INL/EXT-20-58883

Finite Volume Discretization of the Euler Equations in Pronghorn

Sebastian Schunert, Robert Carlsen, Nolan MacDonald, Joshua Hansel, and Alexander Lindsay

The INL is a U.S. Department of Energy National Laboratory operated by Battelle Energy Alliance

June 2020

INL/EXT-20-58883

Finite Volume Discretization of the Euler Equations in Pronghorn

Sebastian Schunert, Robert Carlsen, Nolan MacDonald, Joshua Hansel, and Alexander Lindsay

June 2020

Idaho National Laboratory
Computational Frameworks Department
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

SUMMARY

Modeling flow and heat transfer in high temperature gas reactors (HTGR) requires the ability to model a wide range of flow speeds from slow (natural convection), to intermediate (forced-flow conditions), to supersonic regimes (depressurization) for a wide range of geometries including the pebble bed, upper and lower plenum, and risers. In previous work, Pronghorn has effectively modeled low-to-medium speed flows in scenarios such as the one described in the two-dimensional PBMR-400 benchmark, using its finite-element-based streamline-upwind Petrov-Galerkin (SUPG) stabilized implementation of the Euler equations. However, limitations of this method become apparent when dealing with more complicated geometries (e.g. imposing slip boundary conditions at nodes belonging to two different boundaries) and when gas speeds are fast enough for shocks and supersonic flow to occur. For these problems, the finite-element-based solver lacks robustness and is plagued by slow iterative convergence or even divergence. In order to address these challenges, the Pronghorn code at INL has been updated with new, modified versions of its original equations. The new Pronghorn models are built on the finite volume method with a Harten-Lax-van Leer-Contact (HLLC) Riemann solver based numerical flux method, which (1) allows imposing slip boundary conditions much more robustly and (2) performs well for a wide range of flow speeds. The finite-volume-based flow solver will form the basis for a robust coarse-mesh thermal-hydraulics capability in Pronghorn.

CONTENTS

1 Introduction

Modeling flow and heat transfer in high temperature gas reactors (HTGR) requires the ability to model a wide range of flow speeds from slow (natural convection), to intermediate (forced-flow conditions), to supersonic regimes (depressurization) for a wide range of geometries including the pebble bed, upper and lower plenum, and risers. Pronghorn is a multidimensional, coarse-mesh thermal-hydraulics code developed at Idaho National Laboratory (INL) targeting HTGRs, among other reactor concepts [1]. In previous work, Pronghorn has effectively modeled low-to-medium speed flows in scenarios such as the one described in the two-dimensional PBMR-400 benchmark [2, 3], using its finite-element based SUPG stabilized implementation of the Euler equations. However, limitations of this method become apparent when dealing with more complicated geometries (e.g. imposing slip boundary conditions at nodes belonging to two different boundaries) and when gas speeds are fast enough for shocks and supersonic flow to occur. For these problems, the finite-element-based solver lacks robustness and is plagued by slow iterative convergence or even divergence. In order to address these challenges, the Pronghorn code at INL has been updated with new, modified versions of its original equations. The new Pronghorn models are built on the finite volume method with a Harten-Lax-van Leer-Contact (HLLC) Riemann solver based numerical flux method, which (1) allows imposing slip boundary conditions much more robustly and (2) performs well for a wide range of flow speeds. The finite-volume-based flow solver will form the basis for a robust coarse-mesh thermal-hydraulics capability in Pronghorn.

This new Pronghorn capability was made possible by a new finite-volume (FV) method supported by the Multiphysics Object-Oriented Simulation Environment (MOOSE) framework [4]. The finite volume capability was added, in part, to support this milestone, and more generally to enhance MOOSE's computational fluid dynamics (CFD) capability. The finite volume method was selected because it offers some important features that help support modeling fluid dynamics, including:

- It is locally conservative, providing better accuracy in certain scenarios compared to the continuous finite element (FE) method, which is only guaranteed to be globally conservative.
- There is an abundance of stabilization methods for advection-dominated flow that are relatively straightforward to implement. Stabilization options for the continuous finite element method are more limited and often tricky to implement.
- It offers opportunities for improved performance over the finite element method in certain circumstances.
- Mature techniques for dealing with supersonic flows, shock capturing, and wave/shock propagation are well developed.
- Imposing slip boundary conditions on several boundaries with common vertices is straightforward.

Extensive code development was done in the framework in order to provide a powerfully integrated finite volume capability. While the finite volume method capability developed for MOOSE is not particularly special among existing FV modeling tools, its integration and coupling with MOOSE's multitude of features make it uniquely powerful. The finite volume method implementation in MOOSE provides the ability to run massively parallel simulations, couple finite element and finite volume discretizations of equations together in a single system, and utilize/share existing material property models between FE and FV discretizations, among other advantages.

In the remainder of the report, details of the MOOSE framework finite volume method implementation are discussed, Pronghorn's fluid modeling is described along with details about its HLLC flux method implementation, and a few examples demonstrating this new simulation capability are presented, including:

- A one-dimensional (1D) Sod shock tube problem with comparisons to the analytical solution as well as results from THM – a MOOSE-based 1D code
- A converging-diverging supersonic nozzle with comparison to analytical results
- Supersonic flow over a wedge with comparisons to results from an equivalent Star-CCM+ simulation

The examples focus on near-sonic to super-sonic flows because of the challenging nature of these problems for discretization methods of the Euler equations and to clearly show the advantage that the FV-based solver has over the FE-based solver.

2 Finite Volume Method Design in MOOSE

The FV method differs from the FE method in a a few important ways. In the FV method, each mesh element has its own independent solution approximation. FV element solutions do not share shape function solution contributions via degrees of freedom at shared nodes with neighboring elements, as is commonly done for the FE method. Instead, FV interelement coupling occurs through complementary fluxes at shared element faces. Take for example the advection diffusion equation:

$$\frac{\partial u}{\partial t} - \nabla \cdot k\nabla u + \nabla \cdot u\vec{v} - S = 0$$

For the FE method, we generate a weak form of the equation by multiplying by a test function w and then integrating over the domain Ω. We then use integration by parts and Gauss's divergence theorem to rewrite the diffusion term as the sum of two integrals–one over the domain (i.e. volume integration) and one over the domain boundary (i.e., surface integration) giving us:

$$\int w\frac{\partial u}{\partial t}\partial\Omega + \int \nabla w \cdot k\nabla u \partial\Omega - \int wk\nabla u \cdot \vec{n}\partial\Gamma + \int \nabla w \cdot \vec{v}u\partial\Omega + \int wu\vec{v} \cdot \vec{n}\partial\Gamma - \int wS\partial\Omega = 0.$$

This is then discretized most commonly using node-based shape functions. The surface integrals apply to the domain boundary and represent integrated/flux boundary conditions.

The FV method has a corresponding weak form, but uses a constant test function of 1. The divergence theorem is applied locally, giving surface integrals that not only apply at the domain boundary but also to shared interior element faces. The FV advection-diffusion weak form looks like this:

$$\int w\frac{\partial u}{\partial t}\partial\Omega - \int k\nabla u \cdot \vec{n}\partial\Gamma + \int u\vec{v} \cdot \vec{n}\partial\Gamma - \int S\partial\Omega = 0.$$

Notably, the FV method has element face surface integrals in addition to volume integrals, both of which occur over the entire mesh. The FE method, on the other hand, only requires volume integrations over each element along with surface integrals applied only on the domain boundary. In the FV method, mesh elements' equations are coupled together by numerical fluxes that are calculated on the shared faces between mesh elements. This face-focused coupling drives many of the design decisions made for MOOSE's FV implementation.

2.1 Framework Integration

For researchers, MOOSE provides many built-in capabilities that enhance the quality and quantity of equation modeling. Implementing the FV method in MOOSE while maintaining and integrating with existing capabilities required careful consideration and design. Some of these capabilities are listed below, along with a discussion on how they affected the FV method design and implementation in MOOSE.

2.1.1 Parallelism and Threading

MOOSE is designed to take advantage of parallelism available on modern computers, including supercomputers as well as laptops and desktop workstations. The FV method implementation needed to effectively follow the patterns in MOOSE for scaling to multiple cores. A mesh-face threaded loop was created following the patterns established in the existing finite element threaded loops. This allows operations that involve looping over the mesh faces (e.g., computing the numerical fluxes) to be partitioned into several threads and performed simultaneously. Data structures needed for looping over the mesh faces were carefully coded to account for the parallel mesh partitioning that occurs in both threaded and message passing interface (MPI)-based parallelism.

2.1.2 Simple Kernel Application Programming Interface (API)

MOOSE has several well-established design conventions that users/researchers follow when building models and running simulations. This includes the concept of a physics *kernel* – a generally simple, encapsulated piece of code that describes the mathematical terms of the equations of interest. Due to differences in needs between FE and FV, a dedicated FV kernel system was created.

The FV method uses the Gauss-divergence theorem to convert volume integrals with a divergence operator into surface integrals representing flux of various quantities through faces between mesh cells. Unlike FE kernels, no test/weight function is needed. Coupling between cells occurs through a numerical flux calculation on the shared face. Calculating numerical fluxes requires access to variable values and material properties on both sides of each face. FE kernels, on the other hand, require only one set of volumetric/elemental values for the cell of interest. FV kernels also need to deal with things such as normal face vectors, cross-diffusion correction factors for non-orthogonal meshes, etc. All these differences make it impractical and messy to try to integrate everything into the existing MOOSE kernel system and motivated the decision to create a separate FV kernel system.

Although a separate system was created, the interfaces and APIs couple well and familiarly with other MOOSE systems. FV kernels were integrated cleanly with the material property system. In short, a lot of work was done behind the scenes to ensure that researchers using the FV method capability can do things in all the ways that they are familiar with in MOOSE.

2.1.3 Mesh Adaptivity

MOOSE has the ability to perform automatic refinement in ”interesting” areas of the mesh according to various user-defined metrics. Mesh adaptivity can create hanging nodes where one large element face abuts multiple smaller faces. In order for this to work well with the FV method implementation, data structures for looping over the faces needed to be created with care to account for these hanging nodes and corresponding face asymmetry. This among other accommodations allow the FV implementation to function correctly and seamlessly with mesh adaptivity.

2.1.4 Automatic Differentiation

MOOSE has the ability to automatically track partial derivatives of equation terms with respect to the degrees of freedom of the solution. This facilitates computing a complete Jacobian matrix that assists in efficiently and robustly solving a wide range of multiphysics problems. The FV method is somewhat unique from the FE method in that equation terms (i.e., residuals) in one element are computed using values directly from neighboring elements rather than indirectly through shared nodal degrees of freedom. These interelement residual contributions have been carefully accounted for to maintain the power of automatic differentiation in supplying a correct Jacobian matrix, just like users have come to expect for FE problems.

2.1.5 Simple/Flexible Boundary Conditions

Similar reasoning to that for the FV kernel system motivated the creation of a separate FV boundary condition (BC) system as well. While FV-integrated BCs lack the test/weight functions of FE-integrated BCs, they are somewhat similar to FE-integrated BCs. FV Dirichlet BCs, however, must be implemented completely differently than in FE and strongly motivated the creation of a separate FV BC system.

Dirichlet BCs in an FV method cannot be created by directly setting degrees of freedom like in an FE method because FV solution degrees of freedom do not exist on the mesh boundary. There are various approaches for dealing with this. A ghost-element approach was selected due to its popularity and robustness. In this approach, Dirichlet BCs are implemented as a weak BC. To do this, all the numerical flux kernel terms are applied at the mesh boundary faces by using special solution values for the ghost element. Since flux kernels are calculated using information from cells on both sides of the face, we use the desired Dirichlet BC value to extrapolate a ghost cell value for the side of the face that has no actual mesh cell. Other necessary cell properties are also reflected/mirrored from the existing cell. A design was chosen that allows handling ghost-element creation and use by existing flux kernels automatically for enforcing Dirichlet BCs. This allows use of weak Dirichlet BCs without requiring users to write special BC versions of their kernel terms to handle calculations with the boundary's missing neighbor element. Corresponding care was taken to enable material properties to be evaluated and used on these non-existing mesh elements, removing all need for users to give special consideration to FV Dirichlet BCs.

2.1.6 Equation Coupling

A powerful feature of MOOSE is the ability to seamlessly couple together multiple partial differential equations. This capability has been preserved, allowing FV-dependent variables to be coupled together in exactly the same way. Additionally, the FV system was designed to allow bidirectional coupling between both FE- and FV-dependent variables. The APIs were designed so that users can create generic code for material models and kernels (among other things) and switch seamlessly between coupling to FE- or FV-dependent variables based only on modifications to an input file – no changes to code are necessary. Although some details of this FE-FV coupling are not fully implemented, this FV-FE agnostic coupling ability has great potential to provide interesting research opportunities.

FV-specific dependent variable classes were created alongside the FE-specific classes sharing common base functionality. These classes are responsible for calculating and providing FV cell and face solution values to objects that need them. These classes transparently handle gradient reconstruction and ghost-elements for users, allowing user kernel and boundary condition code to be written similarly to their FE counterparts.

2.2 Implementing Custom Physics

To implement equation terms, users will create a C++ class following normal MOOSE patterns. The following example shows what a simple convective flux kernel term might look like:

```
class MyEquation : public FVFluxKernel
{
 public:
    MyEquation(InputParameters params)
      : FVFluxKernel(params),
        _velocity_elem(getADMaterialProperty<RealVectorValue>("velocity")),
        _velocity_neighbor(getNeighborADMaterialProperty<RealVectorValue>("
    velocity")),
        _density_elem(adCoupledGradient("density")),
        _density_neighbor(adCoupledNeighborGradient("density"))
    {
    }

    virtual ADReal computeQpResidual() override
    {
        ADReal avg_density = (_density_elem[_qp] + _density_neighbor[_qp]) * 0.5;
        ADRealVectorValue avg_velocity = (_velocity_elem[_qp] + _velocity_neighbor
    [_qp]) * 0.5;
        ADReal residual = avg_density * avg_velocity * _normal;
        return residual;
    }

 private:
  const ADMateriaProperty<RealVectorValue> & _velocity_elem;
  const ADMateriaProperty<RealVectorValue> & _velocity_neighbor;
  const ADVariableGradient & _density_elem;
  const ADVariableGradient & _density_neighbor;
};
```

Listing 1: Implementing a flux kernel equation term.

This example demonstrates coupling to material properties as well as other dependent variables. This is very similar to typical FE kernel code in MOOSE. The primary difference is that for flux terms, the residual contribution is evaluated on a mesh face rather than on a mesh element. Consequently, solution values and material properties from elements on both sides of the face must be used in addition to certain face-relevant quantities such as the `_normal` data member representing the face's outward unit normal vector. Other face-relevant geometric information is provided via a `_face_info` variable.

For advective equation terms, different numerical flux methods may be used to account for the advection velocity vector. The following example shows a modified version of the same `computeQpResidual` function that provides basic upwinding:

```
virtual ADReal computeQpResidual() override
{
    ADRealVectorValue vel_dot_normal = _normal * (_velocity_elem[_qp] +
    _velocity_neighbor[_qp]) * 0.5;
    if (vel_dot_normal > 0)
      return vel_dot_normal * _density_elem[_qp];
    return vel_dot_normal * _density_neighbor[_qp];
}
```

Listing 2: Upwind advective flux term.

Support for automatically handling basic, common numerical flux methods can be used like this:

```
virtual ADReal computeQpResidual() override
{
  ADRealVectorValue vel_interface;
  interpolate(InterpMethod::Average, vel_interface, _velocity_elem[_qp],
    _velocity_neighbor[_qp]);
  ADReal density_interface;
  interpolate(InterpMethod::Upwind, density_interface, _density_elem[_qp],
    _density_neighbor[_qp], vel_interface);
  return _normal * vel_interface * density_interface;
}
```

Listing 3: Using built-in interpolation methods.

Support for automatically switching between other flux schemes such as Quadratic Upstream Interpolation for Convective Kinematics (QUICK) and total variance diminishing (TVD) will be added in the near future, making it possible to easily toggle between flux methods from an input file without having to modify any code.

2.3 In progress functionality

Work for supporting various gradient reconstruction methods is under way. This will help provide higher-order numerical flux approximations and better shock-capturing. Support for easy toggling between basic numerical flux schemes including TVD, QUICK, and others has also begun and should be available in the near future. Support for displaced/moving meshes is under development. Another direction of development aims at improving performance of FV simulations – speed is likely to improve by at least a factor of two in the coming months.

3 Pronghorn

Pronghorn is a multidimensional, coarse-mesh thermal-hydraulics code that initially focused on pebble bed reactor applications [1]. It is built on the multiphysics object oriented simulation environment (MOOSE) [4]. Current development efforts focus on extending Pronghorn as a general purpose coarse-mesh thermal-hydraulics code. In contrast to computational fluid dynamics (CFD), coarse-mesh thermal-hydraulics models only model the major geometric features of the fluid-flow geometry and account for small geometric features, turbulence, and boundary layers via appropriate correlations.

The essential idea of the Pronghorn coarse-mesh thermal-hydraulics approach is shown in Fig. 1. On the left is a picture of the HTR-10 reactor. Clearly visible is the pebble cavity and the lower-cone region. The hot helium leaves the core through the small holes at the bottom of the pebble cavity. Around the pebble cavity is the graphite reflector, including the control rod holes, the small absorber sphere holes (KLAK), and the riser channels. On the right of the figure is the corresponding Pronghorn geometry. Large features like the riser and control-rod holes are modeled explicitly, while small features like the outlet holes at the bottom of the bed are not modeled at all. The pressure drop due to the outlet holes are accounted for by appropriate correlations.

This report details the implementation of an FV-based discretization of the Euler equations into Pronghorn. The FV-based implementation (1) enhances the robustness of Pronghorn for modeling complex flow geometries; and (2) it extends the range of Pronghorn's capabilities to model very slow to very fast flows.

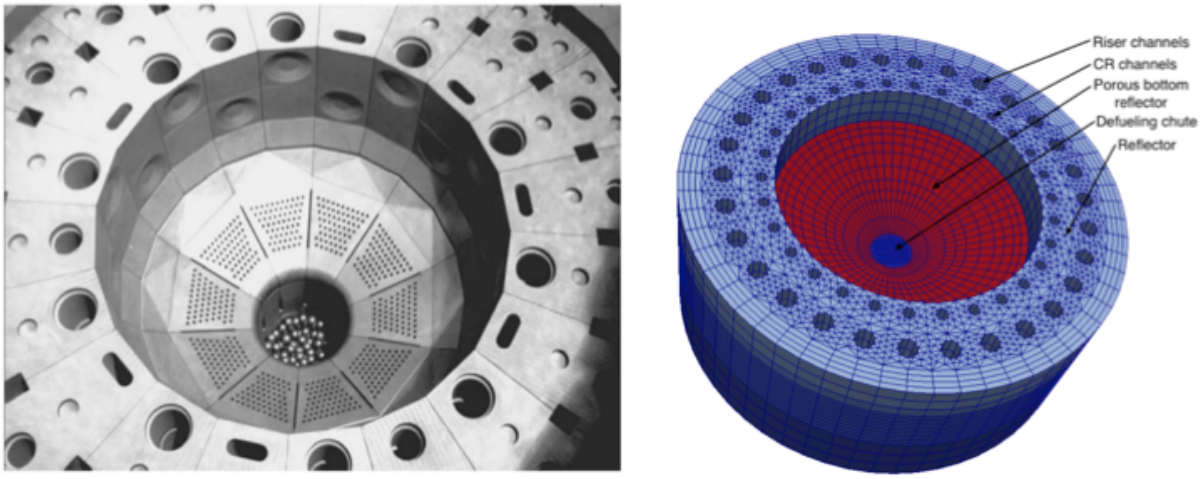


Figure 1: This figure shows the essential idea of the coarse-mesh approach. On the left is a picture of the HTR-10 reactor, on the right of the figure is the corresponding Pronghorn geometry.

3.1 Pronghorn Models

This section introduces the Euler equations that serve as model equations for high-speed flows in Pronghorn. The Euler equations are discretized using the FV method and the HLLC Riemann solver.

3.1.1 Euler equations

High-speed flow in Pronghorn is modeled using the Euler equations [5]. The Euler equations are obtained as an approximation of the Navier-Stokes equations [6] by neglecting viscous shear forces. This approximation is valid outside of the boundary layer. The boundary-layer thickness decreases as $1/\sqrt{\text{Re}}$, where Re is the Reynolds number, so that the boundary layer is very small for nearly inviscid, fast flows. Therefore, the Euler equations apply to the majority of the flow field for nearly inviscid, fast flows.

The Euler equations are given by:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) &= 0 \\ \frac{\partial \rho \vec{v}}{\partial t} + \nabla \cdot (\rho \vec{v} \otimes \vec{v}) + \nabla p &= 0 \\ \frac{\partial \rho E}{\partial t} + \nabla \cdot (\rho \vec{v} H) &= 0, \end{aligned} \tag{1}$$

where $\rho(\vec{r}, t)$ represents fluid density as a function of spatial location $\vec{r}$ and time t, $\vec{v}(\vec{r}, t)$ is the velocity vector, $\otimes$ denotes the tensor product of two vectors, $p(\vec{r}, t)$ is the pressure, $E(\vec{r}, t)$ is the total energy density given by $E = e + \frac{1}{2}\vec{v} \cdot \vec{v}$, $e(\vec{r}, t)$ is the internal energy density, and $H(\vec{r}, t)$ is the total enthalpy density given by $H = h + \frac{1}{2}\vec{v} \cdot \vec{v}$.

It is convenient to write the Euler equations in vector notation by defining the unknown vector $\vec{U}$:

$$\vec{U} = (\rho, \rho\vec{v}, \rho E)^T, \tag{2}$$

and the flux vector $\vec{F}$:

$$\vec{F} = (\rho\vec{v}, \rho\vec{v} \otimes \vec{v} + p\mathbf{I}, \rho\vec{v}H)^T, \tag{3}$$

where $\mathbf{I}$ is the identity matrix. The Euler equations are then given by:

$$\frac{\partial \vec{U}}{\partial t} + \nabla \cdot \vec{F} = 0. \tag{4}$$

Within this report, the ideal gas equation is exclusively used as equation of state (EOS). The ideal gas equation connects density, pressure, and temperature:

$$\frac{p}{\rho} = R_s T, \tag{5}$$

where R_s is the specific gas constant and T is the fluid temperature. In addition, internal energy and enthalpy are connected with temperature by the isochoric and isobaric heat capacity:

$$\begin{aligned} e &= c_v T \\ h &= c_p T, \end{aligned} \tag{6}$$

where c_v and c_p are the constant isochoric and isobaric heat capacities of the fluid. From the heat capacities, the heat capacity ratio (also known as adiabatic index) is computed:

$$\gamma = c_p / c_v. \tag{7}$$

The Euler equations are complemented by a set of boundary conditions. In this report, we adopt the approach of labeling boundary conditions by the physical situation they represent. Relevant boundary conditions for this report are stagnation inlet, supersonic inlet, supersonic outlet, and wall.

- Stagnation inlet boundary conditions: these boundary conditions model inlet conditions when reservoir pressure (i.e., at rest or at stagnation) p_s and temperature T_s of the gas are known. Currently, stagnation boundary conditions are only available for ideal gases. Inlet conditions (indexed by subscript i) for density and total enthalpy are computed from the stagnation conditions by:

$$\begin{aligned} T_i &= T_s - \frac{1}{2c_p}\vec{v} \cdot \vec{v}, \\ p_i &= p_s \left(\frac{T_s}{T_i}\right)^{-\frac{\gamma}{\gamma-1}} \\ \rho_i &= \frac{p_i}{R_s T_i} \\ H_i &= c_v T_i + \frac{1}{2}\vec{v} \cdot \vec{v} + p_i/\rho_i, \end{aligned} \tag{8}$$

where $\vec{v} = \vec{v}(\vec{r}_b, t)$ is the computed value of the velocity evaluated at point $\vec{r}_b$ on the boundary. The stagnation inlet boundary conditions are imposed on the flux vector at the boundary:

$$\vec{n} \cdot \vec{F}_b = [\rho_i \vec{n} \cdot \vec{v}(\vec{r}_b, t), \rho_i \vec{n} \cdot \vec{v}(\vec{r}_b, t) \otimes \vec{v}(\vec{r}_b, t), \rho_i H_i \vec{n} \cdot \vec{v}(\vec{r}_b, t)]^T \tag{9}$$

Note, the velocity is not imposed by the boundary conditions, but is rather taken as the computed velocity evaluated on the boundary. This is sometimes referred to as allowing the velocity to "float" (see Ref. [7] for using the term "float" in this manner).

- Supersonic inlet: characteristic analysis shows that all flow variables must be prescribed on supersonic inlets [7]. The inflow conditions are imposed on the fluxes at the boundary:

$$\vec{n} \cdot \vec{F}_b = [\rho_i \vec{v}_i, \rho_i \vec{v}_i \otimes \vec{v}_i + p_i \mathbf{I}, \rho_i \vec{v}_i H_i]^T (\vec{r}_b, t), \tag{10}$$

  where ρ_i, $\vec{v}_i$, H_i, and p_i are all externally imposed quantities.

- Supersonic outlet: characteristic analysis shows that on supersonic outlets no flow variables can be prescribed [7] (i.e., all flow variables have to "float"). The outflow conditions are imposed on the fluxes at the boundary:

$$\vec{n} \cdot \vec{F}_b = [\rho \vec{v}, \rho \vec{v} \otimes \vec{v} + p \mathbf{I}, \rho \vec{v} H]^T (\vec{r}_b, t), \tag{11}$$

  where the $(\vec{r}_b, t)$ denotes that all quantities in the vector are computed quantities evaluated at location $\vec{r}_b$ and time t.

- Wall boundary conditions: walls are solid obstacles that do not allow flow through them:

$$\vec{n} \cdot \vec{v} = 0, \tag{12}$$

  where $\vec{n}$ is the unit normal on the wall. This condition translates into a condition for $\vec{F}_b$:

$$\vec{n} \cdot \vec{F}_b = [0, \vec{n} p, 0]^T, \tag{13}$$

3.1.2 Integral Form of the Euler Equations and Finite Volume Formulation

The integral form of the Euler equation are the starting point for the finite-volume discretization. It is obtained by integrating Eq. 4 over the extent of a mesh element Ω and using Gauss's theorem:

$$\frac{\partial}{\partial t} \int_\Omega \vec{U} d\Omega + \sum_{j=1}^{J} \int_{S_j} \vec{n}_j \vec{F} \left(\vec{U} \right) dS, \tag{14}$$

where S_j is the j-th face of the element. The average of the solution vector over the element is defined by:

$$\vec{U} = \frac{1}{\Omega} \int_\Omega \vec{U} d\Omega, \tag{15}$$

and the average of the numerical flux over face $\partial\Omega_j$ is defined by:

$$\vec{\vec{F}}_j = \frac{1}{S_j} \int\limits_{\partial\Omega_j} \vec{n}_j \vec{F} dS \tag{16}$$

Then the integral form of the Euler equations is given by:

$$\frac{\partial \vec{U}}{\partial t} + \sum_{j=1}^{J} \vec{\vec{F}}_j S_j = 0. \tag{17}$$

The integral form of the Euler equations, Eq. 17, is exact. However, it is incomplete because it does not provide a means to evaluate $\vec{\vec{F}}_j$ in terms of the adjacent $\vec{U}$. Discretization methods provide formulas to compute the flux over the face j, $\vec{n}_j \vec{\vec{F}}_j$, in terms of the adjacent solution vectors $\vec{U}_{j,L}$ and $\vec{U}_{j,R}$ where L and R stand for left and right, respectively. The finite-volume form of the Euler equations is then given by:

$$\frac{\partial \vec{U}}{\partial t} + \sum_{j=1}^{J} \vec{\vec{F}}_j \left(\vec{U}_{j,L}, \vec{U}_{j,R} \right) S_j = 0. \tag{18}$$

The approximate flux on the boundary, $\vec{\vec{F}}_j \left(\vec{U}_{j,L}, \vec{U}_{j,R} \right)$, is typically referred to as numerical flux. The HLLC Riemann solver is a discretization scheme that provides formulas to compute the numerical fluxes.

3.1.3 HLLC Riemann Solver

The HLLC Riemann solver is a Godunov-type method for the solution of the Euler equations [8]. Godunov's method is the extension of upwinding to nonlinear equations [9]. The advantage of Godunov's approach is that it generalizes upwinding from single linear advection equations to nonlinear systems of equations [5]. The upwinding directions are obtained by solving the Riemann problem for the system of equations under consideration. Thereby, the Riemann problem is a one-dimensional (1D) initial value problem with piecewise constant initial data on the left and right sides [5]. Godunov-type methods are attractive because of the ability to automatically capture shocks at their correct location and because of their intrinsic robustness.

The solution of the Riemann problem is at the core of Godunov-type methods (i.e., the Riemann problem is the kernel step of Godunov-type methods). Obtaining exact solutions of Riemann problems is computationally expensive; therefore, many approximate Riemann solvers were devised. Examples of approximate Riemann solvers are the Roe solver that exactly solves the "linearized"[1] Riemann problem of the Euler equations [10] and the Harten-Lax-van Leer (HLL) Riemann solver that simplifies the Riemann problem to consist of only two waves, namely the shock and rarefaction waves. Many other approximate Riemann solvers exist and are described in Refs. [11, 12] and enclosed references.

The HLLC Riemann solver is an improvement on the HLL solver. The HLL solver tracks only two shock fronts that separate three constant states and neglects the contact discontinuity. The difference between the two shock fronts and the contact discontinuity is that only density changes discontinuously over the contact discontinuity, while pressure and velocity are continuous. The HLLC Riemann solver improves on the HLL Riemann solver by restoring the contact discontinuity in the approximate solution of the Riemann problem [8].

[1] To be more precise, the solution dependent matrix in the quasilinear form of the Euler equations is replaced by the Roe matrix.

A detailed discussion of the derivation of the HLLC Riemann solver is available in Ref. [11]. This report states only the final equations used in the implementation of the HLLC Riemann solver in Pronghorn. The numerical flux is computed via:

$$\vec{F}_j\left(\vec{U}_{j,L},\vec{U}_{j,R}\right)=\begin{cases}\vec{n}\cdot\vec{F}\left(\vec{U}_{j,L}\right) & S_L>0\\ \vec{n}\cdot\vec{F}^*\left(\vec{U}_{j,L}\right) & S_L\leq 0<S_*\\ \vec{n}\cdot\vec{F}^*\left(\vec{U}_{j,R}\right) & S_*\leq 0\leq S_R\\ \vec{n}\cdot\vec{F}\left(\vec{U}_{j,R}\right) & S_R<0.\end{cases}, \tag{19}$$

where S_j with $j=L,R,*$ stands for the wave speeds of the left, right, and contact discontinuity, $\vec{F}\left(\cdot\right)$ is simply given by Eq. 3, while $\vec{F}^*\left(\vec{U}_{j,K}\right)$ for $K=L,R$ is given by:

$$\vec{F}^*\left(\vec{U}_{j,K}\right)=\vec{F}\left(\vec{U}_{j,K}\right)+S_K\left(\vec{U}_{j,*,K}-\vec{U}_{j,K}\right). \tag{20}$$

In Eq. 20, $\vec{U}_{j,*,K}$ is given by:

$$\vec{U}_{j,*,K}=\rho_K\left(\frac{S_K-q_K}{S_K-S_*}\right)\begin{bmatrix}1\\ \vec{n}_j\left(S_*-q_K\right)+\vec{v}_K\\ \frac{E_K}{\rho_K}+\left(S_*-q_K\right)\left(S_*+\frac{p_K}{\rho_K\left(S_K-q_K\right)}\right)\end{bmatrix}, \tag{21}$$

where $q_K=\vec{n}\cdot\vec{v}_K$. The wave speeds S_L, S_R, and S_K are estimated using the formulas given in Ref. [10, 13]:

$$\begin{aligned}S_L&=\min\left(q_L-c_L,q_{\text{Roe}}-c_{\text{Roe}}\right),\\ S_R&=\max\left(q_R+c_R,q_{\text{Roe}}+c_{\text{Roe}}\right),\\ S_*&=\frac{\rho_R q_R(S_R-q_R)-\rho_L q_L(S_L-q_L)+p_L-p_R}{\rho_R(S_R-q_R)-\rho_L(S_L-q_L)},\end{aligned} \tag{22}$$

where subscript Roe indicates Roe-averaged quantities and c denotes the speed of sound. The Roe-averaged quantities are given by:

$$\begin{aligned}v_{\text{Roe}}&=\frac{\sqrt{\rho_L}v_L+\sqrt{\rho_R}v_R}{\sqrt{\rho_L}+\sqrt{\rho_R}}\\ q_{\text{Roe}}&=\frac{\sqrt{\rho_L}q_L+\sqrt{\rho_R}q_R}{\sqrt{\rho_L}+\sqrt{\rho_R}}\\ H_{\text{Roe}}&=\frac{\sqrt{\rho_L}H_L+\sqrt{\rho_R}H_R}{\sqrt{\rho_L}+\sqrt{\rho_R}}\\ h_{\text{Roe}}&=H_{\text{Roe}}-\frac{1}{2}v_{\text{Roe}}^2\\ \rho_{\text{Roe}}&=\sqrt{\rho_L\rho_R}\\ c_{\text{Roe}}&=c(\rho_{\text{Roe}},h_{\text{Roe}}),\end{aligned} \tag{23}$$

where $v_K=\sqrt{\vec{v}_K\cdot\vec{v}_K}$.

4 Demonstration Problems

We compare the results of Pronghorn's FV HLLC implementation to a set of common test problems. The set is comprised of a 1D Sod shock tube [14], a supersonic converging-diverging nozzle problem [7], and supersonic flow over a wedge [11].

4.1 1D Sod Shock Tube

The Sod shock tube problem is a 1D test problem for Euler equation solvers [14]. It consists of a tube of infinite length (taken as $L = 100$ in this work) filled with a gas in two different states on the left and right sides of a diaphragm located at the center of the shock tube at $x_D = 50$. In this work, we follow the initial conditions given in Ref. [14]:

$$p(x, t = 0) = \begin{cases} 1 & \text{if } x < x_D; \\ 0.1 & \text{if } x \geq x_D. \end{cases}$$
$$\rho(x, t = 0) = \begin{cases} 1 & \text{if } x < x_D; \\ 0.125 & \text{if } x \geq x_D. \end{cases}, \tag{24}$$

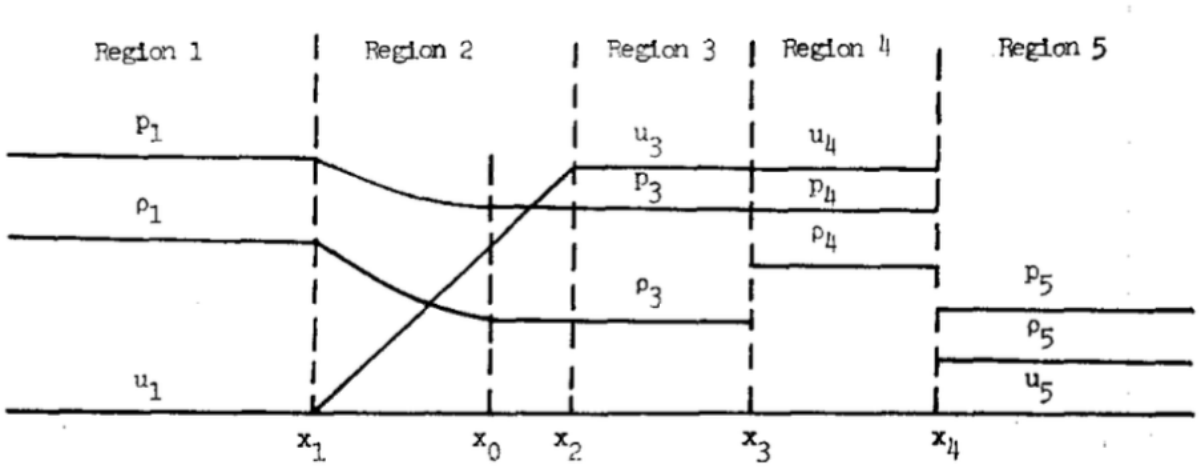
and $v_x = 0$. The fill gas is assumed to be air with a molar weight of $M = 0.029$ kg/mol and a specific heat ratio of $\gamma = 1.4$; the fill gas is assumed to behave like an ideal gas. The problem is initiated by breaking the diaphragm at $t = 0$. Boundary conditions are irrelevant for the shock tube problem because the rarefaction and shock waves do not reach the boundary within the considered time window.

The Euler equations can be solved analytically for the shock-tube problem; for a detailed solution, refer to Ref. [14]. The solution is comprised of five distinct regions separated by the three characteristics of the system (left running rarefaction wave at $v_x - c$, contact discontinuity at v_x, and right running shock discontinuity at $v_x + c$). It is convenient to number the regions from left to right using indices $r = 1, \ldots, 5$. Region $r = 1$ and $r = 5$ remain in their initial states (i.e., the left and right initial states in Eq. 24, respectively). The states in Region 2 to 4 can be computed using the approach described in Refs. [5, 11]. The following characterization of the problem is sufficient for the purpose of this work:

- The state in Regions 4 and 5 is related by the Ranking Hugoniot condition.
- Density and velocity are continuous across the contact discontinuity (Regions 3 and 4).
- The solution in Region 2 is a rarefaction wave.

A sketch of the pressure, density, and velocity (denoted as u in the figure) is adapted from Ref [14] and shown in Fig. 2

Figure 2: Regions developing in the Sod shock-tube problem. Note that velocity v_x is denoted as u in the figure. Picture courtesy Ref. [14].



Pronghorn solutions for Sod's shock-tube problem are compared against the analytical solution and against results obtained with the MOOSE-based thermal-hydraulics module (THM) [15]. THM is a system analysis code that supports solution of the 1D Euler equations. A snapshot of density at $t = 10$ seconds is shown in Fig. 3, for the following simulation cases:

1. FV, HLLC, 1D: Pronghorn using the FV HLLC implementation solving a 1D x-axis aligned shock tube. Two additional two-dimensional (2D) models are prepared with wall boundary conditions applied on top and bottom. The first 2D model keeps the tube axis aligned with the x-axis, while the second test rotates the tube by 45 degrees to test invariance of the solution to rotating the coordinate system. Both 2D models should give the exact same answer as the 1D model.

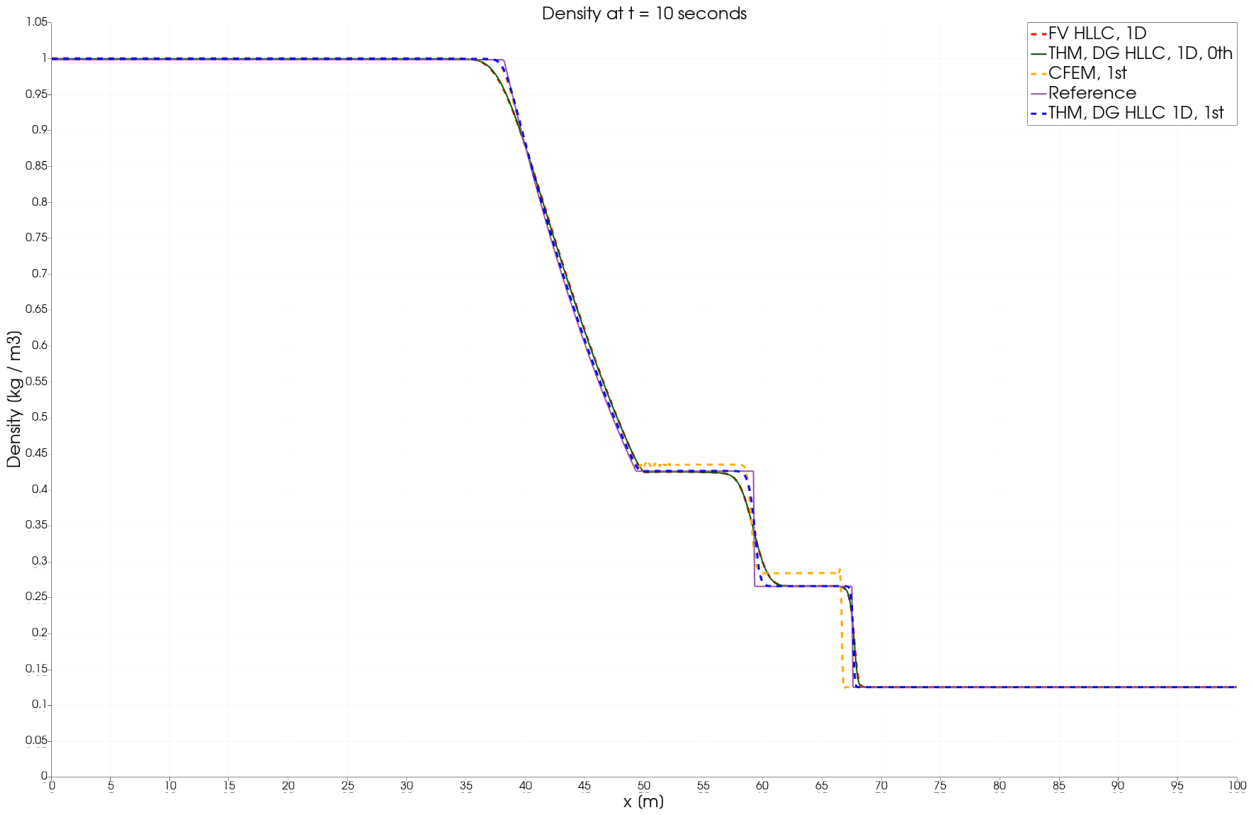
2. THM, DG HLLC, 1D, 0th: THM solution using the discontinuous Galerkin FEM (DGFEM) HLLC implemention. This discretization is equivalent to the Pronghorn finite volume HLLC implementation.

3. THM, DG HLLC, 1D, 1st: THM solution using DGFEM HLLC implemention with linear reconstruction. This method reconstructs the solution variables at the element interfaces when solving the Riemann problem; the reconstruction makes this method second-order accurate.

4. CFEM, 1st: Pronghorn continuous FEM (CFEM) solution using Lagrange first-order shape functions.

All solutions are obtained using $1,000$ elements and a time step of $\Delta t = 0.01$ seconds. An explicit, total variation diminishing Runge-Kutta method of order two is used for all but the CFEM results; the CFEM results are obtained using the implicit backward Euler method.

Figure 3: Comparison of the density at $t = 10$ seconds for the Sod shock-tube problem for a variety of different discretization methods available in the MOOSE framework. The reference results are obtained by analytical solution of the shock-tube problem.



As expected, Cases 1 and 2 from the above enumeration give the exact same density variation along the tube; in addition, the 2D models both give the same answer as Cases 1 and 2. The higher order THM option (THM, DG HLLC, 1D, 1st) gives similar results with the distinct difference that the representation of the discontinuities in the density is sharper. This is expected since higher-order discretization methods represent discontinuities in the solution more sharply (i.e., the solution has a steeper slope across the discontinuity). All discretization methods based on HLLC (DGFEM and FV) agree quite well with the reference solution in two important aspects:

- The wave speeds are predicted correctly. The midpoint of the "smeared" front agrees well with the location of the sharp discontinuity of the reference solution.
- The values of density within each region are predicted accurately.

In contrast, the continuous FEM-based Pronghorn solution predicts the density variation poorly. First, overshoots at the discontinuities and oscillations in Region 3 are visible; second, the plateau values in Regions 3 and 4 are predicted poorly; third, the location of the shock discontinuity is not predicted well.

The shock-tube results indicate that the FV-based HLLC implementation in Pronghorn is capable of accurately resolving Sod's problem.

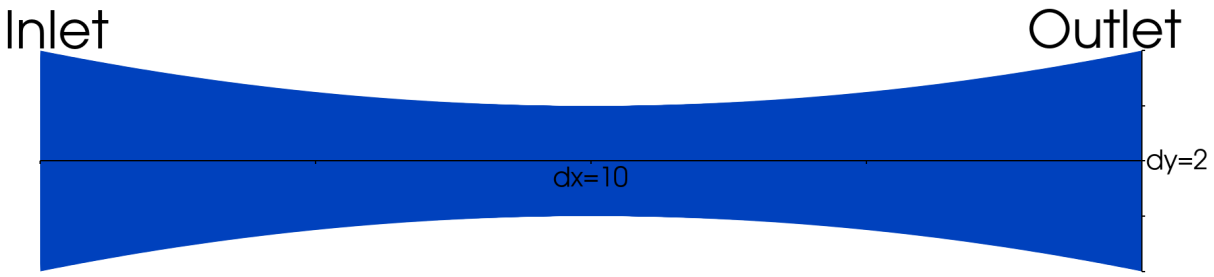
4.2 Supersonic Nozzle

This test problem models supersonic flow through a converging-diverging nozzle. Converging-diverging nozzles are common in rocket engines where they enable supersonic outlet velocities [7]. The test problem can be thought of as an infinitely large reservoir filled with gas (air modeled as an ideal gas) at stagnation pressure $p_s = 1$ and stagnation temperature $T_s = 1$ attached to a converging-diverging nozzle of length $L = 10$ aligned with the x-axis. The cross-sectional area varies along the nozzle axis as:

$$A(x) = 1 + 4\left(\frac{x}{L} - \frac{1}{2}\right)^2. \tag{25}$$

The geometry of the nozzle is depicted in Fig. 4.

Figure 4: Supersonic converging-diverging nozzle geometry.



Stagnation boundary conditions for the mass, momentum, and energy equations are applied to the inlet boundary on the left. The top and bottom boundary (i.e., the arcs) are wall conditions. Supersonic outlet conditions are applied on the right. The outlet boundary conditions enable the flow to be supersonic when leaving the domain because the outlet pressure is not imposed and is allowed to adjust to the interior flow conditions and expansion ratio of the nozzle. The value that the pressure naturally assumes at the outlet is referred to as the natural outflow pressure.

The supersonic nozzle problem is modeled as a 2D problem in Pronghorn. The 2D converging-diverging nozzle problem can be reduced to a quasi 1D problem that does not depend on the y-variable, but allows for variable cross-sectional area $A(x)$. The steady-state, quasi 1D problem is solved analytically and compared to the Pronghorn results. It is noted that the quasi 1D problem provides results that are close but not exactly equal to the 2D problem because it does not account for the non-zero y-velocity component.

The analytical solution is taken from Ref. [7]. The cross sectional area $A(x)$ and the Mach number Ma are related by:

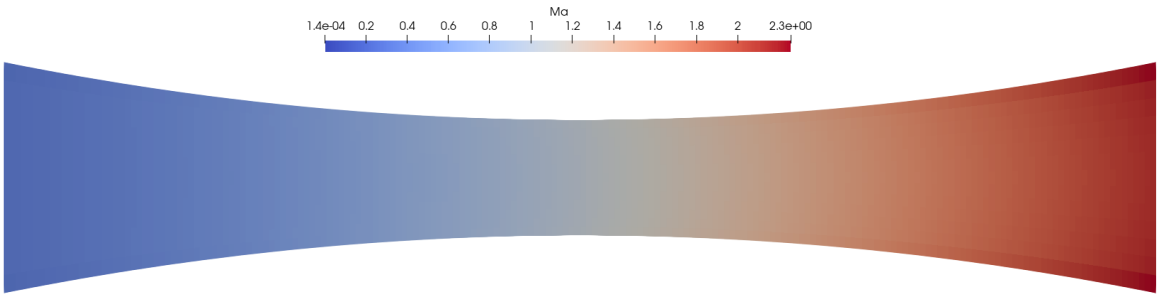
$$\left(\frac{A(x)}{A_t}\right)^2 = \frac{1}{\mathrm{Ma}^2}\left[\frac{2}{\gamma+1}\left(1 + \frac{\gamma-1}{2}\mathrm{Ma}^2\right)\right]^{\frac{\gamma+1}{\gamma-1}}, \tag{26}$$

where A_t is the throat area: $A_t = A(L/2)$. Using Eq. 25, Eq. 26 is numerically solved for $\mathrm{Ma}(x)$. Pressure, density, and temperature are normalized by their respective inlet values and depend solely on the Mach number. For density, one obtains:

$$\frac{\rho(x)}{\rho(x=0)} = \left(1 + \frac{\gamma - 1}{2}\mathrm{Ma}^2\right)^{-1/(\gamma-1)}. \tag{27}$$

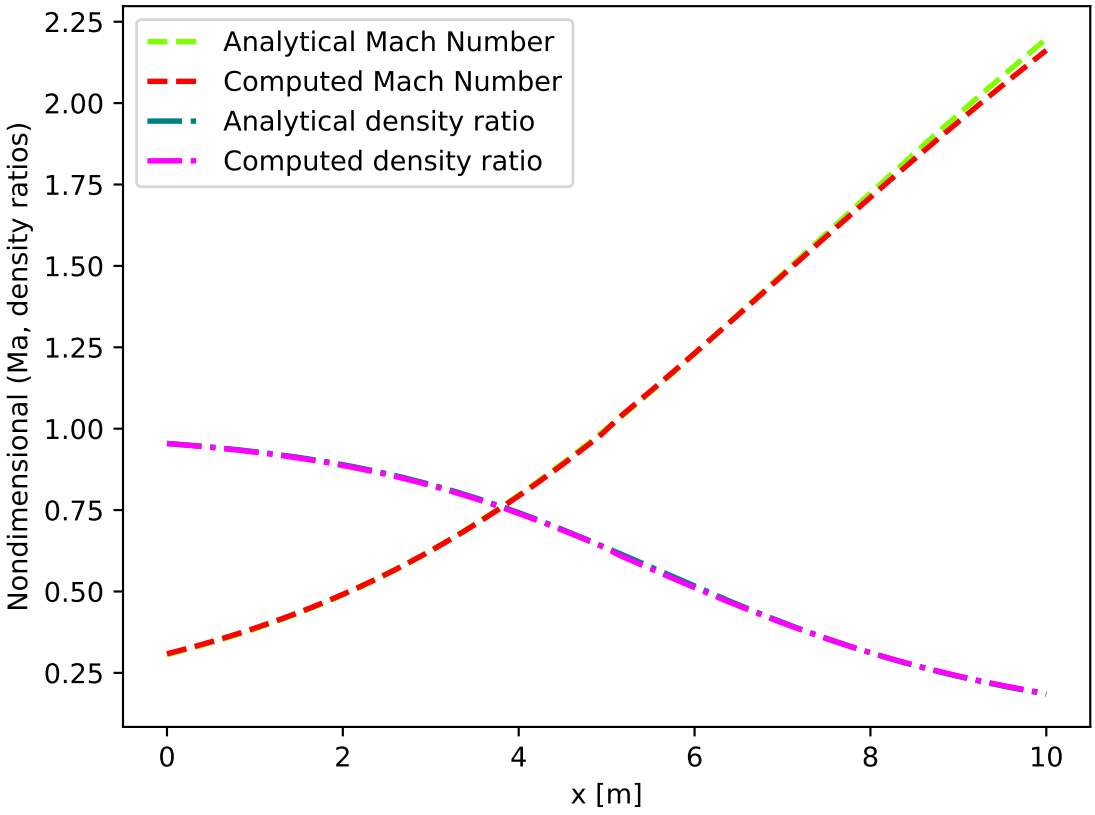
The Pronghorn results are obtained with a mesh comprised of $2,600$ quadrilateral elements and a time step of 0.001 seconds. The explicit, total variation diminishing Runge-Kutta method of order two is used for time integration. Mesh and time step independence of the results have been established by comparison to a mesh with $7,200$ elements and and a time step of 0.0005 seconds. Results are obtained by running a pseudo-transient to an end time of 20 seconds; by this time both Mach number and density have reached a steady-state condition. The computed Mach number is depicted in Fig. 5.

Figure 5: Mach number for the converging-diverging nozzle problem using $2,600$ elements and time step of 0.001 seconds.



For comparing Pronghorn with the quasi 1D analytical results, Pronghorn's Mach number and density are averaged over slices perpendicular to the nozzle axis. Averaged Pronghorn and analytical results are plotted versus x in Fig. 6. Both density ratio and Mach number agree well between the two computations. We attribute the difference between analytical and Pronghorn results to the qualitative difference between the two models (i.e. 1D vs 2D) as mesh and time step independence for the Pronghorn results has been established.

Figure 6: Mach number and density ratio computed with Pronghorn compared with analytical, quasi-1D results.



This example demonstrates that Pronghorn's finite volume implementation can model supersonic flows in multidimensional geometries – relevant for modeling depressurization events in gas-cooled reactors.

4.3 Supersonic Flow Over Wedge

Compressible flow over wedge-shaped obstacles is an important topic of research both for code benchmarking [16] and practical applications ranging from aerospace to explosives applications [17]. In the case of a supersonic flow past a wedge, the Mach number and wedge angle determine the structure of the shock front which can be either a detached normal shock or an attached oblique shock. Normal shocks are defined as shocks perpendicular to the flow direction, while oblique shocks are inclined to the flow direction.

A detached normal shock occurs if the actual wedge angle exceeds the maximum wedge angle $a_{\max}$ (measured in radians) given by:

$$a_{\max} = \frac{4}{3\sqrt{3}(\gamma+1)} \frac{(\mathrm{Ma}^2-1)^{3/2}}{\mathrm{Ma}^2}, \tag{28}$$

where Ma is the free-stream Mach number and as before γ is the heat-capacity ratio. If the actual

wedge angle is smaller than $a_{\max}$, the shock is attached and oblique. The geometry relevant to Eq. 28 is depicted in Fig. 7.

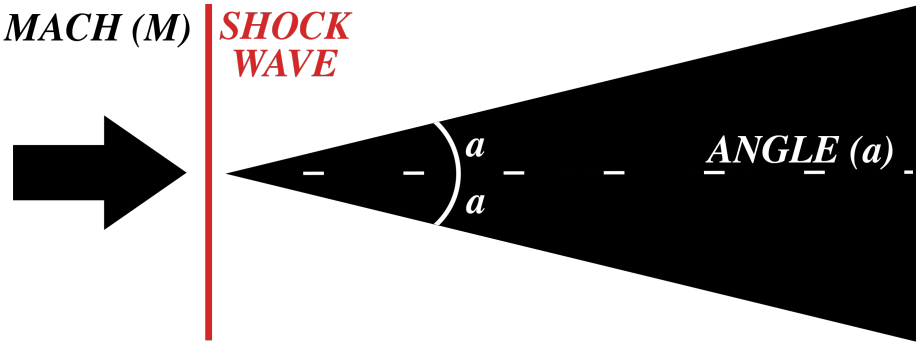


Figure 7: Detached Normal Shock for Flow Over a Wedge.

For demonstrating that Pronghorn is capable of modeling supersonic wedge flows, a test problem described in Ref. [18] is used. The problem domain is shown in Fig. 8. The wedge angle is 23°; the overall length in the x-direction is 0.0123 meters separated into three equal parts: straight segment before the wedge, the wedge, and the straight segment after the wedge; and the height is 0.06 meters. The fluid is air modeled as an ideal gas with a specific heat-capacity ratio of $\gamma = 1.4$ and a specific ideal gas constant of $R = 287$ J/kg K. Pronghorn results for this test problem are compared to solutions obtained with the commercial CFD software, Star-CCM+ (v. 14.04.13-R8) [19] and the open source CFD library OpenFOAM [20].

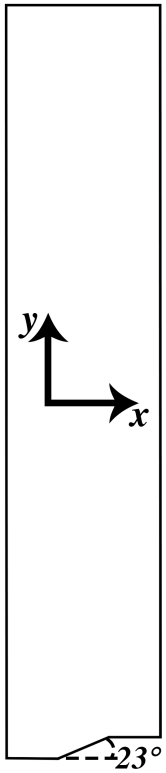


Figure 8: Representative 2D Wedge Geometry

The application of boundary conditions is visualized in Fig. 9: Following the benchmark, the inlet temperature and pressure are fixed at $T = 145.77$ K and 81.1 kPa, respectively, the speed of sound is determined to be 242.01 m/s. The boundary conditions on the left-inlet boundary are chosen to create a Mach number of $M = 1.5$ resulting in an inlet velocity of 363.02 m/s along the x-axis. The boundary conditions of the wedge-shaped obstacle are wall-boundary conditions. The remaining boundaries are modeled as supersonic outlets in Pronghorn and using the "Outlet," "Free Stream," and "Symmetry" boundary conditions in Star-CCM+ and OpenFOAM on the identically named parts of the boundary.

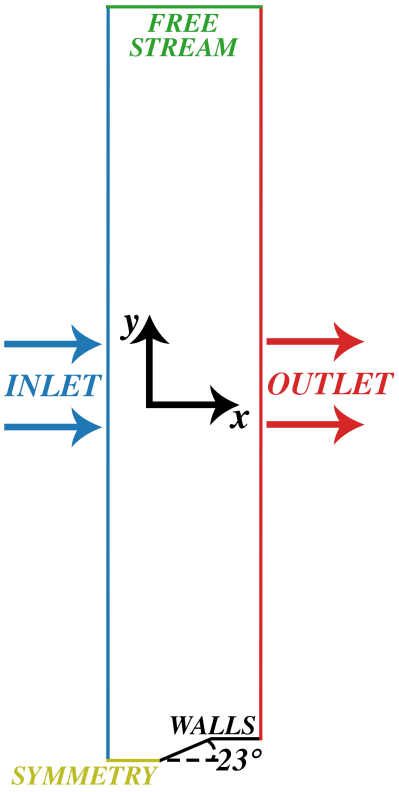


Figure 9: 2D Wedge Boundary Conditions.

Using Eq. 28 and the Mach number of 1.5, we find that the wedge angle of 23° is larger than $a_{\max}$. Consequently, the test problem is expected to develop a detached normal shock.

Star-CCM+ is capable of performing 2D simulations of inviscid ideal gases with an explicit unsteady coupled energy solver. The explicit unsteady solver and coupled explicit energy solver enables the user to define the desired Courant number in order for Star-CCM+ to determine the ideal timestep size. Additionally, coupled flow solvers within Star-CCM+ were chosen to capture sudden variation in density caused by the presence of shockwaves. The OpenFOAM settings for performing the benchmark calculations are adopted from Ref. [18].

For Pronghorn and Star-CCM+, a mesh for the wedge geometry was generated separately. As shown in Fig. 10, both meshes generated for Pronghorn (a) and Star-CCM+ (b) consist of 8000 quadrilateral elements uniformly distributed across the geometry. With 200 divisions vertically and 30 divisions horizontally, the mesh was created with equivalent element sizes across the domain to adequately capture the shockwave. OpenFOAM uses a similar mesh that is not shown in Fig. 10.

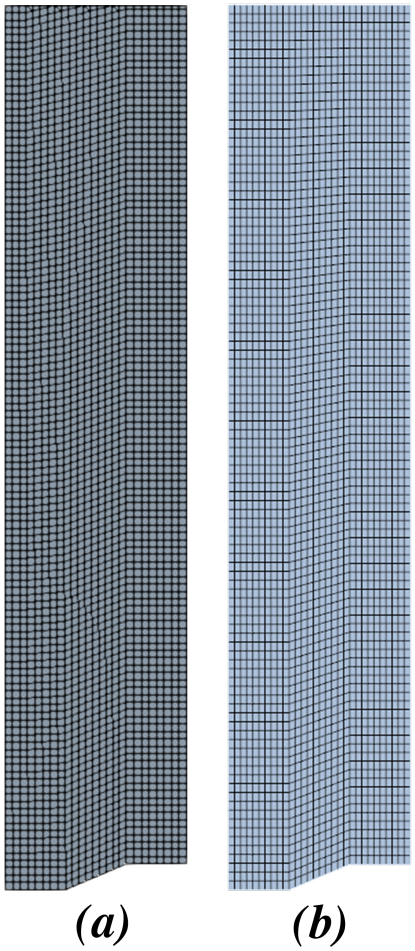


Figure 10: Generated mesh with uniformly distributed quadrilateral elements for (a) Pronghorn and (b) Star-CCM+.

Flow over a wedge is modeled in all codes with the aforementioned adaptive time step for the Courant number to retain a value of 0.5. The simulation is executed to a maximum time of 0.0015 s, because the detached normal shock is sufficiently close to a steady state at this time. A qualitative comparison is performed for the density and pressure predictions of Pronghorn, OpenFOAM, and Star-CCM+. Figure 11 compares the density map computed by Pronghorn (a), OpenFOAM (b) and Star-CCM+ (c), while Fig. 12 shows the pressure map. The overall structure of the density and pressure map agrees well between Pronghorn, OpenFOAM and Star-CCM+. In particular, the shocks are predicted at similar locations. In addition, the magnitude of the pressures and densities in the regions delineated by the shocks matches quite well. As the numerical methods used in the three codes differ, difference are expected on the relatively coarse mesh used for this comparison. Further investigation is warranted for the shock location predicted by Pronghorn when compared with OpenFOAM and Star-CCM+. The latter two codes' predicted shock location is much closer to each other than the Pronghorn shock location.

There is no definite reference solution for the described wedge flow problem and no mesh refinement study was performed to investigate if Pronghorn, OpenFOAM, and Star-CCM+ converge to the same answer. In addition, Star-CCM+ and OpenFOAM have the lion's share of their application in subsonic flow problems so that neither could outright be considered the gold standard

in supersonic flow simulations. At this point, we would like to stress that the initial quantitative comparison of Pronghorn with OpenFOAM and Star-CCM+ is encouraging but further results would need to demonstrate fitness of Pronghorn for analysis of supersonic flow.

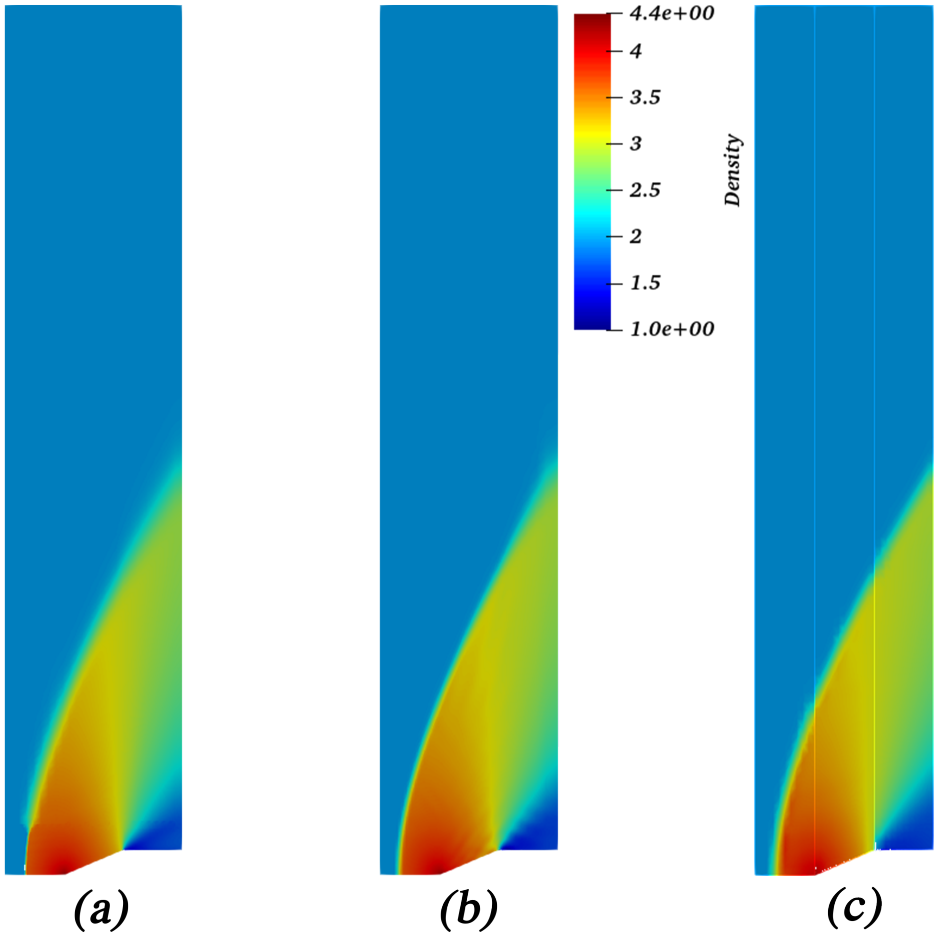


Figure 11: Density contours for (a) Pronghorn, (b) OpenFOAM and (c) Star-CCM+

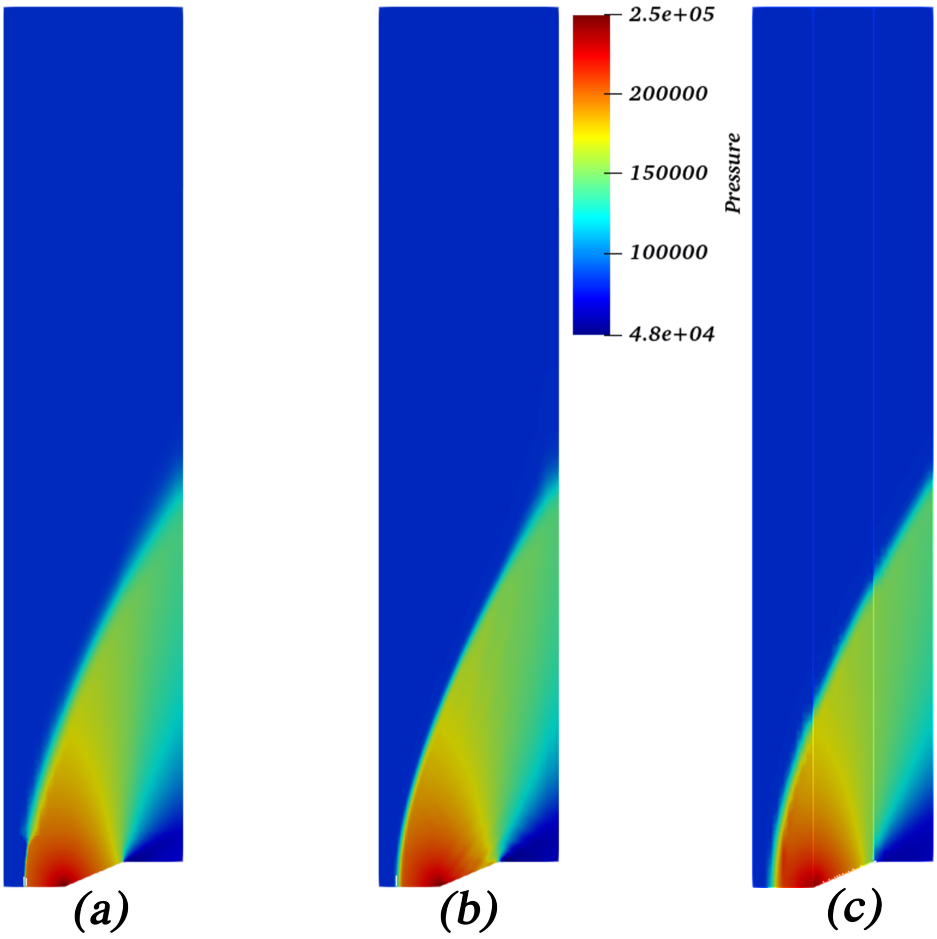


Figure 12: Pressure contours for (a) Pronghorn, (b) OpenFOAM and (c) Star-CCM+

5 Conclusions

Finite volume method support has recently been added to the multiphysics object-oriented simulation environment (MOOSE). In the MOOSE application Pronghorn, we implemented a discretization for the compressible Euler equations based on the FV method and the HLLC Riemann solver. The purpose of this discretization is to form the basis for a robust coarse-mesh thermal-hydraulics capability for all-speed compressible flows in Pronghorn.

The finite-volume HLLC method is tested using three well-known test cases: Sod's shock tube, a supersonic converging/diverging nozzle, and a supersonic flow over a wedge-shaped obstacle. Results are compared to analytical references for the first two test problems and against numerical reference solutions obtained using Star-CCM+ and OpenFOAM for the third. For the Sod shock-tube case, the FV HLLC method demonstrates much better agreement with the analytical solution than Pronghorn's continuous FEM implementation. For the supersonic nozzle and supersonic flow over wedge test cases, the FEM-based Euler discretization is unable to obtain solutions. In contrast, the Pronghorn FV HLLC implementation robustly handles both cases and its results match well with the respective reference solutions.

References

[1] A. Novak, R. Carlsen, S. Schunert, P. Balestra, R. Slaybaugh, and R. Martineau, "Pronghorn: A multidimensional coarse mesh application for advanced reactor thermal-hydraulics," *Nuclear Technology*, vol. submitted, 2020.

[2] S. Schunert, D. Andrš, P. Balestra, R. Carlsen, C. Permann, and R. Martineau, "NEAMS Progress Report on Coupling of Pronghorn and RELAP-7 for Very High Temperature Reactor (VHTR) Applications," Tech. Rep. INL/EXT-19-54495, Idaho National Laboratory, 2019.

[3] P. Balestra, S. Schunert, R. Carlsen, A. Novak, M. DeHart, and R. Martineau, "PBMR-400 Benchmark Solution of Exercise 1 and 2 Using the MOOSE Base Applications: MAMMOTH, Pronghorn," in *Proceedings of PHYSOR*, 2020.

[4] C. J. Permann, D. R. Gaston, D. Andrš, R. W. Carlsen, F. Kong, A. D. Lindsay, J. M. Miller, J. W. Peterson, A. E. Slaughter, R. H. Stogner, and R. C. Martineau, "MOOSE: Enabling massively parallel multiphysics simulation," *SoftwareX*, vol. 11, p. 100430, 2020.

[5] R. LeVeque, *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[6] S. Pope, *Turbulent Flows*. Cambridge University Press, 2000.

[7] J. J.D. Anderson, *Computational Fluid Dynamics - The Basics with Applications*. McGraw Hill, Inc., 1995.

[8] E. Toro, M. Spruce, and W. Speares, "Restoration of the contact surface in the HLL-Riemann solver.," *Shock Waves*, vol. 4, p. 100430, 1994.

[9] S. K. Godunov and I. O. Bohachevsky, "Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics," 1959.

[10] P. Roe, "Approximate riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, vol. 43, no. 2, pp. 357 – 372, 1981.

[11] E. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics - A Practical Introduction*. Springer, 2009.

[12] P. Roe, "Characteristic-based schemes for the Euler equations," *Annual Review of Fluid Mechanics*, vol. 18, pp. 337 – 365, 1986.

[13] B. Einfeldt, C.-D. Munz, P. L. Roe, and B. Sjögreen, "On Godunov-type methods near low densities," *Journal of computational physics*, vol. 92, no. 2, pp. 273–295, 1991.

[14] G. A. Sod, "A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws," *Journal of Computational Physics*, vol. 27, no. 1, pp. 1 – 31, 1978.

[15] Y. Xia, J. E. Hansel, R. A. Berry, D. Andrs, and R. C. Martineau, "Preliminary study on the suitability of a second-order reconstructed discontinuous galerkin method for RELAP-7 thermal-hydraulic modeling," Tech. Rep. INL/EXT-17-43108, Idaho National Laboratory, 2017.

[16] J. Z. Takayama, K., "Shock wave reflection over wedges: a benchmark test for CFD and experiments," *Shock Waves*, vol. 7, 1997.

[17] W. Fickett and W. Davis, *Detonation: Theory and Experiment*. Dover Books on Physics, Dover Publications, 2012.

[18] curiosityFluids, "Mach 1.5 flow over 23 degree wedge – rhoCentralFoam," 2020.

[19] S. CD-adapco, "STAR-CCM+ User Guide Version 14.04," *CD-Adapco: New York, NY, USA*, 2019.

[20] G. Chen, Q. Xiong, P. Morris, E. Paterson, A. Sergeev, and Y. Wang, "OpenFOAM for computational fluid dynamics," *Notices of the American Mathematical Society*, vol. 61, 2014.