LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-405350

Prototype Mixed Finite Element Hydrodynamics Capability in ARES

R. N. Rieben

July 14, 2008

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Prototype Mixed Finite Element Hydrodynamics Capability in ARES *

Robert N. Rieben †

July 9, 2008

Abstract

This document describes work on a prototype Mixed Finite Element Method (MFEM) hydrodynamics algorithm in the ARES code, and its application to a set of standard test problems. This work is motivated by the need for improvements to the algorithms used in the Lagrange hydrodynamics step to make them more robust. We begin by identifying the outstanding issues with traditional numerical hydrodynamics algorithms followed by a description of the proposed method and how it may address several of these longstanding issues. We give a theoretical overview of the proposed MFEM algorithm as well as a summary of the coding additions and modifications that were made to add this capability to the ARES code. We present results obtained with the new method on a set of canonical hydrodynamics test problems and demonstrate significant improvement in comparison to results obtained with traditional methods. We conclude with a summary of the issues still at hand and motivate the need for continued research to develop the proposed method into maturity.

1 Introduction and Motivation

This work is motivated primarily by the desire to make the numerical algorithms which are used in solving the equations of hydrodynamics (namely the inviscid Euler equations) in a Lagrangian frame more robust with respect to grid motion. Our goal is to improve the Lagrangian hydrodynamics algorithms to prevent spurious grid distortions and more importantly, to eliminate artificial symmetry breaking in problems with irregular / non-orthogonal grids.

Originally, this work was driven by the desire to obtain improved treatment to the so called "hourglass" (a.k.a. checkerboard) instabilities which are a common plague among traditional staggered-grid hydrodynamics (SGH) codes. These instabilities are triggered by a point (or delta function) source and occur at the highest spatial frequency supported by the computational grid (namely a single zone or node). They do not go away as the grid is refined and unless they are damped (or removed) by a post-processing "filter" step, they have the potential to grow unchecked as the numerical solution is evolved over time, oftentimes leading to unacceptable Lagrangian grid distortion and eventual termination of a calculation. An example of such an instability is given in Figure 1. The standard solution to this problem is to counteract the spurious accelerations that these modes produce by introducing an "anti-hourglass" force [1]. This simple and efficient method does a reasonable job of keeping these instabilities in check; but this approach is really just a "fix-up" and does not address the fundamental source of the instability. Recently, the method of sub-zonal pressures has been introduced to address this problem [2]. During the early stages of this effort, the method of sub-zonal pressures was investigated as a potential candidate for improving robustness. It was discovered that while this method does appear to suppress certain forms of hourglass instabilities, it is not clear whether this method is suppressing actual physical modes. Furthermore, it was discovered that this method introduces an artificial divergence-free vorticity mode. While it is clear that hourglass instabilities can lead to spurious grid distortion, it is not always clear that they are the sole cause of spurious grid distortion. It is tempting to attribute any spurious grid distortion in a given calculation to "hourglass modes", making them the default scapegoat for a wide variety of mesh instabilities; but in actuality, there are a variety of sources for such spurious grid distortions when using a traditional SGH method in a typical shock hydrodynamics problem (see Figure 2).

*This document describes work for an L3 milestone

†Lawrence Livermore National Laboratory, rieben1@llnl.gov

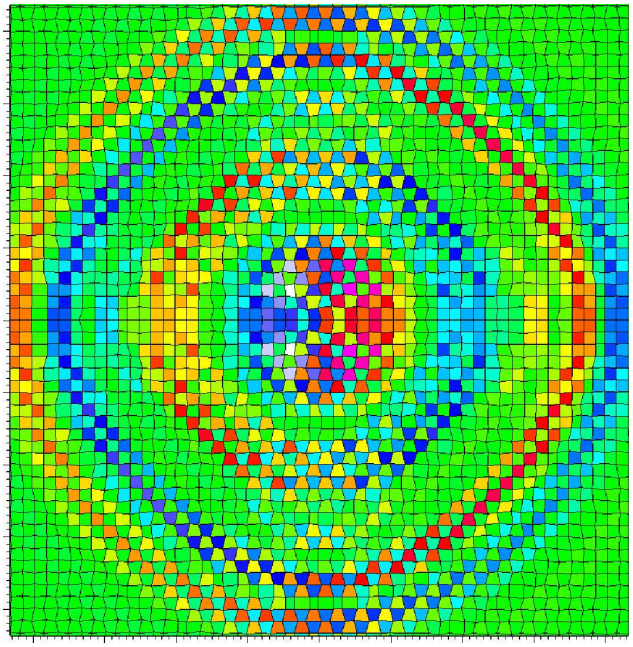

Figure 1: Example of a "checkerboard" instability in the pressure field and corresponding "hourglass" instability in velocity field; excited by applying a time dependent perturbation to a single node at the center. Instability exists at highest spatial frequency of the underlying grid irrespective of mesh resolution and is a result of a fundamental instability in the discrete representation of velocity, pressure and the spatial differential operators which relate them.

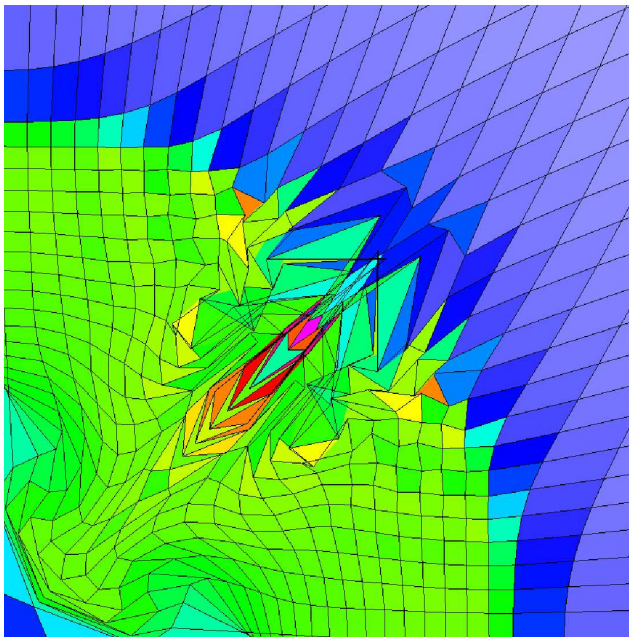

Figure 2: Example of spurious grid distortion encountered when applying standard SGH methods to the Noh problem on an initially orthogonal mesh. There are multiple sources of this grid distortion including hourglass instabilities, inaccuracies in the pressure gradient operator and discretization of the artificial viscosity term. Each of these errors can amplify each other over time, leading to a rapid tangling of the grid.

Perhaps the most outstanding issue with traditional SGH methods is the need to improve numerical symmetry preservation in problems where some degree of symmetry is present (e.g. spherical symmetry), but the underlying computational mesh is not aligned with the shock flow. It is well known that traditional SGH methods break down when the underlying computational mesh is distorted; this limitation is one of the primary motivations for the development of Arbitrary Lagrangian-Eulerian (ALE) methods which keep the computational grid as smooth as possible. In a typical SGH Lagrange step, kinematic vector fields (e.g. acceleration, velocity) are computed at mesh nodes due to a collection of cell centered thermodynamic variables (e.g. pressure, energy, density) [3], [4]. For example, the force acting on a given mesh node due to the gradient of a pressure field is computed using a "control volume" finite differencing technique as illustrated in Figure 3. This simple approach is extremely efficient, but is valid *only* at the center of mass of the control volume. For "well behaved" grids, the control volume center of mass and the node to be accelerated are coincident; however, as the grid is distorted, the control volume center of mass and the mesh node are no longer coincident. This causes artificial "torque" on the node, resulting in spurious, non-physical mesh motion.

The method of artificial viscosity, as originally introduced by [5], is still the most popular method for treating shock waves. The current state of the art is to use a Van Leer type "monotonic limiter"[6] in conjunction with an artificial viscosity to keep the artificial diffusion length of the shock front to a minimum while preventing spurious "ringing" (a.k.a. overshoots and undershoots). The application of such monotonicity limiters is closely related to the use of Riemann solvers to track shock discontinuities. For 1D calculations or cases where the computational grid is perfectly aligned with the flow, such methods give very good results. For general meshes which are not aligned with the flow (in 2D, *r-z* and 3D), such methods are known to cause symmetry breaking (e.g. the Noh problem run on an initially square mesh as shown in Figure 2). It is clear that the artificial viscosity formulation is the biggest (though not the only) culprit in breaking symmetry when shock waves pass over non-aligned grids. Based on numerical evidence, the use of monotonic limiters seems to amplify the symmetry breaking for shock waves passing over non-aligned grids

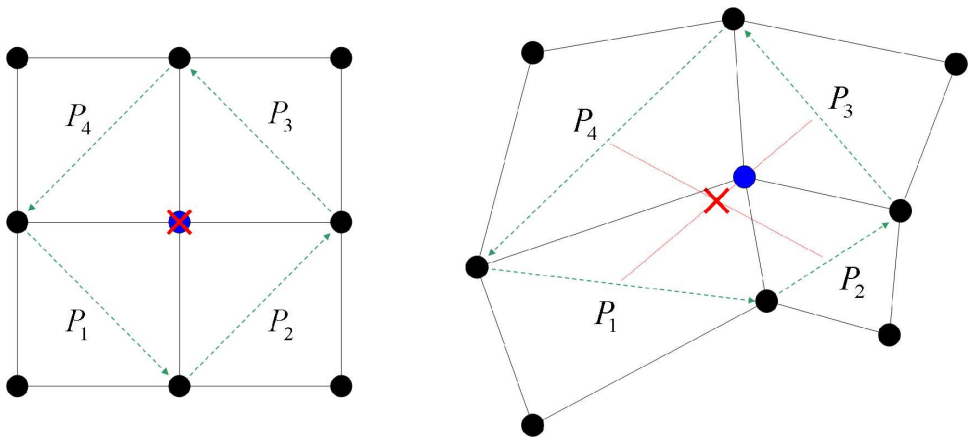

Figure 3: Schematic depiction of the traditional control volume discretization of the pressure gradient operator for computing the force acting on a node. On a well behaved grid the control volume center of mass is coincident with the node we wish to accelerate (*left*). However, as the grid is distorted, the control volume center of mass and nodal coordinate are no longer coincident (*right*.)

(see for example [7] and [8]). Furthermore, the artificial viscosity is typically implemented as a scalar term (with units of pressure) which is added to the bulk pressure in the hydro equations and subsequent internal energy equations. This approach is valid only for the special case of non-vortical flows (i.e. it ignores a shear term). The more general case requires a so called "tensor viscosity" (see for example [9]). Another way of stating this is that the artificial viscous force term involves the $Div(Grad)$ operator acting on the velocity field.

Traditional SGH methods approximate the divergence of the velocity as a simple change in volume with respect to time. This approximation is used to compute the change in internal energy of a given zone due to the kinematic work done on it during the Lagrange step. This approach is referred to as a *PdV* scheme, since the change in energy during the Lagrange step is computed as the pressure, P, multiplied by the change in volume, dV. The change in kinetic energy as computed from the discretization of the pressure gradient is not equal and opposite to this *PdV* based change in internal energy for a given time step, because the discretizations used for *Grad* and *Div* are not compatible with each other. The error in energy conservation for these schemes is therefore bounded by Δt (the discrete time step) for planar (x-y) and for 3D (x-y-z) calculations and will converge to zero under time refinement. For axisymmetric (r-z) calculations, there is another component to this error which is due to the incorrect assumption that the divergence of the velocity is constant in a zone (see [8] for more details) and this error will *not* converge to zero under time or space refinement. Conservation of total numerical energy is required to guarantee convergence / consistency as problem is refined in time and space.

To summarize, the key issues with respect to traditional low-order SGH methods are:

- Inability to calculate accurate pressure gradients on distorted meshes.
- Inability to preserve symmetry in problems where shock wave propagation is not aligned with the computational mesh.
- Presence of high frequency "checkerboard modes" in pressure (resulting in "hourglass modes" in the velocity) that lead to instabilities in time dependent calculations.
- Inability to conserve total energy (kinetic plus internal) algebraically (i.e. to machine precision), especially in axisymmetric coordinates.

We are currently investigating / advocating the use of high order mixed finite element methods for tackling some of these long standing issues. Mixed finite element methods are a promising alternative to traditional staggered grid (finite difference, finite volume based) methods for discretizing the first order spatial differential operators in multi-dimensions, namely *Grad*, *Curl* and *Div* as well as the second order spatial differential operators $Grad(Div)$, $Curl(Curl)$ and $Div(Grad)$. The MFEM excels at

discretizing equations that involve these operators even when the underlying grid geometry (i.e. mesh) is very distorted. This means we can calculate accurate and symmetry preserving pressure gradient operators even on highly distorted grids. We also feel that the key to removing the checkerboard / hourglass instability lies in the use of a *stable* mixed finite element basis function pair for discretizing the pressure and velocity (for more on stability in FEM, see [10]). It is now well known that the bilinear velocity-constant pressure basis (a.k.a. Q1-Q0 on quadrilateral / hexahedral elements, staggered grid hydro) is an unstable mixed finite element pair. Hourglass modes are now known to be caused by the inability of the discrete *Grad* operator (which defines the spatial relationship between velocity and pressure) to capture the null space of the continuum *Grad* operator (i.e. the set of all constants) at the grid level.

There are multiple sets of MFEM basis functions which satisfy the stability condition. From a practical standpoint, there are many things we must consider when choosing a mixed finite element basis pair for hydrodynamics:

- Where are the velocity degrees of freedom defined?
- Where are the pressure degrees of freedom defined?
- What spatial differential operators will we be using?

In a typical hydrodynamics code, all thermodynamic variables (e.g. pressure, energy, density) are discretized as piece-wise constant values at zone centers. To remain consistent with the current methods in the ARES code, we would like to keep our discrete thermodynamic fields as piece-wise constant zone-centered quantities. Therefore, in this work, we utilize so called "divergence conforming" or $H(Div)$ mixed methods. In this approach, the pressure is piece-wise constant in a zone while the velocity is discretized on mesh faces (edges in 2D) using a divergence conforming basis set where the degrees of freedom (i.e. the unknowns) are the normal projections of the velocity on mesh faces. We have developed and implemented both a low order (Raviart-Thomas [11]) and a high order (Brezzi-Douglas-Marini [12]) version of this $H(Div)$ approach for Lagrangian hydrodynamics. Such $H(Div)$ methods require the assembly and solution of a global sparse linear system to solve for the velocity unknowns (the normal projections of the velocity at element interfaces) and this has required the addition of some new software to interface with the HYPRE linear solvers library [13]. The use of an $H(Div)$ conforming basis allows us to generate a discrete version of the *Grad* operator as a sparse, rectangular matrix which maps our piece-wise constant pressure field to the discrete face based velocity field which automatically has the correct range and null spaces. The transpose of this rectangular matrix is the adjoint of the *Grad* operator, namely a discrete version of the *Div* operator (similar to the support operators method of [14]). This natural approach to discretizing *Grad* and *Div* in a consistent manner allows us to preserve conservation of total energy at every Lagrange step in contrast to the non-conservative discretizations used in a traditional SGH method.

For Lagrangian hydrodynamics, a velocity field must somehow be computed at nodes in order to move the computational grid at each discrete time step. The $H(Div)$ MFEM solves for the normal projections of velocity at mesh faces; and once these are known, we can use the velocity basis functions to evaluate the velocity field at any point in a given zone. However, by construction, the $H(Div)$ velocity basis functions only enforce normal continuity of the velocity across zone interfaces, and so there is no guarantee that the velocity field at mesh nodes will be unique. This is an unfortunate downside to such $H(Div)$ methods in the context of Lagrangian hydrodynamics – it means we need some form of "averaging" or interpolation to convert our face based representation of velocity into a node based representation. The process by which the face based velocity unknowns are computed using an $H(Div)$ MFEM are naturally free of hourglass modes (i.e. they are always orthogonal to the set of discrete divergence free velocity modes); however, the process by which nodal velocity fields are computed is not constrained in this manner and therefore some form of hourglass projection / damping is still required.

For treating shock wave propagation, we advocate the traditional approach of adding a monotonically limited artificial viscosity term to the Euler equations. We believe that the key to preserving symmetry in problems where shock waves propagate through meshes that are not aligned with the flow lies in the proper discretization of this viscous term (and careful treatment of the limiter in multi-dimensions) as well as the pressure gradient operator. In this work, we consider only the case of a scalar artificial viscosity (i.e. a scalar q which is simply added to the bulk pressure P in the hydro equations) and show how the $H(Div)$ MFEM can be used to make improvements to this traditional approach. However, we emphasize that this approach is valid only for the special case of non-vortical flows (i.e. it ignores a shear term). The more general case requires a so called "tensor viscosity" and will therefore require a discrete version of the $Div(Grad)$ operator acting on the velocity field. Another drawback to the $H(Div)$ approach is

that such an operator is not well defined for this function space – again, this is because the fundamental assumption of working in an $H(Div)$ space is that only the normal components of the velocity are continuous across element boundaries; a $Div(Grad)$ operator requires that all components (tangential and normal) of the velocity field be continuous across element boundaries. This realization along with the previous discussion concerning the need of obtaining hourglass-free nodal fields suggests that an $H(Div)$ method is not the most appropriate MFEM basis to use and that perhaps a different choice of basis functions would yield even better results; however, it is not yet clear which choice is the best for Lagrangian shock hydrodynamics. We will discuss this idea in greater detail at the end of this document in Section 5.

2 Theoretical Discussion

In this section we introduce the set of continuum partial differential equations (PDEs) we are interested in solving, namely the inviscid Euler equations in a Lagrangian (a.k.a. co-moving, material) reference frame. We describe the key properties that these equations posses and their connection to the physical principles from which they are derived. We begin the discretization by first describing what we mean by "Mixed Finite Element Methods." We cast the relevant equations in variational form and apply a rigorous Galerkin procedure to reduce our continuum equations to a set of semi-discrete (i.e. no time discretization is yet applied) ordinary differential equations (ODEs).

2.1 Notational Conventions

Before we begin our discussion of the equations, we introduce the notational conventions that will be used throughout this document. For *continuum* fields, we will adopt the conventions as outlined in Table 1. All vector fields will be designated with an arrow, with individual spatial components denoted with subscripts x and y (for 2D).

Notation / Symbol	**Description**
$\vec{x} = \{x, y\}$	Lagrangian coordinate
$\vec{v} = \{v_x, v_y\}$	Velocity vector field
$\vec{a} = \{a_x, a_y\}$	Acceleration vector field
$\vec{f} = \{f_x, f_y\}$	Force vector field
ρ	Density (mass/volume)
m	Mass
V	Volume
e	Internal energy per mass
P	Scalar pressure
KE	Kinetic energy
IE	Internal energy
E	Total energy (kinetic plus internal)
Ω	Spatial domain
$\partial\Omega$	Boundary of spatial domain
$\hat{n}$	A boundary normal vector

Table 1: Notational conventions used for *continuum* variables

As we introduce the MFEM discretization process and the notion of *discrete* fields, we will use the notation conventions as outlined in Table 2. For this case, we will use **bold faced** letters to denote "arrays" (i.e. matrices and vectors) where capital letters will denote matrices and lower case letters will denote vectors. We will use a superscript n to designate a discrete time step. Subscripts will be reserved for indexing spatial quantities. We will adopt the Burton-Caramana-Shashkov notation of [15] and [16] for describing "nodal" values (subscript p) and "zonal" values (subscript "z"). We also introduce the notion of a discrete

basis for vector fields (e.g. a discrete velocity basis), which we will denote as $\vec{w}_i$. The subscript in this case will be a a "degree of freedom" (DOF) index, i.e. the number of unique degrees of freedom in the basis function expansion.

Notation / Symbol	Description
Δt	A discrete unit of time (a.k.a. "time step")
Ω_z	A discrete volume of space (a.k.a. "zone")
$\vec{w}_i$	A discrete basis for vector fields
ϕ_i	A discrete basis for scalar fields
$\_^n$	A superscript denoting a discrete integer time step
$\__z$	A subscript denoting a particular "zonal" value
$\__p$	A subscript denoting a particular point (a.k.a "nodal") value
$\__{i,j}$	Subscripts denoting "degree of freedom" indices for inner products
$\hat{\_}$	An accent denoting definition with respect to a reference coordinate system
$\mathbf{J}_z$	The Jacobian matrix for zone z
$\mathbf{M}_z$	The "mass" matrix for zone z
$\mathbf{S}_z$	The "stiffness" matrix for zone z
$\mathbf{D}_z$	The "derivative" matrix for zone z
$\mathbf{v}_z$	The velocity DOF vector for zone z
$\mathbf{p}_z$	The pressure DOF vector for zone z
$\mathbf{M}$	The global (assembled) "mass" matrix
$\mathbf{S}$	The global (assembled) "stiffness"
$\mathbf{D}$	The global (assembled) "derivative" matrix
$\mathbf{v}$	The global (assembled) velocity DOF vector (array)
$\mathbf{p}$	The global (assembled) pressure DOF vector (array)

Table 2: Notational conventions used for *discrete* variables

2.2 The Euler Equations in a Lagrangian Frame

The equations of hydrodynamics in a Lagrangian (a.k.a. co-moving or material) reference frame are described by a set of four unknowns: velocity $\vec{v}$, density ρ, internal energy e, and pressure P; and a set of four equations, known as the Euler equations:

$$\text{Momentum Conservation:} \quad \rho \frac{\partial \vec{v}}{\partial t} = -\vec{\nabla} P \tag{1}$$

$$\text{Mass Conservation:} \quad \frac{1}{\rho} \frac{\partial \rho}{\partial t} = -\vec{\nabla} \cdot \vec{v} \tag{2}$$

$$\text{Energy Conservation:} \quad \rho \frac{\partial e}{\partial t} = -P \vec{\nabla} \cdot \vec{v} \tag{3}$$

$$\text{Equation of State:} \quad P = EOS(\rho, e) \tag{4}$$

These equations describe the interplay between kinematics and thermodynamics for a collection of materials distributed over a given spatial domain Ω. The connection between these two physical processes in a given material is the so called Equation of State (EOS). The EOS is a material response function which determines the net pressure exerted by a differential volume of a certain material as a function of how much mass it contains (its density) and its internal energy. We define the total mass in the spatial domain Ω to be

$$m \equiv \int_\Omega \rho \tag{5}$$

Furthermore, the total kinetic energy in the spatial domain Ω is defined as

$$KE = \frac{1}{2} \int_\Omega \rho \vec{v} \cdot \vec{v} \tag{6}$$

The total internal energy in the spatial domain Ω is defined as

$$IE = \int_{\Omega} \rho e \tag{7}$$

The total energy contained in the spatial domain Ω is thus

$$E = KE + IE \tag{8}$$

If the spatial domain Ω contains no energy sources (or sinks) and there is no flux of energy and/or mass out of the boundary of the domain, $\partial\Omega$, then the total energy contained in Ω is a constant for all time

$$\frac{\partial E}{\partial t} = 0 \tag{9}$$

2.3 The Compatible Discretization Philosophy

Our goal is to solve the continuum equations of (1) - (4) by replacing the continuum differential operators and fields which are valid for all points in space and time, with discrete analogues which are valid on discrete volumes (a.k.a zones) and discrete points in time (a.k.a. time steps). If our discretization process is consistent and convergent, then our discrete solution will converge to the continuum solution in the limit that the discrete volumes go to zero (i.e. mesh refinement) and the discrete time steps go to zero (i.e. time refinement). The notion of "compatibility" extends the idea of consistency and convergence by adding additional constraints / requirements to the numerical discretizations based on properties of the corresponding continuum partial differential equations (PDEs). A compatible (a.k.a. "mimetic") spatial discretization method is said to "inherit or mimic fundamental properties of the continuum PDE such as topology, conservation, symmetries and ... maximum principles [10]." Such methods have proven highly successful in several fields in computational physics, including electromagnetics [17], [18], incompressible flow [19] and magnetohydrodynamics [20]. The idea is becoming popular in numerical shock hydrodynamics. In simple terms, the goal of compatibility is to reproduce all of the salient continuum features of differential equations and their corresponding differential operators (both time and space dependent) in a discrete sense. One example is energy conservation. The continuum equations we wish to solve conserve total energy, therefore our subsequent discrete equations should likewise conserve some discrete measure of total energy. Less obvious examples lie in the discretization of first order spatial differential operators like *Grad*, *Curl* and *Div* as well as second order spatial differential operators such as $Grad(Div)$, $Curl(Curl)$ and $Div(Grad)$. For this case, compatibility implies that continuum properties such as $\vec{\nabla} \times (\vec{\nabla} \vec{f}) = 0$ for all vector fields $\vec{f}$ must be reproduced by the discrete differential operators. While seemingly abstract, this property is essential for guaranteeing stability and eliminating certain "spurious" modes.

2.4 Introduction to Mixed Finite Element Methods

What do we mean by Mixed Finite Elements? Loosely speaking, any method which uses multiple sets of basis functions (e.g. bilinear velocity + constant pressure) can be thought of as a mixed method. In this context, we can view the traditional SGH method as a mixed method (in fact, mixed methods are a rigorous, high order generalization of the staggered-grid concept to arbitrary unstructured grids). However, there are several fundamental properties that a true MFEM satisfies:

- We follow the rigorous definition of a finite element as originally set forth by Ciarlet [21]. This means we have concrete definitions for our finite element space (and subsequent basis functions), our degrees of freedom, and our element topology and geometry. For more details on how this process works in a practical setting, see [22].
- Accurate numerical quadrature over zones is key to avoid "variational crimes." This means that we perform all of our zone based integrals using high order quadrature rules. The concept of "nodal mass" that is used in traditional SGH methods is equivalent to mass-lumping, i.e. under integrating the element mass matrix and we feel this contributes to spurious grid distortion in Lagrangian hydro.

- Galilean invariance is achieved through proper discretization in arbitrary curvilinear coordinate systems. This means we build the metric tensor (i.e the Jacobian matrix) directly into our discretizations which removes the dependence of our numerical solution on the regularity of the mesh.
- Improved accuracy and stability is achieved by using higher order methods (e.g. quadratic basis functions for velocity).
- In general this means we need to solve sparse linear systems to obtain our solution unknowns. This is troubling to some, due to the computational cost involved in assembling and solving a linear system at each Lagrange hydro time step. However, for the sake of obtaining improvements in accuracy, we feel it is justified; certainly at this early stage of development. Future work will investigate means of reducing this computational effort.

2.5 Spatial Discretization

We begin by decomposing the spatial domain Ω into a set of non overlapping, discrete volumes called zones (a.k.a. elements). The union of these discrete zones forms the computational mesh, which we will denote as $\tilde{\Omega}$, and is defined as

$$\tilde{\Omega} \equiv \sum_z \Omega_z \tag{10}$$

Note that the computational mesh $\tilde{\Omega}$ is a geometrical approximation to the true geometry of the continuum domain Ω. We now make the very important distinction between zone *topology* and *geometry*. The topology of the zone defines its connectivity, for example triangular zones vs. quadrilateral zones. In simple terms, the topology of a zone is defined by its number of unique "vertices" and how they are connected. The geometry of a zone is determined by the locations of these vertices in some coordinate space. For example, we typically consider the case where the discrete zones Ω_z are quadrilaterals (hexahedrons) formed by connecting straight lines to the vertices. However, when using a mixed finite element method, we are not restricted to this choice – we can use triangles (tetrahedrons). Furthermore, we are not restricted to using straight lines, we can readily incorporate curved surfaces provided we have higher order geometrical information (i.e. instead of straight lines connecting two vertices, we could use a quadratic function which passes through three vertices, see [17] for specific examples of this idea). The topology of our zone determines the explicit form of the basis functions we will use (see Section 2.6). The geometry of the zone (i.e. straight lines, curved lines, etc ...) determines the order of the local-to-global (a.k.a. parametric) mapping which defines the Jacobian matrix of the zone (see Section 2.7).

2.5.1 Discrete Momentum Conservation

We begin our spatial discretization process by applying a variational formulation to the momentum conservation equation. We do this by transforming the momentum equation into a weighted volume integral equation. We multiply (1) by some vector valued test function $\vec{w}$ and integrate over the finite element mesh $\tilde{\Omega}$ to obtain the variational form

$$\int_{\tilde{\Omega}} (\rho \frac{\partial \vec{v}}{\partial t}) \cdot \vec{w} = -\int_{\tilde{\Omega}} (\vec{\nabla} P) \cdot \vec{w} \tag{11}$$

Now we perform integration by parts on the right hand side of (11) and apply the Gauss divergence theorem to obtain

$$\int_{\tilde{\Omega}} (\rho \frac{\partial \vec{v}}{\partial t}) \cdot \vec{w} = \int_{\tilde{\Omega}} (\vec{\nabla} \cdot \vec{w}) P - \oint_{\partial \tilde{\Omega}} P(\vec{w} \cdot \hat{n}) \tag{12}$$

where $\hat{n}$ is the outward pointing unit normal vector of the surface $\partial\tilde{\Omega}$. Now we assume a piece wise polynomial representation of the fields $\vec{v}$ and P over the mesh $\tilde{\Omega}$ of the particular form

$$\vec{v}(\vec{x},t) \approx \sum_i v_i(t)\vec{w}_i(\vec{x}) \tag{13}$$

$$P(\vec{x},t) \approx \sum_i p_i(t)\phi_i(\vec{x}) \tag{14}$$

$$for\ \vec{x} \in \Omega_z$$

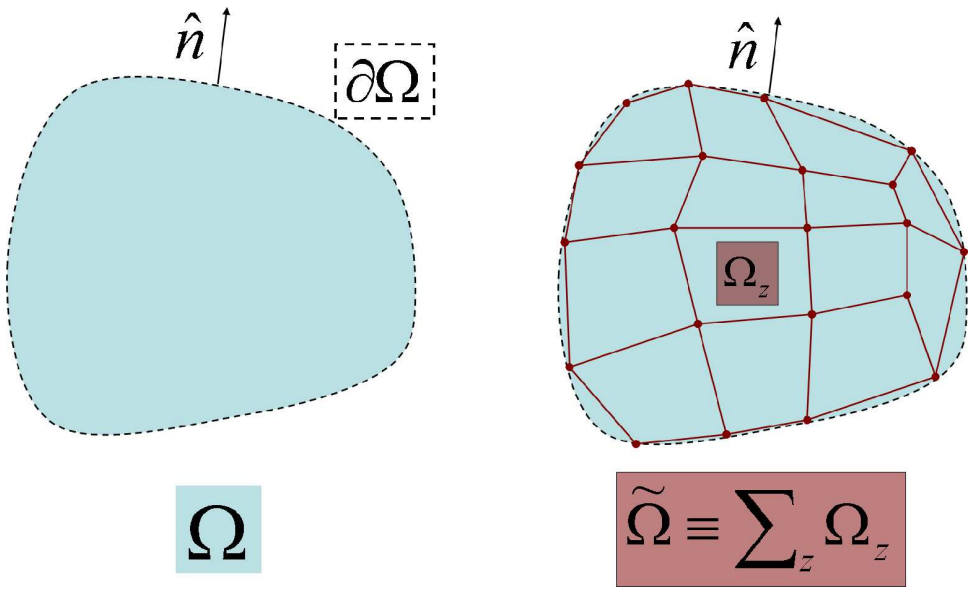

Figure 4: Schematic depiction of a continuum spatial domain Ω and its geometric approximation by means of a finite element mesh $\tilde{\Omega}$ consisting of a set of discrete quadrilateral volumes, or zones Ω_z.

Note that we have introduced a separation of variables in the basis function expansions of (13) and (14), where the coefficients of the expansion (a.k.a. "degrees of freedom" or DOF) depend only on time while the basis functions depend only on space.

As mentioned in Section 1, we would like the discrete representation of pressure to be a piece-wise constant zone centered value. For this simple representation, the basis function expansion for pressure in (14) takes on the specific form:

$$P(\vec{x},t) \approx p(t) \tag{15}$$

Therefore, for a given zone Ω_z, the basis function expansion of pressure is constant in space. Now we apply Galerkin's method to our variational formulation of (12), which means that we use our velocity basis functions $\vec{w}_i$ from (13) as our testing function. If we test against every basis function in the expansion and ignore the boundary intergal term, we obtain following linear system of ordinary differential equations (ODEs)

$$\int_{\tilde{\Omega}} (\rho \frac{\partial v_i}{\partial t})(\vec{w}_i \cdot \vec{w}_j) = \int_{\tilde{\Omega}} (\vec{\nabla} \cdot \vec{w}_j) P \tag{16}$$

Now we apply our piece-wise constant representation for pressure from (15) and pull all terms that are not spatially dependent out of the integrals to obtain

$$\frac{\partial v_i}{\partial t} \int_{\tilde{\Omega}} \rho(\vec{w}_i \cdot \vec{w}_j) = p \int_{\tilde{\Omega}} (\vec{\nabla} \cdot \vec{w}_j) \tag{17}$$

For clarity, we prefer to write the linear system of (17) in terms of matrices and vectors as

$$\mathbf{M} \frac{\partial}{\partial t} \mathbf{v} = \mathbf{D}^T \mathbf{p} \tag{18}$$

where $\mathbf{M}, \mathbf{D}, \mathbf{v}$ and $\mathbf{p}$ are global matrices and vectors (a.k.a. arrays) which are assembled over the entire mesh $\tilde{\Omega}$ from the contributions of each individual zone as

$$\begin{aligned} \mathbf{M} &= Assemble(\mathbf{M}_z) \\ \mathbf{D} &= Assemble(\mathbf{D}_z) \\ \mathbf{v} &= Assemble(\mathbf{v}_z) \\ \mathbf{p} &= Assemble(\mathbf{p}_z) \end{aligned}$$

The process of global assembly is analogous to the concept of "nodal accumulation" that is used in a traditional SGH code where a quantity at a node is defined to be the sum of contributions from all of the zones which share this node. For the case of $H(Div)$ basis functions (described in Section 2.6) which have unknowns defined on mesh faces, assembly is simply the process by which a quantity at a face is defined to be the sum of contributions from all of the zones which share that face. This process requires the notion of a global face indexing scheme and this is discussed in Section 3.4.

We define the "mass matrix" for zone z to be

$$(\mathbf{M}_z)_{i,j} \equiv \int_{\Omega_z} \rho(\vec{w}_i \cdot \vec{w}_j) \tag{19}$$

This matrix is symmetric positive definite (SPD) by construction, and has a dimension N_v by N_v, where N_v denotes the number of coefficients used in the basis function expansion for velocity of (13). We define the so called "derivative" matrix as

$$(\mathbf{D}_z)_{i,j} \equiv \int_{\Omega_z} (\vec{\nabla} \cdot \vec{w}_j) \tag{20}$$

This matrix is *rectangular* with dimension N_p by N_v, where N_p denotes the number of coefficients used in the basis function expansion for pressure of (14). Since we have already assumed simple piece-wise constants for pressure in (15), this means that $N_p = 1$ and so our derivative "matrix" is technically a vector, but we choose to write it in this form for the sake of generality. This rectangular derivative matrix is a map between the two discrete representations of velocity and pressure and is a discrete version of the *Div* operator. Its adjoint (or transpose) is a discrete version of the *Grad* operator as seen in (18).

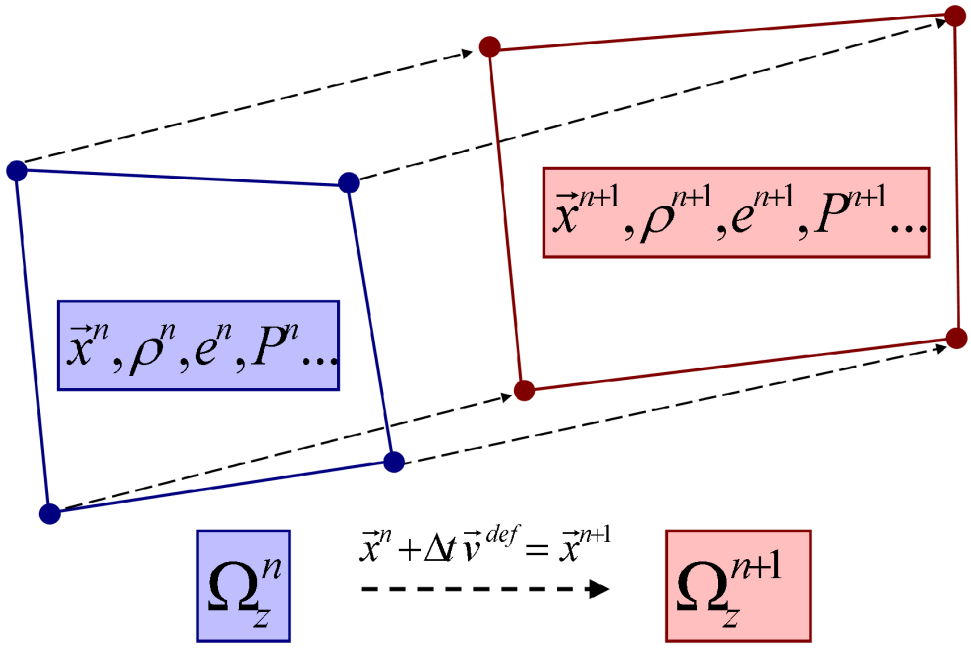

Figure 5: Schematic depiction of the velocity deformation field transforming a zone from one time step n to another at $n+1$ in a time of Δt.

In the Lagrangian description of hydrodynamics, our discrete mesh $\tilde{\Omega} \equiv \sum_z \Omega_z$ moves or "flows" with the material. This means that at some time t (denoted by the integer n) we have a certain geometrical configuration of mesh elements determined by the Lagrangian coordinates $\vec{x}^n$ (a.k.a vertices). The Lagrangian coordinates at time n determine the geometry of our zones Ω_z^n. A principle goal of the Lagrange hydro step is to compute the *accelerations* (due to a collection of various forces) which will cause the material to move a discrete distance $\Delta\vec{x}$ in a discrete amount of time Δt such that at time $t + \Delta t$ we have a different geometrical configuration Ω_z^{n+1}; see Figure 5 for a schematic depiction of this process.

The acceleration is defined as the instantaneous change in velocity

$$\vec{a} \equiv \frac{\partial \vec{v}}{\partial t}$$

We therefore define the velocity *deformation field* as

$$\vec{v}^{def} \equiv \vec{v}^n + \Delta t \vec{a} \tag{21}$$

where Δt is some discrete time step between states n and $n+1$. The acceleration field $\vec{a}$ therefore defines a differential change in velocity that is used to accelerate a zone from its current configuration Ω_z^n to a new configuration Ω_z^{n+1} in a discrete time step Δt. The deformation field is usually prescribed on the vertices of the mesh, but it is (implicitly) assumed that it can be extended to the whole domain. We define the discrete acceleration to be

$$\mathbf{a} \equiv \frac{\partial}{\partial t}\mathbf{v}$$

2.5.2 Discrete Mass Conservation

Now we define the concept of "zonal mass"; this is postulated to be the total mass contained within a discrete volume element Ω_z and is defined by the following integral

$$m_z \equiv \int_{\Omega_z} \rho \tag{22}$$

With this definition, the fundamental postulate of the Lagrangian description of hydrodynamics can be written as

$$\frac{\partial m_z}{\partial t} = 0 \tag{23}$$

This implies that the total mass in a discrete volume element (or zone) is a constant. Stated another way, no mass enters or exits the boundary of a discrete volume element Ω_z regardless of how the geometry of the volume element changes in time. It is important to point out that this postulate places no restriction on how the density within a zone can vary, i.e. the density (mass per unit volume) inside the discrete volume element can be a function of space and time, the only restriction is that it must integrate to a constant mass by means of (23)

To remain consistent with the ARES architecture, we choose to approximate the density with a function which has a constant value, ρ_z in each zone Ω_z. In this case, we can simplify (23) and the principle of zonal mass conservation implies

$$\rho_z^n = \frac{m_z}{V_z^n} \tag{24}$$

where V_z^n is the volume of the zone Ω_z^n. This is the same approach that is used in most traditional SGH codes and is sometimes referred as "mass conservation by fiat" [15]. The change of variables induced by the deformation field $\vec{v}^{def}$ implies

$$\int_{\Omega_z^{n+1}} \rho_z^{n+1} = \int_{\Omega_z^n} \rho_z^{n+1} \left(1 + \Delta t \vec{\nabla} \cdot \vec{v}^{def}\right) + O(\Delta t^2)$$

If we apply the principle of zonal mass conservation for the case of piece-wise constant densities, we get

$$\frac{\rho_z^{n+1} - \rho_z^n}{\Delta t \rho_z^{n+1}} = -\frac{1}{V_z^n} \int_{\Omega_z^n} \vec{\nabla} \cdot \vec{v}^{def} + O(\Delta t)$$

Thus, up to an $O(\Delta t)$ term, zonal mass conservation approximates the continuous mass conservation equation of (2)

$$\frac{1}{\rho}\frac{\partial \rho}{\partial t} = -\vec{\nabla} \cdot \vec{v}$$

2.5.3 Discrete Energy Conservation

Recall from Section 2.2 the total energy in a given spatial domain Ω at time n is defined as

$$E^n = \frac{1}{2}\int_{\Omega^n} \rho^n \vec{v}^n \cdot \vec{v}^n + \int_{\Omega^n} \rho^n e^n$$

The *change* in kinetic energy during a discrete Lagrange time step Δt, induced by the acceleration field $\vec{a}$ on the current spatial configuration Ω^n is

$$\frac{1}{2}\left(\int_{\Omega^n} \rho^n \vec{v}^{def} \cdot \vec{v}^{def} - \int_{\Omega^n} \rho^n \vec{v}^n \cdot \vec{v}^n\right) = \Delta t \int_{\Omega_n} P^n \vec{\nabla} \cdot (\frac{\vec{v}^{def} + \vec{v}^n}{2})$$

Now, assume that the new internal energy satisfies

$$e^{n+1} = e^n - \Delta t \int_{\Omega^n} \frac{P^n}{\rho^n} \vec{\nabla} \cdot (\frac{\vec{v}^{def} + \vec{v}^n}{2})$$

and the new velocity $\vec{v}^{n+1}$ on the *new* configuration Ω^{n+1} is computed such that

$$\int_{\Omega^{n+1}} (\rho^{n+1}\vec{v}^{n+1}) \cdot \vec{v}^{n+1} = \int_{\Omega^n} (\rho^n \vec{v}^{def}) \cdot \vec{v}^{def} \tag{25}$$

Then the total energy is conserved

$$E_{n+1} = E_n$$

Thus, the change in kinetic energy from $\vec{v}^n$ to $\vec{v}^{n+1}$ is exactly compensated by the change in internal energy. To preserve this on the next time step, the velocity should be transfered without introducing any new kinetic energy. This can be ensured by a simple (global) scaling of the $\vec{v}^{n+1}$ vector field on Ω^{n+1}.

2.5.4 Discrete Equation of State

The equation of state is a material response function which specifies the pressure exerted by a given volume of material which contains a specified amount of mass and has a specified internal energy per unit mass. The discrete version of the EOS is simply a piece-wise constant function of the from

$$P_z^n = EOS(\rho_z^n,\, e_z^n) \tag{26}$$

2.6 Mixed Finite Element Basis Functions for $H(Div)$

Here we introduce the specific form of the basis functions we will use for velocity, which we will denote as $\vec{w}_i$, and for pressure, which we will denote as ϕ_i. Again, the subscript i is a "degree of freedom" (or DOF) index which spans the total number of unique basis functions which make up the basis set. The basis functions are said to *span* a particular polynomial space over a discrete spatial domain Ω_z (a.k.a. zone); this is expressed notationally as

$$\vec{w}_i \in \mathbf{W}(\Omega_z)$$

$$\phi_i \in \mathbf{P}(\Omega_z)$$

The specific form of the polynomial space and the number of basis functions required to span it are determined by the topology and geometry of the zone as well as the maximum degree of polynomials we wish to represent exactly with the basis set. For example, suppose we want to represent all *linear* functions of one variable over the one dimensional spatial domain $\Omega_z = [0,1]$. The *topology* of our zone Ω_z is determined by the fact that we have a one dimensional unit of space defined by two "vertices". The *geometry* of our zone Ω_z is determined by the specific locations of these vertices at 0 and 1. This simple domain can be spanned by the polynomial basis $\alpha_0 + \alpha_1 x$, i.e. one constant term with a coefficient α_0, and one linear term with a coefficient α_1. The basis

functions for this case are very simple, namely 1 and x. The degrees of freedom (DOF) are simply the coefficients α_0 and α_1. Suppose we wish to approximate some arbitrary scalar function of one variable, $f(x)$, with our basis over the domain $\Omega_z = [0,1]$. This is accomplished by the following interpolation (a.k.a. basis function expansion)

$$f(x) \approx \sum_i \alpha_i \phi_i(x) \tag{27}$$

For this simple example, the coefficients in the expansion, the DOF α_i, are simply the value of the function f at the end points of the domain Ω_z, namely $\alpha_0(f) = f(0)$ and $\alpha_1(f) = f(1)$. The coefficients of expansion are said to be the *projection* of the function f onto the *dual space* of the discrete basis, and in general they are defined by weighted integrals of the function f over the domain Ω_z (i.e. the degrees of freedom are defined as *linear functionals* in a manner completely analogous to computing the coefficients in a Fourier expansion). This simple example serves to illustrate two key concepts we will be using in this section: first, the notion of a *discrete basis* ϕ_i which spans some polynomial space defined with respect to some discrete spatial domain such that $\phi_i \in \mathbf{P}(\Omega_z)$ and second, the notion of degrees of freedom as a *projection* operation (i.e. linear functionals).

As mentioned previously, we would like to keep our discrete pressure as piece-wise constants at zone centers. In terms of function spaces, this means our discrete pressure basis is very simple, just a single constant value is required. The degree of freedom for this simple basis is the value of the pressure evaluated at the zone center. Having specified a basis for pressure, we now must choose a specific velocity basis, $\vec{w}_i$. However, in order for the MFEM to be stable, we cannot choose this basis arbitrarily. It must be chosen to satisfy the so called Babuska-Brezzi stability condition [23]. Simply stated, the stability condition requires that our velocity basis satisfy the following relation

$$\vec{\nabla} \cdot \mathbf{W}(\Omega) = \mathbf{P}(\Omega) \tag{28}$$

In other words, the basis we choose for velocity must have a constant divergence (since we have restricted ourselves to using piece-wise constants for our discrete pressure basis ϕ_i).

Finally, before we define the basis for velocity, we need to define the topology of the discrete spatial domain it will be defined on. It is at this point where we make the choice as to what the underlying topology of our discrete zones Ω_z will be. To be compatible with the current ARES 2D architecture, we will choose to work with quadrilateral zones. All quadrilateral zones (including zones with curved boundaries) in a physical mesh are topologically equivalent to a reference quadrilateral zone. In order to make integration over the reference zone as simple as possible, we adopt a standard Cartesian coordinate system with an origin at the point $(0,0)$ as our reference coordinate system. Throughout the remainder of this paper, all objects explicitly defined with respect to this reference coordinate system will be accented with a *hat* symbol. Let $\hat{\Omega}_z$ denote the unit quadrilateral such that

$$\hat{\Omega}_z \equiv \{(\hat{x},\hat{y});\ 0 \le (\hat{x},\hat{y}) \le 1\} \tag{29}$$

2.6.1 The Raviart-Thomas Basis Functions

We define the explicit form of the so called Raviart-Thomas (RT) [11] basis functions for a unit quadrilateral in Table 3. These four vector valued basis functions are linear in $\hat{x}$ and $\hat{y}$ (though not complete for all linear functions) and interpolate at element faces, i.e. each basis function has a non-zero normal component along one and only one face in the reference quadrilateral. This can be seen visually by the vector plots of the basis functions shown in Figure 6. Most importantly, the basis has a constant divergence.

The degrees of freedom for these basis functions are the normal components of the velocity along each of the four faces (edges) of the quadrilateral, giving us one DOF per face. Note that there is freedom in choosing the labeling and orientation convention for the explicit form of the degrees of freedom (e.g. the orientation of the face normal vector). We have chosen the explicit functional form of the degrees of freedom as given in Table 4 as our particular convention. There are certain practical concerns that have determined this choice. A schematic depiction of the these DOF and the conventions we have chosen are shown in Figure 7. The use of these low order RT basis functions in the prototype ARES MFEM hydro code are specified by the option "`mfemorder 4`" (i.e. 4 DOF per zone).

Basis Function	x-component	y-component
$\hat{\vec{w}}_1$	0	$1-\hat{y}$
$\hat{\vec{w}}_2$	$\hat{x}$	0
$\hat{\vec{w}}_3$	0	$\hat{y}$
$\hat{\vec{w}}_4$	$1-\hat{x}$	0

Table 3: The Raviart-Thomas basis functions on a reference quadrilateral

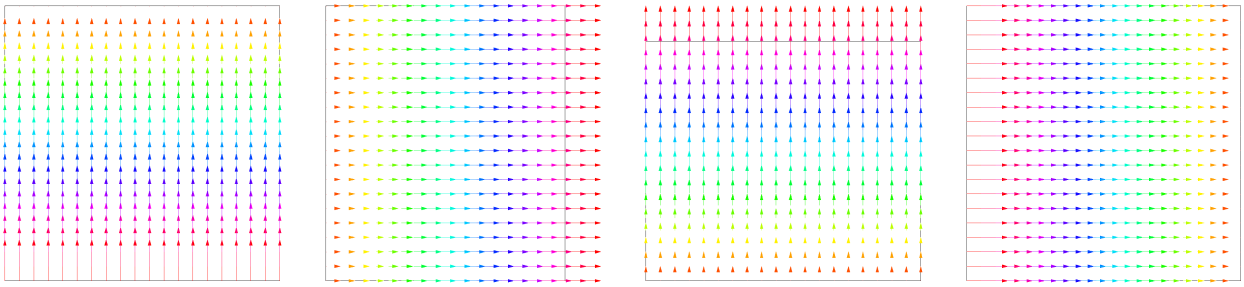

Figure 6: Vector field plots of the four Raviart-Thomas velocity basis functions on a reference quadrilateral.

DOF ID	Functional Form
v_1	$v_1(\vec{f}) = \vec{f}(\frac{\vec{x}_1+\vec{x}_2}{2}) \cdot \hat{n}_1$
v_2	$v_2(\vec{f}) = \vec{f}(\frac{\vec{x}_2+\vec{x}_3}{2}) \cdot \hat{n}_2$
v_3	$v_3(\vec{f}) = \vec{f}(\frac{\vec{x}_3+\vec{x}_4}{2}) \cdot \hat{n}_3$
v_4	$v_4(\vec{f}) = \vec{f}(\frac{\vec{x}_4+\vec{x}_1}{2}) \cdot \hat{n}_4$

Table 4: The Raviart-Thomas degrees of freedom on a reference quadrilateral

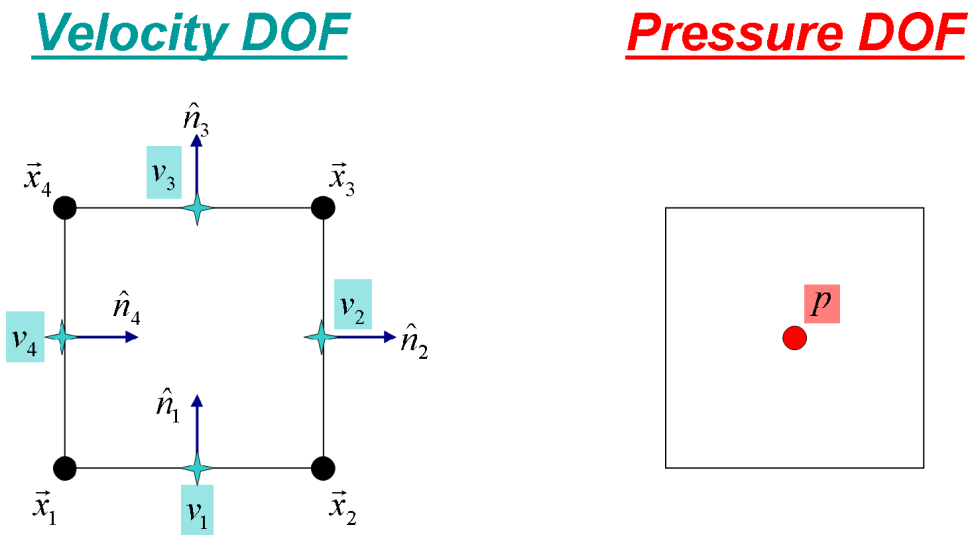

Figure 7: Schematic depiction of the Raviart-Thomas velocity and pressure degrees of freedom.

2.6.2 The Brezzi-Douglas-Marini Basis Functions

We define the explicit form of the so called Brezzi-Douglas-Marini (BDM) [12] basis functions for a unit quadrilateral in Table 5. These eight vector valued basis functions are quadratic in $\hat{x}$ and $\hat{y}$ (though not complete for all quadratic functions) and also interpolate at element faces, i.e. each basis function has a non-zero normal component along one and only one face in the reference quadrilateral. However, because this is a high-order basis, we now have two DOF per face. We provide visual examples of these high order vector basis functions in Figure 8. Again, the critical feature of this basis is that it has a constant divergence. This is accomplished by adding a higher order component to the low order Raviart-Thomas polynomial space that is divergence free, i.e an additional "curl" term. This can be seen in the vector basis plots of Figure 8.

Basis Function	x**-component**	y**-component**
$\hat{\vec{w}}_1$	$(\hat{x}-\hat{x}^2)/2$	$1-\hat{x}-\hat{y}+\hat{x}\hat{y}$
$\hat{\vec{w}}_2$	$-(\hat{x}-\hat{x}^2)/2$	$\hat{x}-\hat{x}\hat{y}$
$\hat{\vec{w}}_3$	$\hat{x}-\hat{x}\hat{y}$	$-(\hat{y}-\hat{y}^2)/2$
$\hat{\vec{w}}_4$	$\hat{x}\hat{y}$	$(\hat{y}-\hat{y}^2)/2$
$\hat{\vec{w}}_5$	$-(\hat{x}-\hat{x}^2)/2$	$\hat{y}-\hat{x}\hat{y}$
$\hat{\vec{w}}_6$	$(\hat{x}-\hat{x}^2)/2$	$\hat{x}\hat{y}$
$\hat{\vec{w}}_7$	$1-\hat{x}-\hat{y}+\hat{x}\hat{y}$	$-(\hat{y}-\hat{y}^2)/2$
$\hat{\vec{w}}_8$	$\hat{y}-\hat{x}\hat{y}$	$(\hat{y}-\hat{y}^2)/2$

Table 5: The BDM basis functions on a reference quadrilateral

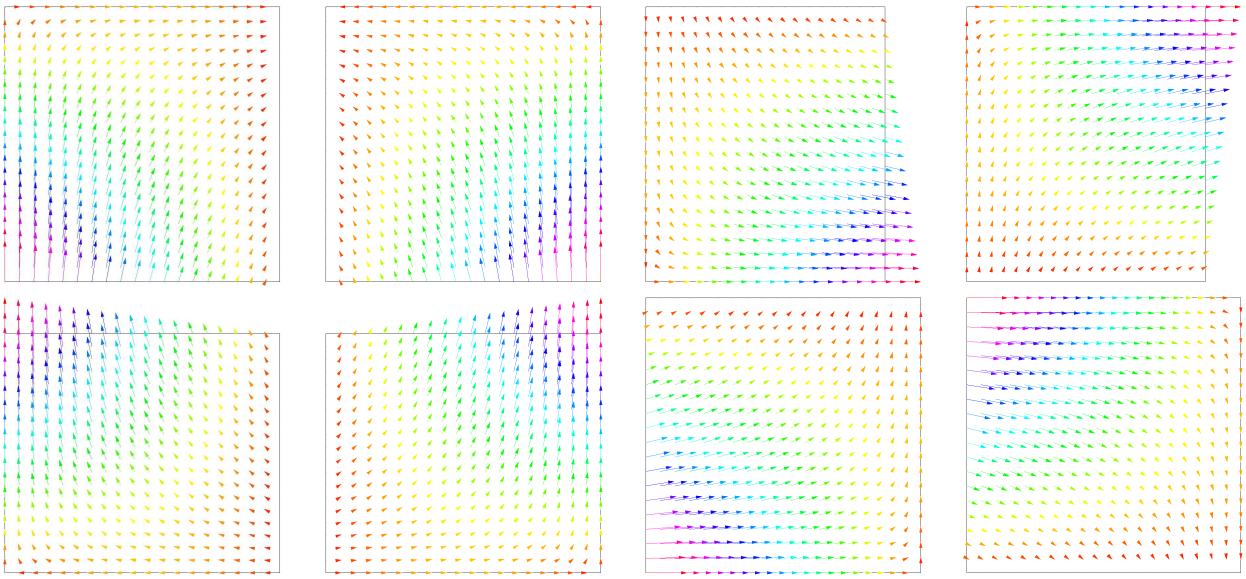

Figure 8: Vector field plots of the eight BDM velocity basis functions on a reference quadrilateral.

Again, there is freedom in choosing the labeling and orientation convention for the explicit form of the degrees of freedom for this basis. We choose to define the degrees of freedom for these basis functions according the convention proposed by [24], and provide the explicit functional forms in Table 6. The DOF represent the value of the velocity at a given vertex, dotted into one of the face normals, giving us two DOF per face. A schematic depiction of the DOF and the conventions we have chosen are shown in Figure 9. The use of these high order BDM basis functions in the prototype ARES MFEM hydro code are specified by the option "`mfemorder 8`" (i.e. 8 DOF per zone).

DOF ID	Functional Form
v_1	$v_1(\vec{f}) = \vec{f}(\vec{x}_1) \cdot \hat{n}_1$
v_2	$v_2(\vec{f}) = \vec{f}(\vec{x}_2) \cdot \hat{n}_1$
v_3	$v_3(\vec{f}) = \vec{f}(\vec{x}_2) \cdot \hat{n}_2$
v_4	$v_4(\vec{f}) = \vec{f}(\vec{x}_3) \cdot \hat{n}_2$
v_5	$v_5(\vec{f}) = \vec{f}(\vec{x}_4) \cdot \hat{n}_3$
v_6	$v_6(\vec{f}) = \vec{f}(\vec{x}_3) \cdot \hat{n}_3$
v_7	$v_7(\vec{f}) = \vec{f}(\vec{x}_1) \cdot \hat{n}_4$
v_8	$v_8(\vec{f}) = \vec{f}(\vec{x}_4) \cdot \hat{n}_4$

Table 6: The BDM degrees of freedom on a reference quadrilateral

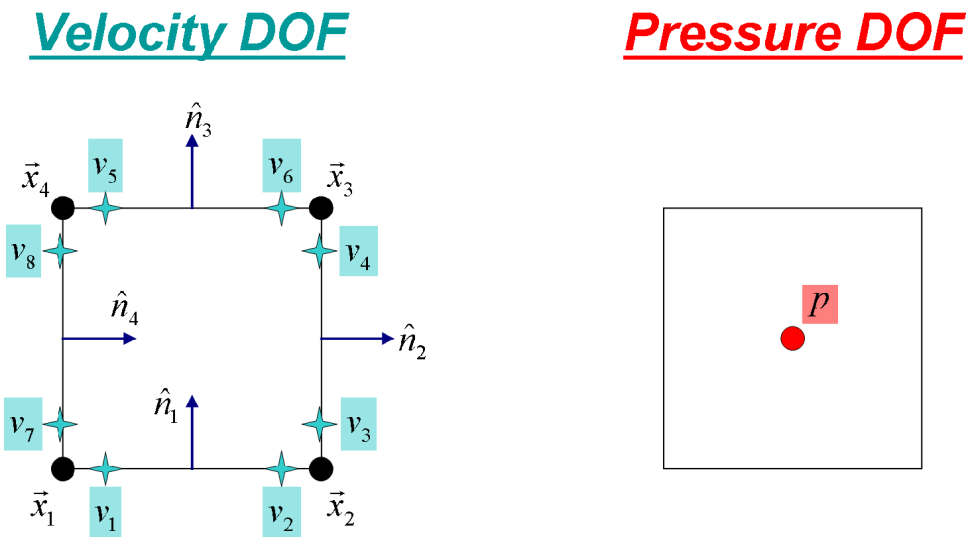

Figure 9: Schematic depiction of the BDM velocity and pressure degrees of freedom.

2.7 The Jacobian Matrix and Coordinate System Invariance

The basis functions and degrees of freedom from Section 2.6 are defined only with respect to the reference zone $\hat{\Omega}_z$. There exists a mapping Φ from the reference zone $\hat{\Omega}_z$ to an actual mesh zone Ω_z. This mapping is functionally written as

$$\vec{x} = \Phi(\hat{\vec{x}}) \tag{30}$$

This mapping is referred to as the local-to-global (a.k.a parametric) mapping and is defined by the geometry of the actual mesh zone. For example, if we restrict ourselves to quadrilaterals consisting of four vertices connected by straight lines, then this is equivalent to a bilinear parametric mapping. For the purposes of this work, this is exactly the mapping we will assume (since this is precisely the type of quadrilateral zone that is permitted in the ARES code); however, we point out that higher order mappings exist which permit curved surfaces. For the specific case of a bilinear parametric map, we have

$$\Phi(\hat{\vec{x}}) = \vec{x}_1(1-\hat{x})(1-\hat{y}) + \vec{x}_2\hat{x}(1-\hat{y}) + \vec{x}_3\hat{x}\hat{y} + \vec{x}_4(1-\hat{x})\hat{y} \tag{31}$$

where $\vec{x}_i$ denotes the Lagrangian coordinate of vertex i. We define the Jacobian matrix (a.k.a. metric tensor) for this mapping as

$$(\mathbf{J}_z)_{i,j} = \frac{\partial x_j}{\partial \hat{x}_i} \tag{32}$$

For the specific case of a bilinear parametric map in 2D, the Jacobian matrix has the form

$$\mathbf{J}_z(\hat{x},\hat{y}) = \begin{pmatrix} (1-\hat{y})(x_2-x_1)+\hat{y}(x_3-x_4) & (1-\hat{y})(y_2-y_1)+\hat{y}(x_3-x_4) \\ (1-\hat{x})(x_4-x_1)+\hat{x}(x_3-x_2) & (1-\hat{x})(y_4-y_1)+\hat{x}(y_3-y_2) \end{pmatrix} \tag{33}$$

where x_i and y_i denote the x and y components of the Lagrangian coordinate of vertex i. Note that the Jacobian matrix is a *function* of $\hat{x}$ and $\hat{y}$ and therefore varies inside of a zone.

A key property of the $H(Div)$ basis functions described in section Section 2.6 is that they "interpolate" at element faces, meaning that they have a non-zero normal component along one and only one face in the reference element. This can be expressed by the relation

$$v_i(\hat{\vec{w}}_j) = \delta_{i,j} \tag{34}$$

where $\delta_{i,j}$ is the Kronecker delta function. We need to preserve this property for an arbitrary change of variables (i.e. for an arbitrary quadrilateral), this is known as *invariance*. The linear functionals which define the velocity degrees of freedom involve dot products with face normals. Surface normal vectors transform *covariantly* upon a change of variables as

$$\vec{n} = |\mathbf{J}_z|\, \mathbf{J}_z^{-1} \hat{n} \tag{35}$$

where $|\mathbf{J}_z|$ denotes the determinant of the Jacobian matrix. Therefore, in order to preserve the invariance property of (34), we must transform the basis functions $\hat{\vec{w}}_i$ in an inverse manner as

$$\vec{w}_i = \frac{1}{|\mathbf{J}_z|}\, \mathbf{J}_z^T \hat{\vec{w}}_i \tag{36}$$

This is known as the Piola transformation and it preserves the normal components of $\hat{\vec{w}}_i$ upon an arbitrary change of variables. An example of this transformation is shown in Figure 10.

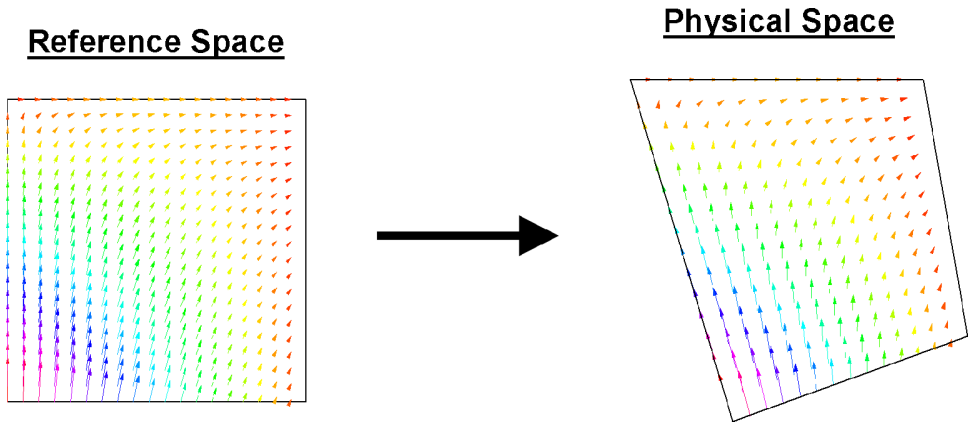

Figure 10: Example of the Piola transformation applied to one of the BDM velocity basis functions on a distorted quadrilateral. Note how the basis fiction maintains its property of having a normal component along one and only one face of the zone.

2.8 Construction of the Mass Matrix

Here we show the details of how the "mass matrix" of (19) is computed on a zone by zone basis using the $H(Div)$ basis functions. The integral over the actual mesh zone Ω_z is transformed to an integral over the reference zone $\hat{\Omega}_z$ by a change of variables as

$$(\mathbf{M}_z)_{i,j} = \int_{\hat{\Omega}_z} \rho_z \left(\frac{1}{|\mathbf{J}_z|}\mathbf{J}_z^T \hat{\vec{w}}_i\right) \cdot \left(\frac{1}{|\mathbf{J}_z|}\mathbf{J}_z^T \hat{\vec{w}}_j\right) |\mathbf{J}_z| \tag{37}$$

This integral is approximated using Gaussian quadrature of a specified order. The quadrature rule is first defined in 1D by a set of k weights w_i and points x_i, where k denotes the order of the quadrature rule. Specifically, we use Gauss-Lobatto quadrature where the endpoints of the 1D reference domain $[0,1]$ are included in the points x_i. This is done to maximize the sparsity of the mass matrix (since the basis functions are also defined at these points). The one dimensional Gauss-Lobatto weights and points on the interval $[0,1]$ for $k = 1,...,5$ are defined in Table 7. To generate the corresponding 2D weights and points for the reference quadrilateral $\hat{\Omega}_z$, we simply take a direct tensor product of the 1D weights and points, generating a total of k^2 weights and points. The order of the quadrature rule can be specified in ARES with the option "`mfemquadorder`". We have determined that the optimal value is $k = 3$, resulting in 9 quadrature points per zone in the calculation of (37). This is the default value used in the code.

Order 1	
$w_1 = 1$	$x_1 = 0.5$
Order 2	
$w_1 = 1/2$	$x_1 = 0.0$
$w_2 = 1/2$	$x_2 = 1.0$
Order 3	
$w_1 = 1/6$	$x_1 = 0.0$
$w_2 = 4/6$	$x_2 = 0.5$
$w_3 = 1/6$	$x_3 = 1.0$
Order 4	
$w_1 = 1/12$	$x_1 = 0.0$
$w_2 = 5/12$	$x_2 = 0.276393$
$w_3 = 5/12$	$x_3 = 0.723607$
$w_4 = 1/12$	$x_4 = 1.0$
Order 5	
$w_1 = 0.05$	$x_1 = 0.0$
$w_2 = 0.272222$	$x_2 = 0.172673$
$w_3 = 0.355556$	$x_3 = 0.5$
$w_4 = 0.272222$	$x_4 = 0.827327$
$w_5 = 0.05$	$x_5 = 1.0$

Table 7: Weights and Points for 1D Gauss-Lobatto Quadrature Rules of Order 1 Through 5

We now point out some interesting properties of the "mass" matrix. The velocity basis functions $\vec{w}_i$ have units of $distance$ while the velocity degrees of freedom v_i have units of $1/time$, such that the basis function expansion

$$\vec{v} \approx \sum_i v_i \vec{w}_i$$

has the correct units of $distance/time$. This means that the "mass" matrix has units of $mass * distance^2$. The mass matrix therefore depends on the current spatial configuration of the mesh (i.e the Lagrange coordinates) and must therefore change every time the Lagrangian mesh is updated (i.e. at every discrete time step). This is in contrast to traditional SGH methods which have a constant "mass matrix" at every cycle (i.e. nodal mass). The discrete kinetic energy in given zone Ω_z is defined as

$$KE_z = \frac{1}{2}\mathbf{v}_z^T \mathbf{M}_z \mathbf{v}_z \tag{38}$$

and therefore has the correct units of $mass * velocity^2$. Note that the discrete kinetic energy of (38) is a direct discretization of the integral relation of (6).

2.9 Construction of the Derivative Matrix

Here we show the details of how the "derivative matrix" of (20) is computed on a zone by zone basis using the $H(Div)$ basis functions. The integral over the actual mesh zone Ω_z is again transformed to an integral over the reference zone $\hat{\Omega}_z$ by a change of variables as

$$(\mathbf{D}_z)_{i,j} = \int_{\hat{\Omega}_z} \vec{\nabla} \cdot \hat{\vec{w}}_j \tag{39}$$

This is a simple calculation due to the fact that the divergence of the basis functions, $\vec{\nabla} \cdot \hat{\vec{w}}_i$ is a constant over any zone and can therefore be pulled from the inside the integral (i.e. this matrix does not need to be recomputed at every time step).

2.10 Semi-Discrete Form of Hydrodynamics Equations

Recall from (18) the global momentum conservation equation is written as

$$\mathbf{M}\frac{\partial}{\partial t}\mathbf{v} = \mathbf{D}^T\mathbf{p}$$

For the case of BDM basis functions, $\mathbf{M}$ is a symmetric positive definite matrix of dimension $2N_{faces}$ by $2N_{faces}$, $\mathbf{v}$ is an array of dimension $2N_{faces}$, $\mathbf{D}$ is a rectangular matrix of dimension N_{zones} by $2N_{faces}$ and $\mathbf{p}$ is an array of dimension N_{zones}, where N_{zones} denotes the total number of zones in the mesh and N_{faces} denotes the total number of unique faces in the mesh. The total discrete kinetic energy in the mesh $\tilde{\Omega}$ is therefore given as

$$KE = \frac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v} \tag{40}$$

We assume a piece-wise constant value for the internal energy per unit mass. The discrete energy equation for a given zone is written as

$$m_z\frac{\partial}{\partial t}e_z = -\mathbf{p}_z\mathbf{D}_z\mathbf{v}_z \tag{41}$$

The total discrete internal energy in the mesh $\tilde{\Omega}$ is therefore given by the sum

$$IE = \sum_z m_z e_z$$

Note that under the assumption of piece-wise constant densities, this is a discrete analogue of (7). Given these definitions, we can define the following generalized discrete energy conservation equation

$$\frac{\partial}{\partial t}\left(\frac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v} + \sum_z m_z e_z\right) = \frac{1}{2}\mathbf{v}^T(\frac{\partial}{\partial t}\mathbf{M})\mathbf{v} \tag{42}$$

Therefore, if $\frac{\partial}{\partial t}\mathbf{M} = 0$ (i.e. the mass matrix does not change over time), we recover conservation of total energy (9) automatically. However, as we pointed out in Section 2.8, this is not the case for our MFEM mass matrix (in fact, we argue that this is not the case for *any* proper mass matrix on quadrialteral / hexahedral grids) and so we see the fundamental reason why the rescaling of velocity is required in (25).

2.11 Mesh Motion, Nodal Velocity and Hourglass Modes

Our velocity field lives in the space $H(Div)$, but in order to move the mesh, we need to somehow obtain our velocity field at mesh vertices (a.k.a. nodes). One major advantage of having explicit basis functions is that once we have solved the discrete momentum equation of (18), we can express the velocity at any point within a zone using a basis function expansion. In particular, we could choose to evaluate the velocity at a particular vertex

$$\vec{v}(\vec{x}_p) = \sum_i v_i \;\; \vec{w}_i(\vec{x}_p)$$

However, the $H(Div)$ velocity representation only enforces normal continuity of the velocity field across zone boundaries, so there is no guarantee that this field will be unique at a given vertex. One way to overcome this is to apply an averaging technique. The simplest case is taking the average of the value of $\vec{v}(\vec{x}_p)$ from every zone which shares the vertex $\vec{x}_p$. This nodal averaging method can be specified in ARES with the option "`mfemnodemethod 0`". This simple approach has proven to be the most robust and is the default option.

A more advanced method is known as a "patch recovery" technique. The basic idea is that we use a local patch of elements (namely all of the elements that share the given vertex $\vec{x}_p$) and perform a least squares averaging of the contribution from each of these zones to the vertex $\vec{x}_p$. Functionally, this is expressed as

$$\tilde{\vec{v}}_p \equiv \sum_z \frac{\int_{\Omega_z} \psi \, \vec{v}}{\int_{\Omega_z} \psi} \tag{43}$$

where $\vec{v}$ is our basis function representation of the velocity and ψ is some interpolating shape function, such as the standard bi-linear shape functions of (31). If we use bi-linear shape functions along with a simple 4 point quadrature rule, then (43) reduces to simple volume weighted averaging. This nodal averaging method can be specified in ARES with the option "`mfemnodemethod 1`". Finally, if we apply a more accurate 9 point quadrature rule, we obtain a higher-order averaging. This nodal averaging method can be specified in ARES with the option "`mfemnodemethod 2`".

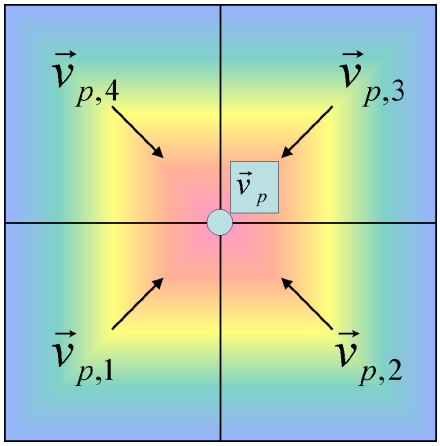

Figure 11: Schematic depiction of the local patch recovery process for computing nodal velocity fields

Unfortunately, this averaging does not pay special attention to the hourglass modes since it is effectively taking our $H(Div)$ represenation of velocity (which does not have hourglass modes) and converting it to a bilinear representation of velocity at nodes (which does have hourglass modes). This is depicted in Figure 12. Therefore, some form of hourglass suppression is still required as a post-processing step. A simple local filter is the following:

$$\vec{v}_{p,hg} = \vec{v}_p - \frac{\beta}{4}\left((\vec{v}_p \cdot \vec{h}_1)\vec{h}_1 + (\vec{v}_p \cdot \vec{h}_2)\vec{h}_2\right)$$

Here $\vec{h}_1$ and $\vec{h}_2$ are the two zonal hourglass modes and β is an appropriately chosen parameter depending on the time step and zone information.

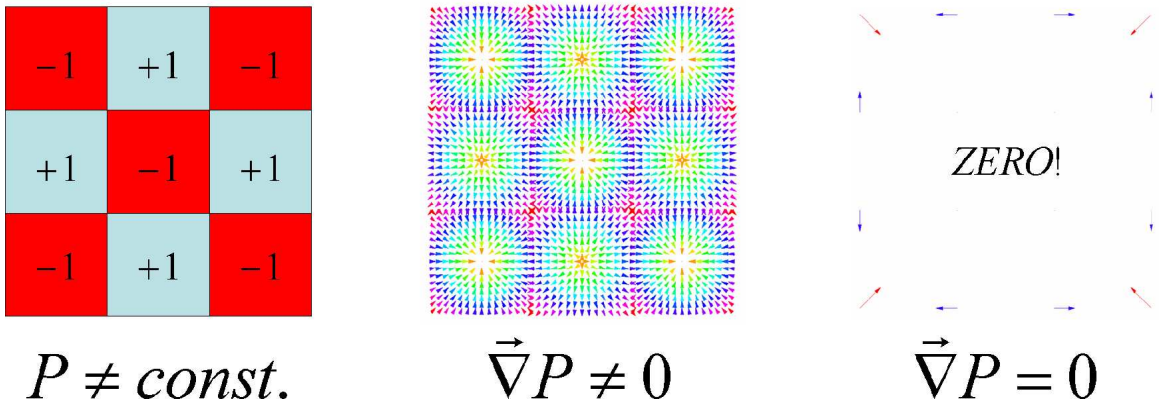

Figure 12: Consider a "checkerboard" pressure field on a simple grid (*left*). The MFEM solution of $\vec{\nabla} P$ evaluated at multiple points in a given zone is clearly non-zero as it should be (*center*). However by performing averaging to get $\vec{\nabla} P$ at nodes , the interior nodal fields are now zero (*right*).

2.12 Time Integration and the Lagrange Step

For the prototype capability described in this document, we have employed a very simple leap frog time integration method for the Lagrange step. The process for a given MFEM Lagrange hydro step (or cycle) is outlined below:

$$
\begin{aligned}
\mathbf{M}^n \mathbf{a} &= \mathbf{D}^T \mathbf{p}^n \\
\mathbf{v}^{def} &= \mathbf{v}^n + \Delta t\ \mathbf{a} \\
\vec{v}_p^{\,def} &= NodalAverage(\mathbf{v}^{def}) \\
\vec{x}_p^{\,n+1} &= \vec{x}_p^{\,n} + \Delta t\ \vec{v}_p^{\,def} \\
V_z^{n+1} &= ComputeVolume(\vec{x}_p^{\,n+1}) \\
\rho_z^{n+1} &= \frac{m_z}{V_z^{n+1}} \\
e_z^{n+1} &= e_z^n - \frac{\Delta t}{m_z} P_z^n \mathbf{D}_z \left(\frac{\mathbf{v}_z^{def} + \mathbf{v}_z^n}{2}\right) \\
P_z^{n+1} &= EOS(\rho_z^{n+1}, e_z^{n+1}) \\
\mathbf{v}^{n+1} &= ProjectAndRescale(\vec{v}_p^{\,def})
\end{aligned}
$$

Note that the overall sequence is very similar to what is traditionally done in a typical SGH code [4]. The key differences are:

- A linear solve is performed to obtain the face based acceleration unknowns (analgous to dividing nodal forces by a "nodal mass").
- A nodal averaging step is performed to compute nodal velocity fields from the face based velocity representation
- The change in internal energy per zone is computed from the face based velocity representation (i.e. a discrete version of velocity divergence).
- The transfer of the face based velocity from the mesh at time n to the newly updated mesh at time $n+1$ (a remap step) requires a rescaling to prevent the introduction of new kinetic energy due the change in the mass matrix.

The method is easily adaptable to a more advanced predictor-corrector time integration scheme, or any other time marching scheme. The results for discrete energy conservation in semi-discrete form of (42) are valid for a generic time discretization process.

2.13 Axisymmetric (*r*-*z*) Formulation

For problems that can be modeled as a body of revolution, we can cast our equations in axisymmetric form by assuming cylindrical coordinates. Consider a three dimensional function of space defined with respect to a cylindrical coordinate system, $f(r,\theta,z)$. The function is said be axisymmetric if the following condition holds true

$$\frac{\partial}{\partial\theta} f(r,\theta,z) = 0$$

Thus, our function f is only spatially varying in the *r*-*z* plane as depicted in Figure 13. For an axisymmetric problem, the total mass in a volume Ω can be computed as follows

$$m \equiv \int_\Omega \rho = 2\pi \int_\Gamma \rho\, r$$

This allows us to cast the variational form of the momentum equation which is usually defined as a full volume integral over the domain Ω, as a simplified integral over a "cut" in the *r*-*z* plane (which we denote as Γ) as follows.

$$2\pi \int_\Gamma r(\rho \frac{\partial \vec{v}}{\partial t}) \cdot \vec{w} = 2\pi \int_\Gamma r(\vec{\nabla} \cdot \vec{w}) P \tag{44}$$

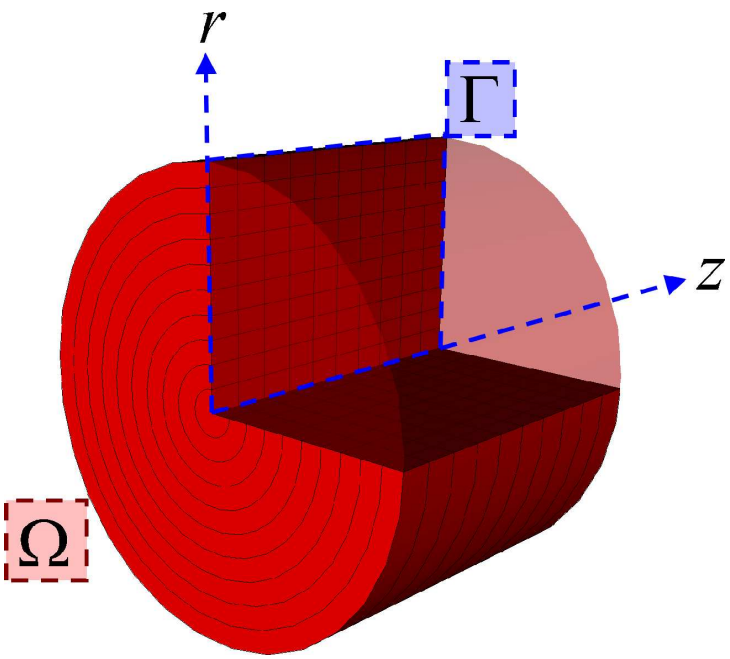

Figure 13: Schematic depiction of the reduction of an axisymmetric problem to a surface integral over a "cut" in the *r*-*z* plane

For axisymmetric functions, the divergence operator is defined as

$$\vec{\nabla} \cdot \vec{f}(r,z) = \frac{\partial}{\partial z} f_z + \frac{\partial}{\partial r} f_r + \frac{f_r}{r} \tag{45}$$

We can simply replace *z* with *x* and *r* with *y* which allows us to write the axisymmetric momentum equation (44) as

$$2\pi \int_\Gamma y(\rho \frac{\partial \vec{v}}{\partial t}) \cdot \vec{w} = 2\pi \int_\Gamma y \left(\frac{\partial}{\partial x} w_x + \frac{\partial}{\partial y} w_y + \frac{w_y}{y} \right) P$$

The $H(Div)$ basis functions have a constant divergence in *x*-*y* space and so this term can be pulled from the intergal to obtain

$$2\pi \int_\Gamma y(\rho \frac{\partial \vec{v}}{\partial t}) \cdot \vec{w} = 2\pi (\frac{\partial}{\partial x} w_x + \frac{\partial}{\partial y} w_y) \int_\Gamma yP + 2\pi \int_\Gamma w_y P$$

This is the correct variational form of the momentum equation under the axisymmetric assumption and this form is coded in ARES for the option "`ifplane 0`." However, there is a fundamental flaw in this formulation. Recall the stability condition of (28), which mandates that for the case of piecewise constant pressures, the divergence of our velocity basis must also be constant. For the $H(Div)$ basis functions we have considered, this is certainly true in 2D *x-y* and also in 3D *x-y-z* space. However, this is *not* true in axisymmetric *r-z* coordinates due to the definition of the *Div* operator in this particular coordinate system. The fundamental flaw here lies in the fact that the *Div* operator in *r-z* space of (45) is very different than the *Div* operator in *x-y* space. This flaw will lead to large and unacceptable errors on and near the axis of symmetry as shown in Section 4. This is in contrast to the *Grad* operator which is functionally equivalent in *x-y* and in *r-z*, which is precisely the reason why the so called Petrov-Galerkin (a.k.a. area weighting) scheme is justified in discretizing the *Grad* operator in methods such as [3].

We point out that this trouble in discretizing the *Div* operator in *r-z* is not unique to the MFEM; in traditional SGH methods, the *Div* operator is approximated as a simple change in volume with respect to time for the internal energy update equation, but this approach is flawed in *r-z* coordinates and causes large errors in energy that do not go away as the mesh is refined in time or space. The discretization error is simply transferred from the momentum equation to the energy equation. We are currently lacking a finite element basis that is specifically designed for axisymmetric problems of this type and therefore, any research into this subject is of great importance.

2.14 Artificial Viscosity and the Treatment of Shocks

The most popular and straightforward method for treating shock waves in a Lagrangian hydrodynamics code is to introduce an artificial viscosity (i.e. a dissipative) term to the Euler equations as originally proposed by [5]. Specifically, we add a viscous force term to the momentum equation and a corresponding energy term to the energy conservation equation to obtain

$$\rho\frac{\partial\vec{v}}{\partial t} = -\vec{\nabla}P + \vec{\nabla}\cdot(\mu\vec{\nabla}\vec{v}) \tag{46}$$

$$\rho\frac{\partial e}{\partial t} = -P\vec{\nabla}\cdot\vec{v} + (\mu\vec{\nabla}\vec{v}):\vec{\nabla}\vec{v} \tag{47}$$

where μ is an artificial diffusion length scale and : denotes tensor contraction. Note that we have considered the general case of a "tensor viscosity" where the viscous force is computed as the divergence of a stress-like tensor computed as the gradient of the velocity field. The artificial viscous force term is a vector field and we can therefore apply a Helmholtz decomposition to obtain

$$\vec{\nabla}\cdot(\mu\vec{\nabla}\vec{v}) = \vec{\nabla}(\mu_{bulk}\vec{\nabla}\cdot\vec{v}) - \vec{\nabla}\times(\mu_{shear}\vec{\nabla}\times\vec{v}) \tag{48}$$

If we ignore the shear term (an assumption which is only valid when $\vec{\nabla}\times\vec{v}=0$), and define the following scalar value

$$q = \mu_{bulk}\vec{\nabla}\cdot\vec{v}$$

then we can rewrite the modified momentum and energy equations as

$$\rho\frac{\partial\vec{v}}{\partial t} = -\vec{\nabla}(P+q) \tag{49}$$

$$\rho\frac{\partial e}{\partial t} = -(P+q)\vec{\nabla}\cdot\vec{v} \tag{50}$$

Note that this is precisely the form of the equations used in traditional SGH codes wich make use of a so called scalar artificial viscosity [5], [4].

A modern scalar artificial viscosity has the form

$$q = \rho\left(c_1C_s|\Delta v|(1-\Psi) + c_2\rho|\Delta v|^2(1-\Psi^2)\right) \tag{51}$$

where c_1 and c_2 are dimensionless constants (the options "`qlin`" and "`qquad`" respectively in ARES), C_s is the local sound speed and Ψ is the so called "monotonic limiter". The goal of the limiter is to act as a shock detector and "switch on" ($\Psi\neq 0$) only in the

vicinity of strong velocity gradients, while remaining off ($\Psi = 0$) when the solution is smoothly varying. The term "monotonic" implies that the limiter is designed to prevent spurious ringing (or "overshoots" and "undershoots"). The details of how such a monotonic limiter is constructed in an SGH code can be found in [4]. We point out that the theory for such monotonic limiters is traditionally limited to $1D$ analysis and as stated in [4], "it is not obvious how to generalize ... to multiple dimensions." We feel that it is precisely this limitation which is responsible for the type of spurious grid distortions shown in Figure 2 and noted in [8] and [7]. However, there is currently a body work that is devoted to addressing this issue for the case of advective transport [25] and this work may prove useful for improving such monotonic limters for artificial viscosities in multiple dimensions.

The term Δv in (51) is known as the "velocity jump" and is defined to be the change in velocity in a given zone in the direction normal to the shock front. In $1D$, it is straightforward to define such a jump term as $\Delta v = dx\frac{\partial v}{\partial x}$ where dx is the width of a zone and $\frac{\partial v}{\partial x}$ is the divergence of velocity in $1D$. In multiple dimensions it is not so straightforward to define. One common approach is to define the velocity jump as a zone centered quantity

$$\Delta v = l\vec{\nabla}\cdot\vec{v} \tag{52}$$

where l is a characteristic length of the given zone. The question is, which length do we use for l? In ARES, the characteristic length in $2D$ is computed as an average value $l = \frac{A}{\Delta x + \Delta y}$ where Δx and Δy are the lengths of the median mesh lines which intersect at the zone center and A is the area of the zone. This approach combined with (52) and (51) *without* a limiter (i.e $\Psi = 0$) is known as the "bulk viscosity" option in ARES and is specified with the option "`ifqmodel 3`." The trouble with this approach is that it does not take the shock direction into account, and so for a quadriateral zone with a high aspect ratio, the length scale is the same regardless of wether the shock wave is traveling along the short direction, the long direction or some arbitrary direction in between.

For the MFEM, it is straightforward to compute the cell centered velocity divergence as the inner product

$$(\vec{\nabla}\cdot\vec{v})_z = \mathbf{D}_z\mathbf{v}_z$$

However, because this is a cell centered scalar value, it does not have any directional information. To overcome this limitation, we tried out the simple idea of defining the velocity jump to be

$$\Delta v = \frac{1}{2}\left((\vec{v}_{23} - \vec{v}_{14})\cdot\frac{d\vec{x}}{|d\vec{x}|} + (\vec{v}_{34} - \vec{v}_{12})\cdot\frac{d\vec{y}}{|d\vec{y}|}\right)$$

where $\vec{v}_{ij}$ is value of the velocity evaluated at the mid-point between the Lagrangian coordinates $\vec{x}_i$ and $\vec{x}_j$ of the zone using the MFEM basis function expansion and $d\vec{x}$ and $d\vec{y}$ are the median zone vectors. We refer to this idea as the "modified" bulk viscosity. We show in Section 4 that this simple idea can lead to improvements, but we do not feel it is a general robust solution to the problem of defining artificial viscosities in multiple dimensions. We feel that the best solution to this problem is to consider the more accurate case of a tensor viscosity coupled with an improved treatment of the "monotonic limiter" term.

3 Description of Code Additions / Modifications

In this section we give a brief, high level overview of the specific code additions that were made in order to implement the MFEM hydro capability in ARES. The vast majority of coding was added in five main source files located in the "`lag`" subdirectory of the ARES source tree:

- `MFEM2D.c` – This file contains all of the core routines for computing zone based MFEM quantities such as Jacobian matrices, local to global transformations, mass and derivative matrices. Also includes routines for projection operations, nodal field reconstruction and zone based artificial viscosity calculations.
- `MFEMLinSys.c` – This file contains mesh level (i.e. loops over domain structures) routines for performing global operations required for a linear solve such as assembling a global mass matrix, a global right hand side and applying boundary conditions. Makes use of the HYPRE unstructured solver interface.

- `MFEMAccel.c` – Extracts the solution from the acceleration operator linear solve and uses it to construct the nodal velocity field which will accelerate the grid. Also computes anti-hourglass forces using various experimental methods.
- `MFEMUpdateGrid.c` – Applies the acceleration to grid nodes and updates the Lagrangian coordinates.
- `MFEMWork.c` – Computes the corresponding thermodynamic work done on the mesh by the Lagrange step in a conservative manner.

In addition to these files, there are several other auxiliary source files that were modified to permit use of the new MFEM hydro algorithm ares, most notably in the `blk/domain.c` file for adding new domain fields, but for the sake of brevity and clarity we have not included these modifications here.

3.1 Zone Specific Routines of `MFEM2D.c`

The bulk of the routines for performing MFEM zone specific operations as described in the Theoretical discussion of Section 2 are in the file `MFEM2D.c`. These routines are called on a zone-by-zone basis, usually from within in a domain-loop. Below are descriptions of the key routines that are used to compute MFEM quantities on a given zone:

```
MFEM2D_jacobian        - Evaluate Jacobian matrix at a point in 2D

MFEM2D_jacobian_inv    - Computes the inverse of a given Jacobian matrix

MFEM2D_loc_to_glob     - Get the global coordinates of a point in reference
                         space

MFEM2D_get_quadrule    - Computes and returns the quadrature weights and
                         points for a specified quadrature rule order

MFEM2D_eval_RT_basis   - Evaluates the RT basis functions at the specified
                         quadrature points

MFEM2D_eval_BDM_basis  - Evaluates the BDM basis functions at the specified
                         quadrature points

MFEM2D_get_RT_massmat  - Get the local RT mass matrix for a given zone

MFEM2D_get_BDM_massmat - Get the local BDM mass matrix for a given zone

MFEM2D_get_BDM_divmat  - Compute the rectangular divergence matrix for a given
                         zone using the BDM basis functions and Gaussian
                         quadrature

MFEM2D_project_RT      - Project a nodal vector field onto the RT space

MFEM2D_project_BDM     - Project a nodal vector field onto the BDM space

MFEM2D_interp          - Interpolate a nodal vector field at zone nodes
                         using the basis functions and a set of degrees of
                         freedom

MFEM2D_get_face_map    - Computes the global IDs of the face degrees of
```

```
                        freedom for a given zone. Only valid for the specific
                        case of a single mesh block

MFEM2D_CalcQDivV      - Calculates a zone centered scalar artifical viscosity
```

3.2 Linear System Specific Routines of `MFEMLinSys.c`

In order to assemble and solve a sparse linear system that is defined over the entire mesh, we need mesh level routines for gathering and scattering our local zone based results into a global linear system object. In addition, we need routines for computing and imposing boundary conditions as well as retrieving zone based solution info from the global solution vector. The file `MFEMLinSys.c` contains such routines and is the primary interface between ARES and the HYPRE unstructured solver interface (discussed in the next section). Below are descriptions of the key routines that are used to assemble global linear system quantities needed for a linear solve and its post-processing:

```
MFEMLinSys_Init        - A function for initializing the HYPRE matrix data
                         used for MFEM hydro

MFEMLinSys_SetShared   - A function for setting the shared degree of
                         freedom information for the HYPRE matrix data
                         (NOTE - this routine is necessary for multi-block
                          caluclations and requires a global face index,
                          and is therefore not yet implemented)

MFEMLinSys_AssembleMat - A function for assembling the linear system matrix
                         for MFEM hydro

MFEMLinSys_AssembleRHS - A function for assembling the linear system right
                         hand side for MFEM hydro

MFEMLinSys_SetBCs      - A function for computing and setting the boundary
                         conditions for MFEM hydro

MFEMLinSys_Solve       - A function for solving the MFEM linear system

MFEMLinSys_NodeAccel   - A function for computing the nodal accelerations
                         from the face based solution vector

MFEMLinSys_Destroy     - A function for cleaning memory and data structures
                         used by HYPRE for MFEM hydro
```

3.3 The `HYPRE_FEI` Solver Routines

The easiest way to interface with the HYPRE linear solvers library for the case of the MFEM hydro algorithm is through the so called finite element interface (FEI). All that is required to make this interface work on any grid is a zone based "DOF index map". This mapping is simply an enumeration of the global DOF indices that a given zone contains. Since our $H(Div)$ MFEM algorithm has velocity unknowns on *faces*, this implies that we need a global face ID for every unique face in our mesh. For a single domain, this mapping is trivial to construct on a single block, and is the approach that was taken for this prototype code. However, as pointed out in the next section, a realistic multi-block (i.e. parallel) calculation will require the notion of global face

indices. The original HYPRE_FEI is written in C++, but there is also a C wrapped version of this interface in the latest version of HYPRE (v2.0.0) that is appropriate for use in ARES. Below are descriptions of the key routines that are used in `MFEMLinSys.c` to interface with the HYPRE linear solvers library:

```
HYPRE_FEI_create               - Creates and initializes the HYPRE_FEI object

HYPRE_FEI_parameters           - Sets linear solver paramters

HYPRE_FEI_initFields           - Defines and initializes the fields of the linear
                                 system (i.e. number of DOF per zone etc ...)

HYPRE_FEI_initElemBlock        - Initializes the connectivity pattern of
                                 a given element block (requires a zone based
                                 DOF index map)

HYPRE_FEI_initElem             - Initializes the connectivity pattern of
                                 a given element (requires a zone based
                                 DOF index map)

HYPRE_FEI_initComplete         - Signifies completion of connectivity specification

HYPRE_FEI_sumInElemMatrix      - Sums a local zone based matrix into the global
                                 matrix

HYPRE_FEI_sumInElemRHS         - Sums a local zone based vector into the global
                                 RHS vector

HYPRE_FEI_loadNodeBCs          - Applies boundary conditions to specified boundary
                                 nodes. (NOTE - the name does NOT imply that only
                                 node based DOF can be set)

HYPRE_FEI_loadComplete         - Signifies completion of boundary condition
                                 specification

HYPRE_FEI_solve                - Launches the linear solver

HYPRE_FEI_getNumBlockActNodes  - Used for retrieving a local zone based solution
                                 vector from the global solution vector

HYPRE_FEI_getBlockNodeIDList   - Used for retrieving a local zone based solution
                                 vector from the global solution vector

HYPRE_FEI_getBlockNodeSolution - Used for retrieving a local zone based solution
                                 vector from the global solution vector

HYPRE_FEI_resetSystem          - Resets the HYPRE_FEI object
```

3.4 Need for Global Face Data Structures

The current prototype MFEM hydro capability described in this document is limited to single block problems. This is due to the fact that a global face indexing scheme is required in order to use face based DOF in conjunction with the HYPRE linear solvers library. As mentioned previously, for a single domain, this mapping is trivial to construct on a single block. However, our ultimate goal is a multi-block (i.e. parallel) calculation and this will require the notion of global face indices. This type of face indexing scheme will be useful for many applications, not just the particular MFEM algorithm as described in this document. As such, it will be worth wile to pursue development of these feature in the ARES code.

4 Results on Some Standard Hydrodynamics Test Problems

In this section we review some results obtained using the newly developed MFEM hydro capability in the ARES code on some standard (or benchmark) hydrodynamics test problems. We compare results obtained with the new method to those obtained with the default ARES hydro algorithm, which we will refer to as the HEMP method [3]. In each case, we point out the beneficial as well as the undesirable features of the newly developed algorithm. Problems run with the "`ifplane 1`" option will be referred to as *x-y* mode problems while those run with the "`ifplane 0`" option will be referred to as *r-z* mode problems.

4.1 The Coggeshall Adiabatic Compression Problem #2

We begin with a variation of a problem originally described in [26]; in particular we apply the analytic solution described as problem #2. This problem describes an adiabatic compression (i.e. no shock waves generated) in one spatial coordinate, r. We run the problem in 2D cylindrical geometry (i.e. *x-y* mode, $k = 1$) on a quarter symmetry polar "ring" mesh with an inner radius $R_i = 0.1$ and an outer radius $R_o = 1.0$. We initialize the density, internal energy and velocity of the problem according to the solution provided in [26]. For the free parameters given in the original problem definition, we choose the values $k = 1, \beta = 1$ and $\rho_0 = 1$ and obtain the following initial values

$$\begin{aligned} \vec{v}(r,0) &= -\frac{3r}{5}\hat{r} \\ \rho(r,0) &= \rho_0 r \\ e(r,0) &= \frac{3}{25}r^2 \end{aligned}$$

We use a simple ideal gas equation of state with $\gamma = 5/3$. Since the problem is adiabatic, there is no need to apply an artificial viscosity and so for this problem we explicitly turn off all artificial viscosity parameters by applying the options "`qlin 0.0`" and "`qquad 0.0`." This implies that for this test problem, all mesh motion will be generated by discrete pressure gradients. This will allow us to investigate the differences between the MFEM discrete acceleration equation and the traditional HEMP approach. We apply a random perturbation to a subset of the interior mesh nodes. In addition to the radial surface at $r = R_i$, we keep the next level of radial nodes unperturbed, leaving a one zone thick ring of unperturbed mesh nodes. The initial mesh, pressure and velocity of the problem are shown in Figure 14 and Figure 15. We apply "wall" boundary conditions (i.e. $\vec{v} \cdot \hat{n} = 0$) to the two symmetry planes and we leave the inner and outer radial surfaces "free". We let the problem run for a total physical time of 0.85 time units with a fixed time step, and run for a total of 600 time steps. This generates a roughly 10-fold radial compression of the initial mesh. In Figure 16 we show a close up of the computational mesh the final time step using the default HEMP based hydro algorithm. Note that the originally unperturbed radial surfaces display a great deal of distortion, breaking the radial symmetry of the problem. This is due to the breakdown of the HEMP gradient operator for highly distorted grids as depicted in Figure 3. The errors introduced by this breakdown in grid acceleration are relatively small, but the cumulative effect of this error over many cycles can lead to a drastic breakdown in symmetry as shown in Figure 16. In Figure 17 we show a close up of the computational mesh at the final time step using the new MFEM hydro algorithm. Note that the originally unperturbed radial surfaces remain unperturbed, thus preserving the radial symmetry of the problem even on a randomly distorted mesh.

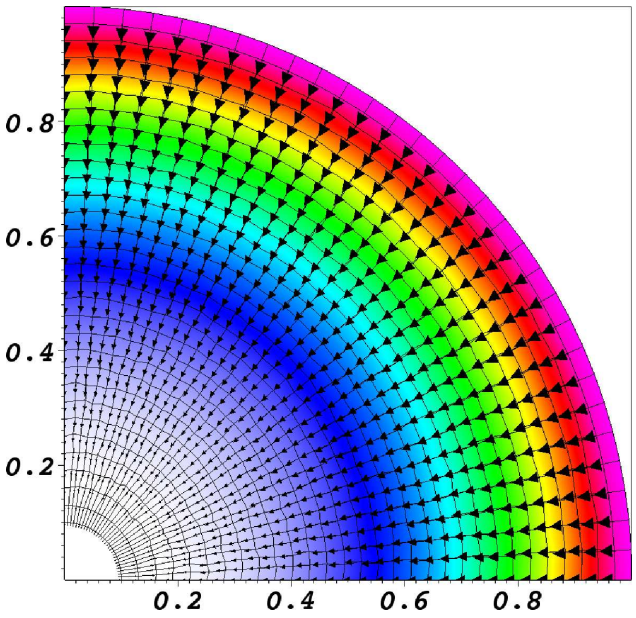

Figure 14: Initial polar "ring" mesh with a random perturbation applied to a subset of the interior mesh nodes, initialized with an analytic pressure and a radial velocity field.

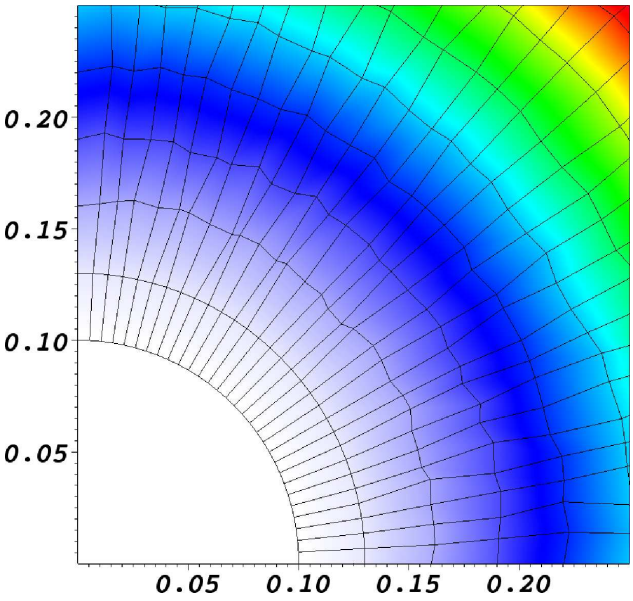

Figure 15: Close-up view of initial mesh. Note how first two radial sets of nodes are left unperturbed.

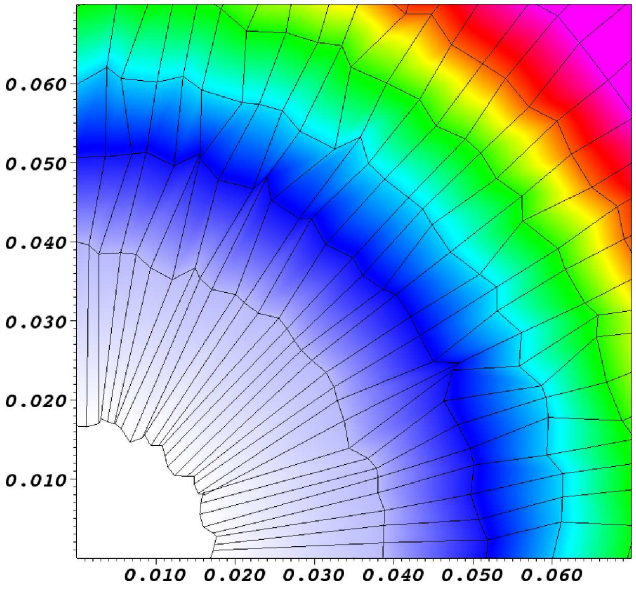

Figure 16: Close-up view of final mesh after 600 cycles using standard HEMP hydro algorithm. Note how the initially unperturbed inner ring is now highly distorted, breaking the radial symmetry of the problem.

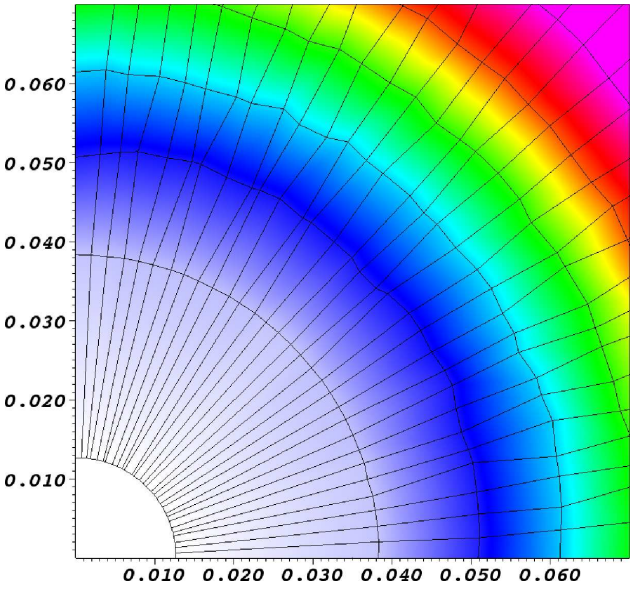

Figure 17: Close-up view of final mesh after 600 cycles using new MFEM hydro algorithm. Note how the initially unperturbed inner ring maintains its radial symmetry even after a roughly 10-fold compression on a randomly distorted mesh.

4.2 The Cylindrical (*x*-*y*) Sedov Problem

Here we investigate the Sedov [27] explosion test problem in planar *x*-*y* mode. The problem consists of an ideal gas ($\gamma = 1.4$) with a delta function source of internal energy deposited at the origin. The sudden release of the energy will create an expanding shock wave, converting the initial internal energy into kinetic energy over time. Total energy should be conserved for all time. We run the problem on a quarter symmetry 60 by 60 zoned Cartesian "box" mesh. The delta function source is approximated by setting the internal energy per mass variable in the corner zone to a large value such that the total integrated energy over the problem domain is $1EU$; however, since we are only meshing a quarter of the entire domain, we need to scale this number by a factor of $1/4$. With this much initial energy, the expanding shock wave should arrive at a radial distance of $r = 1$ at time $t = 1$. For this problem we use the default scalar monotonic artificial viscosity, "`ifqmodel 1`" with "`qlin 0.5`" and "`qquad 0.6666667`" in conjunction with the new MFEM hydro algorithm. In Figure 18 we plot the pressure along with the Lagrangian mesh of the

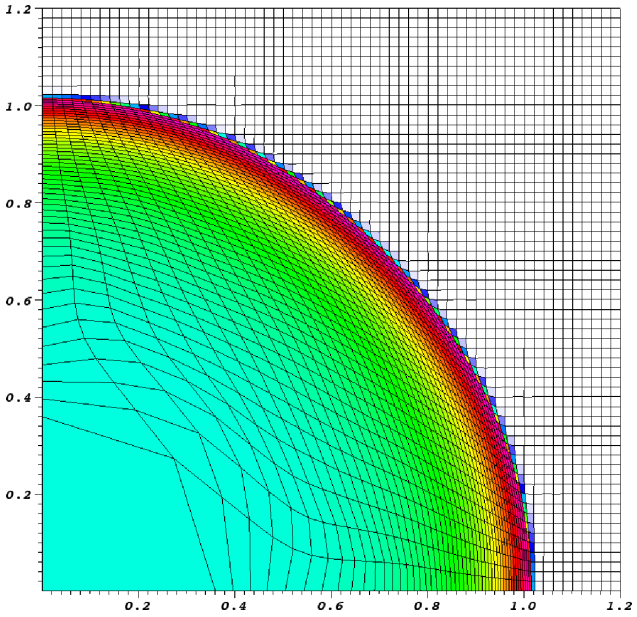

Figure 18: Snapshot of pressure and Lagrangian mesh at final time step using the MFEM hydro algorithm.

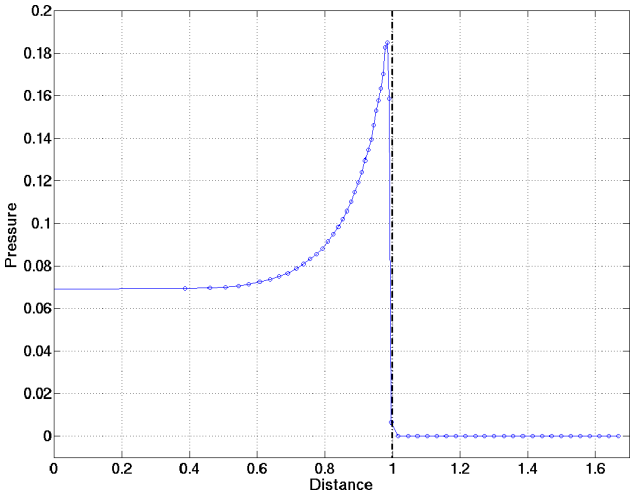

Figure 19: Line-out of pressure at 45 degrees. Note how shock front is in correct location.

problem domain at the final time step. In Figure 19 we plot a "line-out" of the pressure along the 45 degree line. Note that the shock front is very sharp (minimal diffusion caused by the scalar monotonic artificial viscosity) and the shock location is correct. In Figure 20 we plot the kinetic, internal and total energies as a function of time on a log scale in time. Because of the extremely large pressure gradient in early time (due to the delta function energy source), the CFL [28] limited time step is extremely small at early times, thus a log scale is appropriate for viewing the dynamics at early time. Note how total energy is conserved exactly as the initial internal energy is converted to kinetic energy over time. In Figure 21, we compare the total energy as computed by the new MFEM hydro algorithm in comparison to the original HEMP algorithm. Note that total energy is not conserved for the HEMP algorithm (due to the incompatibility of the *PdV* based internal energy equation with the HEMP gradient operator used in the momentum equation.)

4.3 The Spherical (*r*-*z*) Sedov Problem

Here we investigate the Sedov explosion test problem in spherical *r*-*z* mode. The purpose of this example is to illustrate the issues with the MFEM and HEMP formulations in *r*-*z* coordinates as discussed in Section 2.13. The problem is again initialized with a delta function source of energy, the difference this time being that our problem domain is now a sphere.

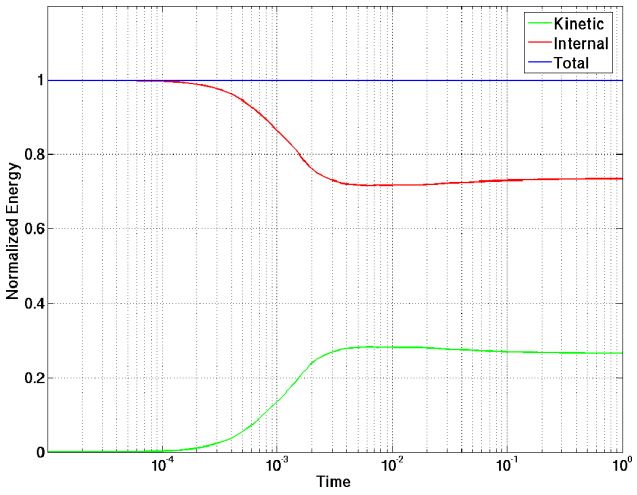

Figure 20: Kinetic, internal and total energies as a function of time on a log scale using the new MFEM hydro algorithm for the cylindrical *x*-*y* Sedov problem.

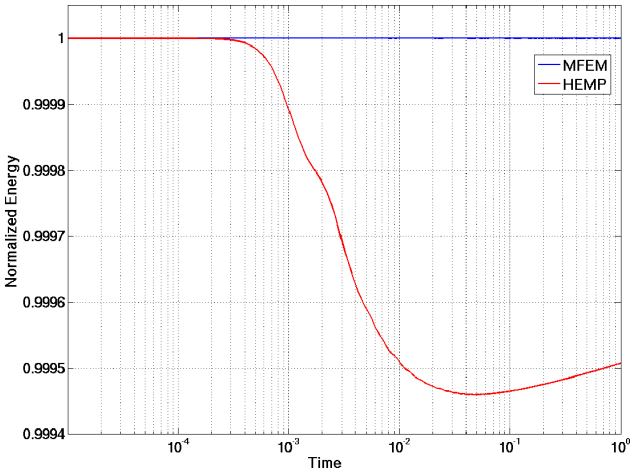

Figure 21: Comparison of total energy as a function of time between the traditional HEMP algorithm and the new MFEM algorithm. Note how the MFEM algorithm conserves energy exactly for all time while the HEMP algorithm loses about 0.06% total energy.

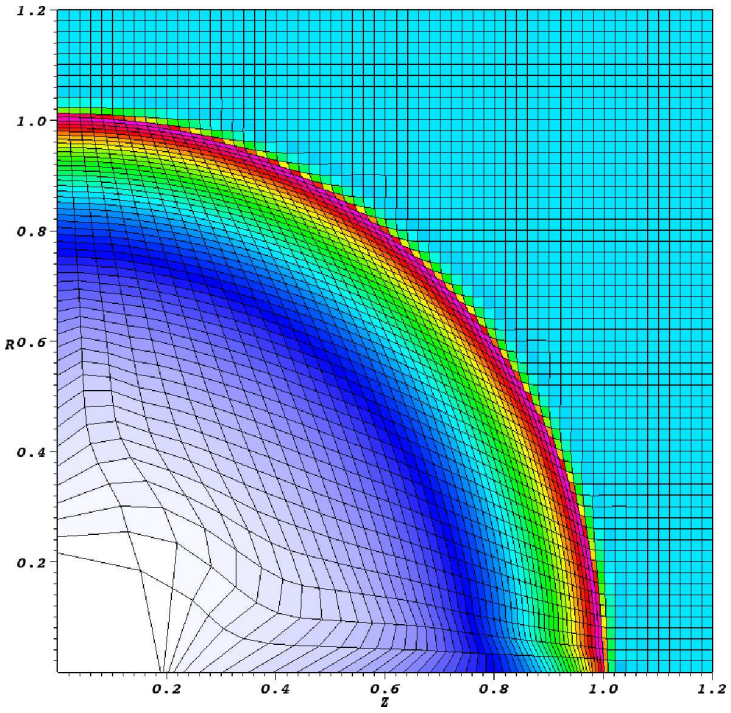

Figure 22: Snapshot of density and Lagrangian mesh at final time step using the MFEM hydro algorithm in *r*-*z* mode. Note the aberration in the density and mesh along the *z*-axis near the shock front.

In Figure 22 we plot the Lagrangian mesh along with the density at the final time step using the MFEM hydro algorithm in *r*-*z* mode. Note the "aberration" in density that appears along the *z*-axis in both the mesh and density. This aberration is most noticeable at the shock front. This is caused by the fundamental (mathematical) error of using a basis defined in planar geometry on a problem in axial geometry as discussed in Section 2.13. The HEMP algorithm does not have this issue because it formulates the *Grad* operator in planar coordinates. However, the HEMP algorithm uses the *dV* approximation for the *Div* operator which leads to large errors in total energy. In Figure 23 we plot the kinetic, internal and total energies as a function of time on a log scale using the MFEM hydro algorithm. Again, note how total energy is conserved exactly as the initial internal energy is converted to kinetic energy over time. In Figure 24, we compare the total energy as computed by the new MFEM hydro algorithm in comparison to the original HEMP algorithm. Note the large gain of roughly 10% in total energy. Again, this is due to the incorrect discretization of the *Div* operator in *r*-*z* coordinates that is used in the HEMP algorithm. It is important to point out that such a large, artificial gain in total energy will affect the speed of the shock wave, causing it to overshoot the final location by a significant amount.

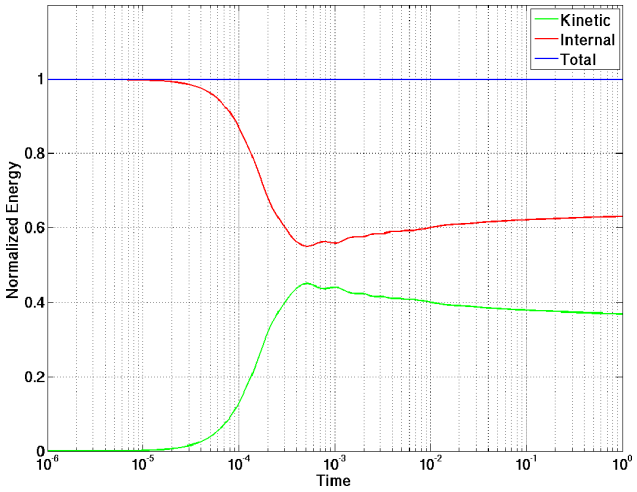

Figure 23: Kinetic, internal and total energies as a function of time on a log scale using the new MFEM hydro algorithm for the spherical *r*-*z* Sedov problem.

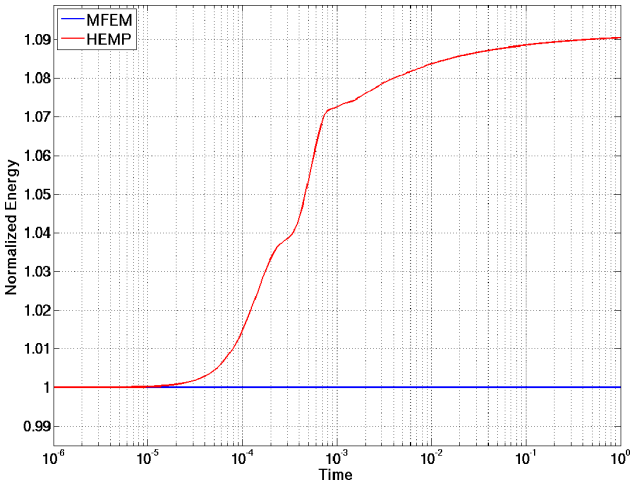

Figure 24: Comparison of total energy as a function of time between the traditional HEMP algorithm and the new MFEM algorithm. Note how the MFEM algorithm conserves energy exactly for all time while the HEMP algorithm gains about 10% total energy.

4.4 The Cylindrical (*x*-*y*) Noh Problem

Here we investigate the Noh [29] implosion test problem. The problem consists of an ideal gas ($\gamma = 5/3$) with a radially directed initial velocity, $\vec{v}(r,0) = -\hat{r}$. This generates a "stagnation" shock wave which propagates radially outward from the origin with a constant speed such that at time $t = 0.6$ the shock front should be at a radial location of $r = 0.2$. Conceptually the opposite of the Sedov explosion test, this test converts initial kinetic energy into internal energy over time, while conserving total energy. We run the problem on a quarter symmetry Cartesian "box" mesh with initially uniform 40 by 40 zoning. In this example we use the new MFEM hydro algorithm in conjunction with the modified "bulk" scalar artificial viscosity that was discussed in Section 2.14 and compare to results obtained using the standard HEMP algorithm in conjunction with the standard "bulk" scalar artificial viscosity, namely "`ifqmodel 3`" with "`qlin 0.25`" and "`qquad 2.0`."

In Figure 25 we plot the density and Lagrangian mesh at the final time step obtained with the standard HEMP algorithm. In Figure 26 we plot the density and Lagrangian mesh at the final time step obtained with the new MFEM algorithm. Both plots use an identical color scale. Note how the post shock density is much more uniform while the "wall heating" and mesh distortion is less significant for the MFEM case. Note also the presence of high frequency "ringing" along the shock front for the HEMP case. In Figure 27 we plot a "line-out" of the density along the 45 degree line. The analytic value of the post-shock density should be

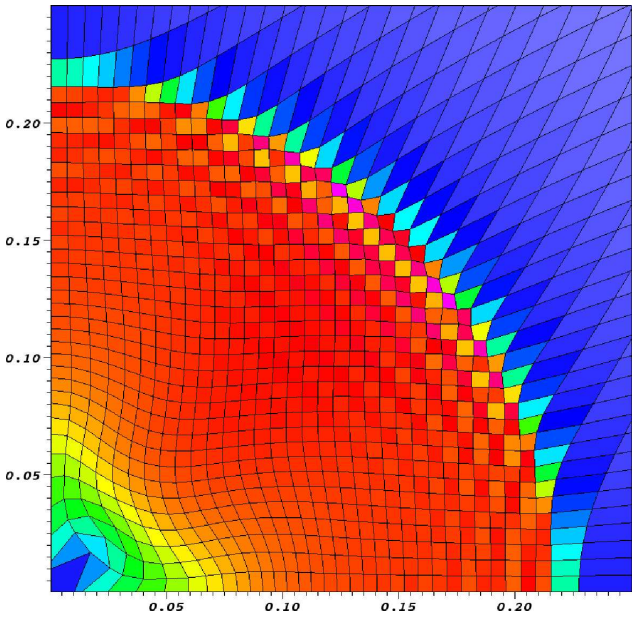

Figure 25: Snapshot of density and Lagrangian mesh at final time step using the standard HEMP hydro algorithm for the cylindrical Noh problem.

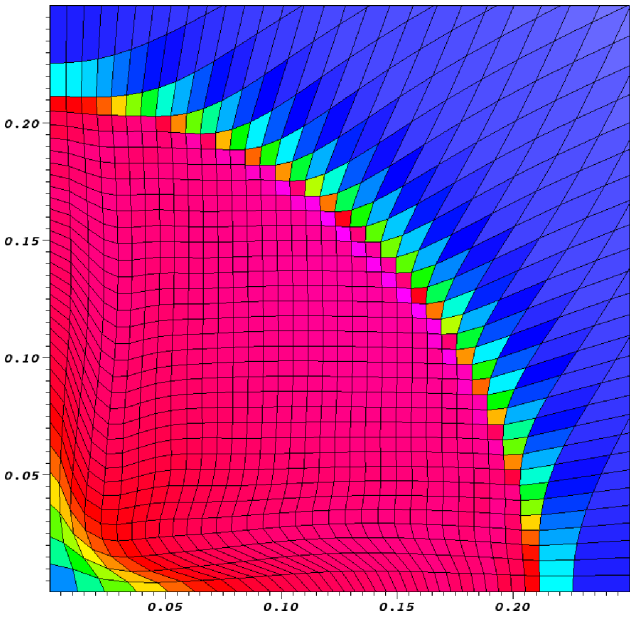

Figure 26: Snapshot of density and Lagrangian mesh at final time step using the new MFEM hydro algorithm for the cylindrical Noh problem.

$\rho = 16$ for the cylindrical Noh problem. Note how the MFEM result is closer to the correct value of 16 and that spurious "ringing" at the shock front is greatly reduced.

In Figure 28 we plot the kinetic, internal and total energies as a function of time on a log scale using the MFEM hydro algorithm. Again, note how total energy is conserved exactly as the initial kinetic energy is converted to internal energy over time. In Figure 29, we compare the total energy as computed by the new MFEM hydro algorithm in comparison to the original HEMP algorithm. Note the small gain of roughly 0.01% in total energy.

4.5 The Planar (*x*-*y*) Saltzmann Piston Problem

The Saltzman piston is a 1D shock tube problem run on a distorted mesh. It is designed to test a code's ability to propagate shock waves along a grid not aligned with the wave. A constant velocity is applied to the boundary nodes at one end of the problem domain for all time, simulating a piston being pushed by an external energy source. This will generate a shock wave that should only travel in the *x*-direction at a constant speed. The initial mesh used for this test is shown in Figure 30. We apply "wall" boundary conditions (i.e. $\vec{v} \cdot \hat{n} = 0$) to the two symmetry planes at $y = 0$ and at $y = 0.1$, and a "fixed" boundary condition ($\vec{v} = 0$) to the end of the piston at $x = 1$.

In this example we use the new MFEM hydro algorithm in conjunction with the modified "bulk" scalar artificial viscosity that was discussed in Section 2.14 as well as the high order nodal reconstruction option, "`mfemnodemethod 2`." It was discovered that the best results could be obtained when using this high order reconstruction method; however, it is discomforting that this method does not universally generate better results for all of the test problems. In Figure 31 we plot the density and Lagrangian mesh at time $t = 0.7$. At this point in time, the post shock density should have a uniform value of $\rho = 4$. In Figure 31 we plot the density and Lagrangian mesh at time $t = 0.8$ where the initial shock wave has bounced off of the fixed wall at $x = 1$ and reversed its propagation direction. At this point in time, the post shock density should have a uniform value of $\rho = 10$.

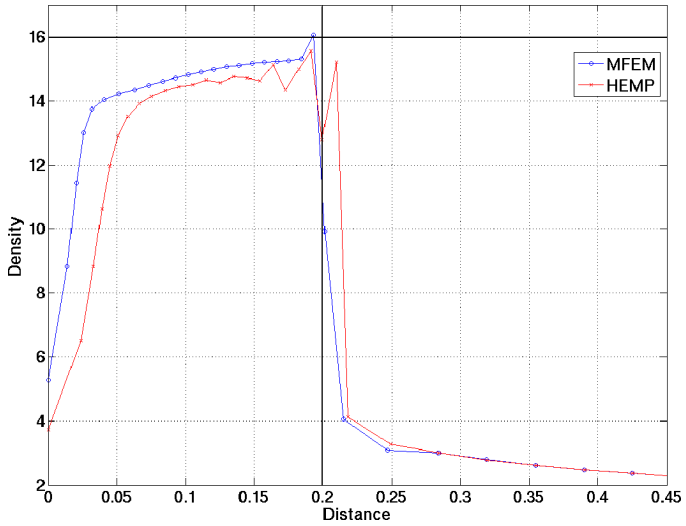

Figure 27: Line-out of density at 45 degrees for both the HEMP and MFEM results. Note how the MFEM result is closer to the correct post-shock density of 16.

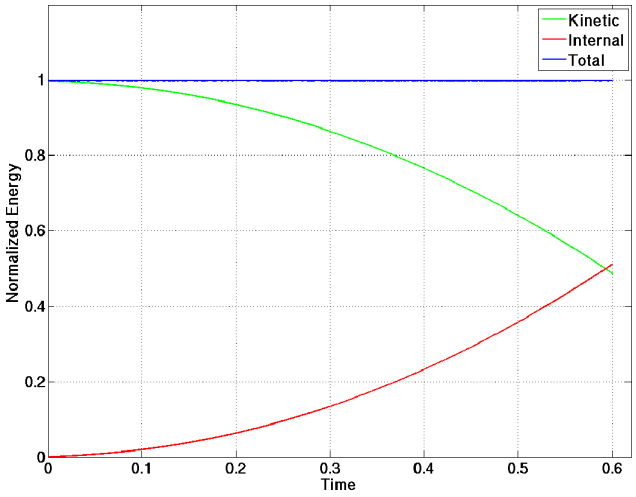

Figure 28: Kinetic, internal and total energies as a function of time using the new MFEM hydro algorithm for the cylindrical *x*-*y* Noh problem.

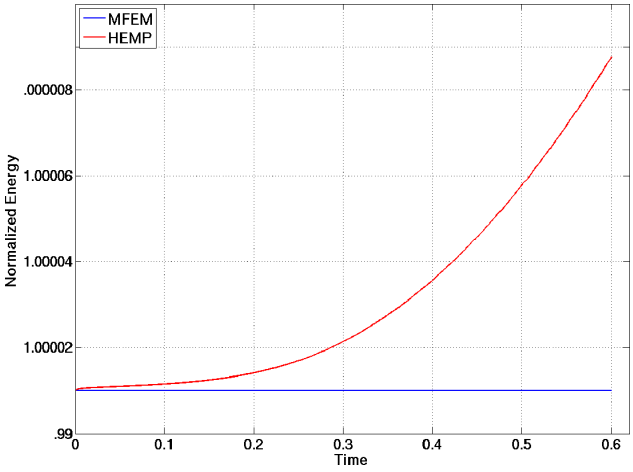

Figure 29: Comparison of total energy as a function time between the traditional HEMP algorithm and the new MFEM algorithm. Note how the MFEM algorithm conserves energy exactly for all time while the HEMP algorithm gains about 0.01% total energy.

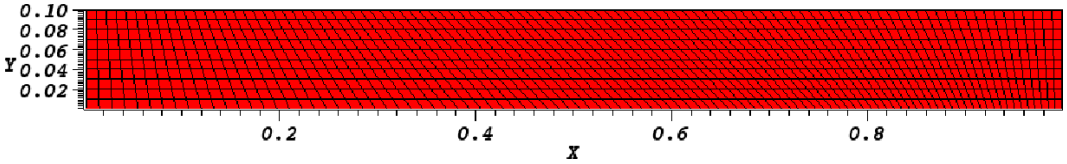

Figure 30: Initial mesh used for the Saltzman piston test.

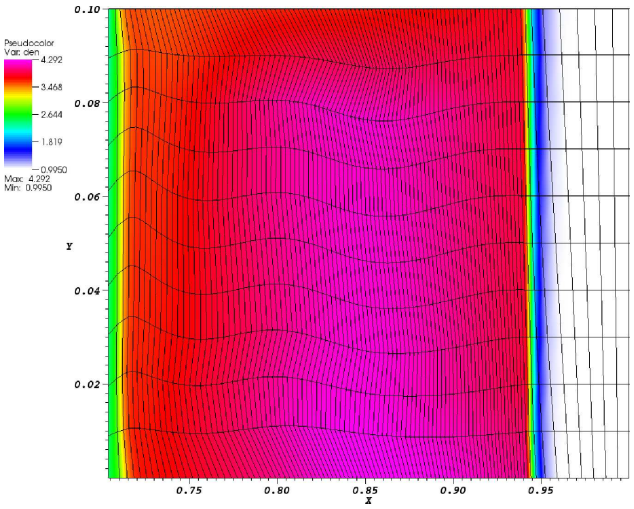

Figure 31: Snapshot of density and Lagrangian mesh for the Saltzman piston problem at time $t = 0.7$ using the new MFEM hydro algorithm.

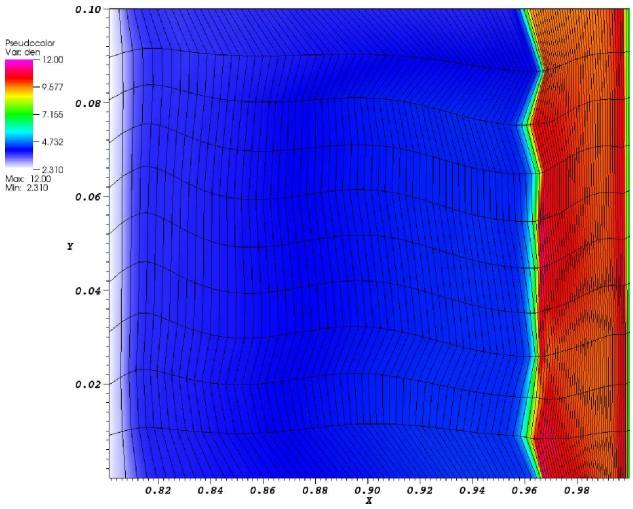

Figure 32: Snapshot of density and Lagrangian mesh for the Saltzman piston problem at time $t = 0.8$ using the new MFEM hydro algorithm.

5 Conclusions and Future Work

In summary, the particular $H(Div)$ MFEM hydrodynamics algorithm presented in this document has some very attractive and promising features. We have shown that accurate pressure gradients can be computed even on distorted grids and that exact energy conservation is possible by discretizing the *Div* and *Grad* operators in a compatible manner. We have identified the fundamental source of hourglass modes as the inability of a discrete *Grad* operator to faithfully reproduce the correct null space of the continuum *Grad* operator (i.e. there exist non-constant "checkerboard" pressure fields for which $\vec{\nabla}P = 0$ in a discrete sense) and that this is a fundamental problem of traditional SGH formulations which use bi-linear representations of velocity at mesh nodes and piece-wise constant pressures at zone centers. We have shown that the MFEM hydro algorithm in combination with a simple modification to the standard bulk artificial viscosity can yield significant improvements in some standard shock hydro test problems.

However, the current $H(Div)$ MFEM hydro algorithm has some limitations that require further investigation. The $H(Div)$ MFEM discrete *Grad* operator does have the correct range and null spaces as its continuum counterpart, but this only provides the normal components of the velocity at mesh faces. In order to move the vertices of the grid during the Lagrange step, we have been forced to develop a nodal averaging process which converts our $H(Div)$ velocity representation into a nodal velocity. Unfortunately, this process is susceptible to hourglass modes as shown in Figure 12. As shown in Section 2.14, the artificial viscosity really needs to be treated as a tensor and not just a simple scalar quantity. Calculating the viscous force vector due to this tensor in multiple dimensions requires a discrete version of the $Div(Grad)$ operator for vector fields. As shown in (48), this operator involves both a *Div* and *Curl*. The $H(Div)$ basis functions have only a well defined divergence (this is essentially the definition of an $H(Div)$ basis) and not a well defined *Curl*, and are therefore not suitable for defining such a differential operator.

These drawbacks suggest that a different choice of MFEM basis functions may be the best solution. We believe that use of the so called Taylor-Hood (a.k.a.$Q2-Q1$ elements) may be a better choice. These basis functions are a stable, high order generalization of the unstable $Q1-Q0$ elements. These elements permit a discretization of the $Div(Grad)$ operator and allow for a natural representation of velocity at mesh vertices. Furthermore, with the right choice of quadrature points, the mass matrix for such a basis can be made diagonal without sacrificing accuracy, thus removing the need for a global linear solve at every Lagrange step. They key obstacle at this point is that this basis requires a higher order representation for pressure, possibly at mesh nodes and is not yet clear how to make this work for Lagrange shock hydrodynamics. Future research into this method should yield some very exciting results.

Regardless of the drawbacks imposed by the particular choice of an $H(Div)$ conforming MFEM basis, a general MFEM approach to Lagrangian hydrodynamics offers several additional advantages that have not been explored in this work, namely:

- Method is naturally valid on unstructured grids
- Method is valid for triangular / tetrahedral elements
- Method is readily extendable to elements with curvilinear boundaries

These advantages will be explored as the current method is developed into maturity. In addition to this, we need to consider the general case of an arbitrary Lagrangian-Eulerian (ALE) discretization and formulate new methods for transport (advection) of state variables during the remap phase; and we need to consider the case of materials with strength (i.e. full stress tensor calculations).

There is much work to be done, but we feel the time is right to make a concerted effort to explore and develop advanced methods for numerical hydrodynamics. To summarize, we feel the following areas warrant further investigation:

- Use of Taylor-Hood elements (and similar variants) for Lagrangian shock hydrodynamics
- Development of a new basis specifically for axisymmetric *r-z* formulation
- Formulation of tensor viscosity in multi-dimensions
- Improved treatment of monotonic limiters in multi-dimensions and unstructured grids
- Use of curvilinear surface elements in Lagrangian shock hydro calculations
- Exploration of potential improvements on unstructured grids using MFEM framework
- Improved time integration methods combined with formal stability analysis

Acknowledgments

I would like to acknowledge the great help of Tzanio Kolev and Panayot Vassilevski for their rigorous mathematical input, their derivations of discrete mass and energy conservation, their ideas concerning the nodal averaging process and their insight into the axisymmetric formulation. This work would not have been possible without their help. I would also like to acknowledge Charles Tong for his help with the custom C-wrapped version of the HYPRE Finite Element Interface. Finally I would like to acknowledge Bob Tipton, Brian Pudliner, B.I. Jun, Tom McAbee, Rob Managan, Misha Shashkov, Pavel Bochev, Andy Barlow and Dan White for their guidance and insight.

References

[1] R. Tipton. Hourglass modes and a new monotonic hourglass filter. Technical report, Lawrence Livermore National Laboratory, December 1999. Unpublished.

[2] E. J. Caramana and M. J. Shashkov. Elimination of artificial grid distortion and hourglass-type motions by means of Lagrangian subzonal masses and pressures. *J. Comput. Phys.*, 142(2):521–561, 1998.

[3] M. L. Wilkins. *Methods in Computational Physics*, volume 3, chapter Calculation of Elastic-Plastic Flow. Academic Press, 1964.

[4] R. Tipton. CALE Lagrange step. Technical report, Lawrence Livermore National Laboratory, October 1990. Unpublished.

[5] J. VonNeumann and R. D. Richtmyer. A method for the numerical calculation of hydrodynamic shocks. *J. Appl. Phys.*, 21:232–237, 1950.

[6] B. VanLeer. Towards the ultimate conservative difference scheme. *J. Comput. Phys.*, 32(1):101–136, 1979.

[7] R. Louberre. On the effect of the different limiters for the tensor artificial viscosity for the compatible lagrangian hydrodynamics scheme. Technical report, Theoretical Division, T-7, Los Alamos National Laboratory, 2005.

[8] R. Rieben. Lagrange hydro step in ARES. In *Proceedings of the Advanced Numerical Methods for Lagrangian Hydrodynamics Workshop*, Lawrence Livermore National Laboratory, Livermore, California, November 2007.

[9] J. C. Campbell and M. J. Shashkov. A tensor artificial viscosity using a mimetic finite difference algorithm. *J. Comput. Phys.*, 172:739–765, 2001.

[10] D. N. Arnold, P. B. Bochev, R. B. Lehoucq, R. A. Nicolaides, and M. J. Shashkov, editors. *Compatible Spatial Discretizations*, volume 142 of *The IMA Volumes in Mathematics and its Applications*. Springer-Verlag, 2006.

[11] P.A. Raviart and J.M. Thomas. A Mixed Finite Element Method for 2^{nd} Order Elliptic Problems. In I. Galligani and E. Mayera, editors, *Mathematical Aspects of the Finite Element Method*, volume 606 of *Lect. Notes. on Mathematics*, pages 293–315. Springer Verlag, 1977.

[12] F. Brezzi, J. Douglas, and L. D. Marini. Two families of mixed finite elements for second order elliptic problems. *Numer. Math.*, 47:217–235, 1985.

[13] R. Falgout et. al. *HYPRE User's Manual*. Lawrence Livermore National Laboratory, Center for Applied Scientific Computing, 2.0 edition, December 2006.

[14] J. E. Morel, R. M. Roberts, and M. J. Shashkov. A local support-operators diffusion discretization scheme. *J. Comput. Phys.*, 144:17–51, 1998.

[15] D. E. Burton. Multidimensional discretizations of conservation laws for unstructured polyhedral grids. Technical Report UCRL-JC-118306, Lawrence Livermore National Laboratory, 1994.

[16] E. J. Caramana, D. E. Burton, M. J. Shashkov, and P. P. Whalen. The construction of compatible hydrodynamics algorithms utilizing conservation of total energy. *J. Comput. Phys.*, 146:227–262, 1998.

[17] R. Rieben, D. White, and G. Rodrigue. A high order mixed vector finite element method for solving the time dependent Maxwell equations on unstructured grids. *J. Comput. Phys.*, 204:490–519, 2005.

[18] R. Rieben and D. White. Verification of high-order mixed FEM solution of transient magnetic diffusion problems. *IEEE Trans. Mag.*, 42(1):25–39, 2006.

[19] P. Bochev. *Least-squares finite element methods for the Stokes and Navier-Stokes equations*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1994.

[20] R. N. Rieben, D. A. White, B. K. Wallin, and J. M. Solberg. An arbitrary Lagrangian-Eulerian discretization of MHD on 3D unstructured grids. *J. Comput. Phys.*, 226:534–570, 2007.

[21] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems.* North-Holland, 1978.

[22] P. Castillo, R. Rieben, and D. White. FEMSTER: An object orienetd class library of higher-order discrete differential forms. *ACM Trans. Math. Soft.*, 31(4):425–457, 2005.

[23] F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods.* Springer Series in Computational Mathematics. Springer Verlag, 1991.

[24] M. F. Wheeler and I. Yotov. *Compatible Spatial Discretizations*, volume 142 of *The IMA Volumes in Mathematics and its Applications*, chapter A Cell-Centered Finite Difference Method on Quadrilaterals. Springer-Verlag, 2006.

[25] D. Kuzmin and S. Turek. High-resolution FEM-TVD schemes based on a fully multidimensional flux limiter. *J. Comput. Phys.*, 198(1):131–158, 2004.

[26] S. V. Coggeshall. Analytic solutions of hydrodynamics equations. *Phys. Fluids A*, 3(5):757–769, 1991.

[27] L. I. Sedov. *Similarity and Dimensional Methods in Mechanics*. CRC Press, tenth edition, 1993.

[28] R. Courant, K. O. Friedrichs, and H. Lewy. Uber die partiellen differenzengleichungen der mathematisches physik. *Math. Ann.*, 100:32–74, 1928.

[29] W. F. Noh. Errors for calculations of strong shocks using an artificial viscosity and an artificial heat flux. *J. Comput. Phys.*, 72(1):78–120, 1987.