000 & \text{for } x = [0, 0.1] \\ \rho = 1, v_x = 0, p = 0.01 & \text{for } x = (0.1, 0.9] \\ \rho = 1, v_x = 0, p = 100 & \text{for } x = (0.9, 1] \end{cases}$$

9.4.2 Problem Setup

The numerical methods used to setup the simulations are summarized in the following table. The simulation was started at $t = 0$, and terminated at $t = 0.038$. The computed density, velocity, and pressure profiles are plotted in Figs. 15, 16, 17, respectively.

Item	Specifics
Governing equations	1D Euler
Fluid properties	Ideal gas
Spatial discretization	Discontinuous Galerkin
Solution polynomial	Piecewise constant (cell-average)
Reconstruction	None
Stabilization scheme	Minmod slope limiter
Numerical flux scheme	HLLC
Execution	Transient
Temporal discretization	Explicit two-step TVD Runge-Kutta
Linear solver	None

9.4.3 Input Files

The input file for this test case can be found at `tests/cnsfv/1d_woodward_colella_blast_wave.i` under the BIGHORN repository.

9.4.4 Mesh Files

The framework generated mesh is used for this test case. No mesh file is required.

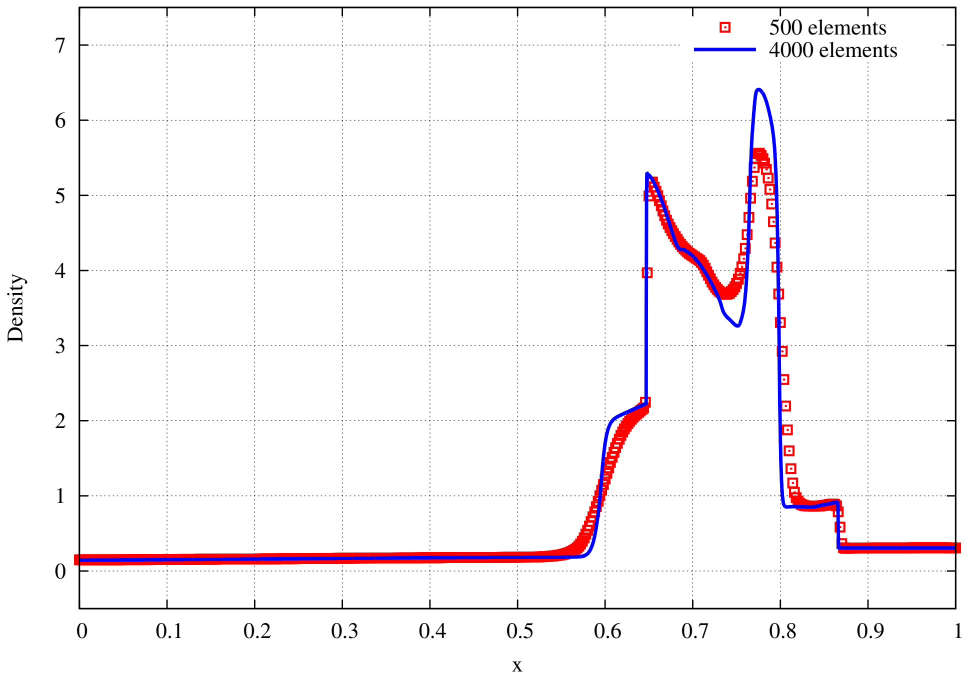


Figure 15. Density profile at $t = 0.038$ for the Woodward–Collela blast wave.

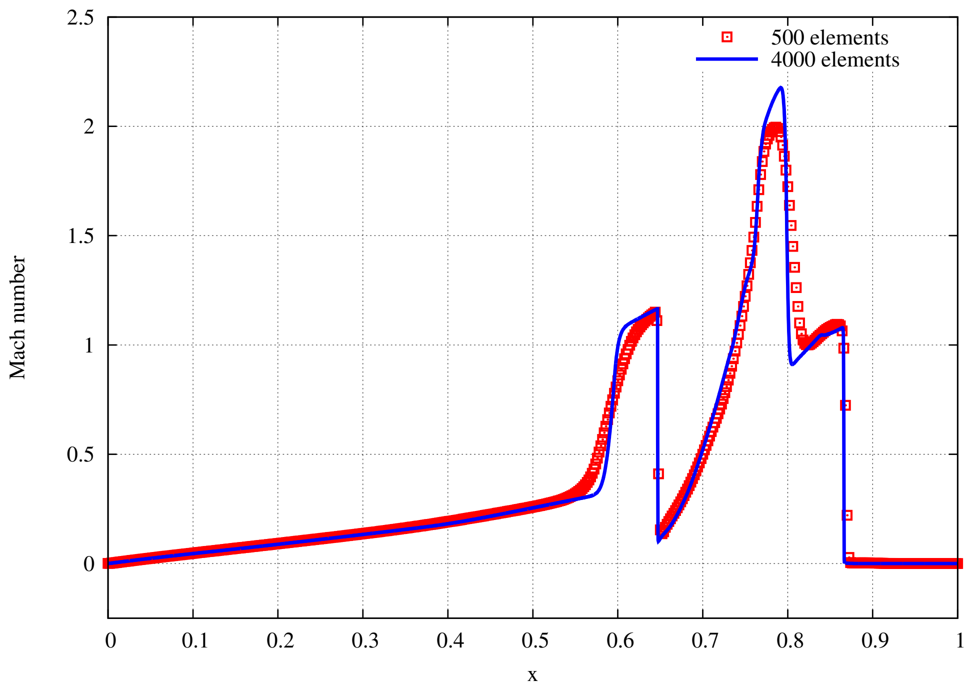


Figure 16. Mach number profile at $t = 0.038$ for the Woodward–Collela blast wave.

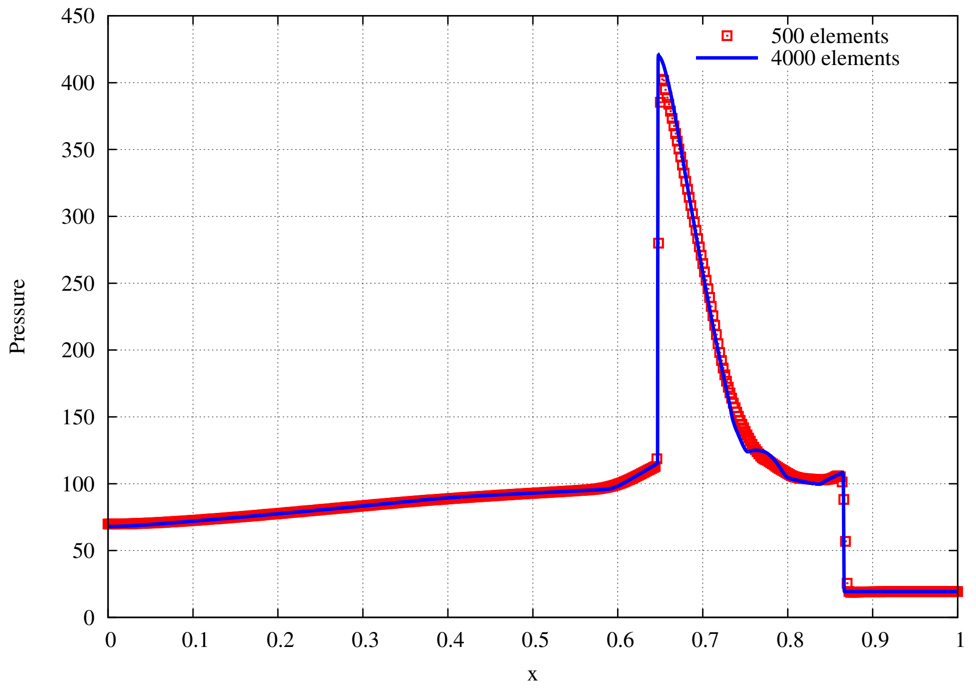


Figure 17. Pressure profile at $t = 0.038$ for the Woodward–Collela blast wave.

9.5 Sedov Blast Wave in 1D

9.5.1 Problem Description

The solution for a blast wave created by a "point explosion" in a uniform medium was derived independently during the World War II by John von Neumann, Leonid Sedov, and by Sir Geoffrey Taylor. The exact solution is specified in detail in [65,66]. This problem contains very low density with strong shocks. In this test case, the governing equations are the 1D Euler equations with a constant ratio of specific heats of $5/3$. The computational domain is bounded between $x = 0$ and $x = 1$. Two meshes that consist of 400 and 800 equi-distant elements respectively, were used in computation. The symmetry condition is imposed at the left boundary, and the free outflow condition at the right boundary. The initial conditions can be described as below:

$$\begin{cases} \rho = 1, v_x = 0, p = 1591549 & \text{for } x = [0, \Delta x] \\ \rho = 1, v_x = 0, p = 1e-8 & \text{for } x = (\Delta x, 1] \end{cases}$$

where Δx means the length of a single element.

9.5.2 Problem Setup

The numerical methods used to setup the simulations are summarized in the following table. The simulation was started at $t = 0$, and terminated at $t = 0.005$. The computed density, velocity, and pressure profiles are plotted in Figs. 18, 19, 20, respectively.

Item	Specifics
Governing equations	1D Euler
Fluid properties	Ideal gas
Spatial discretization	Discontinuous Galerkin
Solution polynomial	Piecewise constant (cell-average)
Reconstruction	None
Stabilization scheme	Minmod slope limiter
Numerical flux scheme	HLLC
Execution	Transient
Temporal discretization	Explicit two-step TVD Runge-Kutta
Linear solver	None

9.5.3 Input Files

The input files for this test case can be found at `tests/cnsfv/1d_sedov_blast_wave.i` and `tests/cnsfv/1d_sedov_blast_wave_dtmin.i` under the BIGHORN repository. They correspond to the two cases of 1) fixed time-step size, and 2) CFL-condition allowed maximum time-step size, respectively.

9.5.4 Mesh Files

The framework generated mesh is used for this test case. No mesh file is required.

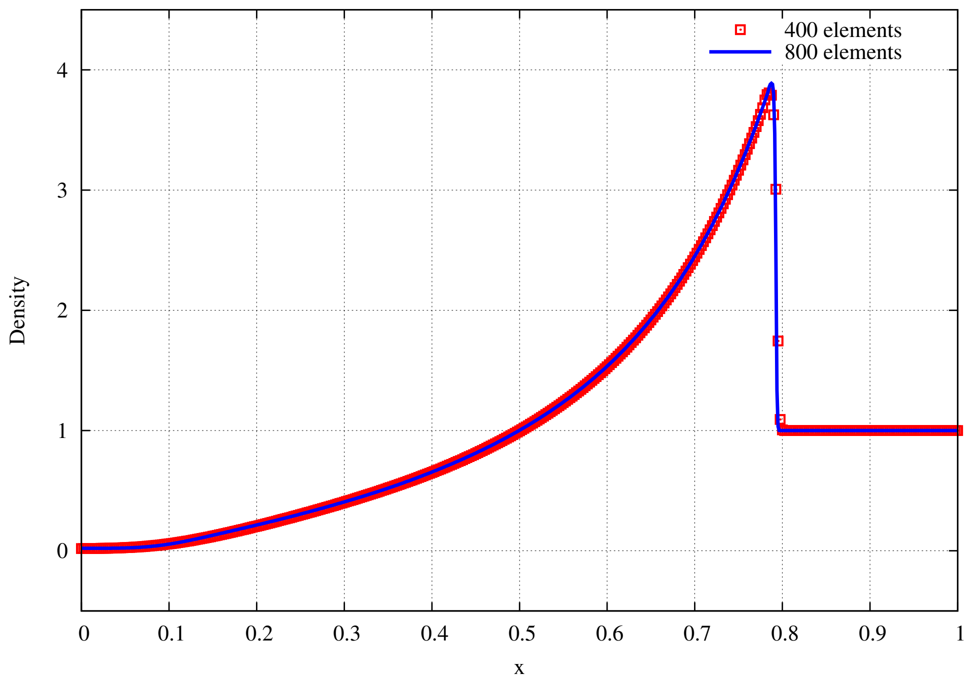


Figure 18. Density profile at $t = 0.005$ for the Sedov blast wave.

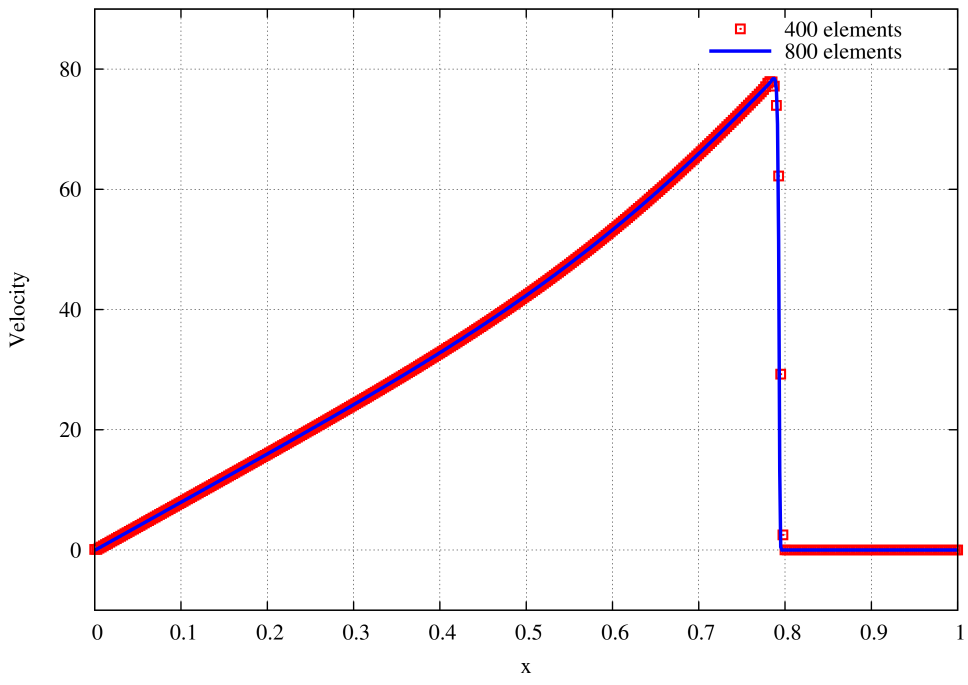


Figure 19. Velocity profile at $t = 0.005$ for the Sedov blast wave.

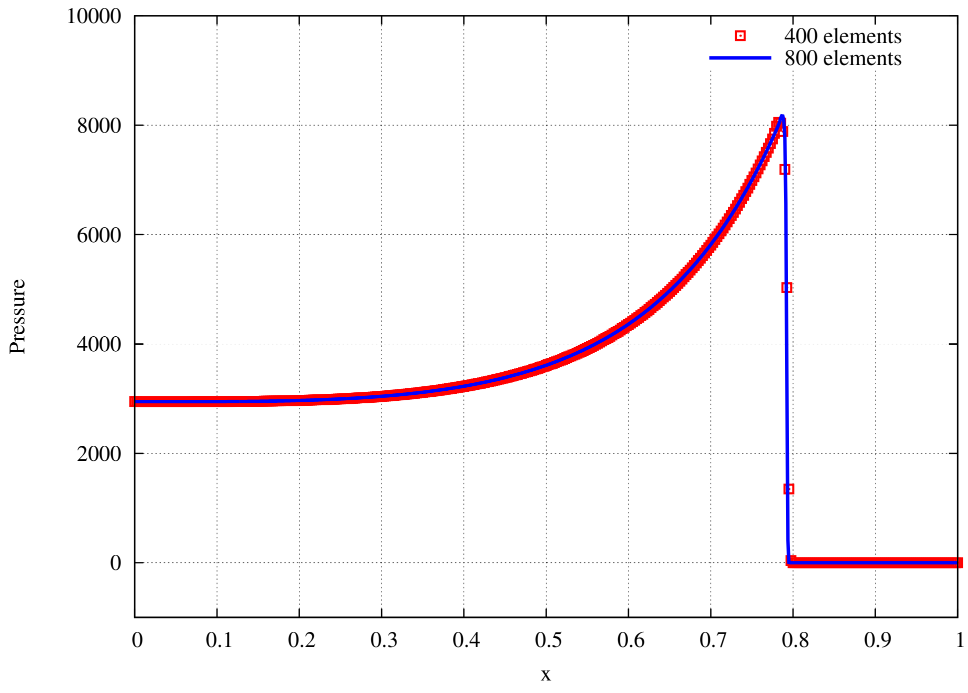


Figure 20. Pressure profile at $t = 0.005$ for the Sedov blast wave.

9.6 Double Rarefaction Wave

9.6.1 Problem Description

This problem was originally introduced by Linde & Roe [67] in order to test the robustness of the Euler solver codes. This test case contains low pressure and low density regions, and is difficult to simulate. The governing equations are the 1D Euler equations with a constant ratio of specific heats of 1.4. The computational domain is bounded between $x = 0$ and $x = 1$. Two meshes that consist of 400 and 800 equi-distant elements respectively, were used in computation. The free outflow condition is imposed at both the left and right boundaries. The initial conditions can be described as below:

$$\begin{cases} \rho = 7, v_x = -1, p = 0.2 & \text{for } x = [-1, 0] \\ \rho = 7, v_x = 1, p = 0.2 & \text{for } x = (0, 1] \end{cases}$$

9.6.2 Problem Setup

The numerical methods used to setup the simulations are summarized in the following table. The simulation was started at $t = 0$, and terminated at $t = 0.6$. The computed density, velocity, and pressure profiles are plotted in Figs. 21, 22, 23, respectively.

Item	Specifics
Governing equations	1D Euler
Fluid properties	Ideal gas
Spatial discretization	Discontinuous Galerkin
Solution polynomial	Piecewise constant (cell-average)
Reconstruction	None
Stabilization scheme	Minmod slope limiter
Numerical flux scheme	HLLC
Execution	Transient
Temporal discretization	Explicit two-step TVD Runge-Kutta
Linear solver	None

9.6.3 Input Files

The input files for this test case can be found at `tests/cnsfv/1d_double_rarefaction_wave.i` under the BIGHORN repository.

9.6.4 Mesh Files

The framework generated mesh is used for this test case. No mesh file is required.

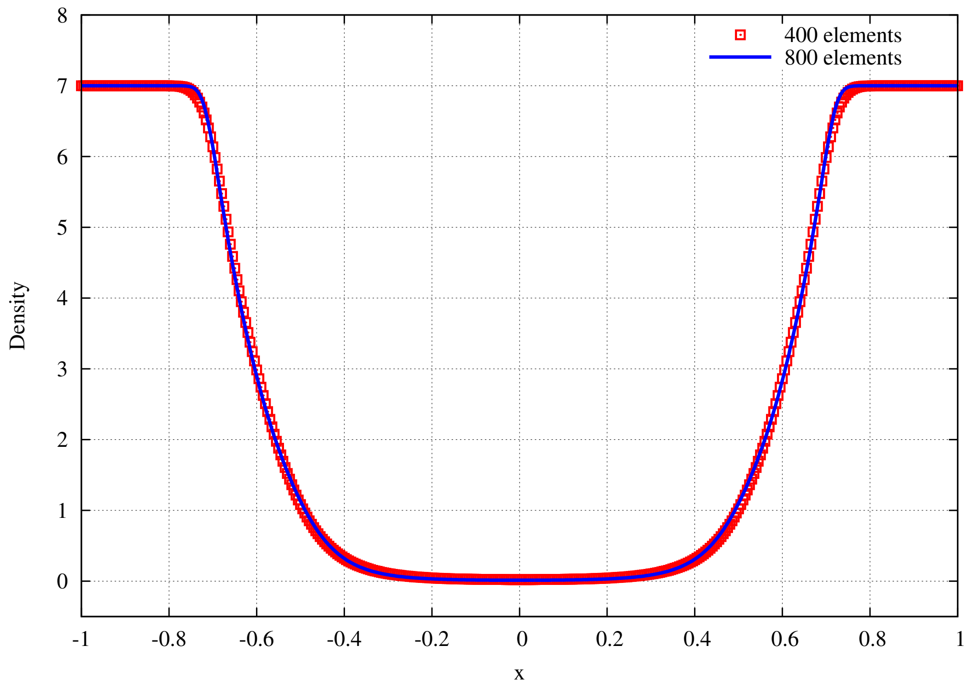


Figure 21. Density profile at $t = 0.6$ for the double rarefaction wave.

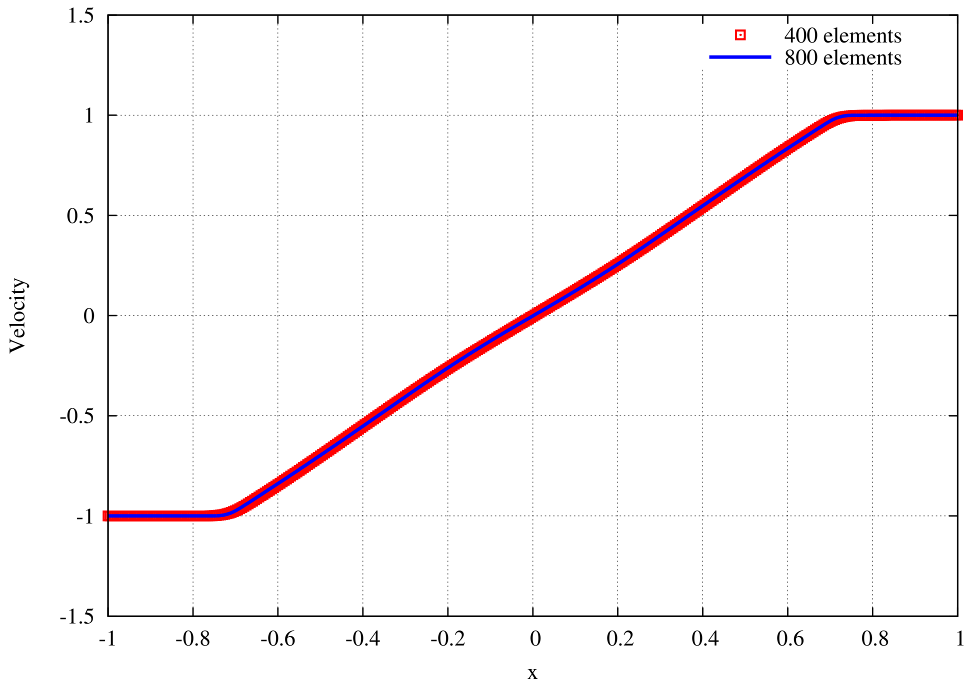


Figure 22. Velocity profile at $t = 0.6$ for the double rarefaction wave.

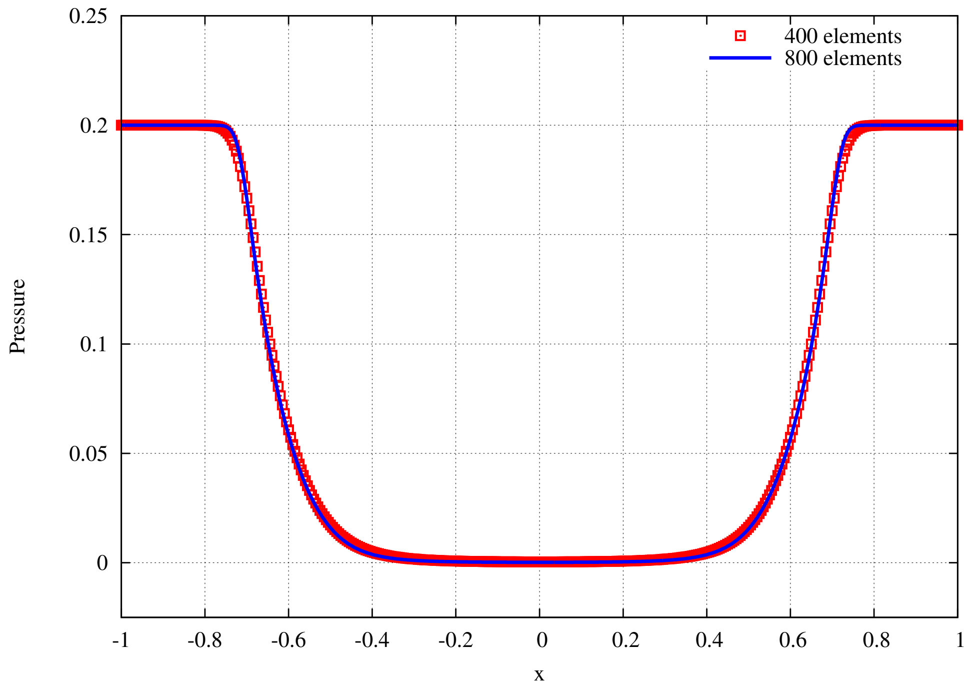


Figure 23. Pressure profile at $t = 0.6$ for the double rarefaction wave.

9.7 Inviscid Flow through a 2D Channel

9.7.1 Problem Description

This problem is aimed at testing the developed numerical methods for the computation of internal flow. In this subsonic flow problem, the geometry is smooth, and so is the flow. Entropy should be a constant in the flow field. The L_2 norm of the entropy error can be used as the indicator of solution accuracy since the analytical solution is unknown. The convergence rate can be expected to be $P+1$, where P is order of the discrete polynomial approximation. The governing equations are the 2D Euler equations with a constant ratio of specific heats of 1.4. The upper and lower walls are slip walls. The computational domain is bounded between $x=-1.5$ and $x=1.5$, and between the bump and $y=0.8$, as shown in Fig. 24. The bump is defined as

$$y = 0.0625e^{-25x^2}.$$

The inflow Mach number is 0.5 with 0 angle of attack. Total pressure and temperature is imposed as an inflow boundary condition on the left boundary, and a constant static pressure is imposed on the outflow boundary.

9.7.2 Problem Setup

The numerical methods used to setup the simulations are summarized in the following table. The

Item	Specifics
Governing equations	2D Euler
Fluid properties	Ideal gas
Spatial discretization	Discontinuous Galerkin
Solution polynomial	Piecewise constant (cell-average)
Reconstruction	Least-squares slope reconstruction
Stabilization scheme	None
Numerical flux scheme	HLLC
Execution	Steady
Temporal discretization	Implicit backward Euler
Linear solver	ILU(0) preconditioned GMRES

main objective is to demonstrate grid convergence of drag on a sequence of successively refined meshes. As shown in Fig. 24, four unstructured quadrilateral meshes were used for calculations. The simulations were started from a uniform free stream Mach number of 0.5 everywhere, and the L_2 norm of the density residual was monitored. The computed steady-state Mach number contours are shown in Fig. 25. The following entropy error was used as the accuracy indicator:

$$Err_{L_2(\Omega)} = \sqrt{\int_\Omega \left(\frac{p}{p_\infty} (\frac{\rho_\infty}{\rho})^\gamma - 1 \right)^2 \mathrm{d}V}$$

The plot of L_2 entropy error vs. length scale h (i.e., $h = (\text{nDOFs})^{-1/2}$) is displayed in Fig. 26, which shows an average order of accuracy of 1.81. The plot of L_2 entropy error vs. time-step is shown in Fig. 27. All the tests reached full convergence within 10 time steps. In Table 2, the raw data is provided in three columns, i.e. h, L_2 error, and work units.

Table 2. Raw data for inviscid flow through a channel.

Mesh	nDOFs	h	L_2 error	Work units
Ref.2	260	0.06201737	0.00386033	
Ref.3	900	0.03333333	0.00133776	
Ref.4	3332	0.01732397	0.00036614	
Ref.5	12804	0.00883745	0.00011455	

9.7.3 Input Files

A set of four input files for this test case can be found at

- `tests/cnsfv/2d_bump_impl_ref2.i`
- `tests/cnsfv/2d_bump_impl_ref3.i`
- `tests/cnsfv/2d_bump_impl_ref4.i`
- `tests/cnsfv/2d_bump_impl_ref5.i`

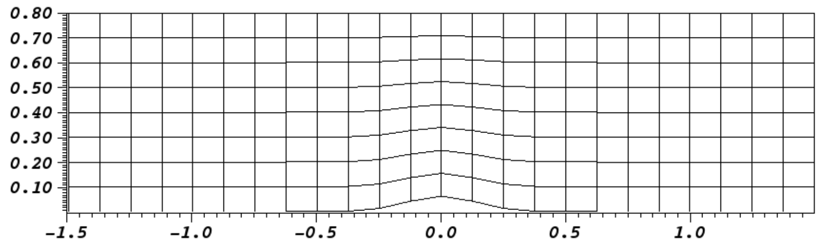
under the BIGHORN repository, respectively.

9.7.4 Mesh Files

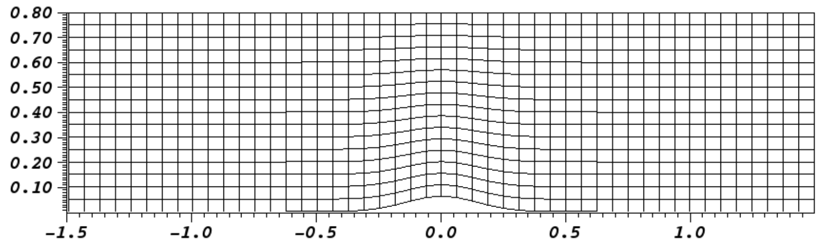
A set of four **GMSH** format mesh files corresponding to the four input files can be found at

- `tests/cnsfv/SmoothBump_quad_ref2_Q1.msh`
- `tests/cnsfv/SmoothBump_quad_ref3_Q1.msh`
- `tests/cnsfv/SmoothBump_quad_ref4_Q1.msh`
- `tests/cnsfv/SmoothBump_quad_ref5_Q1.msh`

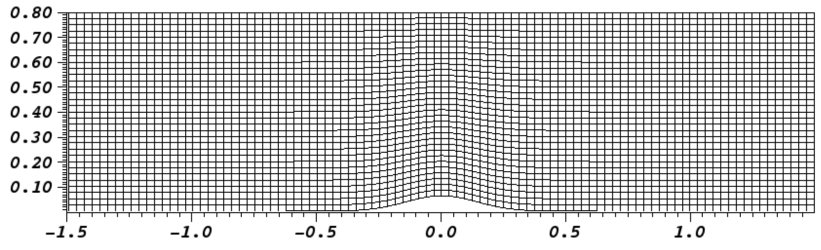
under the BIGHORN repository, respectively.



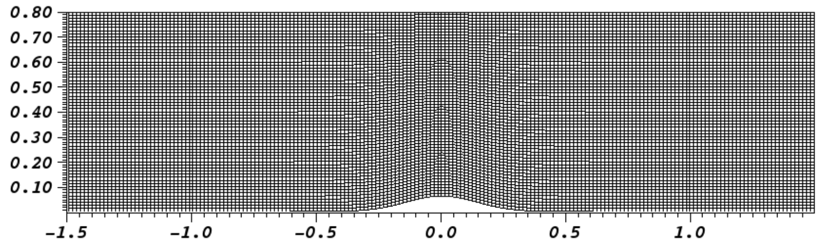
(a) Ref. 2



(b) Ref. 3

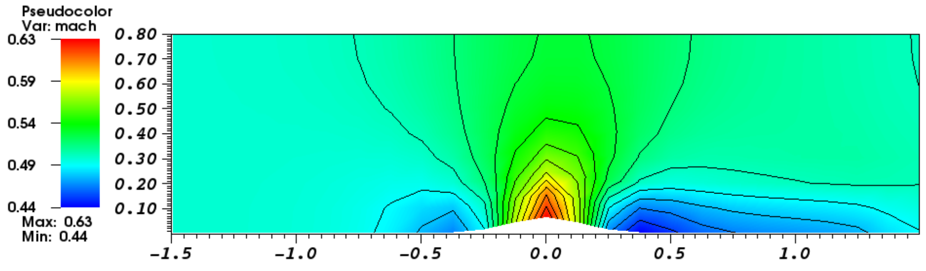


(c) Ref. 4

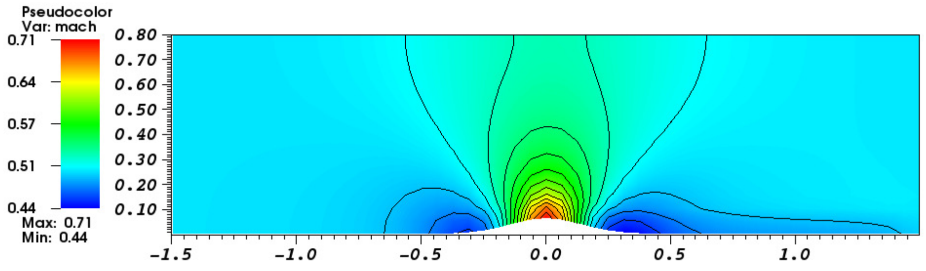


(d) Ref. 5

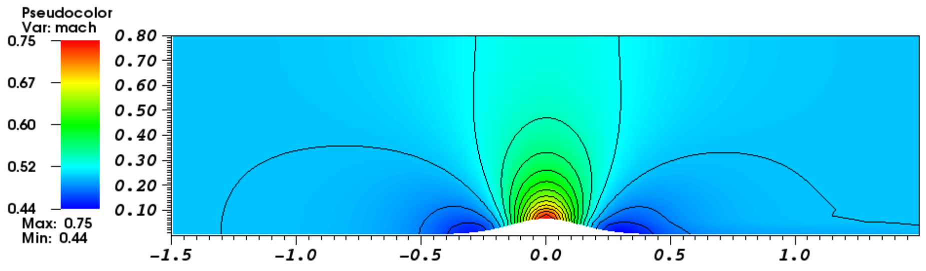
Figure 24. Computational meshes for inviscid flow through a channel.



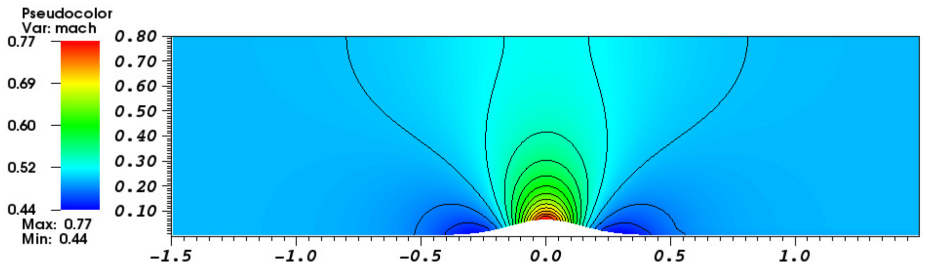
(a) Ref. 2



(b) Ref. 3



(c) Ref. 4



(d) Ref. 5

Figure 25. Steady-state Mach number contours for inviscid flow through a channel.

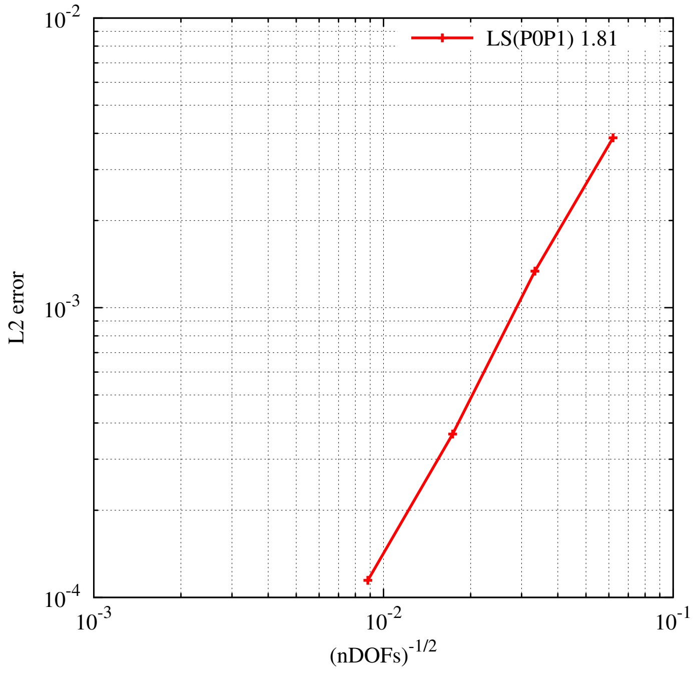


Figure 26. L_2 entropy error vs. length scale h for inviscid flow through a channel.

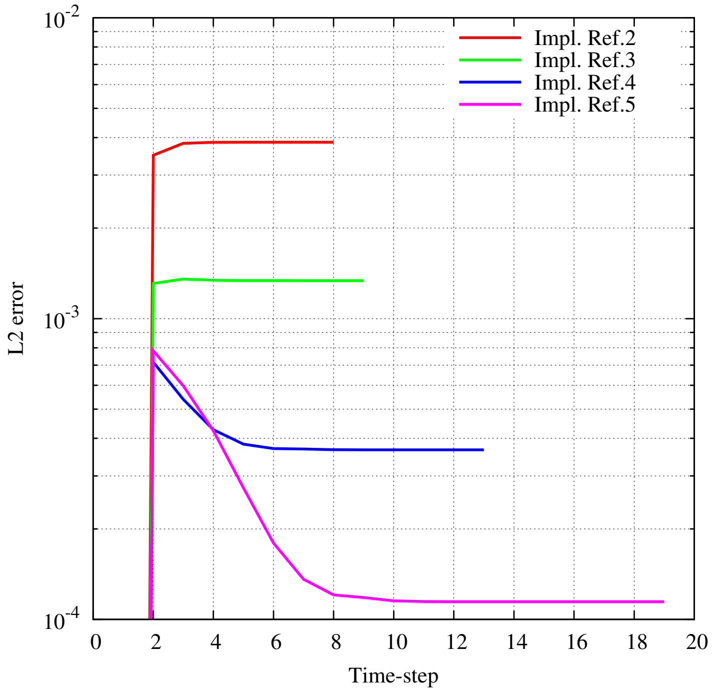


Figure 27. L_2 entropy error vs. time-step for inviscid flow through a channel.

9.8 Inviscid Flow past a 2D Circular Cylinder

9.8.1 Problem Description

This problem is aimed at testing the developed numerical methods for the computation of external flow. In this subsonic flow problem, the geometry is smooth, and so is the flow. Entropy should be a constant in the flow field. The L_2 norm of the entropy error can be used as the indicator of solution accuracy since the analytical solution is unknown. The convergence rate can be expected to be $P+1$, where P is order of the discrete polynomial approximation. The governing equations are the 2D Euler equations with a constant ratio of specific heats of 1.4. The computational domain is bounded between $r = 0.5$ and $r = 20$, as shown in Fig. 28. The infinity-condition Mach number is 0.5 with 0 angle of attack. The slip condition is imposed on the wall surface of the cylinder, and the Riemann invariant condition is imposed on the far-field boundary.

9.8.2 Problem Setup

The numerical methods used for running the simulations in this test case are summarized in the following table. The main objective is to demonstrate grid convergence of drag on a sequence of

Item	Specifics
Governing equations	2D Euler
Fluid properties	Ideal gas
Spatial discretization	Discontinuous Galerkin
Solution polynomial	Piecewise constant (cell-average)
Reconstruction	Least-squares slope reconstruction
Stabilization scheme	None
Numerical flux scheme	HLLC
Execution	Steady
Temporal discretization	Implicit backward Euler
Linear solver	ILU(0) preconditioned GMRES

successively refined meshes. As shown in Fig. 28, three unstructured quadrilateral meshes were used for calculations. Let us take the Lv.2 mesh as an example as to explain how the meshes were generated. First, 8 intervals were specified along the axial direction, with a 0.008 fraction of the axial length of the domain for the first interval adjacent to the cylinder wall. 32 constant intervals were then specified radially, resulting in a mesh that contains 32×8 elements. The Lv.3 and Lv.4 meshes were then generated through successive mesh refinement, where the nodes of the cylinder surface were fitted on the perimeter of the inner circle. The simulations were started from a uniform free stream Mach number of 0.5 everywhere, and the L_2 norm of the density residual was monitored. The computed steady-state Mach number contours are shown in Fig. 29. The following entropy error was used as the accuracy indicator:

$$Err_{L_2(\Omega)} = \sqrt{\int_{\Omega} \left(\frac{p}{p_\infty} (\frac{\rho_\infty}{\rho})^\gamma - 1 \right)^2 \mathrm{d}V}$$

The plot of L_2 entropy error vs. length scale h (i.e., $h = (\mathrm{nDOFs})^{-1/2}$) is displayed in Fig. 30, which shows an average order of accuracy of 1.86. The plot of L_2 entropy error vs. time-step is shown in Fig. 31. All the tests reached full convergence within 40 time steps. In Table 3, the raw data is provided in three columns, i.e. h, L_2 error, and work units.

Table 3. Raw data for inviscid flow past a circular cylinder.

Mesh	nDOFs	h	L_2 error	Work units
Lv.2	32×8	0.062500	0.03654315	
Lv.3	64×16	0.031250	0.00954726	
Lv.4	128×64	0.015625	0.00277384	

9.8.3 Input Files

A set of three input files using explicit time integration and a set of three input files using implicit time integration can be found at

- `tests/cnsfv/2d_cyln_expl_lv2.i`
- `tests/cnsfv/2d_cyln_expl_lv3.i`
- `tests/cnsfv/2d_cyln_expl_lv4.i`

and

- `tests/cnsfv/2d_cyln_impl_lv2.i`
- `tests/cnsfv/2d_cyln_impl_lv3.i`
- `tests/cnsfv/2d_cyln_impl_lv4.i`

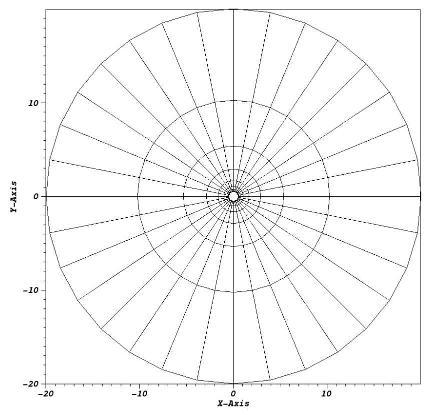
under the BIGHORN repository, respectively.

9.8.4 Mesh Files

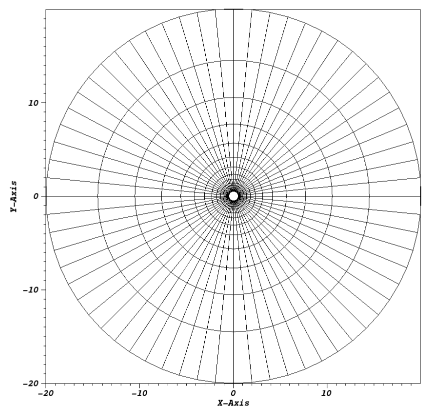
A set of three Exodus-II format mesh files corresponding to either set of the three input files can be found at

- `tests/cnsfv/2d_cyln_mesh_lv2.e`
- `tests/cnsfv/2d_cyln_mesh_lv3.e`
- `tests/cnsfv/2d_cyln_mesh_lv4.e`

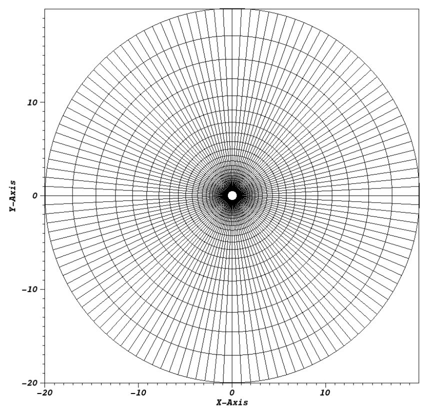
under the BIGHORN repository, respectively.



(a) Lv. 2

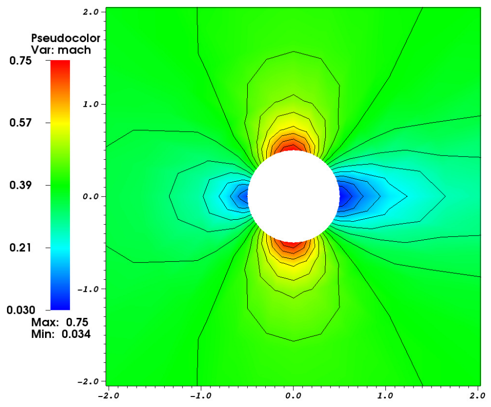


(b) Lv. 3

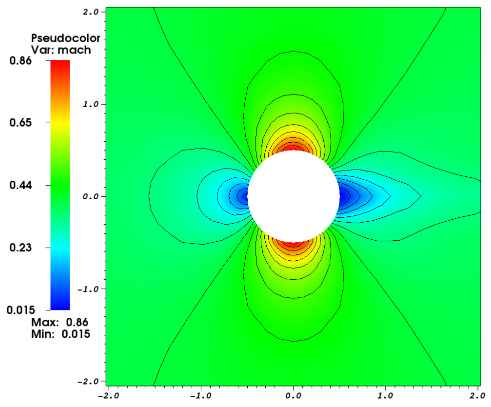


(c) Lv. 4

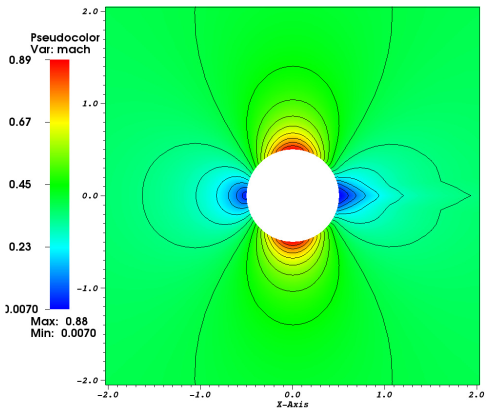
Figure 28. Computational meshes for inviscid flow past a cylinder.



(a) Lv. 2



(b) Lv. 3



(c) Lv. 4

Figure 29. Steady-state Mach number contours for inviscid flow past a cylinder.

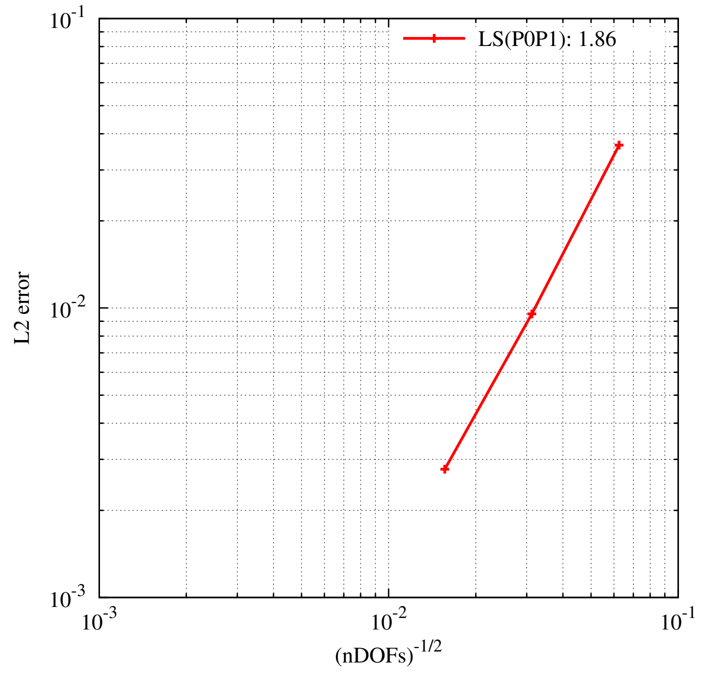


Figure 30. L_2 entropy error vs. length scale h for inviscid flow past a cylinder.

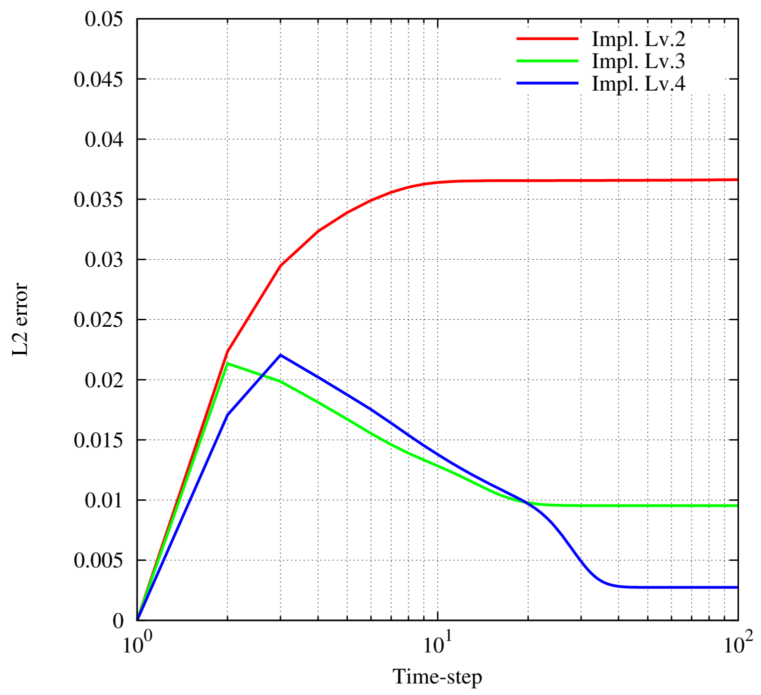


Figure 31. L_2 entropy error vs. time-step for inviscid flow past a cylinder.

9.9 A 2D Mach-3 Wind Tunnel with a Step

9.9.1 Problem Description

This 2-D test problem was originally introduced in [68], which has proven to be a useful test for a large number of methods over the decades. We set up this test case by following the description in [64]. The problem begins with uniform Mach 3 flow in a wind tunnel containing a step. The wind tunnel is 1 length unit high in the y-direction, and 3 length units long in the x-direction. The step is 0.2 length unit high, and is located 0.6 length unit from the left end of the tunnel. The governing equations are the 2D Euler equations with a constant ratio of specific heats of 1.4. The computational domain is shown in Fig. 32. The infinity conditions are density 1.0, velocity 3.0 with 0 angle of attack, and pressure $1/1.4$. The slip condition is imposed on the wall surface of the tunnel, and the Riemann invariant condition is imposed on the left-end inflow boundary and the right-end outflow boundary.

9.9.2 Problem Setup

The numerical methods used for running the simulations in this test case are summarized in the following table. The simulations were started at $t = 0.0$ and terminated at $t = 4.5$. The infinity

Item	Specifics
Governing equations	2D Euler
Fluid properties	Ideal gas
Spatial discretization	Discontinuous Galerkin
Solution polynomial	Piecewise constant (cell-average)
Reconstruction	Least-squares slope reconstruction
Stabilization scheme	Min-max slope limiter
Numerical flux scheme	HLLC
Execution	Transient
Temporal discretization	Explicit two-step TVD Runge-Kutta
Linear solver	None

conditions mentioned above were used as initial conditions everywhere in the domain. Note that the coner of the step is the center of a rarefaction fan and hence is a singular point of the flow field. Nothing special was done at this singular point in this test case. Consequently the flow field in this corner region is seriously affected by large numerical errors generated in the neighborhood of this singular point. These errors case a boundary layer of a bout one zone in thickness to form above the step in the wind tunnel. Shocks then interact with this boundary layer, and the qualitative nature of the flow field in the tunnel is altered more or less dramatically, depending upon the the spatial scheme and the mesh used. Some special boundary conditions were used in [64], in an attempt to particularly minimize numerical errors generated at the corner of the step for this problem. The time evelution, up to $t = 4$, of the density distributionin the wind tunnel is displayed in Figs. 33–39.

In addition, the transient pressure and Mach number distributions are displayed in Figs. 40 and 41, respectively. Notice that the flow field at $t = 4$ is still unsteady. As shown in [69], a steady-state flow field of this problem develops by $t = 12$, which has very little structure.

9.9.3 Input Files

The input file for this test case can be found at `tests/cnsfv/2d_mach3step.i` under the BIGHORN repository.

9.9.4 Mesh Files

The Exodus-II format mesh file can be found at `tests/cnsfv/2d_mach3step.e` under the BIGHORN repository.

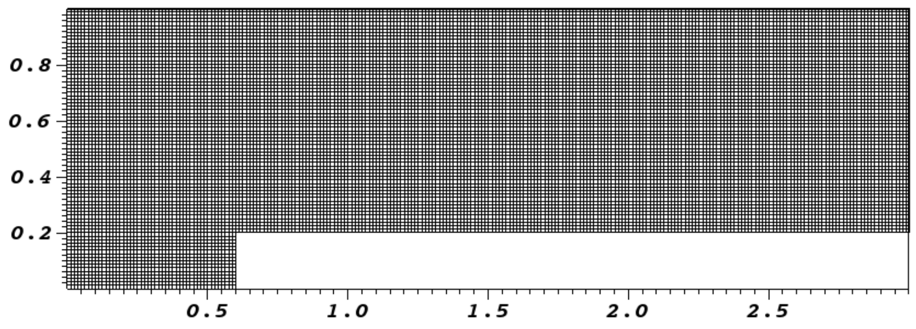


Figure 32. The unstructured quadrilateral mesh for a Mach-3 wind tunnel with a step.

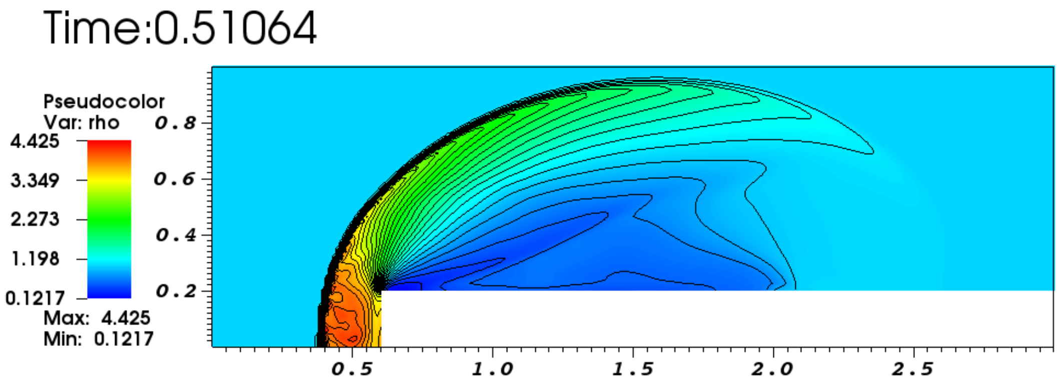


Figure 33. Density contours at about $t = 0.5$ for a Mach-3 wind tunnel with a step.

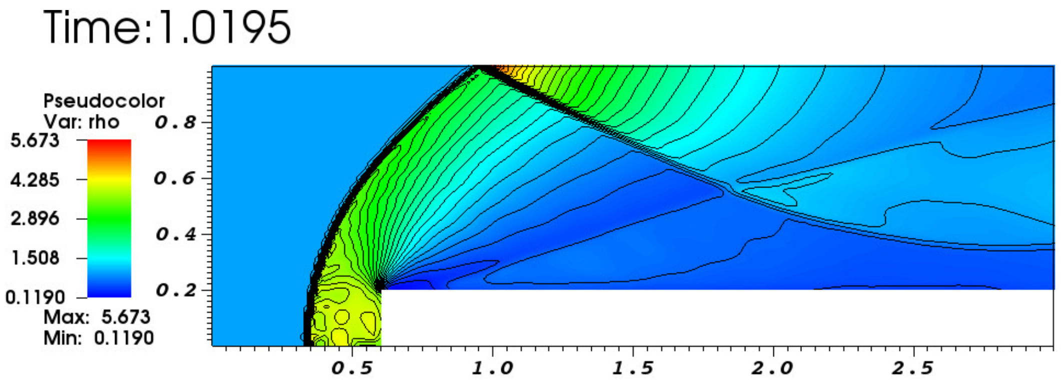


Figure 34. Density contours at about $t = 1.0$ for a Mach-3 wind tunnel with a step.

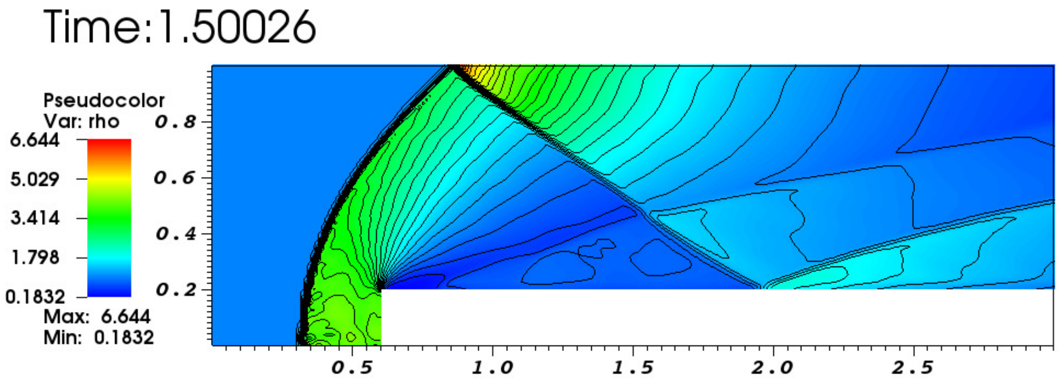


Figure 35. Density contours at about $t = 1.5$ for a Mach-3 wind tunnel with a step.

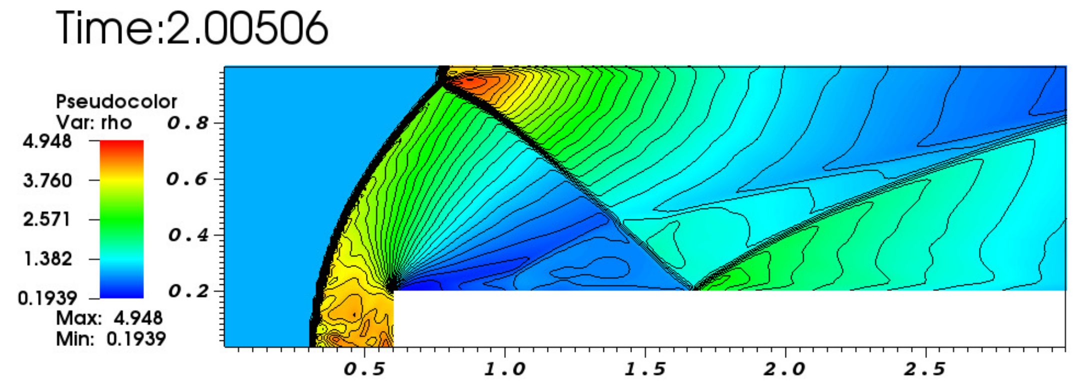


Figure 36. Density contours at about $t = 2.0$ for a Mach-3 wind tunnel with a step.

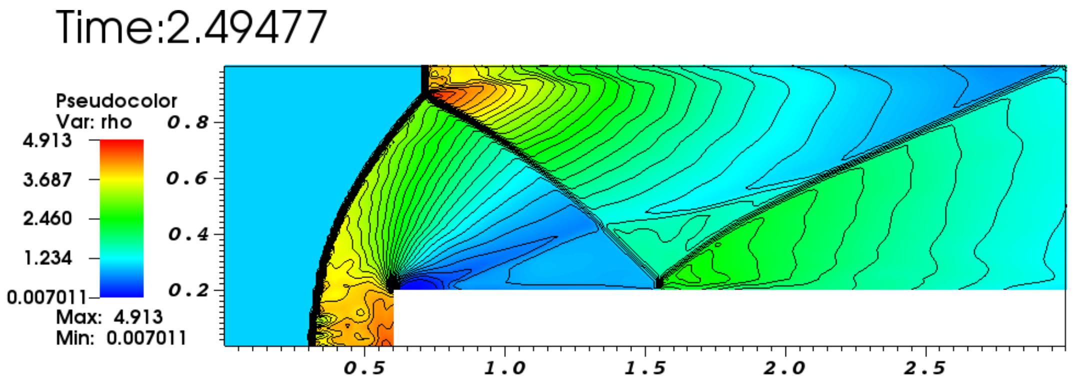


Figure 37. Density contours at about $t = 2.5$ for a Mach-3 wind tunnel with a step.

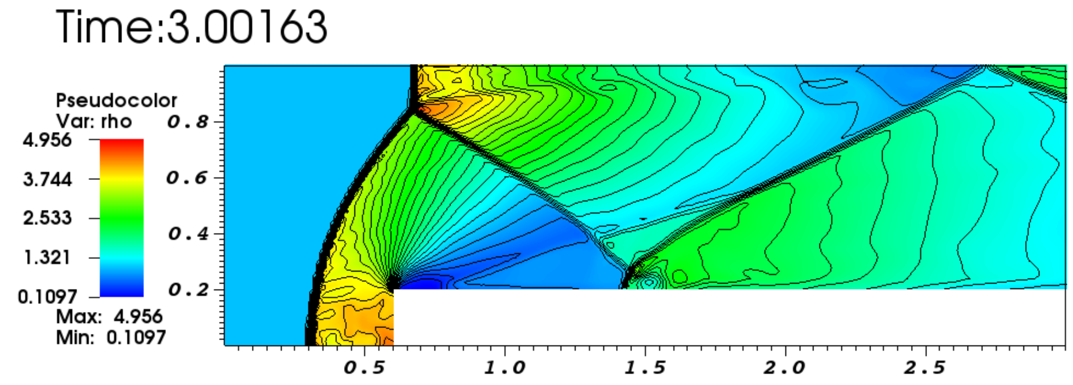


Figure 38. Density contours at about $t = 3.0$ for a Mach-3 wind tunnel with a step.

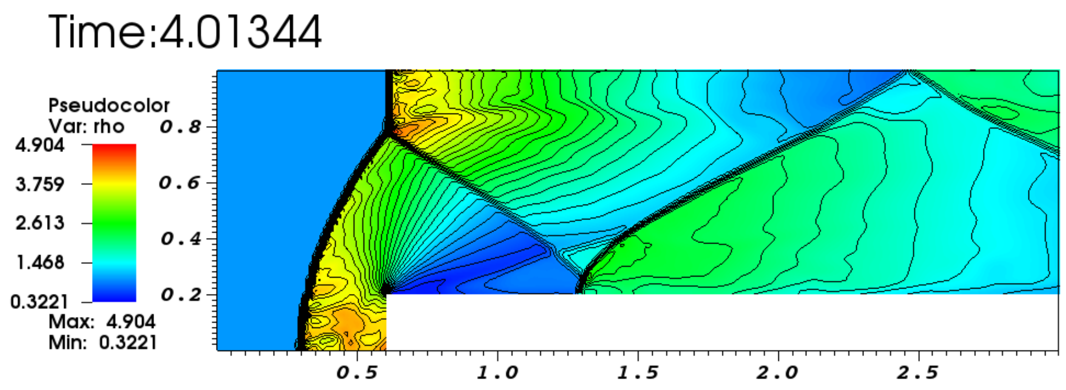


Figure 39. Density contours at about $t = 4.0$ for a Mach-3 wind tunnel with a step.

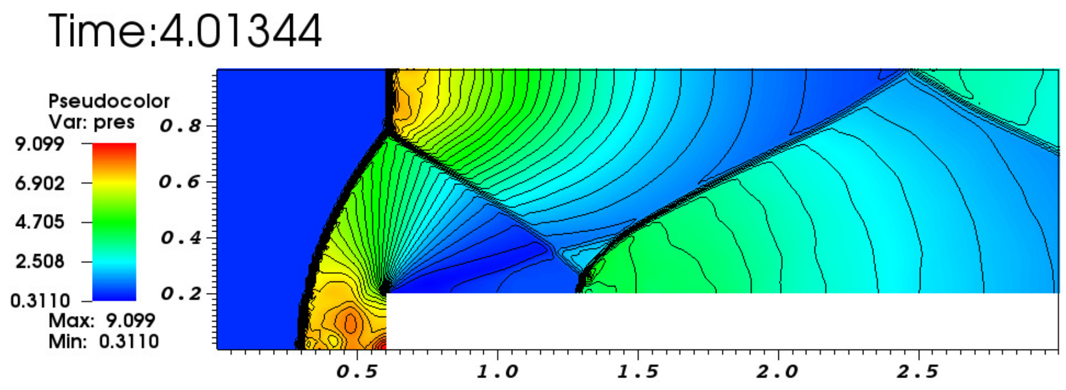


Figure 40. Pressure contours at about $t = 4.0$ for a Mach-3 wind tunnel with a step.

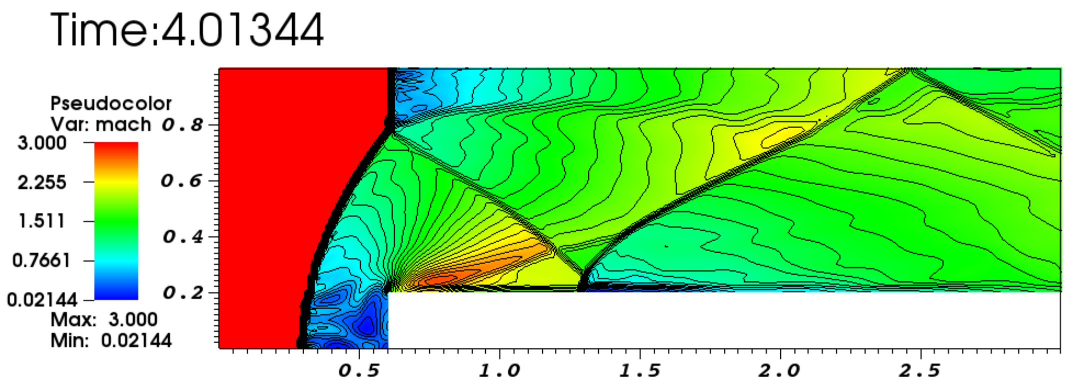


Figure 41. Mach number contours at about $t = 4.0$ for a Mach-3 wind tunnel with a step.

9.10 Inviscid Bow Shock Upstream of a Blunt Body in Supersonic Flow

9.10.1 Problem Description

Taken from the third baseline inviscid test case (BI3) of the 4th International Workshop on High-Order CFD Methods (https://how4.cenaero.be/), this case is designed to isolate testing of the shockcapturing properties of schemes using the detached bow shock upstream of a two-dimensional blunt body in supersonic flow. The governing equations are the 2D Euler equations with a constant ratio of specific heats of 1.4. The computational domain is shown in Fig. 42. The geometry is a flat center section, with two constant radius sections top and bottom. The flat section is 1 unit length, and each radius is $1/2$ unit length. While the flow is symmetric top and bottom, a full domain is computed to support potentially spurious behavior. The aft section of the body is not included to avoid developing an unsteady wake. The infinity conditions are density 1, velocity 4 with 0 angle of attack, and pressure $1/1.4$. Since the incoming freestream is at Mach 4, the inflow and outflow are both supersonic, so Dirichlet and Neumann boundary conditions respectively can be prescribed. The solid surface uses a standard impermeable wall specification ($\mathbf{V} \cdot \mathbf{n} = 0$). In our configuration, the slip condition is imposed on the wall surface of the blunt body (SideSet 1), and the Riemann invariant condition is imposed on the inflow boundary (SideSet 4) and the outflow boundary (SideSet 2 and 3).

9.10.2 Problem Setup

The numerical methods used for running the simulations in this test case are summarized in the following table. A series of three meshes, Lv. 0 (`nelem = 2220`, `npoin = 2325`), Lv. 1 (`nelem = 7776`, `npoin = 7975`), and Lv. 2 (`nelem = 29008`, `npoin = 29403`), are provided by the workshop, as shown in Fig. 43. The meshes are not hierarchical. At each refinement the clustering near the shock location and the surface in increased. These meshes all cluster around the asymptotic shock location (i.e. at coarse resolutions the computed shock will be in the incorrect location relative to the mesh, but should converge to the predicted location). The mesh is designed so that a

Item	Specifics
Governing equations	2D Euler
Fluid properties	Ideal gas
Spatial discretization	Discontinuous Galerkin
Solution polynomial	Piecewise constant (cell-average)
Reconstruction	Least-squares slope reconstruction
Stabilization scheme	Min-max slope limiter
Numerical flux scheme	HLLC
Execution	Steady
Temporal discretization	Explicit Euler
Linear solver	None

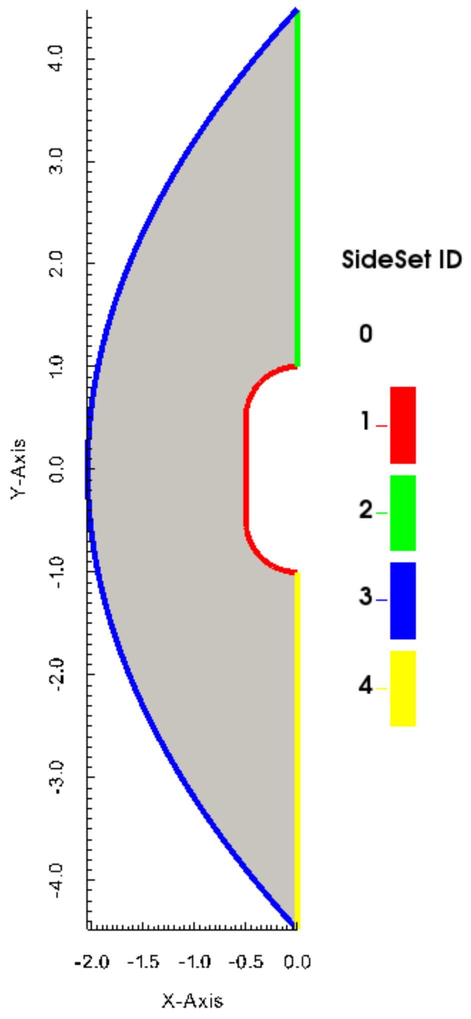


Figure 42. Computational domain for inviscid bow shock upstream of a blunt body in Mach-4 supersonic flow.

single element / cell / stencil straddles the asymptotic shock location. The stead-state density, pressure, and Mach numbers contours are plotted in Figs. 44, 45, 46, respectively. Over the contours obtained on the Lv. 0 mesh, a wide shock interface can be seen, due to the relative coarseness of the Lv. 0 mesh in the shock region. After the mesh is locally successively refined in the shock region, a very thin shock interface is resolved on the Lv. 2 mesh. Furthermore, the density, pressure, and Mach number values computed on the cell vertices are plotted for $x \in [-1.5, -1.2]$ along the symmetry line of $y = 0$, as shown in Figs. 47, 48, 49, respectively. Grid convergence of the oscillation-free solution profiles can be observed in each of those figures, where the shock interface become sharper and sharper as more refined meshes are used.

9.10.3 Input Files

A set of three input files using explicit time integration can be found at

- `tests/cnsfv/2d_bowshock_expl_lv0.i`
- `tests/cnsfv/2d_bowshock_expl_lv1.i`
- `tests/cnsfv/2d_bowshock_expl_lv2.i`

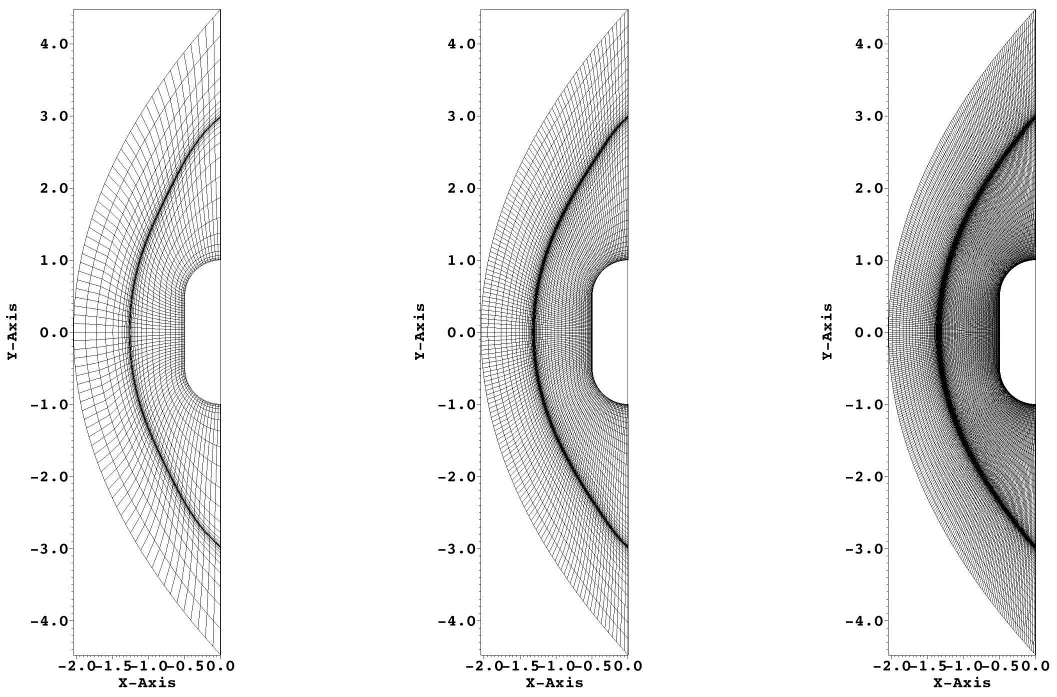
under the BIGHORN repository.

9.10.4 Mesh Files

A set of three **GMSH** format mesh files corresponding to either set of the three input files can be found at

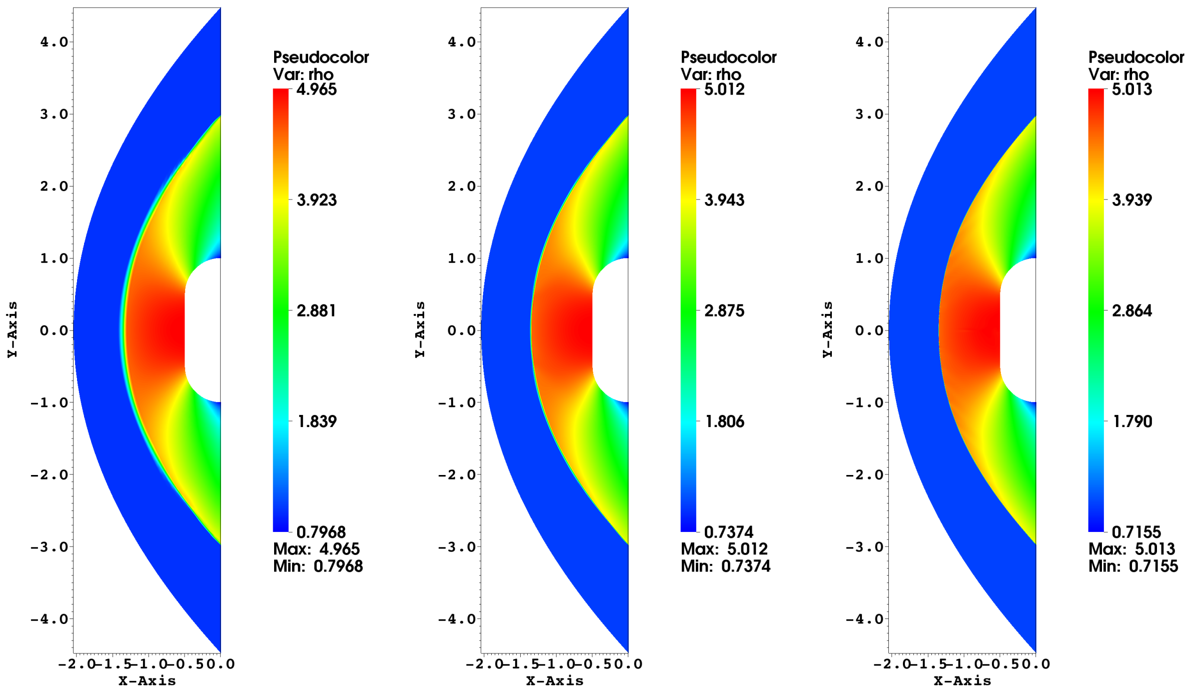
- `tests/cnsfv/2d_bowshock_mesh_lv0.msh`
- `tests/cnsfv/2d_bowshock_mesh_lv1.msh`
- `tests/cnsfv/2d_bowshock_mesh_lv2.msh`

under the BIGHORN repository.



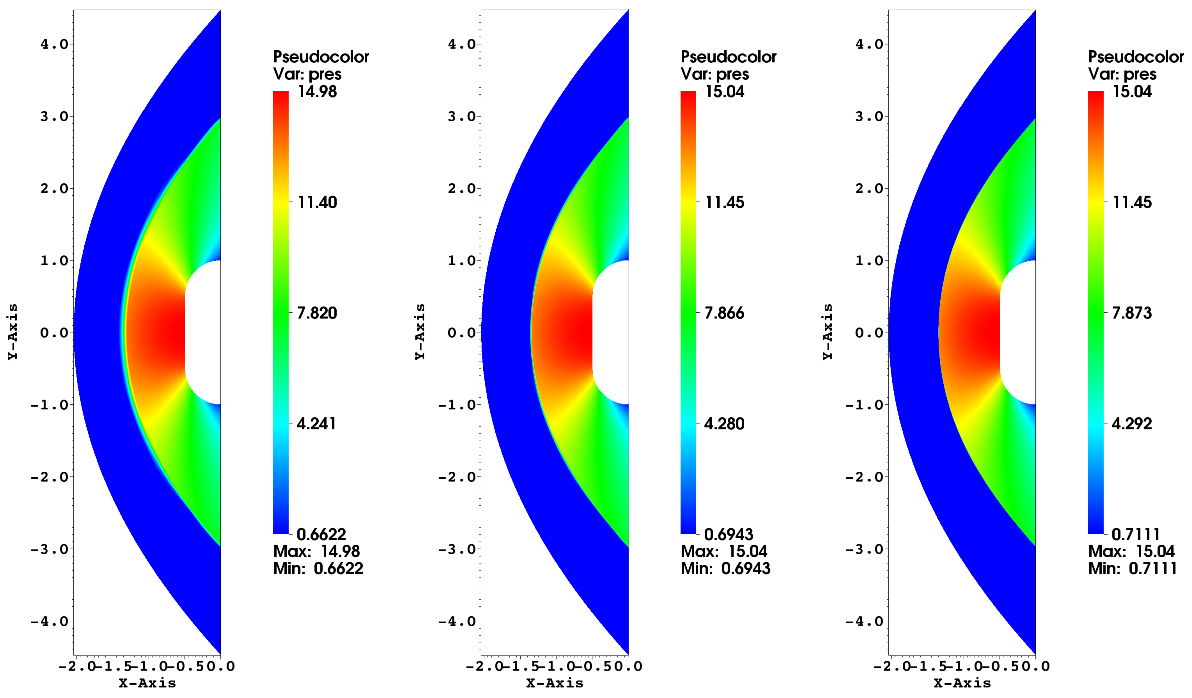
(a) Lv. 0 **(b)** Lv. 1 **(c)** Lv. 2

Figure 43. A set of three successively refined unstructured quadrilateral meshes for inviscid bow shock upstream of a blunt body in Mach-4 supersonic flow.



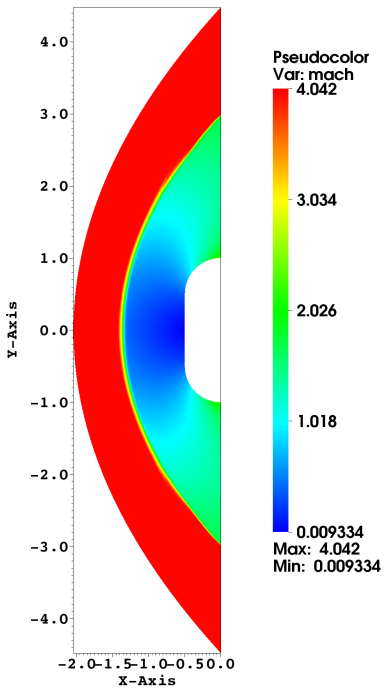
(a) Lv. 0 **(b)** Lv. 1 **(c)** Lv. 2

Figure 44. Steady-state density contours for inviscid bow shock upstream of a blunt body in Mach-4 supersonic flow.

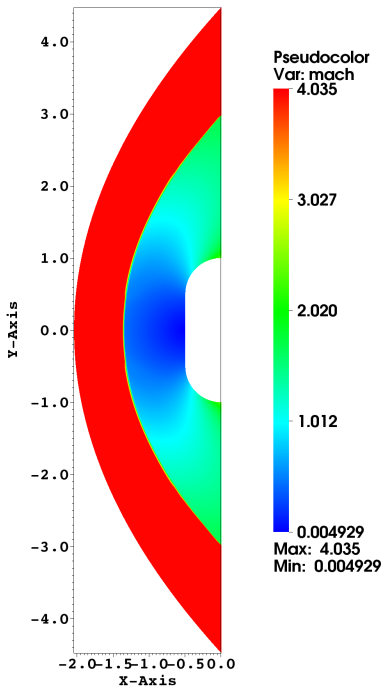


(a) Lv. 0 **(b)** Lv. 1 **(c)** Lv. 2

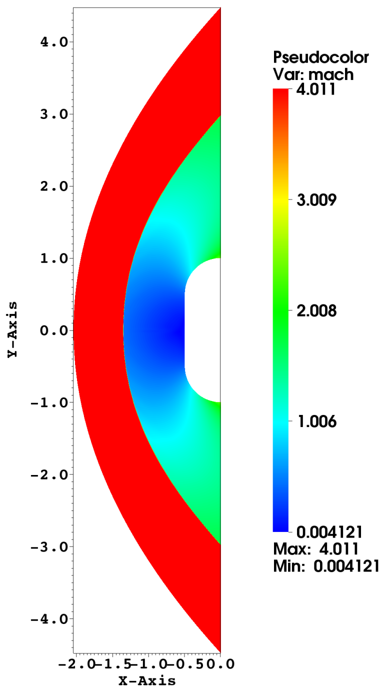
Figure 45. Steady-state pressure contours for inviscid bow shock upstream of a blunt body in Mach-4 supersonic flow.



(a) Lv. 0



(b) Lv. 1



(c) Lv. 2

Figure 46. Steady-state Mach number contours for inviscid bow shock upstream of a blunt body in Mach-4 supersonic flow.

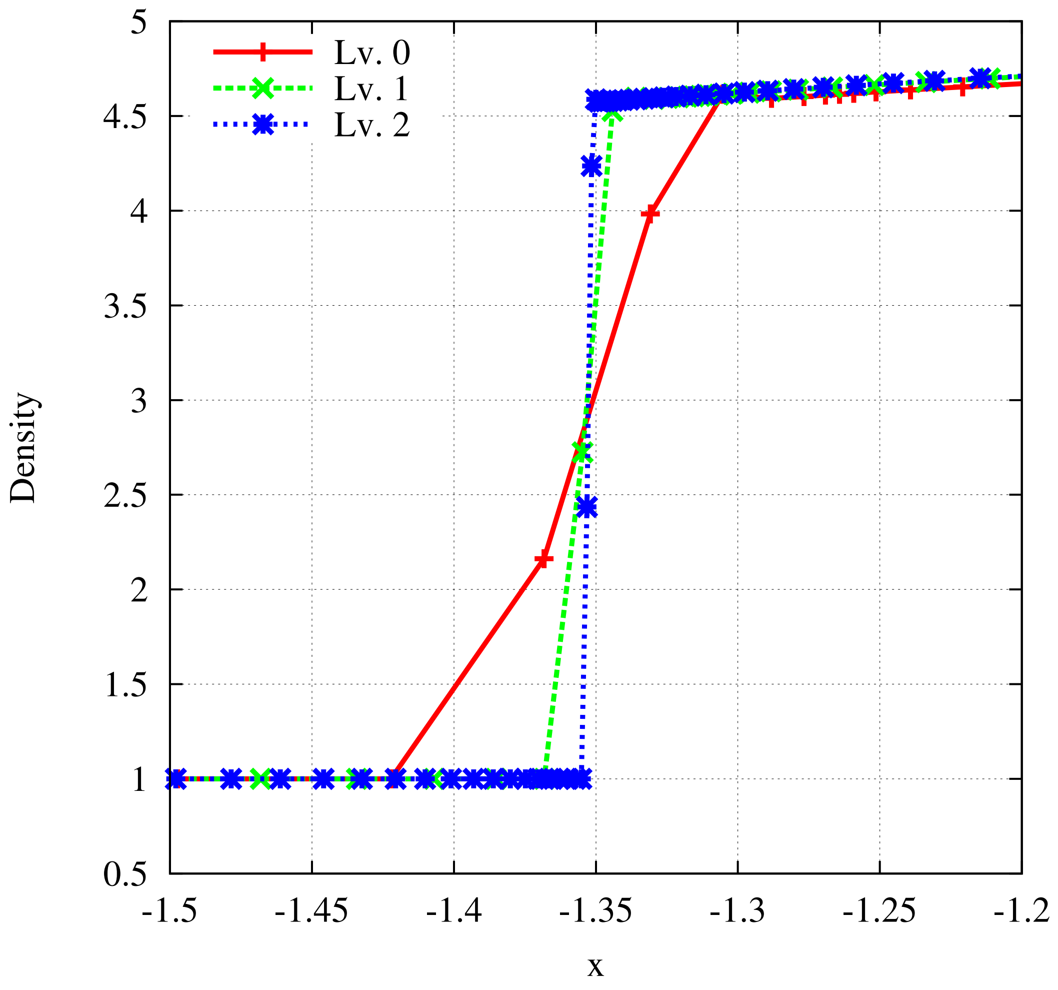


Figure 47. Steady-state density profile along the symmetry line of $y = 0$ for inviscid bow shock upstream of a blunt body in Mach-4 supersonic flow.

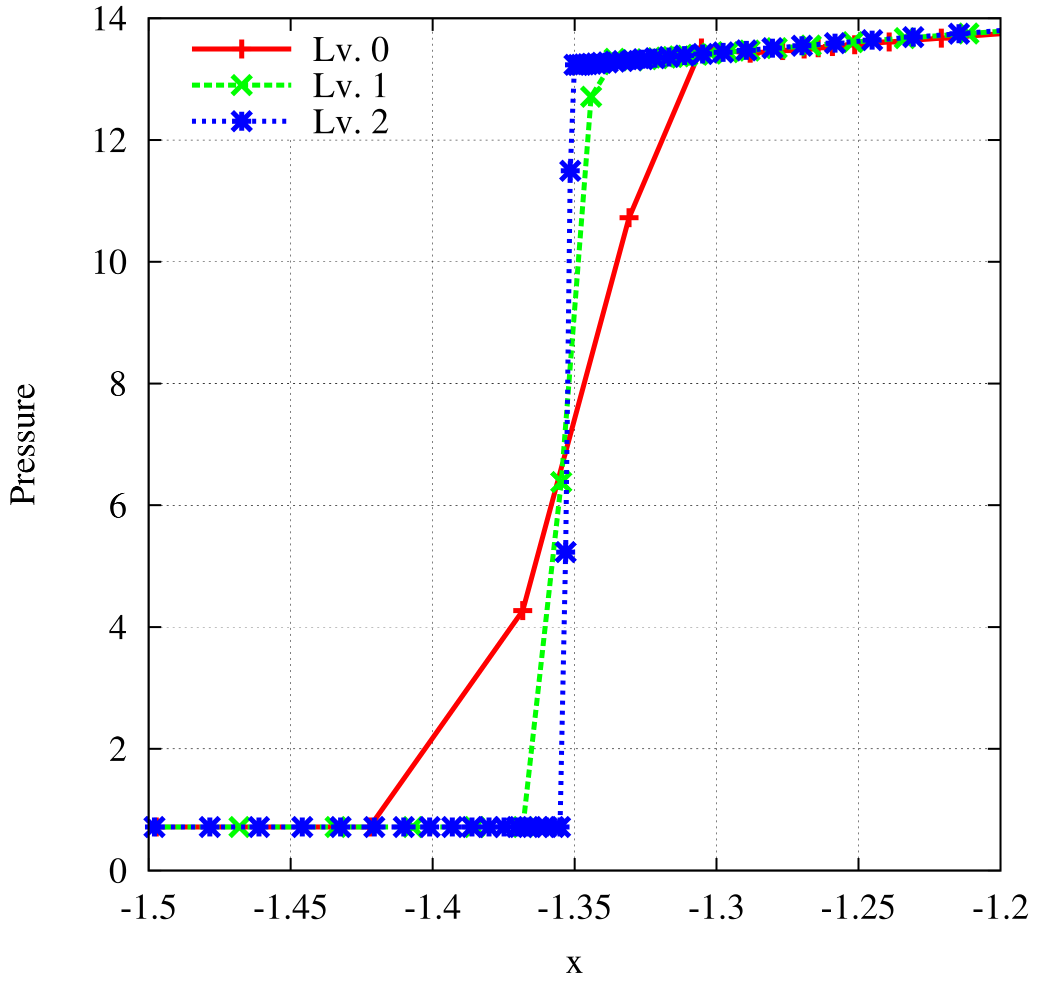


Figure 48. Steady-state pressure profile along the symmetry line of $y = 0$ for inviscid bow shock upstream of a blunt body in Mach-4 supersonic flow.

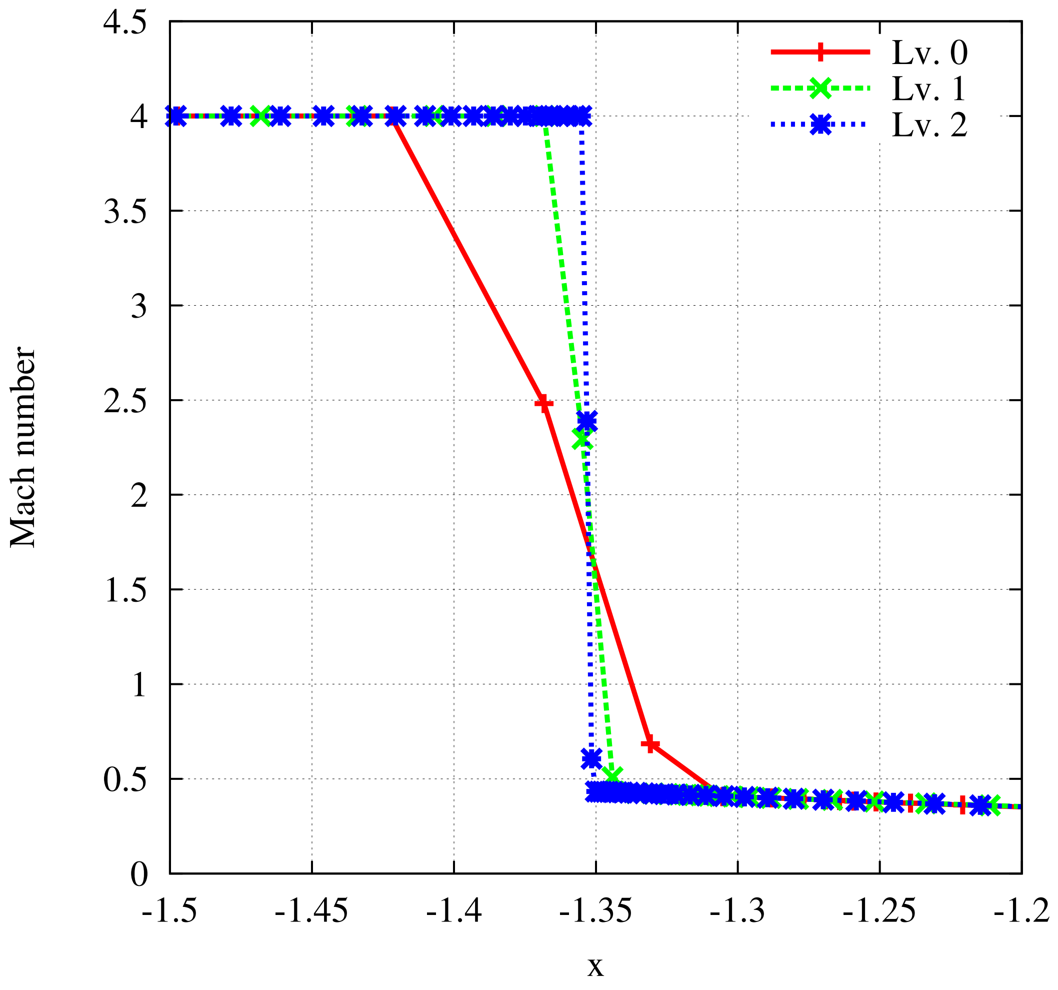


Figure 49. Steady-state Mach number profile along the symmetry line of $y = 0$ for inviscid bow shock upstream of a blunt body in Mach-4 supersonic flow.

9.11 Mach-3 Supersonic Flow Over a Wedge

9.11.1 Problem Description

This is a standard problem in compressible, invicid shock theory. The details about this problem can be found in Chapter 4 of [70]. For this problem, one can use the Rankine-Hugoniot relations to compute the exact fluid state just behind the oblique shock [70]. Only half of the configuration is modeled due to the symmetry nature of this problem. The governing equations are the 2D Euler equations with a constant ratio of specific heats of 1.4. Fig. 50 displays the computational domain for simulation. The wedge has a half-angle of 15°, and a free-stream Mach-3 flow (with $\rho_{\infty} = 1$ and $p_{\infty} = 1/1.4$) is imposed on the inflow boundary (SideSet 2). The slip condition is imposed on the symmetry boundary and also on the wall surface of the wedge (SideSet 1). The Riemann invariant condition is imposed on the outflow boundary (SideSet 3).

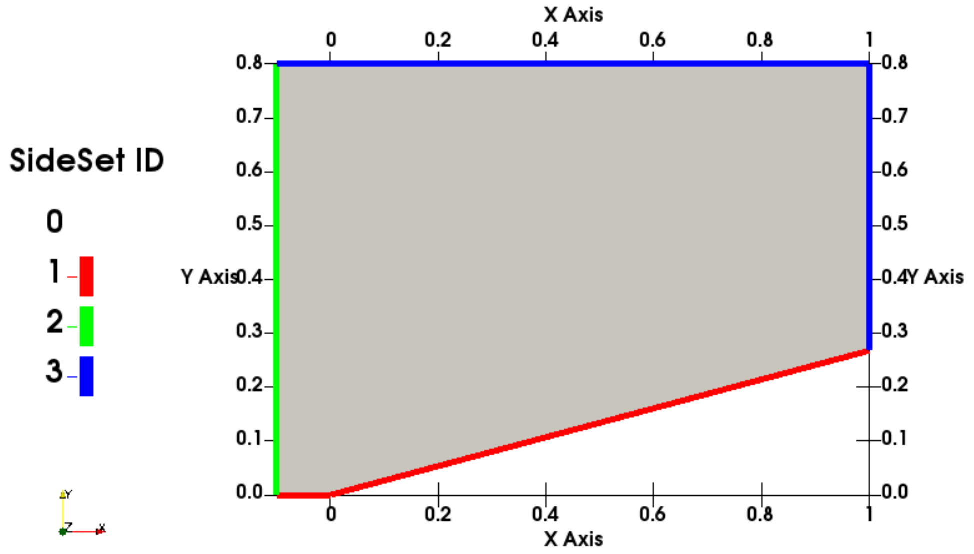


Figure 50. Computational domain for inviscid Mach-3 supersonic flow over a wedge (`SideSet` = 1 is solid wall, `SideSet` = 2 is inflow boundary, and `SideSet` = 3 is outflow boundary).

9.11.2 Problem Setup

The numerical methods used for running the simulations in this test case are summarized in the following table. An unstructured triangular mesh consisting of 11024 elements and 5656 points was used for simulations in this test case, as shown in Fig. 51. The mesh is isotropic and relatively coarse for this problem. Three spatial discretization methods, the first-order DG (P0), the second-order rDG (P0P1) with least-squares (L-S) slope reconstruction and min-max slope limiting, and

Item	Specifics
Governing equations	2D Euler
Fluid properties	Ideal gas
Spatial discretization	Discontinuous Galerkin
Solution polynomial	Piecewise constant (cell-average)
Reconstruction	Least-squares slope reconstruction
Stabilization scheme	Min-max / WENO slope limiter
Numerical flux scheme	HLLC
Execution	Steady
Temporal discretization	Explicit Euler
Linear solver	None

the second-order rDG (P0P1) with L-S slope reconstruction and WENO slope limiting, were used in simulations, respectively. The following equation can be used to calculate the entropy on each element or grid point:

$$S = \frac{p}{p_\infty}(\frac{\rho_\infty}{\rho})^\gamma - 1$$

The stead-state density, pressure, Mach number, and entropy contours are plotted in Figs. 52, 53, 54, and 55, respectively. Overall, it can be seen that the shock region simulated by the two rDG (P0P1) methods is thinner than that by the DG (P0). However, notice that the rDG (P0P1) method with the min-max slope limiting had difficulty to fully converge the flow field in the post-shock region, resulting in non-smoothness in its numerical solution. Such non-smoothness cannot be seen very clearly in density, pressure and Mach number contours, but is pretty remarkable in the entropy contour in Fig. 55(b). In comparison, the rDG (P0P1) method with WENO slope limiting resulted in fully converged smooth solution in the post-shock region, as shown in the entropy contour in Fig. 55(c). Furthermore, the computed density, pressure, Mach number, and entropy values were plotted for $x \in [0,1]$ along the horizontal line of $y = 3.5$, as shown in Figs. 56, 57, 58, and 59, respectively. Notice that the second-order methods present sharper shock capturing than their first-order counterpart. Again, the non-smoothness of the minmax slope limited rDG (P1P2) solution in the post-shock region can be observed in Fig. 59, where the magnitude of oscillation is significant. Finally, the density residuals versus time steps for the two rDG (P0P1) methods are presented in Fig. 60, where the residual obtained by the WENO limited rDG (P0P1) reached the level of 10^{-12}, but the residual obtained by the minmax limited rDG (P0P1) stalled at the level of 10^{-4}.

9.11.3 Input Files

The input file for this test case can be found at `tests/cnsfv/2d_obliqueshock_expl_weno.i` under the BIGHORN repository.

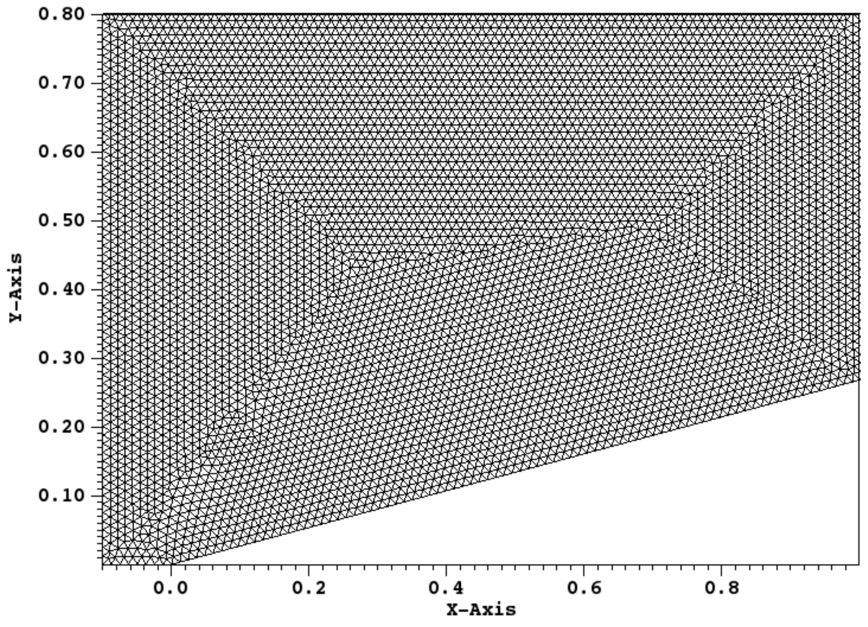
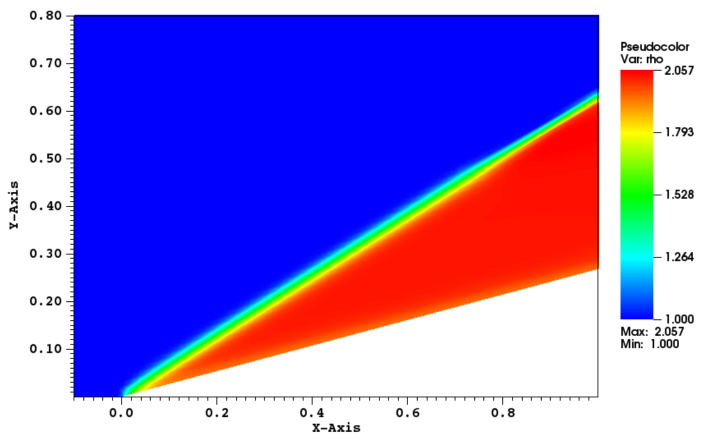


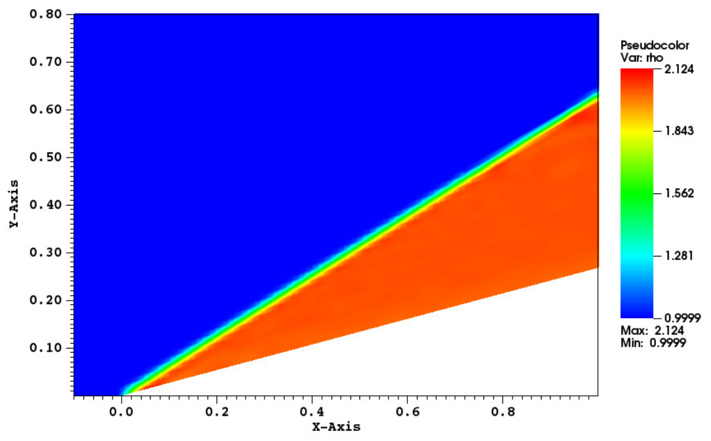
Figure 51. An unstructured triangular mesh (`nelem` = 11024, `npoin` = 5656) for inviscid Mach-3 supersonic flow over a wedge.

9.11.4 Mesh Files

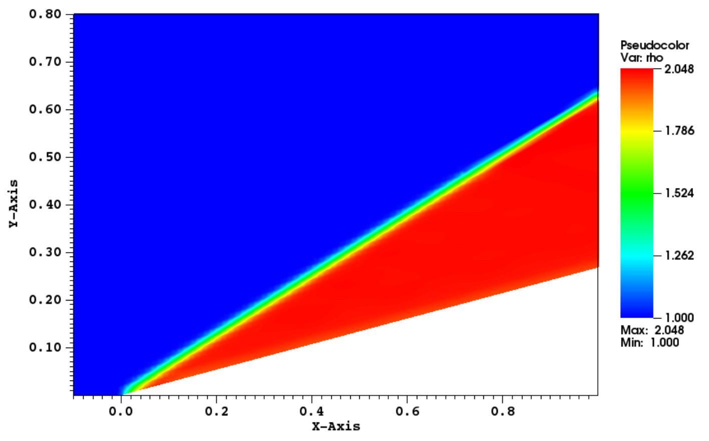
The Exodus-II format mesh file can be found at `tests/cnsfv/2d_obliqueshock_mesh.e` under the BIGHORN repository.



(a)

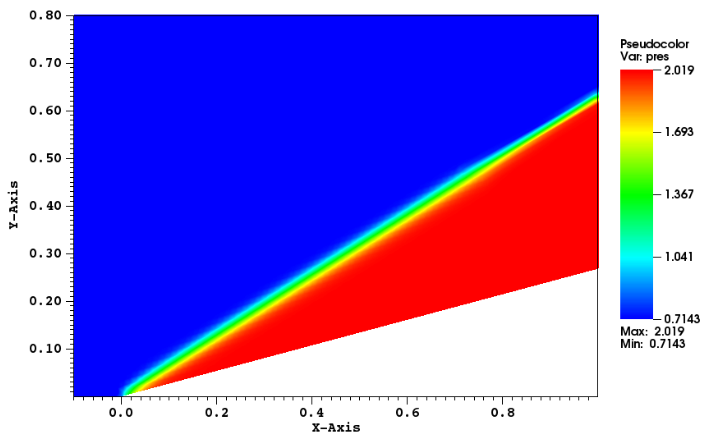


(b)

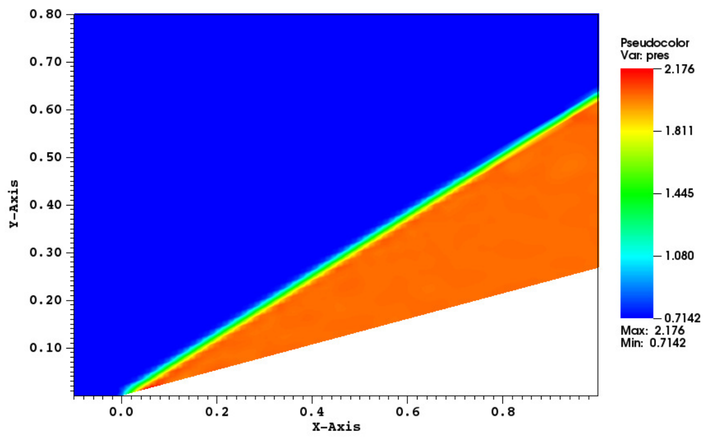


(c)

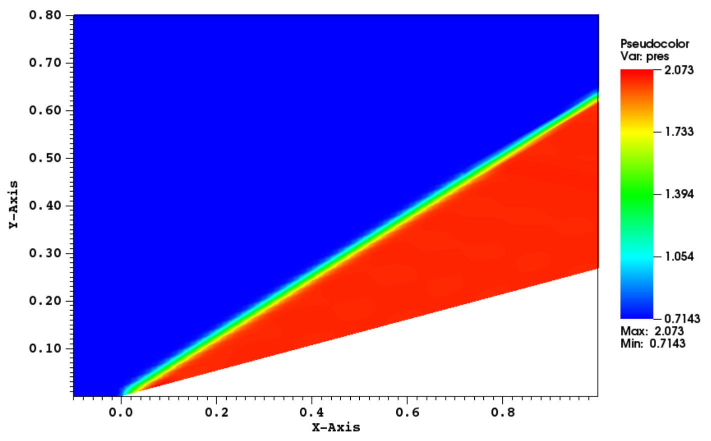
Figure 52. Steady-state density contours for inviscid Mach-3 supersonic flow over a wedge: (a) DG(P0), (b) rDG(P0P1) with L-S slope reconstruction and minmax slope limiting, (c) rDG(P0P1) with L-S slope reconstruction and WENO slope limiting.



(a)

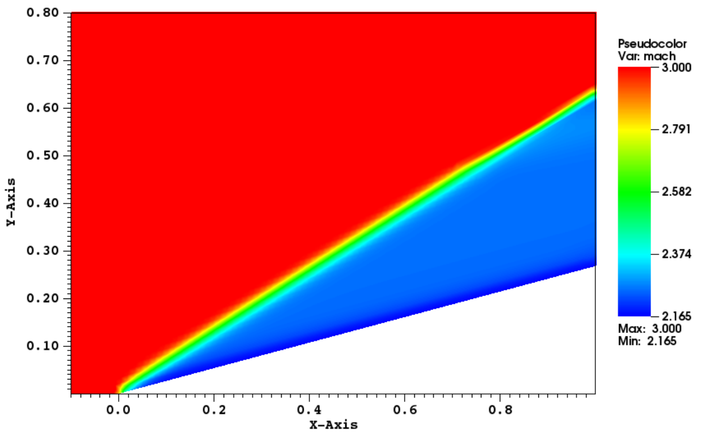


(b)

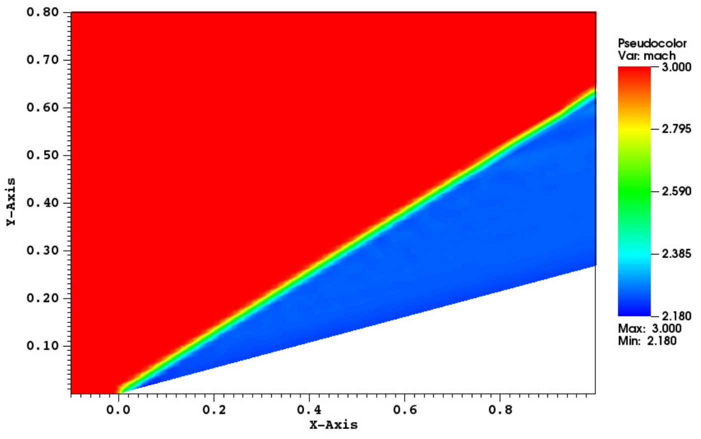


(c)

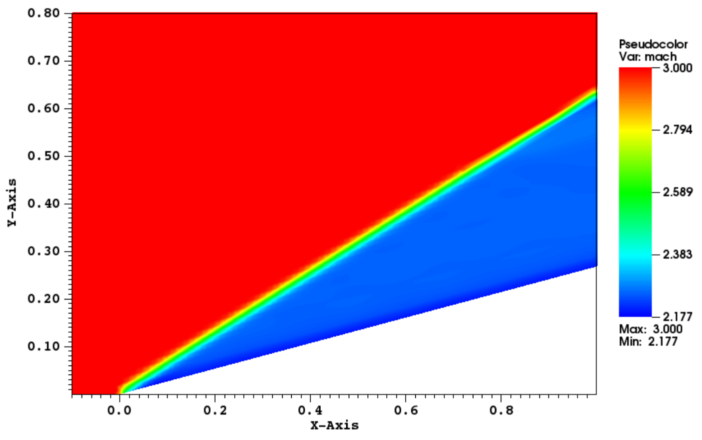
Figure 53. Steady-state pressure contours for inviscid Mach-3 supersonic flow over a wedge: (a) DG(P0), (b) rDG(P0P1) with L-S slope reconstruction and minmax slope limiting, (c) rDG(P0P1) with L-S slope reconstruction and WENO slope limiting.



(a)

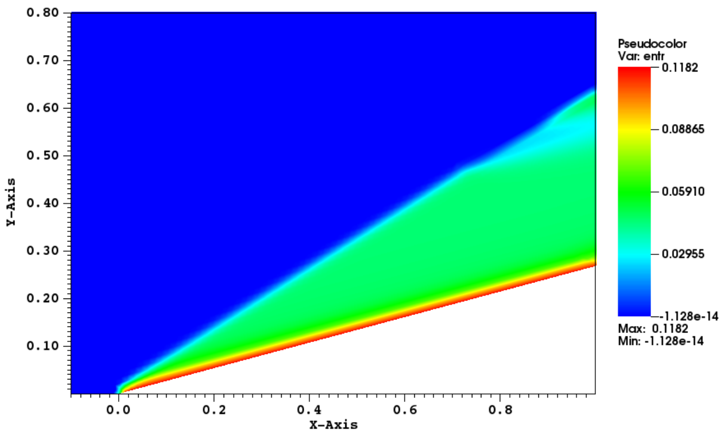


(b)

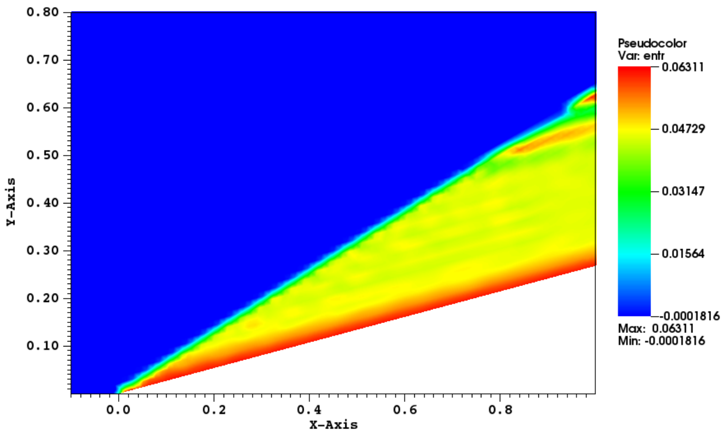


(c)

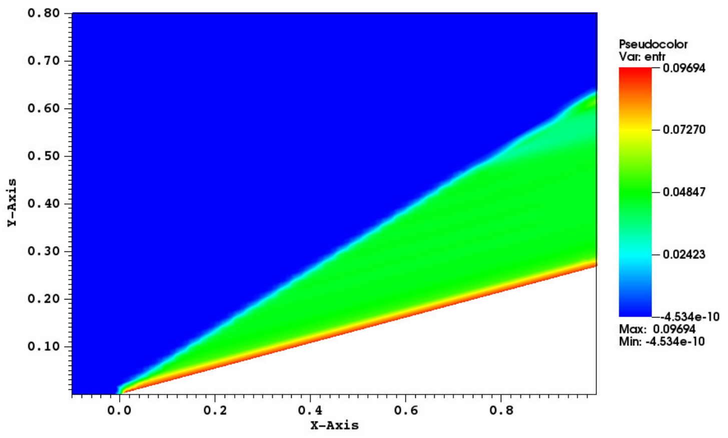
Figure 54. Steady-state Mach number contours for inviscid Mach-3 supersonic flow over a wedge: (a) DG(P0), (b) rDG(P0P1) with L-S slope reconstruction and minmax slope limiting, (c) rDG(P0P1) with L-S slope reconstruction and WENO slope limiting.



(a)



(b)



(c)

Figure 55. Steady-state entropy contours for inviscid Mach-3 supersonic flow over a wedge: (a) DG(P0), (b) rDG(P0P1) with L-S slope reconstruction and minmax slope limiting, (c) rDG(P0P1) with L-S slope reconstruction and WENO slope limiting.

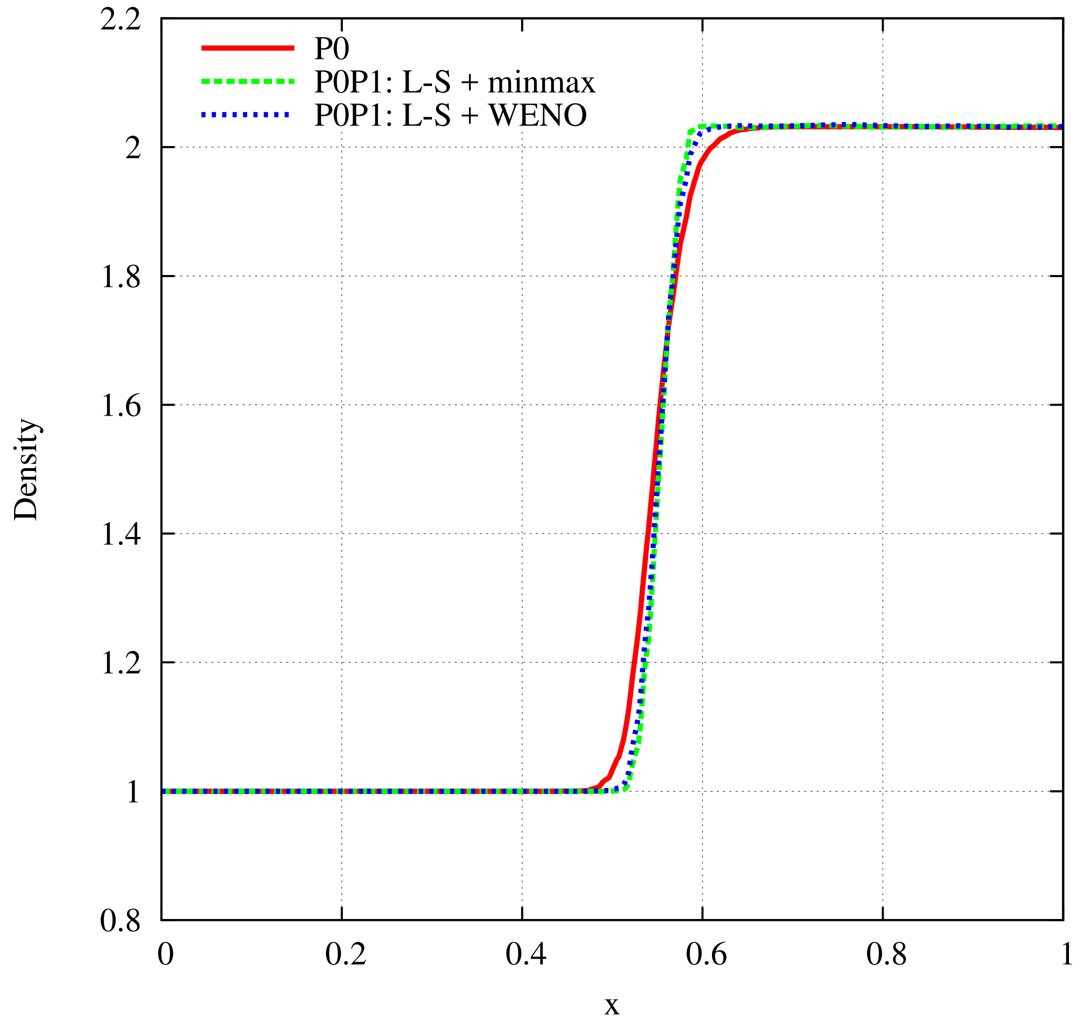


Figure 56. Plot of computed density versus x-coordinate along the line $y = 3.5$.

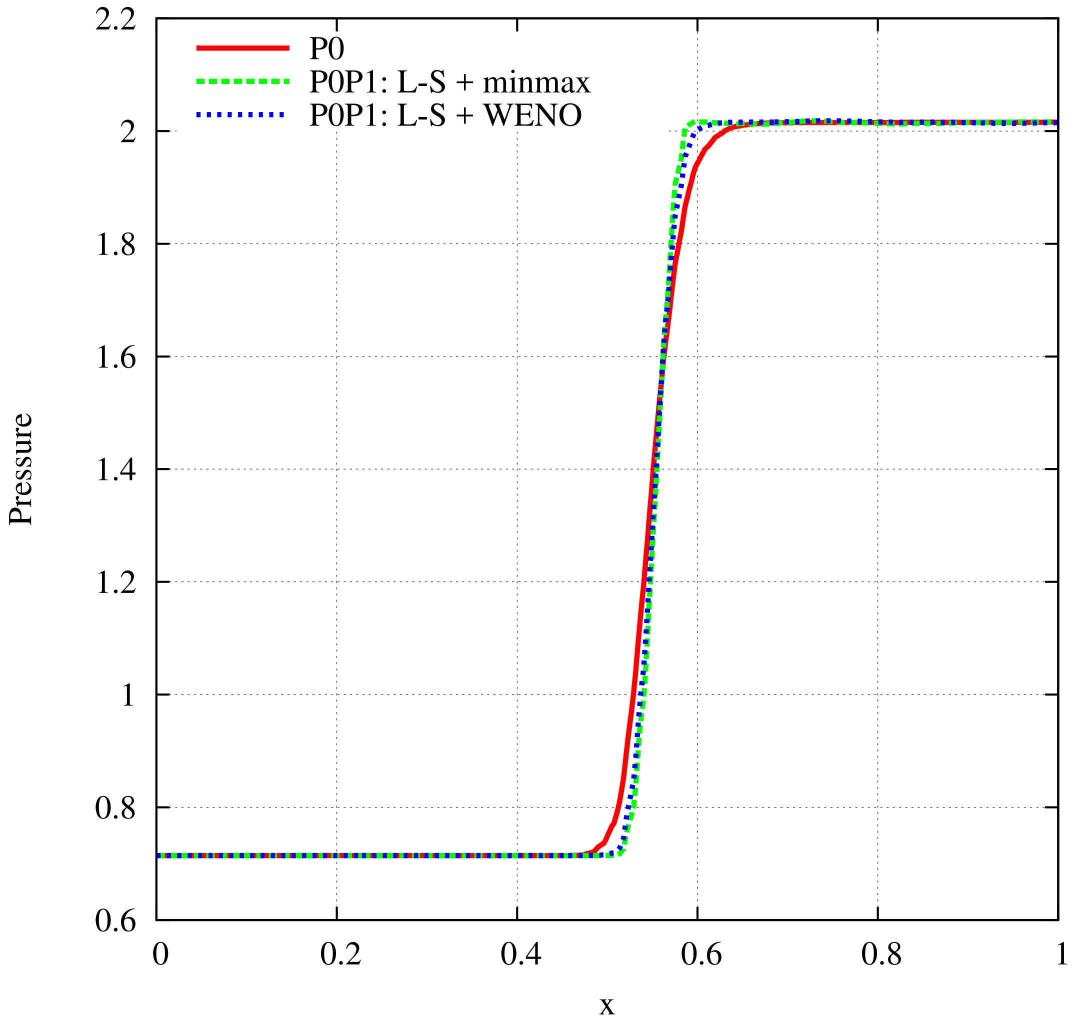


Figure 57. Plot of computed pressure versus x-coordinate along the line $y = 3.5$.

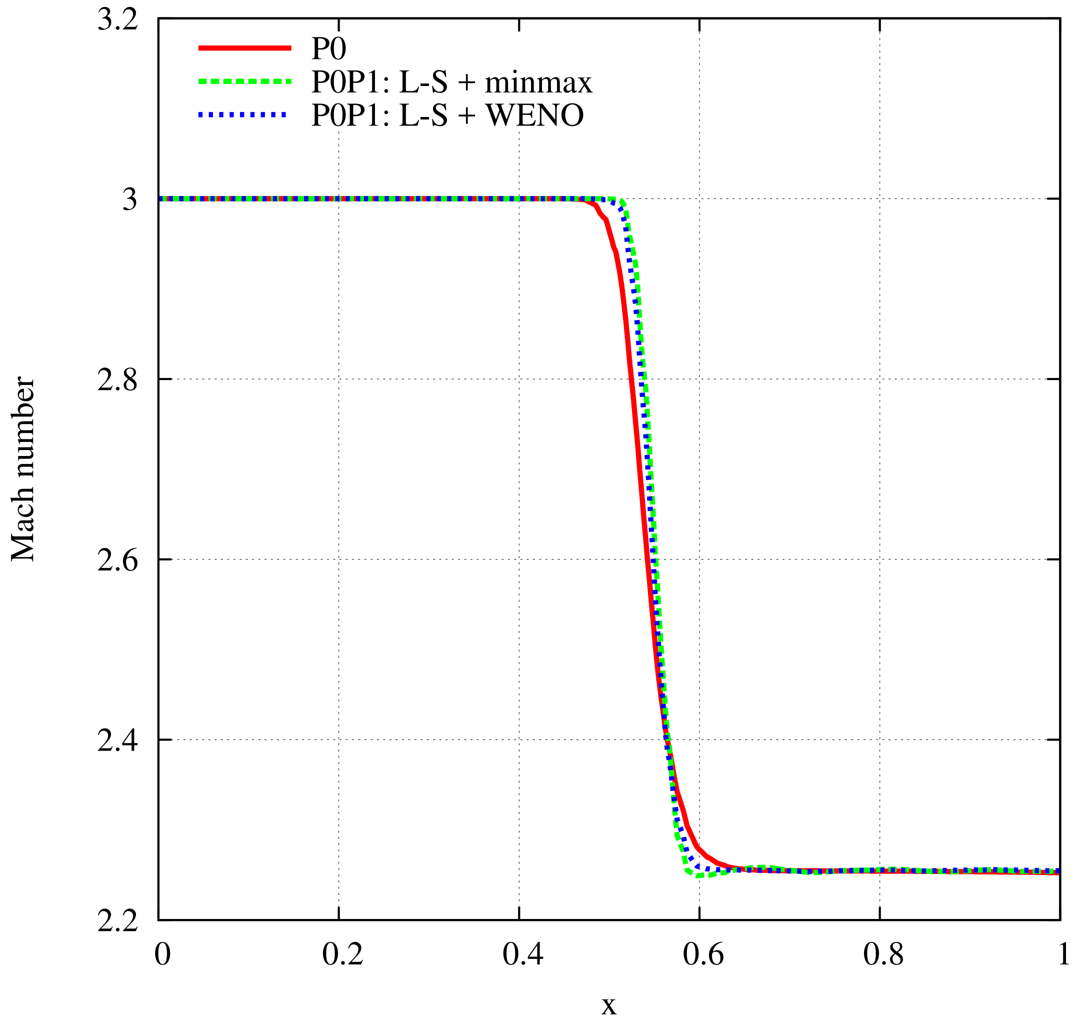


Figure 58. Plot of computed Mach number versus x-coordinate along the line $y = 3.5$.

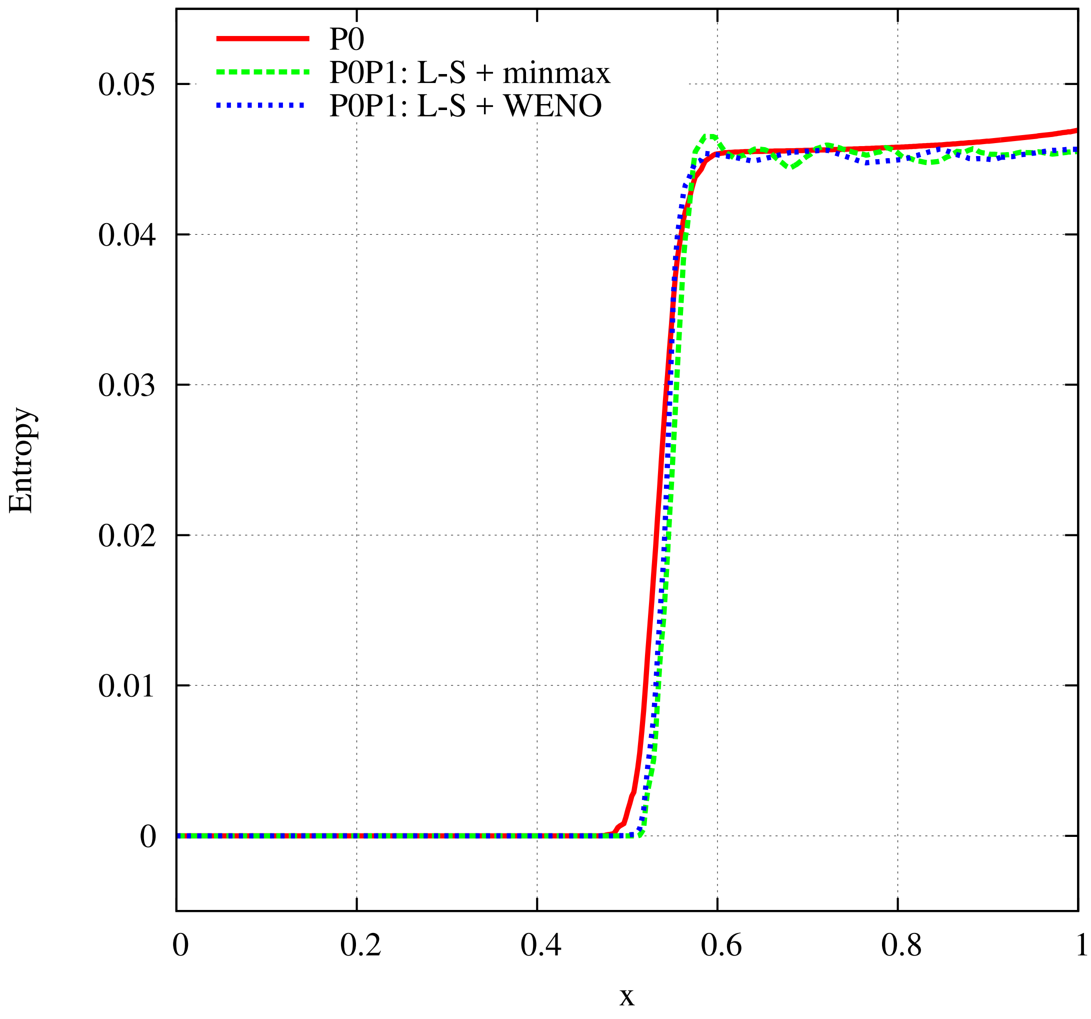


Figure 59. Plot of computed entropy versus x-coordinate along the line $y = 3.5$.

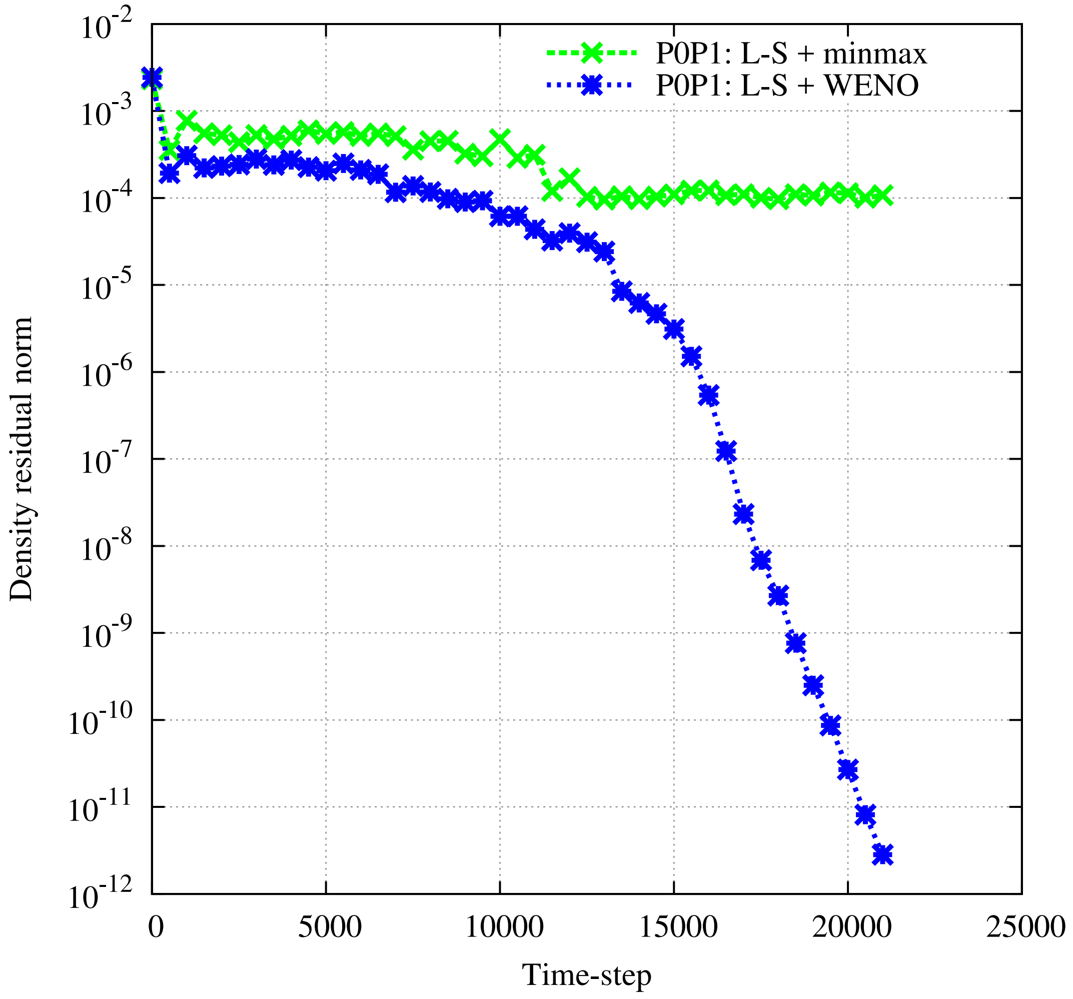


Figure 60. Comparison of computed density residual norm versus time-step between rDG (P0P1) with L-S + minmax and rDG (P0P1) with L-S + WENO.

References

[1] PG Buningt. A 3-D chimera grid embedding technique. In *7th Computational Physics Conference*, AIAA Paper No. 1985-1523, Cincinnati, Ohio, United States, July 1985. American Institute of Aeronautics and Astronautics.

[2] JA Benek, TL Donegan, and NE Suhs. Extended Chimera grid embedding scheme with application to viscous flows. In *8th Computational Fluid Dynamics Conference*, AIAA Paper No. 1987-1126, Honolulu, Hawaii, United States, June 1987. American Institute of Aeronautics and Astronautics.

[3] Rainald Löhner and Paresh Parikh. Generation of three-dimensional unstructured grids by the advancing-front method. *International Journal for Numerical Methods in Fluids*, 8(10):1135–1149, 1988.

[4] Shahyar Pirzadeh. Unstructured viscous grid generation by the advancing-layers method. *AIAA journal*, 32(8):1735–1737, 1994.

[5] Dmitri Sharov, Hong Luo, Joseph D Baum, and Rainald Löhner. Unstructured navier–stokes grid generation at corners and ridges. *International Journal for Numerical Methods in Fluids*, 43(6-7):717–728, 2003.

[6] Donald Keith Clarke, HA Hassan, and MD Salas. Euler calculations for multielement airfoils using cartesian grids. *AIAA Journal*, 24(3):353–358, 1986.

[7] William J. Coirier and Kenneth G. Powell. An Accuracy Assessment of Cartesian-Mesh Approaches for the Euler Equations. *Journal of Computational Physics*, 117(1):121 – 131, 1995.

[8] Marsha J Berger and Randall J Leveque. An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. In *9th Computational Fluid Dynamics Conference*, AIAA Paper No. 1989-1930, Buffalo, New York, United States, June 1989. American Institute of Aeronautics and Astronautics.

[9] MJ Aftosmis, JE Melton, and MJ Berger. Adaptation and surface modeling for cartesian mesh methods. In *12th Computational Fluid Dynamics Conference*, AIAA Paper No. 1995-1725, San Diego, California, United States, June 1995. American Institute of Aeronautics and Astronautics.

[10] Hong Luo, Joseph D Baum, and Rainald Löhner. A hybrid cartesian grid and gridless method for compressible flows. *Journal of Computational Physics*, 214(2):618–632, 2006.

[11] NP Weatherill. Mixed structured-unstructured meshes for aerodynamic flow simulation. *Aeronautical Journal*, 84(934):111–123, 1990.

[12] DJ Mavriplis and V Venkatakrishnan. A unified multigrid solver for the Navier-Stokes equations on mixed element meshes. *International Journal of Computational Fluid Dynamics*, 8(4):247–263, 1997.

[13] Kazuhiro Nakahashi, Dmitri Sharov, Shintaro Kano, and Masatoshi Kodera. Applications of unstructured hybrid grid method to high-Reynolds number viscous flows. *International Journal for Numerical Methods in Fluids*, 31(1):97–111, 1999.

[14] William J Coirier and Philip CE Jorgenson. A mixed volume grid approach for the Euler and Navier-Stokes equations. In *34th Aerospace Sciences Meeting*, AIAA Paper No. 1996-762, Reno, Nevada, United States, June 1996. American Institute of Aeronautics and Astronautics.

[15] Andreas Haselbacher and Jiri Blazek. Accurate and efficient discretization of Navier-Stokes equations on mixed grids. *AIAA Journal*, 38(11):2094–2102, 2000.

[16] Andreas Haselbacher, James J McGuirk, and Gary J Page. Finite volume discretization aspects for viscous flows on mixed unstructured grids. *AIAA journal*, 37(2):177–184, 1999.

[17] L Sbardella and M Imregun. An efficient discretization of viscous fluxes on unstructured mixed-element grids. *Communications in numerical methods in engineering*, 16(12):839–849, 2000.

[18] ZJ Wang. A Quadtree-based adaptive Cartesian/Quad grid flow solver for Navier-Stokes equations. *Computers & Fluids*, 27(4):529–549, 1998.

[19] N.T. Frink and S.Z. Pirzadeh. Tetrahedral finite-volume solutions to the Navier-Stokes equations on complex configurations. *International Journal for Numerical Methods in Fluids*, 31(1):175–187, 1999.

[20] F Haider, J-P Croisille, and B Courbet. Stability analysis of the cell centered finite-volume MUSCL method on unstructured grids. *Numerische Mathematik*, 113(4):555–600, 2009.

[21] 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.

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

[23] Changqing Hu and Chi-Wang Shu. Weighted essentially non-oscillatory schemes on triangular meshes. *Journal of Computational Physics*, 150(1):97–127, 1999.

[24] Michael Dumbser and Martin Käser. Arbitrary high order non-oscillatory finite volume schemes on unstructured meshes for linear hyperbolic systems. *Journal of Computational Physics*, 221(2):693–723, 2007.

[25] 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.

[26] Oliver Friedrich. Weighted essentially non-oscillatory schemes for the interpolation of mean values on unstructured grids. *Journal of computational physics*, 144(1):194–212, 1998.

[27] WR Wolf and JLF Azevedo. High-order ENO and WENO schemes for unstructured grids. *International Journal for Numerical Methods in Fluids*, 55(10):917–943, 2007.

[28] Wanai Li and Yu-Xin Ren. High-order k-exact WENO finite volume schemes for solving gas dynamic Euler equations on unstructured grids. *International Journal for Numerical Methods in Fluids*, 70(6):742–763, 2012.

[29] Timothy J Barth and Dennis C Jespersen. The design and application of upwind schemes on unstructured meshes. In *27th Aerospace Sciences Meeting*, AIAA Paper No. 89-0366, Reno, Nevada, United States, January 1989. American Institute of Aeronautics and Astronautics.

[30] Timothy J Barth. Recent developments in high order k-exact reconstruction on unstructured meshes. In *31st Aerospace Sciences Meeting*, AIAA Paper No. 1993-668, Reno, Nevada, United States, January 1993. American Institute of Aeronautics and Astronautics.

[31] Randall J LeVeque. *Finite-Volume Methods for Hyperbolic Problems*, volume 31. Cambridge University Press, 2002.

[32] Philip L Roe. Some contributions to the modelling of discontinuous flows. In *Large-Scale Computations in Fluid Mechanics*, pages 163–193, 1985.

[33] Bram Van Leer. Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection. *Journal of computational physics*, 23(3):276–299, 1977.

[34] 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.

[35] B. Cockburn, G. Karniadakis, and Shu C. W. The Development of Discontinuous Galerkin Method. In *Discontinuous Galerkin Methods, Theory, Computation, and Applications. Edited by B. Cockburn, G. E. Karniadakis, and C. W. Shu. Lecture Notes in Computational Science and Engineering*, volume 11, pages 5–50. Springer, 2000.

[36] B. Cockburn and C. W. Shu. The Runge-Kutta Discontinuous Galerkin Method for conservation laws V: Multidimensional System. *Journal of Computational Physics*, 141:199–224, 1998.

[37] Roger Alexander. Design and implementation of DIRK integrators for stiff systems. *Applied Numerical Mathematics*, 46(1):1–17, 2003.

[38] Morten Rode Kristensen, John Bagterp Jørgensen, Per Grove Thomsen, and Sten Bay Jørgensen. An ESDIRK method with sensitivity analysis capabilities. *Computers & Chemical Engineering*, 28(12):2695–2707, 2004.

[39] Hester Bijl, Mark H Carpenter, Veer N Vatsa, and Christopher A Kennedy. Implicit time integration schemes for the unsteady compressible Navier–Stokes equations: laminar flow. *Journal of Computational Physics*, 179(1):313–329, 2002.

[40] Christiaan M Klaij, Jaap JW van der Vegt, and Harmen van der Ven. Pseudo-time stepping methods for space–time discontinuous Galerkin discretizations of the compressible Navier–Stokes equations. *Journal of Computational Physics*, 219(2):622–643, 2006.

[41] Li Wang and Dimitri J Mavriplis. Implicit solution of the unsteady Euler equations for high-order accurate discontinuous Galerkin discretizations. *Journal of Computational Physics*, 225(2):1994–2015, 2007.

[42] F Bassi, L Botti, A Colombo, A Ghidoni, and F Massa. Linearly implicit Rosenbrock-type Runge–Kutta schemes applied to the Discontinuous Galerkin solution of compressible and incompressible unsteady flows. *Computers & Fluids*, 118:305–320, 2015.

[43] Xiaodong Liu, Yidong Xia, Hong Luo, and Lijun Xuan. A Class of Rosenbrock Methods for a Reconstructed Discontinuous Galerkin Method for the Unsteady Compressible Flows. In *22nd AIAA Computational Fluid Dynamics Conference*, AIAA Paper No. 2015-2448, Dallas, Texas, United States, June 2015. American Institute of Aeronautics and Astronautics.

[44] Richard C Martineau and Ray A Berry. The pressure-corrected ICE finite element method for compressible flows on unstructured meshes. *Journal of Computational Physics*, 198(2):659–685, 2004.

[45] Ray A Berry. Notes on the pcice method: Simplification, generalization, and compressibility properties. *Journal of Computational Physics*, 215(1):6–11, 2006.

[46] Philip L Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372, 1981.

[47] Meng-Sing Liou and Christopher J Steffen. A new flux splitting scheme. *Journal of Computational Physics*, 107(1):23–39, 1993.

[48] Meng-Sing Liou. A sequel to ausm: AUSM$^+$. *Journal of Computational Physics*, 129(2):364–382, 1996.

[49] Meng-Sing Liou. A sequel to AUSM, Part II: AUSM$^+$-up for all speeds. *Journal of Computational Physics*, 214(1):137–170, 2006.

[50] M-S Liou, C-H Chang, Loc Nguyen, and Theo G Theofanous. How to solve compressible multifluid equations: a simple, robust, and accurate method. *AIAA Journal*, 46(9):2345–2356, 2008.

[51] P Batten, MA Leschziner, and UC Goldberg. Average-state Jacobians and implicit methods for compressible viscous and turbulent flows. *Journal of computational physics*, 137(1):38–78, 1997.

[52] Hong Luo, Joseph D Baum, and Rainald Lohner. Extension of Harten-Lax-van Leer Scheme for Flows at All Speeds. *AIAA journal*, 43(6):1160–1166, 2005.

[53] Jack R Edwards. A low-diffusion flux-splitting scheme for Navier-Stokes calculations. *Computers & Fluids*, 26(6):635–659, 1997.

[54] Jack R Edwards and Meng-Sing Liou. Low-diffusion flux-splitting methods for flows at all speeds. *AIAA journal*, 36(9):1610–1617, 1998.

[55] P Batten, N Clarke, C Lambert, and DM Causon. On the choice of wavespeeds for the HLLC Riemann solver. *SIAM Journal on Scientific Computing*, 18(6):1553–1570, 1997.

[56] Bernd Einfeldt, Claus-Dieter Munz, Philip L Roe, and Björn Sjögreen. On Godunov-type methods near low densities. *Journal of computational physics*, 92(2):273–295, 1991.

[57] Sherrie L Krist, Robert T Biedron, and Christopher L Rumsey. CFL3D User's Manual (Version 5.0). Technical Report NASA/TM-1998-208444, NASA Langley Research Center, Hampton, Virginia, United States, 1998.

[58] Jan-Reneé Carlson. Inflow/Outflow Boundary Conditions with Application to FUN3D. Technical Report NASA/TM-2011-217181, NASA Langley Research Center, Hampton, Virginia, United States, 2011.

[59] Gianmarco Mengaldo, Daniele De Grazia, J Peiro, Antony Farrington, F Witherden, PE Vincent, and SJ Sherwin. A Guide to the Implementation of Boundary Conditions in Compact High-Order Methods for Compressible Aerodynamics. In *7th AIAA Theoretical Fluid Mechanics Conference*, AIAA Paper No. 2014-2923, Atlanta, Georgia, United States, June 2014. American Institute of Aeronautics and Astronautics.

[60] V. Daru and C. Tenaud. High order one-step monotonicity-preserving schemes for unsteady compressible flow calculations. *Journal of Computational Physics*, 193(2):563–594, 2004.

[61] QL Zeng, NU Aydemir, FS Lien, and T Xu. Comparison of implicit and explicit ausm-family schemes for compressible multiphase flows. *International Journal for Numerical Methods in Fluids*, 77(1):43–61, 2015.

[62] Jun Zhu, Xinghui Zhong, Chi-Wang Shu, and Jianxian Qiu. Runge-kutta discontinuous galerkin method with a simple and compact hermite weno limiter. *Communications in Computational Physics*, 19(04):944–969, 2016.

[63] ZJ Wang, Krzysztof Fidkowski, Rémi Abgrall, Francesco Bassi, Doru Caraeni, Andrew Cary, Herman Deconinck, Ralf Hartmann, Koen Hillewaert, HT Huynh, et al. High-order CFD methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013.

[64] 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.

[65] Leonid Ivanovich Sedov. *Similarity and Dimensional Methods in Mechanics*. CRC Press, 1993.

[66] Viktor Pavlovich Korobeinikov. *Problems of Point Blast Theory*. Springer Science & Business Media, 1991.

[67] Timur Linde and Philip L Roe. Robust Euler codes. In *13th AIAA Computational Fluid Dynamics Conference*, AIAA Paper No. 1997-2098, Snowmass Village, Colorado, United States, June 1997. American Institute of Aeronautics and Astronautics.

[68] Ashley F Emery. An evaluation of several differencing methods for inviscid fluid flow problems. *Journal of Computational Physics*, 2(3):306–331, 1968.

[69] 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.

[70] John David Anderson. *Modern Compressible Flow: With Historical Perspective*, volume 12. McGraw-Hill New York, 1990.

A Flowchart

A.1 Main Code Structure

Algorithm 2 The main code structure

```
 1: procedure MESH CONNECTIVITY
 2:     getElemSurrPoint()              ▷ Find elements surrounding each grid point
 3:     getElemSurrElem()               ▷ Find elements surrounding each element
 4:     getFaceElemConn()               ▷ Build face-element connectivity
 5: end procedure
 6: procedure MESH DATA
 7:     getFaceData()                   ▷ Calculate face centroid and normals
 8:     getElemDate()                   ▷ Calculate element centroid and volume
 9:     getInverseElemMassMat()         ▷ Calculate the inverse of element mass matrix
10:     getElemMassMat()                ▷ Calculate the element mass matrix
11: end procedure
12: setICs()                            ▷ Set up initial conditions
13: procedure TIME INTEGRATION
14:     for iTime = 1 to nTime do
15:         predictDT()                 ▷ Calculate the allowable timestep size
16:         explicitTVDRK()             ▷ Perform explicit TVD-RK time stepping
17:         currentTime = currentTime + DT
18:         if execType = Steady then
19:             calcL2Norm()            ▷ Calculate the L2 norm for steady-state flow
20:         end if
21:         calcResidual()              ▷ Calculate the residual vector
22:         mpiBcastResidual()          ▷ MPI communications
23:         if currentTime $\geq$ maxTime then
24:             Exit                    ▷ Exit time loop if duration is reached
25:         end if
26:     end for
27: end procedure
```

Idaho National Laboratory