ENTROPY VISCOSITY METHOD FOR LAGRANGIAN HYDRODYNAMICS AND CENTRAL SCHEMES FOR MEAN FIELD GAMES

A Dissertation

by

VLADIMIR TOMOV

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Bojan Popov
Co-Chair of Committee,	Jim E. Morel
Committee Members,	Jean-Luc Guermond
	Raytcho Lazarov
Head of Department,	Emil J. Straube

May 2014

Major Subject: Mathematics

ABSTRACT

In this dissertation we consider two major subjects. The primary topic is the Entropy Viscosity method for Lagrangian hydrodynamics, the goal of which is to solve numerically the Euler equations of compressible gas dynamics. The second topic is concerned with applications of second order central differencing schemes to the Mean Field Games equations.

The Entropy Viscosity method discretizes all kinematic and thermodynamic variables by high-order finite elements and solves the resulting discrete problem on a computational mesh that moves with the material velocity. The method is based on two major concepts. The first one is producing high order convergence rates for smooth solutions even with active viscosity terms. This is achieved by using high order finite element spaces and, more importantly, entropy based viscosity coefficients that clearly distinguish between smooth and singular regions. The second concept is providing control over oscillations around contact discontinuities as well as oscillations in shock regions. Achieving this requires adding extra viscosity terms in a way that the resulting system is still in agreement with generalized entropy inequalities, the minimum principle on the specific entropy and the general requirements for artificial tensor viscosities like orthogonal transformation invariance, radial symmetry, Galilean invariance, etc. We define a fully-discrete finite element algorithm and present numerical results on model Lagrangian hydro problems. We also discuss possible extensions of the method, e.g. length scale independent viscosity coefficients, incorporating mass diffusion into the mesh motion, and handling of different materials. In addition we present approaches to the different stages of arbitrary Lagrangian-Eulerian (ALE) methods, which can be used to extend the Entropy Vis-

cosity method. That is, we discuss mesh relaxation by harmonic smoothing schemes, advection based solution remap, and multi-material zones treatment.

The Mean Field Games (MFG) equations describe situations in which a large number of individual players choose their optimal strategy by considering global (but limited) incentive information that is available to everyone. The resulting system consists of a forward Hamilton-Jacobi equation and a backward convection-diffusion equation. We propose fully discrete explicit second order staggered finite difference schemes for the two equations and combine these schemes into a fixed point iteration algorithm. We discuss the second order accuracy of both schemes, their interaction in time, memory issues resulting from the forward-backward coupling, stopping criteria for the fixed point iteration, and parallel performance of the method.

To my family.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Bojan Popov. His mentoring, patience and encouragement made this dissertation possible. I will always remember what he has done for me.

I owe special thanks to Tzanio Kolev, Veselin Dobrev and Robert Rieben for giving me the opportunity to work in their team for the last two summers. This allowed me to learn many new things and to meet many interesting people.

I am also grateful to Dr. Jean-Luc Guermond for all the discussions, his precise comments and constructive criticisms about my main project. He set high research standards and he guided me to meet those standards.

I am indebted to Dr. Raytcho Lazarov for encouraging me to start this Ph. D. program five years ago. I deeply appreciate his belief in me. I greatly value his advices and support.

I would like to acknowledge Dr. Jim Morel, Dr. Marvin Adams, Dr. Wolfgang Bangerth, Dr. Andrea Bonito, and Dr. Vivette Girault for their courses, conversations and comments, which have been crucial to my education and helped me build my perception of nuclear engineering and numerical analysis.

Most importantly, none of this would have been possible without the help of my family. I am deeply grateful to my wife Maria, my two young daughters Borislava and Veronica, my mother Yanka and father Zdravko, and my two brothers Borislav and Stanimir for the constant love, care and support for all these years.

I am also thankful to the former student Dr. Orhan Mehmetoglu for being a great office mate!

Last but not least, I would like to thank the faculty and staff at the Mathemat-

ics Department of Texas A&M University for providing an effective and enjoyable learning environment.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Mean Field Games

The Mean Field Games (MFG) equations describe situations that arise in Economics, Finance or other related subjects, namely a large number of individual players choose their optimal strategy by considering global (but limited) incentive information that is available to everyone. As time evolves, each player's actions alters the incentive information which leads to changes in the players' strategies. The mathematical model of such problems has first been introduced by Lions and Lasry in [40], and MFG video lectures of Lions can be found in [46]. The MFG system in 1D can be written as the following forward-backward system of equations:

$$\frac{\partial u}{\partial t} + H\left(\frac{\partial u}{\partial x}\right) = f(x,m) + \sigma\frac{\partial^2 u}{\partial^2 x}, \tag{1.1}$$

$$\frac{\partial m}{\partial t} + \frac{\partial}{\partial x}\left[H'\left(\frac{\partial u}{\partial x}\right)m\right] = -\sigma\frac{\partial^2 m}{\partial^2 x}, \tag{1.2}$$

$$u(x,0) = u_0(x), \quad m(x,T) = m_T(x), \ m > 0, \int_\Omega m \, dx = 1 \text{ for all } t \in [0,T],$$

where m is a distribution of players, u is an incentive function, and σ is a volatility factor. We give further details and a heuristic derivation of the above equations is Subsection 1.1.1. Notice that equation (1.1) is forward in time, while equation (1.2) is backward in time. Equation (1.1) is a forward Hamilton-Jacobi (FHJ) equation for u with a source and a diffusion term. The source $f(x,m)$ describes how the players' actions affect the incentive information. Equation (1.2) is a backward convection-diffusion (BCD) equation for m with a diffusion term. The advection term $H'\left(\frac{\partial u}{\partial x}\right)m$ describes how the incentive function influences each player's actions.

Central schemes are the standard tool for numerical approximation of hyperbolic conservation equations, the first such scheme is introduced by Lax in [42]. Their main feature is simplicity since they don't involve Riemann solvers, and their structure allows efficient parallelization. Utilizing these schemes for general convection-diffusion equations is straightforward, and by exploiting the general Hamilton-Jacobi equations' relation to conservation laws, see [11], we can apply central schemes to those equations as well.

1.1.1 Physical Motivation

Here we discuss the physical meaning of the system (1.1), (1.2). However we consider the case of having a backward in time equation for u, and a forward equation for m, so that the initial condition is on $m(x,0)$, and the final condition is on $u(x,T)$. Although time direction makes no difference mathematically, the case we discuss is the more intuitive setting.

Suppose we have some domain, say $[0,1]$, and a player (or agent) that is at location $x(0)$ at $t=0$. By moving in the domain, this player wants to minimize a cost function $A(x(T))$, which is defined with respect to every position in the domain at final time $t=T>0$. Furthermore, there is a transportation cost function $B(v(t))$, which is the price at moving with velocity v at time $t\in[0,T]$, $v=x'(t)$. Then total cost is the sum of the final location cost and the transportation cost, namely

$$A(x(T))+\int_0^T B(v(t))\,dt. \tag{1.3}$$

The player's goal is to choose a trajectory $x(t)$ that minimizes the above expression.

Now we define an optimal cost function $u(x_0, t_0)$ by

$$u(x_0, t_0) = \inf \left[A(x(T)) + \int_{t_0}^{T} B(v(t))\ dt \right],$$

where the infimum is taken over all possible paths $x(t), t \in [t_0, T]$, starting from x_0. By this definition we have the final time condition $u(x(T), T) = A(x(T))$. Now we will show that this optimization problem leads to a backward Hamilton-Jacobi equation.

Suppose the player is at position $x(t)$, and he moves with his optimal velocity v for some infinitesimal time dt. By the definition of u we get

$$u(x, t) = u(x + vdt, t + dt) + B(v(t))dt.$$

Taylor expansion to first order for the right-hand side gives

$$u(x, t) = u(x, t) + dt \left(v\frac{\partial u}{\partial x} + \frac{\partial u}{\partial t} \right) + B(v(t))dt. \tag{1.4}$$

Since v was the optimal velocity, then v must be the minimizer of

$$v\frac{\partial u}{\partial x} + B(v(t)). \tag{1.5}$$

Under the assumption that B is convex (see the remark at the end of the subsection), and after using the Legendre transform

$$H(p) = \sup_v vp - B(v), \tag{1.6}$$

it can be derived that the minimum of (1.5) is $-H(\frac{\partial}{\partial x}u(x, t))$, and the velocity that

produces this minimum is $v = -H'(\frac{\partial}{\partial x}u(x,t))$. Then equation (1.4) takes the form

$$\frac{\partial u}{\partial t} - H\left(\frac{\partial u}{\partial x}\right) = 0. \tag{1.7}$$

We may also have noise in the player's motion, e.g. moving from a position $x(t)$ with velocity v for time dt may result in being at $x(t) + vdt + \sigma' dB_t$, where dB_t is infinitesimal Brownian motion, and σ' is the amount of noise. Then the analogue of (1.4) is

$$u(x,t) = u(x,t) + dt\left(v\frac{\partial u}{\partial x} + \frac{\partial u}{\partial t}\right) + B(v(t))dt + \frac{(\sigma')^2}{2}\frac{\partial^2 u}{\partial^2 x}dt,$$

where the additional term does not depend on the velocity v, hence we can repeat the same argument. Then we get a Hamilton-Jacobi equation with a diffusion term, namely

$$\frac{\partial u}{\partial t} - H\left(\frac{\partial u}{\partial x}\right) = -\sigma\frac{\partial^2 u}{\partial^2 x}, \tag{1.8}$$

where $\sigma := \frac{1}{2}(\sigma')^2$. This equation is solved backwards in time since we know $u(x,T)$.

Now suppose the domain contains infinitely many players, all of them trying to minimize the same total cost function. We define $m(x,t)$ to be the normalized density function of players, so that $\int m(x,t)\,dx = 1, \forall t \in [0,T]$. The players' initial positions $m(x,0)$ are known. The players are trying to minimize the same total cost function, hence players that are at the same location all have the same optimal velocity. Then the flux of players at a point x at time t is $m(x,t)v(x,t)$. If we have noise in the players' motion as defined above, then by the Fick's first law the diffusive flux is $-\frac{1}{2}(\sigma')^2\frac{\partial m}{\partial x}$. Then the rate of change in player density for any $[a,b] \subset [0,1]$ is

$$\frac{\partial}{\partial t}\int_a^b m\,dx = \int_a^b \frac{\partial}{\partial x}\left(-mv + \sigma\frac{\partial m}{\partial x}\right)\,dx,$$

leading to the differential form

$$\frac{\partial m}{\partial t} - \frac{\partial}{\partial x}\left(mH'\left(\frac{\partial u}{\partial x}\right)\right) = \sigma\frac{\partial^2 m}{\partial^2 x}, \tag{1.9}$$

where we have used the already derived expression for the optimal velocity, namely $v = -H'(\frac{\partial}{\partial x}u(x,t))$. Equation (1.9) is a convection-diffusion equation, and it is solved forward in time since we know $m(x,0)$.

The final step is to allow the total cost function of each player to depend on the distribution $m(x,t)$ of all other players. This is done by changing (1.3) to

$$A(x(T)) + \int_0^T B(v(t))\ dt + \int_0^T f(m(x(t),t))\ dt,$$

where f is additional cost, depending on the density of players at the current position. For example, if f is an increasing function, then players would prefer to stay away from each other. The new cost function changes equation (1.8) to

$$\frac{\partial u}{\partial t} - H\left(\frac{\partial u}{\partial x}\right) = -f(m) - \sigma\frac{\partial^2 u}{\partial^2 x}. \tag{1.10}$$

Our final system (or Mean Field Game) is equations (1.10) and (1.9) with prescribed data on $u(x,T)$ and $m(x,0)$. The backward equation (1.10) tells us what the players want, and the forward equation (1.9) tells us what they actually get.

An intuitive example for the above situation is the case of fire in a crowded place. Everybody wants to reach the exit (this acts as a location cost function at final time), but reaching the exits is difficult (this acts as a moving cost function).

Remark In order to have well-defined Legendre transform in (1.6), we assume that the movement cost function $B(v)$ is convex and even. If we have a maximization

problem with respect to a movement cost function $B(v)$, then we can make the same derivation as above for the minimization of $-B(v)$ and $-f(x,m)$. However, in the maximization case, $B(v)$ is assumed to be concave.

Example Here we give some physical intuition about the problem presented in Subsection 2.5.1. It uses

$$f(x,m) = -16\left(x-\frac{1}{2}\right)^2 - 0.1\max(0,\min(5,m)),$$
$$H\left(\frac{\partial u}{\partial x}\right) = -\frac{1}{2}\left(\frac{\partial u}{\partial x}\right)^2, \qquad x \in [0,1].$$

In this case H is concave and we have a maximization problem. Then the function f can viewed as an incentive (instead of cost), and f tells us that players have biggest incentive at $x = \frac{1}{2}$. At the same time f is decreasing with respect to m, hence the incentive decreases as the density of players increases. The players' optimal velocity is $-H'(\frac{\partial u}{\partial x}) = \frac{\partial u}{\partial x}$, so that equation (1.9) becomes

$$\frac{\partial m}{\partial t} + \frac{\partial}{\partial x}\left(m\frac{\partial u}{\partial x}\right) = \sigma\frac{\partial^2 m}{\partial^2 x},$$

meaning that the players move towards the increasing values of u. We conclude that players want to be in the middle of the domain, but they prefer to stay away from other players.

1.2 Systems of Hyperbolic Conservation Laws

In this work we consider systems of hyperbolic conservation laws of the form

$$\begin{aligned} \frac{\partial \boldsymbol{v}}{\partial t} + \sum_{m=1}^{d} \frac{\partial \boldsymbol{f^m}(\boldsymbol{v})}{\partial x_m} &= 0, \\ \boldsymbol{v}(\boldsymbol{x},0) &= \boldsymbol{v}_0(\boldsymbol{x}), \end{aligned} \tag{1.11}$$

where $\boldsymbol{x} = (x_1...x_d)$, $\boldsymbol{v}(\boldsymbol{x}, t) = (v_1...v_M)^T$ is the vector of unknowns, and $\boldsymbol{f^1}(\boldsymbol{v})...\boldsymbol{f^d}(\boldsymbol{v})$ are fluxes for each direction, $\boldsymbol{f^m}(\boldsymbol{v}) = (f_1^m...f_M^m)^T$. It is assumed that each $\boldsymbol{f^m}$ is at least C^1. For smooth solution $\boldsymbol{v}$, the system (1.11) can also be written as

$$\frac{\partial \boldsymbol{v}}{\partial t} + \sum_{m=1}^{d} \boldsymbol{A^m}(\boldsymbol{v}) \frac{\partial \boldsymbol{v}}{\partial x_m} = 0, \tag{1.12}$$

where

$$\boldsymbol{A^m}(\boldsymbol{v}) = \boldsymbol{f_v^m}, \qquad A_{ij}^m = \frac{\partial f_i^m}{\partial v_j}. \tag{1.13}$$

The system (1.11) is called hyperbolic if the matrix

$$\sum_{m=1}^{d} k_m \boldsymbol{A^m}(\boldsymbol{v}) \tag{1.14}$$

has real eigenvalues and a complete set of eigenvectors for all real k_m and every $\boldsymbol{v}$. This means that the system can be linearized about a fixed solution $\boldsymbol{v}$ and, hence, hyperbolicity acts as a necessary condition for well posedness of the system with initial data near $\boldsymbol{v}$.

1.2.1 Convention for Vector and Tensor Operations

We use the usual convention for vector and tensor operations. That is, for column vectors $\boldsymbol{a} = (a_1...a_d)^T, \boldsymbol{b} = (b_1...b_d)^T$ and order 2 tensors $\mathbb{g}, \mathbb{h}$ with entries g_{ij}, h_{ij} where $i, j = 1...d$ we have:

$$\boldsymbol{a} \otimes \boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^T, \qquad (\nabla \boldsymbol{a})_{ij} = \frac{\partial a_j}{\partial x_i} \qquad (\nabla \cdot \mathbb{g})_i = \sum_{j=1}^{d} \frac{\partial g_{ji}}{\partial x_j},$$

$$(\boldsymbol{a} \cdot \mathbb{g})_i = \sum_{j=1}^{d} a_j g_{ji}, \qquad (\mathbb{g} \cdot \boldsymbol{a})_i = \sum_{j=1}^{d} g_{ij} a_j, \qquad \mathbb{g} : \mathbb{h} = \sum_{ij} g_{ij} h_{ij}.$$

The following standard identities are going to be used throughout this thesis:

$$\nabla \cdot (\boldsymbol{a} \otimes \boldsymbol{b}) = \boldsymbol{a} \cdot \nabla \boldsymbol{b} + \boldsymbol{b} \nabla \cdot \boldsymbol{a}, \qquad \nabla \cdot (\mathbb{g} \cdot \boldsymbol{a}) = \mathbb{g} : \nabla \boldsymbol{a} + (\nabla \cdot \mathbb{g}) \cdot \boldsymbol{a},$$

$$\left[\nabla \left(\frac{\boldsymbol{a}^2}{2} \right) \right] \cdot \boldsymbol{b} = \boldsymbol{a} \cdot (\boldsymbol{b} \cdot \nabla \boldsymbol{a}) .$$

1.2.2 Characteristic Curves and Loss of Regularity

In this subsection we make a brief overview of some specific features of solutions to (1.11). Suppose we have smooth initial data so that we can write the system (1.11) in the form (1.12). Then looking at the $\mathbb{R}^d \times \mathbb{R}$ space-time plane, for any $i = 1...M$, the unknown v_i is constant along the characteristic curves

$$\left(\frac{\partial f_i^1(\boldsymbol{v})}{\partial x_1} \ ... \ \frac{\partial f_i^d(\boldsymbol{v})}{\partial x_d}, 1 \right)$$

since its derivative in this direction is zero. Regions where different characteristic curves approach each other correspond to compression, and regions where the different characteristic curves move away from each other correspond to expansion. However if characteristics start to intersect, we are led to a multivalued solution. Since multivalued solutions are not physical, one has to define a discontinuous solution according to some physical considerations. These discontinuities are called shocks.

Example A typical example for crossing of characteristics is the scalar 1D Burgers' equation. In this example we start with smooth data, but the solutions becomes discontinuous in finite time:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} \left(\frac{u^2}{2} \right) = 0, \qquad u(x,0) = \frac{1}{1+x^2}.$$

For smooth u we have

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0,$$

meaning that u is constant on the characteristic curves $(u, 1)$. We can express these curves in terms of x and t by using

$$\frac{\partial x(t)}{\partial t} = u(x(t), t) = u(x, 0) \Rightarrow x(t) = x + tu(x, 0),$$

hence the solution u can be written as

$$u\left(x + \frac{t}{1+x^2}, t\right) = \frac{1}{1+x^2}.$$

However if we take two of these curves, say the ones starting at $x = 0$ and $x = 1$, we see that they intersect at $t = 2$, leading to a multi-valued solution (actually the first intersection between any two characteristics occurs at earlier time).

A contact regions is a region where the solution is discontinuous, but the underlying characteristic curves are parallel. In this case there is neither compression, nor expansion.

Example The simples example of a contact discontinuity is the linear transport equation

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \qquad u(x, 0) = \begin{cases} u_l & \text{if } x < 0, \\ u_r & \text{if } x > 0. \end{cases}$$

where $a > 0$ is a constant. The characteristic lines are $x(t) = x + at$ and the solution is

$$u(x, t) = \begin{cases} u_l & \text{if } x < at, \\ u_r & \text{if } x > at. \end{cases}$$

The solution moves and remains discontinuous, but there is no compression or expansion.

1.2.3 Weak Solutions

Hyperbolicity, together with additional regularity requirements on $\boldsymbol{A^m}$, can be used to build local in time, continuously differentiable solutions, see [50]. Due to the nonlinearities of (1.11), however, as discussed in the previous subsection, solutions which are initially smooth may become discontinuous within finite time. In order to define solutions globally in time we remove continuity assumptions on $\boldsymbol{v}$ by considering the equations in distributional sense.

Definition 1.2.1 *A function $\boldsymbol{v} \in L^1_{loc}(\mathbb{R}^d \times [0,\infty))$ is a weak solution of (1.11), if*

$$\int_0^\infty \int_{\mathbb{R}^d} \left(\frac{\partial \boldsymbol{w}}{\partial t} \cdot \boldsymbol{v} + \sum_{m=1}^{d} \frac{\partial \boldsymbol{w}}{\partial x_m} \cdot \boldsymbol{f^m}(\boldsymbol{v}) \right) d\boldsymbol{x}dt = 0 \tag{1.15}$$

for every smooth vector function $\boldsymbol{w}(\boldsymbol{x},t)$ of M components with compact support within $\mathbb{R}^d \times (0,\infty)$.

Remark For simplicity, equation (1.15) does not take into account initial and boundary conditions.

The concept of weak solutions does not guarantee uniqueness. In some cases the weak form (1.15) can admit infinitely many weak solutions.

Example We consider the scalar 1D Burgers' equation

$$\frac{\partial u}{\partial t} + \left(\frac{u^2}{2} \right)_x = 0, \qquad u(x,0) = \begin{cases} 0 & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases}$$

Then we can show that for every $0 < \alpha < 1$, a weak solution is

$$u_\alpha(x,t) = \begin{cases} 0 & \text{if } x < \frac{\alpha t}{2}, \\ \alpha & \text{if } \frac{\alpha t}{2} \leq x < \frac{(1+\alpha)t}{2}, \\ 1 & \text{if } x \geq \frac{(1+\alpha)t}{2}. \end{cases}$$

Since this is a 1D problem, we can check whether the above functions are weak solutions by verifying that equation (1.11) is satisfied in the smooth regions (which in this case is trivial), and checking the Rankine-Hugoniot jump conditions on the two lines of discontinuity. These conditions state that if we have a shock on the line $x = \eta t$ so that

$$u(x,t) = \begin{cases} u^+ & \text{if } x > \eta t, \\ u^- & \text{if } x < \eta t, \end{cases}$$

then the shock speed η is given by

$$\eta = \frac{f(u^+) - f(u^-)}{u^+ - u^-}. \tag{1.16}$$

All of the given functions contain two shocks with speeds $\eta_1 = \frac{\alpha}{2}, \eta_2 = \frac{(1+\alpha)}{2}$ for which the Rankine-Hugoniot conditions hold, hence all of them are weak solutions.

1.2.4 Viscosity Limit and Entropy Inequality

The non-uniqueness of weak solutions requires a mechanism for determining whether a weak solution is physical. A weak solution $\boldsymbol{v}$ of (1.11) is called admissible in the vanishing viscosity sense if it is the limit of solutions $\boldsymbol{v}^\delta$ in L^1_{loc} as $\delta \to 0$ where

$\boldsymbol{v}^\delta$ are the solutions of the viscous equation

$$\frac{\partial \boldsymbol{v}^\delta}{\partial t} + \nabla \cdot \boldsymbol{f}(\boldsymbol{v}^\delta) = \delta \Delta \boldsymbol{v}^\delta. \tag{1.17}$$

If continuously differentiable solutions of (1.17) can be proved to converge in L^1_{loc}, then one can show that the limit will satisfy (1.15). While the convergence has been established for the cases of $M = d = 1$ [37, 42] and $M = 2, d = 1$ [13, 14, 15], the result for the case of a general system is still an open problem. However, we assume the solutions of (1.17) converge as $\delta \to 0$ and we use the limit as our criterion for a "physical" weak solution.

With this assumption, equation (1.17) is used to deduce other conditions which are easier to verify numerically. To do that we first define a function called "entropy", which is a generalization of the thermodynamical entropy.

Definition 1.2.2 *A function $S(\boldsymbol{v})$ is an entropy function for the system* (1.11)*, if:*

1. *S satisfies $S^T_{\boldsymbol{v}}(\boldsymbol{f}^{\boldsymbol{m}}_{\boldsymbol{v}}) = (F^m_{\boldsymbol{v}})^T, m = 1...d$, where the vector $\boldsymbol{F}(\boldsymbol{v})$ is called entropy flux.*

2. *S is a convex function of $\boldsymbol{v}$.*

Now we use this definition in order to derive the so-called "entropy inequality" admissibility condition originally introduced by Lax [43], Krushkov [37]. Let $\boldsymbol{v}^\delta$ be a continuously differentiable solution of (1.17). Then we multiply (1.17) by $\left(\frac{\partial S}{\partial \boldsymbol{v}^\delta}\right)^T$ on the left to obtain

$$\left(\frac{\partial S}{\partial \boldsymbol{v}^\delta}\right)^T \frac{\partial \boldsymbol{v}^\delta}{\partial t} + \sum_{m=1}^{d} \left(\frac{\partial S}{\partial \boldsymbol{v}^\delta}\right)^T \boldsymbol{f}^{\boldsymbol{m}}_{\boldsymbol{v}} \boldsymbol{v}^\delta_{x_m} = \delta \sum_{m=1}^{d} \left(\frac{\partial S}{\partial \boldsymbol{v}^\delta}\right)^T \frac{\partial^2 \boldsymbol{v}^\delta}{\partial^2 x_m}, \tag{1.18}$$

where for simplicity we have split the right-hand side into d one-dimensional expressions. Note that

$$\frac{\partial^2 S(\boldsymbol{v}^\delta)}{\partial^2 x_m} = \frac{\partial}{\partial x_m}\left(\frac{\partial S}{\partial \boldsymbol{v}^\delta}\cdot\frac{\partial \boldsymbol{v}^\delta}{\partial x_m}\right) = \left(\frac{\partial S}{\partial \boldsymbol{v}^\delta}\right)^T \frac{\partial^2 \boldsymbol{v}^\delta}{\partial^2 x_m} + \left(\frac{\partial \boldsymbol{v}^\delta}{\partial x_m}\right)^T \frac{\partial^2 S}{\partial^2 \boldsymbol{v}^\delta}\frac{\partial \boldsymbol{v}^\delta}{\partial x_m}.$$

Then (1.18) becomes

$$\frac{\partial S(\boldsymbol{v}^\delta)}{\partial t} + \sum_{m=1}^{d}(F_{\boldsymbol{v}}^m)^T \boldsymbol{v}_{x_m}^\delta = \delta \sum_{m=1}^{d}\left[\frac{\partial^2 S(\boldsymbol{v}^\delta)}{\partial^2 x_m} - \left(\frac{\partial \boldsymbol{v}^\delta}{\partial x_m}\right)^T \frac{\partial^2 S}{\partial^2 \boldsymbol{v}^\delta}\frac{\partial \boldsymbol{v}^\delta}{\partial x_m}\right],$$

where the right-hand side's last term is nonnegative since $S(\boldsymbol{v}^\delta)$ is convex, it has a positive semidefinite Hessian. Hence

$$\frac{\partial S(\boldsymbol{v}^\delta)}{\partial t} + \sum_{m=1}^{d}\frac{\partial \boldsymbol{F}^m(\boldsymbol{v}^\delta)}{\partial x_m} \le \delta \sum_{m=1}^{d}\frac{\partial^2 S(\boldsymbol{v}^\delta)}{\partial^2 x_m}.$$

Now we go to a weak form in order to pass to the limit $\delta \to 0$. We multiply the last equation by a nonnegative scalar smooth function φ with compact support and integrate by parts to obtain

$$\int\int S(\boldsymbol{v}^\delta)\frac{\partial \varphi}{\partial t} + \boldsymbol{F}(\boldsymbol{v}^\delta)\cdot\nabla\varphi \; d\boldsymbol{x}dt \ge -\delta \int\int \sum_{m=1}^{d} S(\boldsymbol{v}^\delta)\frac{\partial^2 \varphi}{\partial^2 x_m}\; d\boldsymbol{x}dt.$$

If $\boldsymbol{v}^\delta \to \boldsymbol{v}$ in L^1_{loc} as $\delta \to 0$, then we get

$$\int\int S(\boldsymbol{v}^\delta)\frac{\partial \varphi}{\partial t} + \boldsymbol{F}(\boldsymbol{v}^\delta)\cdot\nabla\varphi \; d\boldsymbol{x}dt \ge 0$$

for every smooth nonnegative test function φ. This leads to the following admissibility condition:

Entropy inequality: A weak solution of (1.11) is admissible (or entropy-admissible)

if it satisfies the inequality

$$\frac{\partial S(\boldsymbol{v})}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{v}) \leq 0 \tag{1.19}$$

in the sense of distributions for every pair of convex entropy and entropy flux $(S, \boldsymbol{F})$.

Remark In the scalar case, it was shown in [4] that entropy-admissible solutions exists and are unique.

1.3 The Compressible Euler Equations

In this subsection we state the Euler equations of compressible gas dynamics and explain the physical interpretation of each equation. The dependent variables in the system are the material density ρ (mass/volume), the material velocity $\boldsymbol{u}$ (length/time) and the material specific energy e (energy/mass). The system consists of the following equations:

$$\frac{\partial}{\partial t}\rho + \nabla \cdot (\boldsymbol{u}\rho) = 0, \tag{1.20}$$

$$\frac{\partial}{\partial t}(\rho \boldsymbol{u}) + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) + \nabla p = 0, \tag{1.21}$$

$$\frac{\partial}{\partial t}\left(\rho e + \frac{1}{2}\rho \boldsymbol{u}^2\right) + \nabla \cdot \left[\boldsymbol{u}\left(\rho e + \frac{1}{2}\rho \boldsymbol{u}^2 + p\right)\right] = 0, \tag{1.22}$$

The material's pressure p (force/area) and temperature T are determined by equation of state (EOS), which relates them to the material's density and specific internal energy:

$$p = p(\rho, e).$$

In this work we use ideal gas equation of state, which gives the following p and T:

$$p = (\gamma - 1)\rho e, \qquad T = (\gamma - 1)e. \tag{1.23}$$

where γ is the ratio of constant pressure and constant volume heat capacities. Instead of using the equation for the total energy (1.22), a common practice is to work with the equation for the internal energy. It is derived by taking a dot product of (1.21) with $\boldsymbol{u}$ and subtracting the result from (1.22). The resulting equation is

$$\frac{\partial}{\partial t}(\rho e) + \nabla \cdot (\boldsymbol{u}\rho e) + p\nabla \cdot \boldsymbol{u} = 0. \tag{1.24}$$

As we see in the following subsections, equations (1.20) - (1.22) express the universal conservation principle that the time rate of change of some conserved quantity within a control volume is equal to the source rates minus the sink rates. Equations (1.20), (1.21), (1.22) stand for mass, momentum and total energy conservation, respectively.

1.3.1 Conservation Principles

1.3.1.1 Conservation of Mass

Now we explain the physical meaning of equation (1.20). We integrate (1.20) over a control volume V and obtain

$$\frac{\partial}{\partial t}\int_V \rho \, dV = -\oint \rho \boldsymbol{u} \cdot \boldsymbol{n} \, dA, \tag{1.25}$$

where $\boldsymbol{n}$ is the surface's outward unit normal. The left-hand side in (1.25) represents the rate of change of the mass within V. The term $\rho\boldsymbol{u}$ (mass/area-time) is called mass flux. The right-hand side of (1.25) gives the net rate at which mass enters the control volume V, namely it's incoming mass minus outgoing mass per unit time.

1.3.1.2 Conservation of Momentum

Here we explain the physical meaning of equation (1.21). We integrate (1.21) over a control volume V and obtain

$$\frac{\partial}{\partial t}\int_V \rho u_m \, dV = -\oint \rho u_m \boldsymbol{u}\cdot\boldsymbol{n} \, dA - \int_V \frac{\partial}{\partial x_m} p \, dV, \qquad m = 1...d. \tag{1.26}$$

The left-hand side has units (momentum/time) and it represents the rate of change of the momentum in direction m within V. The term $\rho u_m \boldsymbol{u}$ (momentum/area-time) is momentum flux in direction m, hence the right-hand side's first term is the net rate at which the m-th component of momentum enters the volume. The spatial derivative of p has units (force/volume), hence the right-hand side's second term is the total force applied on the material in V. But force has units of (momentum/time), hence that term can be viewed as total rate of change of momentum within the control volume V due to pressure gradients. We conclude that the right-hand side of (1.26) is the rate of change of momentum due to the boundary flux and pressure gradients.

1.3.1.3 Conservation of Total Energy

Here we explain the physical meaning of equation (1.22). We integrate (1.22) over a control volume V and obtain

$$\frac{\partial}{\partial t}\int_V \left(\rho e + \frac{1}{2}\rho \boldsymbol{u}^2\right) \, dV = -\oint (\rho e + \frac{1}{2}\rho \boldsymbol{u}^2)\boldsymbol{u}\cdot\boldsymbol{n} \, dA - \int_V \nabla\cdot(p\boldsymbol{u}) \, dV. \tag{1.27}$$

The left-hand side has units (energy/time) and it represents the rate of change of the total energy within V. The terms $\rho e \boldsymbol{u}$ and $(\frac{1}{2}\rho \boldsymbol{u}^2)\boldsymbol{u}$ are respectively internal energy flux and kinetic energy flux, hence the right-hand side's first term is the net rate at which internal and kinetic energy enter the volume V. In order to understand the

second term in the right-hand side of (1.27), we split it in two parts by

$$-\nabla \cdot (p\boldsymbol{u})\ dV = -\nabla p \cdot \boldsymbol{u}\ dV - p\nabla \cdot \boldsymbol{u}\ dV.$$

As discussed after equation (1.26), the term $-\nabla p\ dV$ is the total force applied to the material inside dV. Then $-\nabla p \cdot \boldsymbol{u}\ dV$ becomes the kinetic energy rate of change for the material inside dV (using that if a force $\boldsymbol{F}$ is applied to a rigid body, the change of kinetic energy of that body over a path $\boldsymbol{s}$ is $\boldsymbol{F} \cdot \boldsymbol{s}$). The term $\nabla \cdot \boldsymbol{u}\ dV$ has units (volume/time) and represents the volume change of dV due to the material motion. Volume changes of dV cause changes in its internal energy. The term $p\nabla \cdot \boldsymbol{u}\ dV$ represents the internal energy change due to compression or expansion of the material. We conclude that the right-hand side of (1.27) represents the rate of change of total energy due to boundary fluxes of internal and kinetic energy, changes of kinetic energy due to pressure gradients, and changes in internal energy arising from compression or expansion of the material.

1.3.2 Entropy Quantities

Now we derive an entropy S that is in agreement with Definition 1.2.2. We note that the first condition of that definition is equivalent (after taking a dot product of (1.11) with $S_{\boldsymbol{v}}$) to:

$$\frac{\partial S}{\partial t} + \nabla \cdot \boldsymbol{F} = 0 \tag{1.28}$$

for every smooth solution $\boldsymbol{v}$.

We follow the approach first introduced by Harten in [29]. For EOS $p = (\gamma - 1)\rho e$ and $T = (\gamma - 1)e$, we define the quantity

$$s := \log(p\rho^{-\gamma}) = \log((\gamma - 1)e\rho^{1-\gamma}) = \log((\gamma - 1)e) + (1 - \gamma)\log\rho, \tag{1.29}$$

which we call "specific entropy". Notice that this is not exactly the physical specific entropy (it doesn't satisfy $Tds = de - \frac{p}{\rho^2}d\rho$). Its derivatives with respect to density and energy are

$$\frac{\partial s}{\partial \rho} = \frac{1-\gamma}{\rho}, \quad \frac{\partial s}{\partial e} = \frac{1}{e}. \tag{1.30}$$

Assuming a smooth solution, multiply (1.20) by $\frac{\partial s}{\partial \rho}$, (1.24) by $\frac{\partial s}{\partial e}$ and add the resulting equations:

$$\left(\frac{\partial s}{\partial \rho}\frac{\partial \rho}{\partial t} + \frac{\partial s}{\partial e}\frac{\partial e}{\partial t}\right) + \boldsymbol{u}\cdot\left(\frac{\partial s}{\partial \rho}\nabla\rho + \frac{\partial s}{\partial e}\nabla e\right) + \nabla\cdot\boldsymbol{u}(\cancel{\frac{\partial s}{\partial \rho}\rho + \frac{\partial s}{\partial e}T}) = 0,$$

$$\frac{\partial s}{\partial t} + \boldsymbol{u}\cdot\nabla s = 0.$$

Now we multiply by $f'(s)$ where f is any scalar differentiable function of s to obtain

$$\frac{\partial f(s)}{\partial t} + \boldsymbol{u}\cdot\nabla f(s) = 0.$$

Then we multiply the last equation by ρ, (1.20) by $f(s)$ and add them to obtain

$$\frac{\partial}{\partial t}(\rho f(s)) + \nabla\cdot(\boldsymbol{u}\rho f(s)) = 0. \tag{1.31}$$

Then taking

$$S := -\rho f(s), \qquad \boldsymbol{F} := -\boldsymbol{u}\rho f(s)$$

satisfies the condition (1.28). This S will be convex, if and only if (see [30]):

$$f'(s) > 0, \quad f'(s)\frac{\gamma-1}{\gamma} + f''(s) > 0. \tag{1.32}$$

A family of functionals that satisfies (1.32) is:

$$f(s) = \frac{\gamma+\alpha}{\gamma-1}\exp\left(\frac{s}{\gamma+\alpha}\right) \Rightarrow f'(s) = \frac{1}{\rho(\gamma-1)}(p\rho^{\alpha})^{\frac{1}{\gamma+\alpha}}$$

where $\alpha > 0$ is some constant. Then we have the following family of entropies:

$$S = -\rho f(s) = \frac{\gamma+\alpha}{1-\gamma}(p\rho^{\alpha})^{\frac{1}{\gamma+\alpha}}, \tag{1.33}$$

so that S satisfies both conditions of Definition 1.2.2 for any $\alpha > 0$.

Remark If we had a source Q of internal energy, then (1.31) becomes:

$$\frac{\partial}{\partial t}(\rho f(s)) + \nabla\cdot(\rho f(s)) - Qs_e f'(s) = 0,$$

$$\frac{\partial S}{\partial t} + \nabla\cdot S + \frac{Qf'(s)}{e} = 0. \tag{1.34}$$

Remark The amount of violation of (1.31) or (1.34) is called "entropy production". It could be non-zero only in regions of discontinuities of the solution. We use this quantity to scale our viscosity terms, it is our shock detector (and hence the name "Entropy Viscosity method").

Remark We usually choose $\alpha = 1$ in the definition of S (1.33). For smaller values of α the explicit density dependence decreases, for example $\alpha = 0$ would imply that S doesn't change in contact regions (since the pressure doesn't change). In that case the entropy production there is zero and we don't add viscosity in contacts. However the choice of $\alpha = 0$ makes S not strictly convex and hence the entropy inequality (1.19) is invalid.

Remark We have a minus sign in the definitions of S and F, because we want to be consistent with (1.32). The specific entropy used by Harten and Lax in [30] is the

one from (1.29) multiplied by a negative constant, namely it is $\log(\rho e^{1/(1-\gamma)})$ (which is minus the physical specific entropy).

1.3.3 Minimum Principle on the Specific Entropy

In this subsection we derive an additional admissibility condition for the Euler system, namely the minimum principle on the specific entropy, originally derived by Tadmor in [58]. It states the following:

Minimum principle on the specific entropy: If $\boldsymbol{v}(\boldsymbol{x},t)$ is a weak solution of (1.11) that satisfies the entropy inequality (1.19), then for the specific entropy s defined in (1.29) we have

$$\operatorname*{Ess\,inf}_{\boldsymbol{x}\in\Omega} s(\boldsymbol{x},t) \geq \operatorname*{Ess\,inf}_{\boldsymbol{x}\in\Omega+tU} s(\boldsymbol{x},0), \tag{1.35}$$

for all $t>0$ and any domain Ω, where U is the maximal speed $|\boldsymbol{u}|$ in Ω. We go over the derivation of this admissibility condition for completeness.

First we show that if $\boldsymbol{v}(\boldsymbol{x},t)$ is a weak solution of (1.11) that satisfies the entropy inequality (1.19), then for non-positive functions $f(s)$ satisfying (1.32), we have

$$\int_{\boldsymbol{x}\in\Omega} \rho(\boldsymbol{x},t)h(s(\boldsymbol{x},t))\ d\boldsymbol{x} \geq \int_{\boldsymbol{x}\in\Omega+tU} \rho(\boldsymbol{x},0)h(s(\boldsymbol{x},0))\ d\boldsymbol{x}. \tag{1.36}$$

To show this we integrate the entropy inequality (1.19) over the space-time cone

$$C = \{\boldsymbol{x}\in\Omega+(t-\tau)U \mid 0\leq\tau\leq t\}.$$

Recall that $\boldsymbol{F} = -\boldsymbol{u}\rho h(s)$, then the resulting inequality is

$$\int_C \frac{\partial \rho h(s)}{\partial t} + \nabla\cdot(\boldsymbol{u}\rho h(s))\ d\boldsymbol{x}d\tau \geq 0,$$

$$\int_C \nabla^* \cdot (\boldsymbol{u}\rho h(s), \rho h(s))^T \; d\boldsymbol{x}d\tau \geq 0,$$

where in the last integral we have the divergence in the space-time plane. Hence we obtain

$$\int_{\partial C} \rho h(s) \left(n_0 + \sum_{m=1}^{d} u_m n_m \right) \; d\boldsymbol{x}d\tau \geq 0, \tag{1.37}$$

where $\boldsymbol{n}^* = (n_1, ...n_d, n_0)^T$ is the unit outward normal to the boundary of C, n_0 is the component in the time direction. At $\tau = 0$ we have $\boldsymbol{n}^* = (0, ...0, -1)$ and the integral in (1.37) becomes the negative of the right-hand side of (1.36). At $\tau = t$ we have $\boldsymbol{n}^* = (0, ...0, 1)$ and the integral in (1.37) becomes the left-hand side of (1.36). Then, (1.37) can be written as

$$\int_{\boldsymbol{x}\in\Omega} \rho(\boldsymbol{x}, t)h(s(\boldsymbol{x}, t)) \; d\boldsymbol{x} - \int_{\boldsymbol{x}\in\Omega+tU} \rho(\boldsymbol{x}, 0)h(s(\boldsymbol{x}, 0)) \; d\boldsymbol{x} \geq$$
$$- \int_{\partial\_C} \rho h(s) \left(n_0 + \sum_{m=1}^{d} u_m n_m \right) \; d\boldsymbol{x}d\tau,$$

where ∂_C is the boundary of C without $\tau = 0, \tau = 1$. Since $h(s) \leq 0$ by assumption, the inequality (1.36) will hold if

$$n_0 + \sum_{m=1}^{d} u_m n_m \geq 0.$$

On the side surfaces of C we have vectors of the type

$$\left(\pm \frac{x_1}{|\boldsymbol{x}|} U, ... \pm \frac{x_d}{|\boldsymbol{x}|} U, \mp 1 \right),$$

hence the outward normal vector to the side surfaces has the form

$$(n_1 ... n_d, n_0)^T = \frac{1}{\sqrt{1 + U^2}} \left(\frac{x_1}{|\boldsymbol{x}|}, ... \frac{x_d}{|\boldsymbol{x}|}, U \right).$$

Then we obtain

$$n_0 + \sum_{m=1}^{d} u_m n_m = \frac{1}{\sqrt{1+U^2}} \left(U + \sum_{m=1}^{d} \frac{u_m x_m}{|\boldsymbol{x}|} \right) \geq \frac{1}{\sqrt{1+U^2}} \left(U - \sum_{m=1}^{d} \frac{u_m^2}{|\boldsymbol{u}|} \right) \geq 0,$$

where we used that U is the maximal speed and the standard inequality

$$\left| \boldsymbol{u} \cdot \frac{\boldsymbol{x}}{|\boldsymbol{x}|} \right| \leq \boldsymbol{u} \cdot \frac{\boldsymbol{u}}{|\boldsymbol{u}|}.$$

This confirms that the inequality (1.36) holds for non-positive functions $f(s)$ that satisfy (1.32).

Remark The above derivation is correct for smooth functions $f(s)$. A more rigorous derivation would involve taking integrals involving smooth test functions in order to obtain the result in distributional sense, but we skip this for simplicity, see [28].

In order to obtain the result (1.35) we choose a special $h(s)$ given by

$$h(s) = \min(s - s_0, 0), \qquad s_0 = \operatorname*{Ess\,inf}_{\boldsymbol{x} \in \Omega + tU} s(\boldsymbol{x}, 0).$$

This function is non-positive and agrees with (1.32), hence (1.36) applies

$$\int_{\boldsymbol{x} \in \Omega} \rho(\boldsymbol{x}, t) \min(s(\boldsymbol{x}, t) - s_0, 0) \, d\boldsymbol{x} \geq \int_{\boldsymbol{x} \in \Omega + tU} \rho(\boldsymbol{x}, 0) \min(s(\boldsymbol{x}, 0) - s_0, 0) \, d\boldsymbol{x}.$$

Here the right integral is zero by the definition of s_0. Then the left integral must also be zero since $h(s) \leq 0$. Hence for almost every $\boldsymbol{x} \in \Omega$ we must have

$$s(\boldsymbol{x}, t) \geq s_0 = \operatorname*{Ess\,inf}_{\boldsymbol{x} \in \Omega + tU} s(\boldsymbol{x}, 0),$$

which implies (1.35).

1.4 Numerical Methods for Lagrangian Hydrodynamics

The first numerical calculations of the compressible Euler equations involve direct application of the Rankine-Hugoniot jump conditions (1.16). That is, the domain is considered as a union of smooth surfaces divided by discontinuities, and the boundary conditions for each surface are supplied by the Rankine-Hugoniot equations. This approach is very difficult to implement since the shocks are moving relative to the mesh. Their motion is controlled by the non-linear equations and the jump conditions, requiring long trial and error computations at each step in time.

The first artificial viscosity approach, known as the Q-method, is originally proposed by Von Neumann and Richtmeyer in [59]. The shock regions are diffused by introducing a scalar dissipative artificial term that converts a certain amount of mechanical energy into heat. The effect of the artificial term is negligible in smooth regions. The thickness of the resulting diffused shock layers is of the same order as the interval length used in the numerical calculation, so that the Rankine-Hugoniot jump conditions are satisfied at near distances from the shocks. This, or extensions of it, are the approaches used in most numerical methods for solving compressible Euler equations.

Numerical methods for hydrodynamic equations are classified in two categories: Eulerian and Lagrangian hydrocodes. In the Eulerian case, the method uses a fixed computational mesh through which the fluid moves. Lagrangian methods use computational meshes that move with the material. The advantages of this over the Eulerian approach are the following:

- The grid motion provides a natural form of mesh adaptivity. The cells become smaller in compression regions, and larger where the material expands. This is less expensive than adaptive strategies with respect to a fixed mesh, where

control quantities are computed to indicate where to modify the mesh, then the mesh is recomputed and the solutions are mapped to the new mesh.

- Lagrangian methods allow easy coupling to other physical models, for which quantities are defined with respect to a non-moving material. An example of this is the coupling of the compressible Euler equations to neutron or radiation transport equations, where interaction cross-sections are defined with respect to material at rest.

- Lagrangian methods allow multi-material simulations which keep materials pure. If material interfaces are aligned with the cell boundaries at initial time, then those material interfaces will stay on the moving cell boundaries throughout the whole computation (unless some external diffusion procedure is added).

The great advantage of Eulerian hydrocodes over the Lagrangian is that they allow turbulent flows. In general they are also more robust under perturbations in the solution. Vorticity regions cause tangling of the moving discrete mesh or too much compression of cells, causing the computational time step to go to zero. A way to overcome this is the arbitrary Lagrangian-Eulerian (ALE) technique, e.g., [6, 32, 35, 12], where the mesh is evolved by a Lagrangian method until its quality deteriorates, then the current mesh is adjusted according to some quantities that measure its quality, followed by remap of the solution to the new mesh.

In this work we are interested in Lagrangian numerical methods. There are three major approaches for solving compressible Euler equations in Lagrangian frame:

1. Staggered grid hydrodynamics (SGH): thermodynamic variables (density, pressure and internal energy) are approximated by piecewise constant values at each

cell, kinematic variables (velocity, acceleration) are defined at the cell vertices, e.g., [60]. Classical SGH methods use the already mentioned Q-method in order to add artificial viscosity to the equations.

2. Cell-centered hydrodynamics (CCH): all variables are piecewise constant values at each cell, approximate Riemann solvers determine velocities at cell vertices, e.g., [48, 49]. These solvers are usually Godunov-type methods which introduce sufficient diffusion near shocks.

3. Finite element methods (FEM): all variables are approximated by finite element functions. Fully discrete methods are obtained by introducing Galerkin weak formulations and using high-order time integration, e.g., [20, 16]. These methods usually employ tensor artificial viscosities, e.g., [56, 36].

Both SGH and CCH methods often suffer from the so-called "mesh imprinting" phenomenon, which is caused by numerical inaccuracies in the computation of the spatial gradients. That results in symmetry breaking and spurious grid vorticity, e.g., [22]. In addition to that, SGH methods can generate the so-called "hourglass modes", see [23], which come from the fact the numerical solution doesn't see arbitrary scaled hourglass modes added to the velocity and pressure fields. Such modes grow in time, leading to spurious mesh distortion. An extension to the classical SGH and CCH methods is the compatible hydro approach, see [8, 9]. It introduces the notions of sub-zonal corner masses and sub-zonal corner forces which are used to update the internal energy in a way that conserves the total energy, and to resist the generation of hourglass modes. Another way to improve the classical SGH methods is to replace the originally used Q-method by edge and tensor artificial viscosity formulations, e.g., [10, 56, 36, 57].

FEM methods allow approximations with functions of higher polynomial degree

on each cell, which leads to high-order convergence in smooth regions, sub-zonal resolution, curvilinear meshes, improvement in symmetry preservation and reduction of mesh distortion. These methods are usually general with respect to the choice of finite element spaces and the order of the used time integrators. FEM methods can sometimes be viewed as a generalization of existing compatible hydro methods, see [5].

2. CENTRAL SCHEMES FOR MEAN FIELD GAMES

The main idea of this section is to modify and apply the existing explicit second order central schemes to each individual MFG equation and then to combine them into a second order fixed point iteration algorithm. We present the following contributions:

- We derive a fully discrete explicit second order staggered finite difference scheme for the FHJ equation (1.1) in Subsection 2.1. The algorithm we propose is a modification of the method derived by Lin and Tadmor in [45].

- We derive a fully discrete explicit second order staggered finite difference scheme for the BCD equation (1.2) in Subsection 2.2. The scheme is based on the classical Nessyahu-Tadmor scheme from [51].

- Both schemes are combined into a fixed point iteration algorithm that solves the MFG equations in Subsection 2.3. We also describe how the two schemes interact in time, memory issues and stopping criteria.

- We provide theoretical arguments which confirm that the numerical schemes for the FHJ and BCD equations produce second order accuracy in Subsection 2.4.

- Numerical results, convergence rates and strong scaling test of our parallel algorithm are presented in Subsection 2.5

- We compare our approach to some already existing MFG numerical algorithms in Subsection 2.6.

2.1 Discretization of the Forward Hamilton-Jacobi Equation

First we note that Hamilton-Jacobi equations are closely related to conservation laws. If we consider the two equation types

$$\frac{\partial u}{\partial t} + H\left(\frac{\partial u}{\partial x}\right) = 0, \qquad u(x,t) = u_0(x), \tag{2.1}$$

$$\frac{\partial \varphi}{\partial t} + \frac{\partial F(\varphi)}{\partial x} = 0, \qquad \varphi(x,0) = \varphi_0(x), \tag{2.2}$$

then $u(x,t)$ is the unique physical solution (also called viscosity solution) of (2.1) if and only if $\varphi(x,t) = \frac{\partial}{\partial x}u(x,t)$ is the unique physical solution (also called entropy solution) of the conservation law (2.2) with flux $F(\varphi) = H\left(\frac{\partial u}{\partial x}\right)$ and initial condition $\varphi_0(x) = u_0(x)$. Details about this relation can be found in [11] and extension to multiple dimensions is in [34]. Using this idea, schemes that are initially created for conservation laws can be applied to Hamilton-Jacobi equations, e.g. [38, 44, 45, 53, 47, 33]. In this subsection we use the same approach and derive a modified version of the scheme presented in [45] for the FHJ equation (1.1).

We discretize our domain Ω by the grid points $x_j = j\Delta x$. The discrete points in time are $t_n = n\Delta t_{hj}$, note that here we march forward in time. Let u_j^n be the approximate value of $u(x_j, t_n)$. We think of our discrete approximation as a continuous, piecewise quadratic function with values u_j^n at the grid points x_j. Its first and second spatial derivatives are defined as follows:

$$(\hat{u}_x)_{j+\frac{1}{2}}^n := \frac{u_{j+1}^n - u_j^n}{\Delta x}, \tag{2.3}$$

$$(\hat{u}_{xx})^n_{j+\frac{1}{2}} := \frac{1}{\Delta x}\text{minmod}\Bigg[\theta\left((\hat{u}_x)^n_{j+\frac{3}{2}} - (\hat{u}_x)^n_{j+\frac{1}{2}}\right), \\ \frac{1}{2}\left((\hat{u}_x)^n_{j+\frac{3}{2}} - (\hat{u}_x)^n_{j-\frac{1}{2}}\right), \\ \theta\left((\hat{u}_x)^n_{j+\frac{1}{2}} - (\hat{u}_x)^n_{j-\frac{1}{2}}\right)\Bigg], \tag{2.4}$$

where "minmod" is a nonlinear limiter that guarantees non-oscillatory behavior of the scheme, and $\theta \in [1, 2]$, see [51, 38]. Its definition is

$$\text{minmod}(a_1, a_2, ...) := \begin{cases} \min_j(a_j) & \text{if } a_j > 0 \quad \forall j, \\ \max_j(a_j) & \text{if } a_j < 0 \quad \forall j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.5}$$

Then for $x \in [x_j, x_{j+1}]$ we define the discrete interpolant

$$\hat{u}(x, t_n) := u^n_j + (\hat{u}_x)^n_{j+\frac{1}{2}}(x - x_j) + \frac{1}{2}(\hat{u}_{xx})^n_{j+\frac{1}{2}}(x - x_j)(x - x_{j+1}). \tag{2.6}$$

As we explain in Subsection 2.2, let $\hat{m}(x, t)$ be the approximation of $m(x, t)$ and $m^n_{j+\frac{1}{2}}$, $(\hat{m}_x)^n_{j+\frac{1}{2}}$ be the value and first spatial derivative of $\hat{m}(x_{j+\frac{1}{2}}, t_n)$. Suppose we already have the values u^n_j, then the next staggered values in time are derived by integrating (1.1) over $[t_n, t_{n+1}]$ and evaluating at $x_{j+\frac{1}{2}}$:

$$u^{n+1}_{j+\frac{1}{2}} = \hat{u}(x_{j+\frac{1}{2}}, t_n) + \int_{t_n}^{t_{n+1}} \Bigg(- H(\hat{u}_x(x_{j+\frac{1}{2}}, t)) \\ + f(x_{j+\frac{1}{2}}, \hat{m}(x_{j+\frac{1}{2}}, t)) + \sigma\hat{u}_{xx}(x_{j+\frac{1}{2}}, t)\Bigg)\, dt. \tag{2.7}$$

At this point we use the relation of our FHJ problem to conservation laws, namely

in our case $\hat{u}_x$ satisfies the conservation law

$$\frac{\partial}{\partial t}(\hat{u}_x) + \frac{\partial}{\partial x}H(\hat{u}_x) = \frac{\partial}{\partial x}f(x, \hat{m}), \tag{2.8}$$

where we ignore the diffusion term since our approach does not handle third derivatives of $\hat{u}$. Equation (2.8) has finite propagation speed, which means that under a standard hyperbolic CFL condition on the time step

$$\frac{\Delta t_{hj}}{\Delta x} \max_x |H'(\hat{u}_x)| \leq \frac{1}{2}, \tag{2.9}$$

our interpolant's spatial derivatives $\hat{u}_x, \hat{u}_{xx}$ remain well-defined around $x_{j+\frac{1}{2}}$ for $t \in [t_n, t_{n+1}]$. Then we can safely use a quadrature rule for the integral in (2.7). We apply a midpoint rule where the midpoint values in time of $\hat{u}_x$ are computed by Taylor expansion that uses the time derivative from equation (2.8), namely

$$\begin{aligned}(\hat{u}_x)^{n+\frac{1}{2}}_{j+\frac{1}{2}} = (\hat{u}_x)^n_{j+\frac{1}{2}} + \frac{\Delta t_{hj}}{2}\Bigg[&- H'\left((\hat{u}_x)^n_{j+\frac{1}{2}}\right)(\hat{u}_{xx})^n_{j+\frac{1}{2}} \\ &+ f_x(x_{j+\frac{1}{2}}, m^n_{j+\frac{1}{2}}) + f_m(x_{j+\frac{1}{2}}, m^n_{j+\frac{1}{2}})(\hat{m}_x)^n_{j+\frac{1}{2}}\Bigg].\end{aligned} \tag{2.10}$$

After we apply the midpoint rule and substitute (2.10), (2.6) into (2.7) we get the following forward staggered scheme for the FHJ equation (1.1):

$$\begin{aligned}u^{n+1}_{j+\frac{1}{2}} =& \frac{1}{2}\left(u^n_j + u^n_{j+1}\right) - \frac{(\Delta x)^2}{8}(\hat{u}_{xx})^n_{j+\frac{1}{2}} \\ &+ \Delta t_{hj}\left[-H\left((\hat{u}_x)^{n+\frac{1}{2}}_{j+\frac{1}{2}}\right) + f(x_{j+\frac{1}{2}}, m^{n+\frac{1}{2}}_{j+\frac{1}{2}}) + \sigma\frac{(\hat{u}_x)^n_{j+\frac{3}{2}} - (\hat{u}_x)^n_{j-\frac{1}{2}}}{2\Delta x}\right],\end{aligned} \tag{2.11}$$

where for the σ term instead of using $(\hat{u}_{xx})^{n+\frac{1}{2}}_{j+\frac{1}{2}}$ computed by (2.4) at $t_{n+\frac{1}{2}}$ (which requires a lot of operations), we apply a simple central difference at time t_n for the

second derivative. This approach provides reduction of computational cost and is sufficient to achieve second order accuracy (see Subsection 2.4).

The time step $\Delta t_{hj} = t_{n+1} - t_n$ for this scheme must take into account not only the hyperbolic CFL condition (2.9), but also the presence of the Laplace term, namely

$$\Delta t_{hj} := \min\left(\frac{c\Delta x}{\max_j \left|H'\left((\hat{u}_x)^n_{j+\frac{1}{2}}\right)\right|}, \frac{c(\Delta x)^2}{\sigma}\right), \tag{2.12}$$

where c is a CFL constant, we usually use 0.4. The term involving σ is derived from positivity preservation: if we suppose $f = H = \hat{u}_{xx} = 0$ and $u^n_j \geq 0\ \forall j$ in (2.11), then we enforce $u^{n+1}_{j+\frac{1}{2}} \geq 0$ by

$$\frac{1}{2}\left(u^n_j + u^n_{j+1}\right) - \sigma\frac{\Delta t}{2\Delta x^2}\left(u^n_j + u^n_{j+1}\right) \geq 0,\ \forall j \Rightarrow \Delta t \leq \frac{\Delta x^2}{\sigma}.$$

The expression (2.12) is recomputed before each time step, because the dependence of (2.8) on $\hat{m}$ causes changes in the maximum of H'.

Remark Depending on σ, in (2.12) we may have $\Delta t = O(\Delta x)$ or $\Delta t = O(\Delta x^2)$. We say that our simulation is in "hyperbolic regime" when $\Delta t = O(\Delta x)$, and we say that our simulation is in "parabolic regime" when $\Delta t = O(\Delta x^2)$. In hyperbolic regime we have $\sigma = O(\Delta x)$, but in parabolic regime we have $\sigma = O(1)$.

Remark We can define a simpler version of (2.11) by choosing continuous, piecewise linear approximation with values u^n_j at the grid points x_j. This corresponds to the choice $(\hat{u}_{xx})^n_{j+\frac{1}{2}} = 0$, instead of the definition (2.4). Then we can apply the above derivation by using a left-point rule for the integral in equation (2.7) and obtain the

scheme

$$\begin{aligned} u_{j+\frac{1}{2}}^{n+1} =& \frac{1}{2}\left(u_j^n + u_{j+1}^n\right) \\ &+ \Delta t_{hj}\left[-H\left((\hat{u}_x)_{j+\frac{1}{2}}^n\right) + f(x_{j+\frac{1}{2}}, m_{j+\frac{1}{2}}^n) + \sigma\frac{(\hat{u}_x)_{j+\frac{3}{2}}^n - (\hat{u}_x)_{j-\frac{1}{2}}^n}{2\Delta x}\right]. \end{aligned} \tag{2.13}$$

This scheme uses the same time step computation as in (2.12), and requires much less operations than (2.11). However, as derived later, the scheme (2.13) does not produce second order convergent method.

2.2 Discretization of the Backward Convection-Diffusion Equation

In this subsection we derive a modification of the central scheme presented in [51] to discretize the BCD equation (1.2). We use the same spatial grid points $x_j = j\Delta x$ as in Subsection 2.1. However the discrete points in time are different, we consider $t_k = k\Delta t_{cd}$, note that in this algorithm we march backwards in time. We think of our discrete approximation as a piecewise linear function $\hat{m}$ where $m_{j+\frac{1}{2}}^k$ is its average value for the cell $[x_j, x_{j+1}]$ (or the value at $x_{j+\frac{1}{2}}$). The spatial derivative $(\hat{m}_X)_{j+\frac{1}{2}}$ at the point $x_{j+\frac{1}{2}}$ is constructed using the uniformly non-oscillatory (UNO) limiter introduced in [31]:

$$\begin{aligned} (\hat{m}_X)_{j+\frac{1}{2}}^k := \frac{1}{\Delta x}\text{minmod}\Bigg(& m_{j+\frac{1}{2}}^k - m_{j-\frac{1}{2}}^k + \frac{1}{2}\text{minmod}(\Delta^2 m_{j-\frac{1}{2}}^k, \Delta^2 m_{j+\frac{1}{2}}^k), \\ & m_{j+\frac{3}{2}}^k - m_{j-\frac{1}{2}}^k - \frac{1}{2}\text{minmod}(\Delta^2 m_{j+\frac{1}{2}}^k, \Delta^2 m_{j+\frac{3}{2}}^k)\Bigg), \end{aligned} \tag{2.14}$$

$$\Delta^2 m_{j+\frac{1}{2}}^k := m_{j+\frac{3}{2}}^k - 2m_{j+\frac{1}{2}}^k + m_{j-\frac{1}{2}}^k,$$

and we sometimes use the minmod limiter that doesn't need as many values, namely

$$(\hat{m}_x)^k_{j+\frac{1}{2}} := \frac{1}{\Delta x}\text{minmod}\left(m^k_{j+\frac{1}{2}} - m^k_{j-\frac{1}{2}}, m^k_{j+\frac{3}{2}} - m^k_{j+\frac{1}{2}}\right). \tag{2.15}$$

Then for $x \in [x_j, x_{j+1}]$ the approximation function $\hat{m}$ has the form

$$\hat{m}(x, t_k) = m^k_{j+\frac{1}{2}} + \frac{1}{\Delta x}(x - x_{j+\frac{1}{2}})(\hat{m}_X)^k_{j+\frac{1}{2}}. \tag{2.16}$$

Suppose we already have the values $m^k_{j+\frac{1}{2}}$, then the next staggered values in time, going backwards, are obtained by integrating (1.2) over $[t_k, t_{k-1}]$ and $[x_{j+\frac{1}{2}}, x_{j+\frac{3}{2}}]$:

$$\begin{aligned} &\int_{x_{j+\frac{1}{2}}}^{x_{j+\frac{3}{2}}} \hat{m}(x, t_{k-1}) - \hat{m}(x, t_k)\ dx \\ &+ \int_{t_k}^{t_{k-1}} \left[H'(\hat{u}_x(x_{j+\frac{3}{2}}, t))\hat{m}(x_{j+\frac{3}{2}}, t) - H'(\hat{u}_x(x_{j+\frac{1}{2}}, t))\hat{m}(x_{j+\frac{1}{2}}, t)\right] dt = \\ &\qquad - \sigma \int_{t_k}^{t_{k-1}} \hat{m}_X(x_{j+\frac{3}{2}}, t) - \hat{m}_X(x_{j+\frac{1}{2}}, t)\ dt. \end{aligned} \tag{2.17}$$

Similar to Subsection 2.1, the BCD equation (1.2) has a finite speed of propagation, hence with the standard hyperbolic CFL condition on the time step

$$\frac{\Delta t_{cd}}{\Delta x} \max_x |H'(\hat{u}_x)| \leq \frac{1}{2}, \tag{2.18}$$

the value of $\hat{m}$ and its spatial derivative $\hat{m}_x$ remain well-defined around $x_{j+\frac{1}{2}}$ for $t \in [t_{n-1}, t_n]$. Then we can safely use a quadrature rule for the time integrals in (2.17). We apply a midpoint rule where the midpoint values in time of $\hat{m}$ are computed by

Taylor expansion that uses the time derivative from equation (1.2), namely

$$m_{j+\frac{1}{2}}^{k-\frac{1}{2}} = m_{j+\frac{1}{2}}^{k} + \frac{\Delta t_{cd}}{2}\Bigg[H''\left((\hat{u}_x)_{j+\frac{1}{2}}^{k}\right)(\hat{u}_{xx})_{j+\frac{1}{2}}^{k} m_{j+\frac{1}{2}}^{k} + H'\left((\hat{u}_x)_{j+\frac{1}{2}}^{k}\right)(\hat{m}_x)_{j+\frac{1}{2}}^{k}\Bigg], \tag{2.19}$$

where we ignore the diffusion term and use the less sharp $\hat{m}_x$ instead of $\hat{m}_X$ without affecting the method's second order accuracy, see Subsection 2.4. After we apply the midpoint rule and substitute (2.19) for the time integrals of (2.17), and use (2.16) for the space integral of (2.17) we get the following backward staggered scheme for the BCD equation (1.2):

$$\begin{aligned} m_{j+1}^{k-1} =& \frac{1}{2}\left(m_{j+\frac{1}{2}}^{k} + m_{j+\frac{3}{2}}^{k}\right) - \frac{\Delta x}{8}\left((\hat{m}_X)_{j+\frac{3}{2}}^{k} - (\hat{m}_X)_{j+\frac{1}{2}}^{k}\right) \\ &+ \Delta t_{cd}\Bigg(\frac{H'\left((\hat{u}_x)_{j+\frac{3}{2}}^{k-\frac{1}{2}}\right) m_{j+\frac{3}{2}}^{k-\frac{1}{2}} - H'\left((\hat{u}_x)_{j+\frac{1}{2}}^{k-\frac{1}{2}}\right) m_{j+\frac{1}{2}}^{k-\frac{1}{2}}}{\Delta x} \\ &\quad + \sigma\frac{m_{j+\frac{5}{2}}^{k} - m_{j+\frac{3}{2}}^{k} - m_{j-\frac{1}{2}}^{k} + m_{j-\frac{3}{2}}^{k}}{2\Delta x^2}\Bigg), \end{aligned} \tag{2.20}$$

where m_{j+1}^{k-1} is the average for the staggered cell $[x_{j+\frac{1}{2}}, x_{j+\frac{3}{2}}]$, and the computation of $(\hat{u}_x)_{j+\frac{3}{2}}^{k-\frac{1}{2}}, (\hat{u}_x)_{j+\frac{1}{2}}^{k-\frac{1}{2}}$ is discussed in Subsection 2.3.1. Similar to the approach in (2.11), for the σ term instead of using the difference between the midpoint values $(\hat{m}_X)_{j+\frac{3}{2}}^{k-\frac{1}{2}}$ and $(\hat{m}_X)_{j+\frac{1}{2}}^{k-\frac{1}{2}}$, we apply a standard central difference at time t_k. Doing this allows us to reduce computational cost while maintaining second order accuracy, see Subsection 2.4.

The time step $\Delta t_{cd} = t_k - t_{k-1}$ for this scheme must take into account not only

the hyperbolic CFL condition (2.18), but also the presence of the Laplace term:

$$\Delta t_{cd} := \min\left(\frac{c\Delta x}{\max_j \left|H'\left((\hat{u}_x)^k_{j+\frac{1}{2}}\right)\right|}, \frac{c(\Delta x)^2}{\sigma}\right), \tag{2.21}$$

where the derivation of the term involving σ and the CFL constant c are the same as in (2.12). This expression is recomputed before each time step, because the maximum of H' changes.

Remark The scheme (2.20) preserves initial mass up to boundary conditions. For the case of periodic boundary conditions, mass is conserved exactly on discrete level, but for any other type of boundary conditions (Dirichlet, constant extensions, etc.) the preservation is only on continuous level. For such cases the mass error decreases under mesh refinement with linear rate.

Remark We can define a simpler version of (2.20) by choosing a piecewise constant function $\hat{m}$ where $m^k_{j+\frac{1}{2}}$ is its value at $x_{j+\frac{1}{2}}$. This corresponds to the choice $(\hat{m}_X)^k_{j+\frac{1}{2}} = (\hat{m}_x)^k_{j+\frac{1}{2}} = 0$, instead of the definitions (2.14) and (2.15). Then we can apply the above derivation by using a left-point rule for the integral in equation (2.17) and obtain the scheme

$$\begin{aligned} m^{k-1}_{j+1} =& \frac{1}{2}\left(m^k_{j+\frac{1}{2}} + m^k_{j+\frac{3}{2}}\right) \\ &+ \Delta t_{cd}\left(\frac{H'\left((\hat{u}_x)^k_{j+\frac{3}{2}}\right) m^k_{j+\frac{3}{2}} - H'\left((\hat{u}_x)^k_{j+\frac{1}{2}}\right) m^k_{j+\frac{1}{2}}}{\Delta x}\right. \\ &\qquad \left. + \sigma \frac{m^k_{j+\frac{5}{2}} - m^k_{j+\frac{3}{2}} - m^k_{j-\frac{1}{2}} + m^k_{j-\frac{3}{2}}}{2\Delta x^2}\right). \end{aligned} \tag{2.22}$$

This scheme uses the same time step computation as in (2.21), and requires much less operations than (2.20). However, as derived later, the scheme (2.22) does not

produce second order convergent method.

2.3 Fixed Point Iteration

In this subsection we combine the two presented algorithms into a fixed point iteration.

2.3.1 Interaction Between the Equations

First we explain how the schemes (2.11), (2.20) obtain values in time from each other. Looking at the forward scheme (2.10), (2.11), suppose we know the values $m^k_{j+\frac{1}{2}}, m^{k-2}_{j+\frac{1}{2}}$ for all j where $t_k \geq t_n \geq t_{k-2}$. Then we use the following second order interpolation in time:

$$m^n_{j+\frac{1}{2}} := m^{k-2}_{j+\frac{1}{2}} + \frac{m^k_{j+\frac{1}{2}} - m^{k-2}_{j+\frac{1}{2}}}{t_k - t_{k-2}}(t_n - t_{k-2}). \tag{2.23}$$

It's important to note that the values used in (2.23) have the same cell staggering, namely the values $m^k_{j+\frac{1}{2}}, m^{k-2}_{j+\frac{1}{2}}$ are defined at all points $x_{j+\frac{1}{2}}$, while the values $m^{k-1}_{j+\frac{1}{2}}$ are undefined, because evolution from $\hat{m}(x, t_k)$ to $\hat{m}(x, t_{k-1})$ would define the values of $\hat{m}(x, t_{k-1})$ only at the grid points x_j. The derivative $(\hat{m}_x)^n_j$ in (2.10) is computed by combining (2.23) and (2.14), and the value $m^{n+\frac{1}{2}}_{j+\frac{1}{2}}$ in (2.11) is computed by applying (2.23) at time $t_{n+\frac{1}{2}}$.

The same approach is used when we consider the backward scheme (2.19), (2.20): suppose we know the values $u^n_{j+\frac{1}{2}}, u^{n+2}_{j+\frac{1}{2}}$ for all j where $t_{n+2} \geq t_k \geq t_n$. Then $u^k_{j+\frac{1}{2}}$ is defined by

$$u^k_{j+\frac{1}{2}} := u^n_{j+\frac{1}{2}} + \frac{u^{n+2}_{j+\frac{1}{2}} - u^n_{j+\frac{1}{2}}}{t_{n+2} - t_n}(t_k - t_n). \tag{2.24}$$

Again, note that the values used in (2.24) have the same cell staggering, namely the values $u^{n+2}_{j+\frac{1}{2}}, u^n_{j+\frac{1}{2}}$ are defined at all points $x_{j+\frac{1}{2}}$, while the values $u^{n+1}_{j+\frac{1}{2}}$ are undefined,

because evolution from $\hat{u}(x,t_n)$ to $\hat{u}(x,t_{n+1})$ would define the values of $\hat{u}(x,t_{n+1})$ only at the grid points x_j. The derivatives $(\hat{u}_x)^k_{j+\frac{1}{2}}, (\hat{u}_{xx})^k_{j+\frac{1}{2}}$ in (2.19), (2.21) are computed by combining (2.24), (2.3) and (2.4), and the ones in (2.20) are obtained by applying (2.24) at $t_{k-\frac{1}{2}}$ and (2.3).

2.3.2 Difference Norms

In order to use a fixed point iteration, we need to define norms for measuring difference between consecutive solutions. We motivate our choice by some theoretical results from [40]. For the cases of periodic or Dirichlet boundary conditions for m and u, the solution of (1.1), (1.2) is unique, if f is monotone in L^2 and H is strictly convex i.e.

$$\int_\Omega (f(x,m_1) - f(x,m_2))(m_1 - m_2)\, dx \geq 0, \quad \forall m_1, \forall m_2,$$
$$H(p+q) - H(p) - H'(p)q \geq 0, \quad \forall p, q \in \mathbb{R}, \text{ equality implies } q = 0.$$

We go over the proof of the above statement and show that it can be modified for the case when $(-f)$ is monotone in L^2 and $(-H)$ is strictly convex.

Suppose we have two different solutions u_1, u_2 of (1.1), and two different solutions m_1, m_2 that satisfy (1.2). We consider the equations (1.1) for u_1, u_2, multiply them by $(m_1 - m_2)$ and subtract the two resulting equations to obtain

$$\begin{aligned} \frac{\partial(u_1 - u_2)}{\partial t}(m_1 - m_2) + \left(H\left(\frac{\partial u_1}{\partial x}\right) - H\left(\frac{\partial u_2}{\partial x}\right)\right)(m_1 - m_2) = \\ (f(m_1) - f(m_2))(m_1 - m_2) + \sigma \frac{\partial^2 (u_1 - u_2)}{\partial^2 x}(m_1 - m_2). \end{aligned} \tag{2.25}$$

We apply the same procedure to the two equations (1.2) for m_1, m_2 to obtain

$$\begin{aligned}&\frac{\partial(m_1-m_2)}{\partial t}(u_1-u_2)\\&+\frac{\partial}{\partial x}\left(H'\left(\frac{\partial u_1}{\partial x}\right)m_1-H'\left(\frac{\partial u_2}{\partial x}\right)m_2\right)(u_1-u_2)=\\&-\sigma\frac{\partial^2(m_1-m_2)}{\partial^2 x}(u_1-u_2).\end{aligned}\tag{2.26}$$

If we only consider the terms in (2.25), (2.26) that involve time derivatives and integrate them over $t\in[0,T]$ we obtain

$$\int_0^T\frac{\partial(u_1-u_2)}{\partial t}(m_1-m_2)\,dt+\int_0^T\frac{\partial(m_1-m_2)}{\partial t}(u_1-u_2)\,dt=$$

$$\cancel{(u_1-u_2)(m_1-m_2)|_0^T}-\int_0^T\frac{\partial(m_1-m_2)}{\partial t}(u_1-u_2)\,dt+\int_0^T\frac{\partial(m_1-m_2)}{\partial t}(u_1-u_2)=0,$$

since $m_1(x,T)=m_2(x,T)$ and $u_1(x,0)=u_2(x,0)$. Also, if we only consider the σ terms in (2.25), (2.26) and integrates them over the domain Ω we obtain

$$\int_\Omega\sigma\frac{\partial^2(u_1-u_2)}{\partial^2 x}(m_1-m_2)-\int_\Omega\sigma\frac{\partial^2(m_1-m_2)}{\partial^2 x}(u_1-u_2)=$$

$$\cancel{\left[\frac{\partial(u_1-u_2)}{\partial x}(m_1-m_2)\right]_{\partial\Omega}}-\int_\Omega\sigma\frac{\partial(u_1-u_2)}{\partial x}\frac{\partial(m_1-m_2)}{\partial x}$$

$$-\cancel{\left[\frac{\partial(m_1-m_2)}{\partial x}(u_1-u_2)\right]_{\partial\Omega}}+\int_\Omega\sigma\frac{\partial(m_1-m_2)}{\partial x}\frac{\partial(u_1-u_2)}{\partial x}=0,$$

since we consider periodic or Dirichlet boundary conditions. Then we integrate

(2.25), (2.26) over $t \in [0,T], x \in \Omega$ to obtain

$$\int_0^T \int_\Omega \left[\left(H\left(\frac{\partial u_1}{\partial x}\right) - H\left(\frac{\partial u_2}{\partial x}\right)\right)(m_1 - m_2) \right.$$
$$\left. - \left(H'\left(\frac{\partial u_1}{\partial x}\right) m_1 - H'\left(\frac{\partial u_2}{\partial x}\right) m_2 \right) \frac{\partial (u_1 - u_2)}{\partial x} \right] dxdt =$$
$$\int_0^T \int_\Omega (f(m_1) - f(m_2))(m_1 - m_2)\, dxdt.$$

We can rearrange the terms in the form

$$\begin{aligned} &\int_0^T \int_\Omega (f(m_1) - f(m_2))(m_1 - m_2) \\ &\quad + m_2 \left(H\left(\frac{\partial u_1}{\partial x}\right) - H\left(\frac{\partial u_2}{\partial x}\right) - H'\left(\frac{\partial u_2}{\partial x}\right) \frac{\partial (u_1 - u_2)}{\partial x} \right) \\ &\quad + m_1 \left(H\left(\frac{\partial u_2}{\partial x}\right) - H\left(\frac{\partial u_1}{\partial x}\right) - H'\left(\frac{\partial u_1}{\partial x}\right) \frac{\partial (u_2 - u_1)}{\partial x} \right) \, dxdt = 0. \end{aligned} \tag{2.27}$$

Convexity of H and monotonicity of f imply that all three terms in (2.27) are non-negative, hence they must vanish. Then the strict convexity of H implies $\frac{\partial u_1}{\partial x} = \frac{\partial u_2}{\partial x}$, hence the equations (1.2) for m_1 and m_2 use the same flux. It is well known that convection-diffusion equations have an unique solution, hence $m_1 = m_2$. This implies that the sources, in equations (1.1) for u_1 and u_2, are equal, hence $u_1 = u_2$ by using the classical uniqueness results for Hamilton-Jacobi equations.

On the other hand, if $(-f)$ is monotone and $(-H)$ is strictly convex, we can apply the same argument by saying that all three terms in (2.27) are non-positive, hence they must vanish. All numerical tests we present use monotone $(-f)$ and strictly convex $(-H)$.

Under additional assumptions on H, f and u_0, there exist smooth or weak solutions, see [40]. Then for $\sigma \to 0$ there exists a unique solution s.t. u is Lipschitz and m is a probability measure. Based on these statements, we use the L^∞ norm for $\hat{u}$

and the following norm for $\hat{m}$:

$$||\hat{m}^{i+1}(x,t) - \hat{m}^i(x,t)||_* = \int_\Omega \left| \int_0^x (\hat{m}^{i+1}(s,t) - \hat{m}^i(s,t))\, ds \right| \, dx, \tag{2.28}$$

where $\hat{m}^i(x,t), \hat{u}^i(x,t)$ are the solutions obtained after the i-th iteration.

2.3.3 Final Algorithm

We are ready to state the complete algorithm:

1. $\hat{m}^0(x,t)$ is initialized by the values of $m_T(x)$ at every point $x_{j+\frac{1}{2}}$, let $i = 0$.
2. $\hat{u}^{i+1}(x,t)$ is computed by the algorithm from Subsection 2.1 using $\hat{m}^i(x,t)$.
3. $\hat{m}^{i+1}(x,t)$ is computed by the algorithm from Subsection 2.2 using $\hat{u}^{i+1}(x,t)$.
4. if convergence is achieved, namely

$$||\hat{m}^{i+1}(x,0) - \hat{m}^i(x,0)||_* < \varepsilon \quad \text{and} \quad ||\hat{u}^{i+1}(x,T) - \hat{u}^i(x,T)||_\infty < \varepsilon$$

then we stop, the solution is $\hat{m}^{i+1}, \hat{u}^{i+1}$. Otherwise $i = i + 1$, go to 2.

The tolerance we usually use is $\varepsilon = 10^{-6}$. Notice that the algorithm is fully explicit and it doesn't involve any matrix computations.

2.3.4 Memory Usage

The memory problem is the following: values computed from steps 2 and 3 must be kept in memory in order to be used for the next iteration of the other equation (the values obtained in step 2 are used in step 3 and vice versa). If our time steps are in parabolic regime, meaning $\Delta t_{hj}, \Delta t_{cd} = O(\Delta x^2)$, and we store all values in time, then the space-time memory consumption would be $O(\Delta x^{-3})$. If the time steps

are in hyperbolic regime, meaning $\Delta t_{hj}, \Delta t_{cd} = O(\Delta x)$, the problem doesn't exist, because the space-time consumption is the standard $O(\Delta x^{-2})$.

We notice that values of $\hat{m}$ used in the FHJ scheme (2.11) and values of $\hat{u}$ used in the BCD scheme (2.20) are already scaled in a sense by Δt. Since our goal is to achieve $O(\Delta x^2)$ convergence rates, then in parabolic regime it is sufficient to provide accuracy of order $O(\Delta t)$ for these interpolated values, because this would give LTE of order $O(\Delta t^2)$ or GTE of order $O(\Delta t) = O(\Delta x^2)$. This means that storing only $O(\Delta x^{-1})$ instead of $O(\Delta t^{-1})$ values in time, for the parabolic regime, will preserve the second order accuracy and keep the space-time memory consumption to $O(\Delta x^{-2})$. Hence in both regimes storing only $O(\Delta x^{-1})$ values in time is sufficient for second order accuracy.

2.4 Convergence Properties

Here we justify our expectations of second order accuracy in L^∞ and the choices of specific limiters (2.4), (2.14), (2.15). For the time being we refer to $\Delta t_{hj}, \Delta t_{cd}$ just as Δt since this argument doesn't focus on the differences between the two. In order to produce global truncation errors (GTE) of at most $O(\Delta x^2)$ for both hyperbolic and parabolic regimes, we need local truncation errors (LTE) of sizes at most $O(\Delta x^4), O(\Delta x^2 \Delta t), O(\Delta x \Delta t^2)$ or $O(\Delta t^3)$.

First we discuss why the diffusion terms are ignored in the half-time equations (2.10), (2.19). Since the half-time values are already scaled in a sense by $O(\Delta t)$ in (2.11), (2.20), and the diffusion terms are scaled by an additional $O(\Delta t)$ in (2.10), (2.19), then these terms' influence in the final LTE is at most $O(\sigma \Delta t^2)$ which results in GTE of at most $O(\sigma \Delta t)$. If we are in parabolic regime, then $O(\Delta t) = O(\Delta x^2), \sigma = O(1)$ and ignoring the diffusion terms doesn't affect the desired accuracy. If we are in hyperbolic regime, then $\sigma = O(\Delta x)$ and the diffusion terms affect the LTE as

$O(\Delta t^2 \Delta x)$, hence they can be ignored again.

2.4.1 Second Order Accuracy of the Hamilton-Jacobi Scheme

Now we consider FHJ equation (1.1), a centered difference for $\frac{\partial}{\partial t}u(x, t_{n+\frac{1}{2}})$ gives us the midpoint method:

$$u(x, t_{n+1}) = u(x, t_n) + \Delta t \frac{\partial}{\partial t} u(x, t_{n+\frac{1}{2}}) + O(\Delta t^3), \Rightarrow$$

$$\begin{aligned} u(x, t_{n+1}) = u(x, t_n) + \Delta t \Bigg[& - H\left(\frac{\partial}{\partial x} u(x, t_{n+\frac{1}{2}})\right) \\ & + f\left(x, m(x, t_{n+\frac{1}{2}})\right) + \sigma \frac{\partial^2}{\partial^2 x} u(x, t_{n+\frac{1}{2}}) \Bigg] + O(\Delta t^3). \end{aligned} \tag{2.29}$$

Suppose all values u_j^n are exact for every x_j at a fixed time t_n. We can also interpolate $\hat{m}$ values at times $t_n, t_{n+\frac{1}{2}}$ up to at least $O(\Delta x^2)$ accuracy with equation (2.23) as explained in Subsection 2.3.1. Comparing (2.11) and (2.29), we see that acceptable LTE are achieved if

$$(\hat{u}_{xx})_{j+\frac{1}{2}}^n = \frac{\partial^2}{\partial^2 x} u(x_{j+\frac{1}{2}}, t_n) + O(\Delta x^2), \tag{2.30}$$

$$\frac{1}{2}\left(u_j^n + u_{j+1}^n\right) - \frac{(\Delta x)^2}{8}(\hat{u}_{xx})_{j+\frac{1}{2}}^n = u(x_{j+\frac{1}{2}}, t_n) + O(\Delta x^4), \tag{2.31}$$

$$(\hat{u}_x)_{j+\frac{1}{2}}^{n+\frac{1}{2}} = \frac{\partial}{\partial x} u(x_{j+\frac{1}{2}}, t_{n+\frac{1}{2}}) + O(\Delta x^2) \quad [\text{or} \quad O(\Delta t \Delta x)], \tag{2.32}$$

$$m_{j+\frac{1}{2}}^{n+\frac{1}{2}} = m(x_{j+\frac{1}{2}}, t_{n+\frac{1}{2}}) + O(\Delta x^2), \tag{2.33}$$

$$\sigma \frac{(\hat{u}_x)_{j+\frac{3}{2}}^n - (\hat{u}_x)_{j-\frac{1}{2}}^n}{2\Delta x} = \sigma \frac{\partial^2}{\partial^2 x} u(x_{j+\frac{1}{2}}, t_{n+\frac{1}{2}}) + O(\Delta x^2) \quad [\text{or} \quad O(\Delta t \Delta x)]. \tag{2.34}$$

Condition (2.30) is satisfied by the limiter (2.4). Then (2.31) comes from taking Taylor expansions of $u(x_j, t_n), u(x_{j+1}, t_n)$ at $x_{j+\frac{1}{2}}$. Condition (2.33) is guaranteed by

our time interpolation. In order to verify (2.32), we need to look at the half-time step equation (2.10). Then we see that condition (2.32) holds if:

$$(\hat{u}_x)^n_{j+\frac{1}{2}} = \frac{\partial}{\partial x} u(x_{j+\frac{1}{2}}, t_n) + O(\Delta x^2), \tag{2.35}$$

$$m^n_{j+\frac{1}{2}} = m(x_{j+\frac{1}{2}}, t_n) + O(\Delta x), \tag{2.36}$$

$$(\hat{m}_x)^n_{j+\frac{1}{2}} = \frac{\partial}{\partial x} m(x_{j+\frac{1}{2}}, t_n) + O(\Delta x). \tag{2.37}$$

(2.35) is clearly satisfied by (2.3), (2.36) is true by the time interpolation properties, and (2.37) follows from (2.36) and the properties of the minmod limiter (2.15). Finally, for the left side of (2.34) we have:

$$\begin{aligned} \sigma \frac{(\hat{u}_x)^n_{j+\frac{3}{2}} - (\hat{u}_x)^n_{j-\frac{1}{2}}}{2\Delta x} &= \sigma \left(\frac{\partial^2}{\partial^2 x} u(x_{j+\frac{1}{2}}, t_n) + O(\Delta x^2) \right) \\ &= \sigma \left(\frac{\partial^2}{\partial^2 x} u(x_{j+\frac{1}{2}}, t_{n+\frac{1}{2}}) + O(\Delta t) + O(\Delta x^2) \right). \end{aligned}$$

If we are in parabolic regime, then $\Delta t = O(\Delta x^2)$ and (2.34) holds. If we are in hyperbolic regime, meaning $\sigma = O(\Delta x), \Delta t = O(\Delta x)$, then the error on the right side becomes $O(\Delta t \Delta x)$, hence the condition holds again.

Thus we have verified all conditions (2.30)-(2.37), which means that the central difference scheme (2.11) provides second order accuracy in L^∞. Hence we can state the following proposition:

Proposition 2.4.1 *The FHJ scheme* (2.11) *has GTE of order* $O(\Delta x^2)$.

Remark If we consider the simplified scheme (2.13), we immediately see, in the spirit of equation (2.31), that not considering second derivatives of $\hat{u}$ results in approximating $u(x_{j+\frac{1}{2}}, t_n)$ by the term $\frac{1}{2}\left(u^n_j + u^n_{j+1}\right)$. By standard Taylor expansion

at $x_{j+\frac{1}{2}}$ we have

$$\frac{1}{2}\left(u_j^n + u_{j+1}^n\right) = u(x_{j+\frac{1}{2}}, t_n) + O(\Delta x^2),$$

hence this term gives us LTE of size $O(\Delta x^2)$. This means that in hyperbolic regime we obtain GTE of size $O(\Delta x)$, and in parabolic regime the scheme doesn't converge since the GTE is $O(1)$.

2.4.2 Second Order Accuracy of the Convection-Diffusion Scheme

Now we consider BCD equation (1.2), a centered difference for $\frac{\partial}{\partial t}m(x, t_{k-\frac{1}{2}})$ gives us the midpoint method:

$$m(x, t_{k-1}) = m(x, t_k) - \Delta t \frac{\partial}{\partial t} m(x, t_{k-\frac{1}{2}}) + O(\Delta t^3) \Rightarrow$$

$$\begin{aligned} m(x, t_{k-1}) = m(x, t_k) + \Delta t \Bigg[\frac{\partial}{\partial x} \left(H' \left(\frac{\partial}{\partial x} u(x, t_{k-\frac{1}{2}}) \right) m(x, t_{k-\frac{1}{2}}) \right) \\ + \sigma \frac{\partial^2}{\partial^2 x} m(x, t_{k-\frac{1}{2}}) \Bigg] + O(\Delta t^3). \end{aligned} \tag{2.38}$$

Suppose all values $m_{j+\frac{1}{2}}^k$ are exact for every $x_{j+\frac{1}{2}}$ at a fixed time t_k. We can also interpolate $\hat{u}$ values at times $t_k, t_{k-\frac{1}{2}}$ up to at least $O(\Delta x^2)$ accuracy with equation (2.24) as explained in Subsection 2.3.1. Comparing (2.20) and (2.38), we see that acceptable LTE are achieved if

$$\frac{1}{\Delta x}\left((\hat{m}_X)_{j+\frac{3}{2}}^k - (\hat{m}_X)_{j+\frac{1}{2}}^k \right) = \frac{\partial^2}{\partial^2 x} m(x_{j+1}, t_k) + O(\Delta x^2), \tag{2.39}$$

$$\frac{1}{2}\left(m_{j+\frac{1}{2}}^k + m_{j+\frac{3}{2}}^k \right) - \frac{\Delta x}{8}\left((\hat{m}_X)_{j+\frac{3}{2}}^k - (\hat{m}_X)_{j+\frac{1}{2}}^k \right) = m(x_{j+1}, t_k) + O(\Delta x^4), \tag{2.40}$$

$$(\hat{u}_x)_{j+\frac{1}{2}}^{k-\frac{1}{2}} = \frac{\partial}{\partial x} u(x_{j+\frac{1}{2}}, t_{k-\frac{1}{2}}) + O(\Delta x^2), \tag{2.41}$$

$$m_{j+\frac{1}{2}}^{k-\frac{1}{2}} = m(x_{j+\frac{1}{2}}, t_{k-\frac{1}{2}}) + O(\Delta x^2) \quad [\text{or} \quad O(\Delta t \Delta x)], \tag{2.42}$$

$$\sigma\frac{m^k_{j+\frac{5}{2}} - m^k_{j+\frac{3}{2}} - m^k_{j-\frac{1}{2}} + m^k_{j-\frac{3}{2}}}{2\Delta x^2} = \sigma\frac{\partial^2}{\partial^2 x} m(x_{j+1}, t_{k-\frac{1}{2}}) + O(\Delta x^2) \quad [\text{or} \quad O(\Delta t \Delta x)]. \tag{2.43}$$

Condition (2.39) is satisfied by the UNO limiter (2.14). The choice of UNO is important, since condition (2.39) is not true for the minmod limiter (2.15). Then (2.40) comes from taking Taylor expansions of $m(x_{j+\frac{1}{2}}, t_k), m(x_{j+\frac{3}{2}}, t_k)$ at x_{j+1}. The left side of condition (2.41) is computed by doing time interpolations, followed by applying (2.3) on them, hence the condition is true by the time interpolation properties and (2.35). In order to verify (2.42), we need to look at the half-time step equation (2.19). Then we see that condition (2.42) holds if:

$$(\hat{u}_x)^k_{j+\frac{1}{2}} = \frac{\partial}{\partial x} u(x_{j+\frac{1}{2}}, t_k) + O(\Delta x), \tag{2.44}$$

$$(\hat{u}_{xx})^k_{j+\frac{1}{2}} = \frac{\partial^2}{\partial^2 x} u(x_{j+\frac{1}{2}}, t_k) + O(\Delta x), \tag{2.45}$$

$$(\hat{m}_x)^k_{j+\frac{1}{2}} = \frac{\partial}{\partial x} m(x_{j+\frac{1}{2}}, t_k) + O(\Delta x). \tag{2.46}$$

Condition (2.44) follows immediately from (2.41), and (2.45) follows from time interpolation properties and (2.30). Condition (2.46) is guaranteed by the minmod limiter (2.15). Finally, the argument about condition (2.43) is the same as the one for condition (2.34).

Thus we have verified all conditions (2.39)-(2.46), hence the central difference scheme (2.20) provides second order accuracy in L^∞. Hence we can state the following proposition:

Proposition 2.4.2 *The BCD scheme* (2.20) *has GTE of order* $O(\Delta x^2)$.

Remark If we consider the simplified scheme (2.22), we immediately see, in the spirit of equation (2.40), that not considering first derivatives of $\hat{m}$ results in approx-

imating $m(x_{j+1}, t_k)$ by the term $\frac{1}{2}\left(m^k_{j+\frac{1}{2}} + m^k_{j+\frac{3}{2}}\right)$. By standard Taylor expansion at x_{j+1} we have

$$\frac{1}{2}\left(m^k_{j+\frac{1}{2}} + m^k_{j+\frac{3}{2}}\right) = m(x_{j+1}, t_k) + O(\Delta x^2),$$

hence this term gives us LTE of size $O(\Delta x^2)$. This means that in hyperbolic regime we obtain GTE of size $O(\Delta x)$, and in parabolic regime the scheme doesn't converge since the GTE is $O(1)$.

2.5 Numerical Tests

In this subsection we first show results that are in agreement with the 1D results obtained in [24]. Then we test the convergence properties of the algorithm on a manufactured smooth test case. Finally we demonstrate some computational features of our algorithm. For all tests our CFL constant is $c = 0.4$.

2.5.1 Test Problem 1

We first examine a test case presented in [24]: it models a maximization problem (see Subsection 1.1.1), so that the players see increasing incentive in the middle of the domain, but at the same time they prefer to be away from other players:

$$f(x,m) = -16\left(x - \frac{1}{2}\right)^2 - 0.1\max(0, \min(5, m)), \qquad H\left(\frac{\partial u}{\partial x}\right) = -\frac{1}{2}\left(\frac{\partial u}{\partial x}\right)^2,$$
$$m_0(x) = \frac{1.0}{1.1}\left[1.0 + 0.2\cos\left(\pi\left(2x - \frac{3}{2}\right)\right)^2\right], \qquad u_T(x) = 0.0.$$

Notice that the system discussed in [24] is forward in time with respect to m and backward with respect to u, namely it is in the form (1.10), (1.9). In order to simulate the same test case, but with reversed time, we solve the form (1.1), (1.2) by taking

the same expressions for f and H, but we switch the initial and final conditions:

$$m_T(x) = \frac{1.0}{1.1}\left[1.0 + 0.2\cos\left(\pi\left(2x - \frac{3}{2}\right)\right)^2\right], \qquad u_0(x) = 0.0.$$

The domain is $[0, 1]$, the volatility is $\sigma = 0.5$, the final time is $T = 0.5$ and the boundary conditions are $\frac{\partial u}{\partial x} = \frac{\partial m}{\partial x} = 0$ on both ends. Since in this example σ is big compared to Δx, we optimize memory usage by saving the solutions of $\hat{m}, \hat{u}$ for only 400 time steps (out of 50 000 steps). On Figure 2.1 we show the distribution of players m at final and initial times. On Figure 2.2 we show the incentive function u and its gradient $\frac{\partial u}{\partial x}$ at final time. The result is computed on 400 cells, the fixed point iteration converges on the fifth loop. We observe that our results are in agreement with the ones in [24].

For this problem's boundary conditions our algorithm preserves mass only on continuous level. The difference between initial and final mass converges to zero linearly under refinement. For the presented simulation on 400 cells the difference is $9.85e^{-3}$.

2.5.2 Test Problem 2

The purpose of this example is to verify the method's ability to obtain second order convergence rate for a smooth problem. We use a similar setup as in Test Problem 1, but we initialize $m_T(x)$ by a C^1 function with compact support:

$$m_T(x) = \begin{cases} 4.0\sin^2\left(2\pi\left(x - \frac{1}{4}\right)\right) & x \in [\frac{1}{4}, \frac{3}{4}], \\ 0 & \text{otherwise.} \end{cases}$$

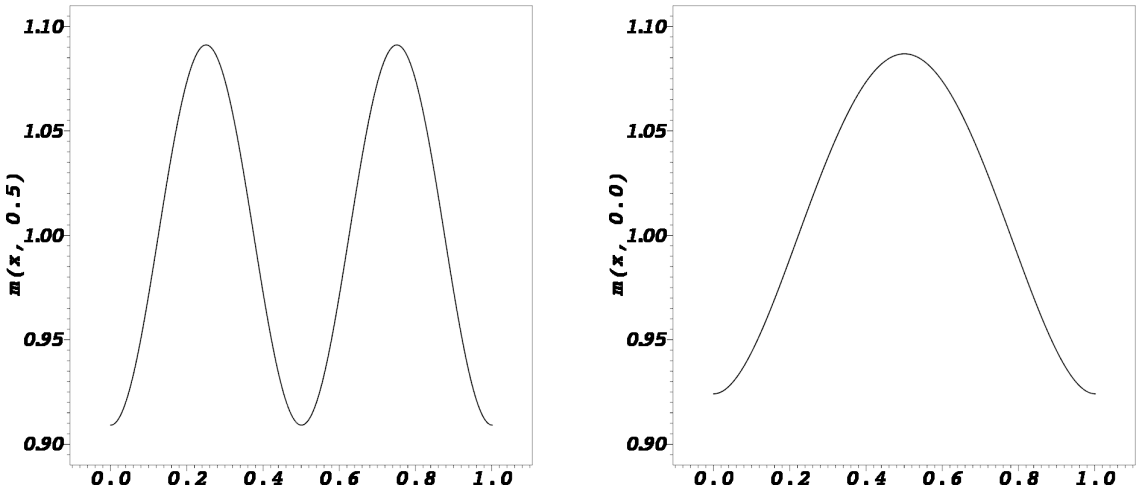

Figure 2.1: Plot of $m_T(x)$ (on the left side) and the solution $m(x, 0.0)$ (on the right side) computed on 400 cells for Test Problem 1.

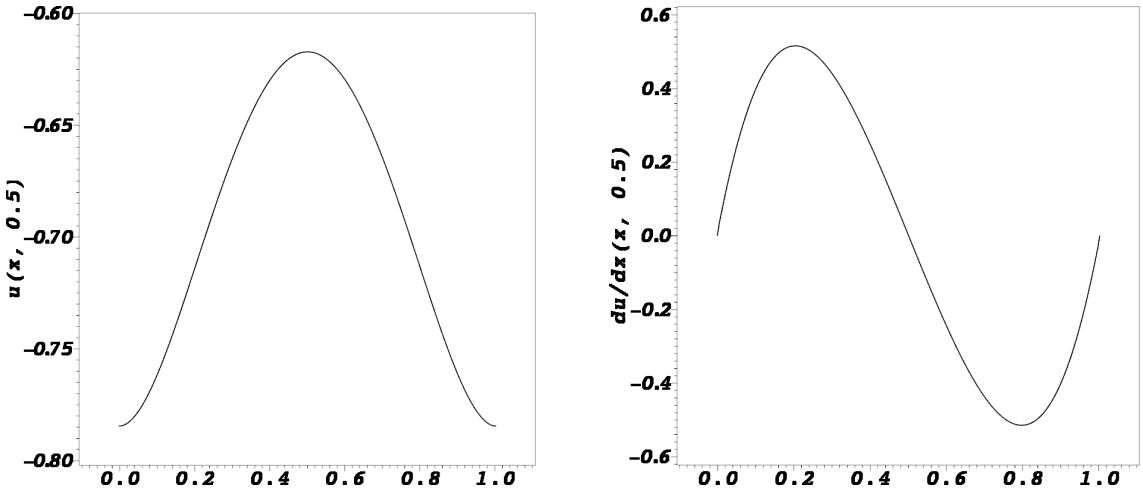

Figure 2.2: Solution for u (on the left side) and $\frac{\partial u}{\partial x}$ (on the right side) computed on 400 cells for Test Problem 1.

and we keep u smooth by using a similar source:

$$u_0(x) = 0.0, \qquad f(x, m) = 3.0 m_T(x) - \min(4.0, m), \qquad H\left(\frac{\partial u}{\partial x}\right) = -\frac{1}{2}\left(\frac{\partial u}{\partial x}\right)^2.$$

The domain is $(0, 1)$, the volatility is $\sigma = 0.05$ and in order to keep the solution smooth enough we use a final time $T = 0.05$. We compute convergence speed by considering a reference solution calculated on 3000 cells. Each simulation optimizes memory usage by storing only Δx^{-1} solutions in time. On Figure 2.3 we show the distributions of players m at final and initial times. On Figure 2.4 we show the incentive functions u and their gradients $\frac{\partial u}{\partial x}$ at final time. In Table 2.1 we show convergence speeds for the L^∞ and L^1 norms, and mass preservation. The presented norms are computed by dividing the domain in 10 000 cells, comparing the end points of each cell to obtain the L^∞ norm, and applying a 3-point Gauss quadrature rule in each cell to obtain the L^1 norm and the mass. We observe the expected second order in L^∞ and L^1, and the linear dissipation of the mass error. The mass error for the reference solution is 3.28E-9.

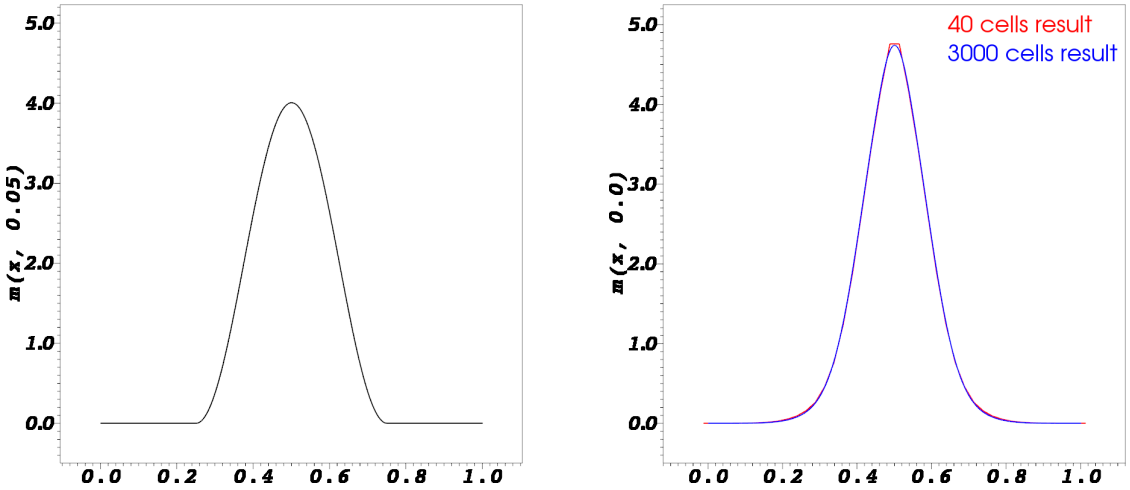

Figure 2.3: Plot of $m_T(x)$ (on the left side) and the solution $m(x, 0.0)$ (on the right side) computed on 40 and 3000 cells for Test Problem 2.

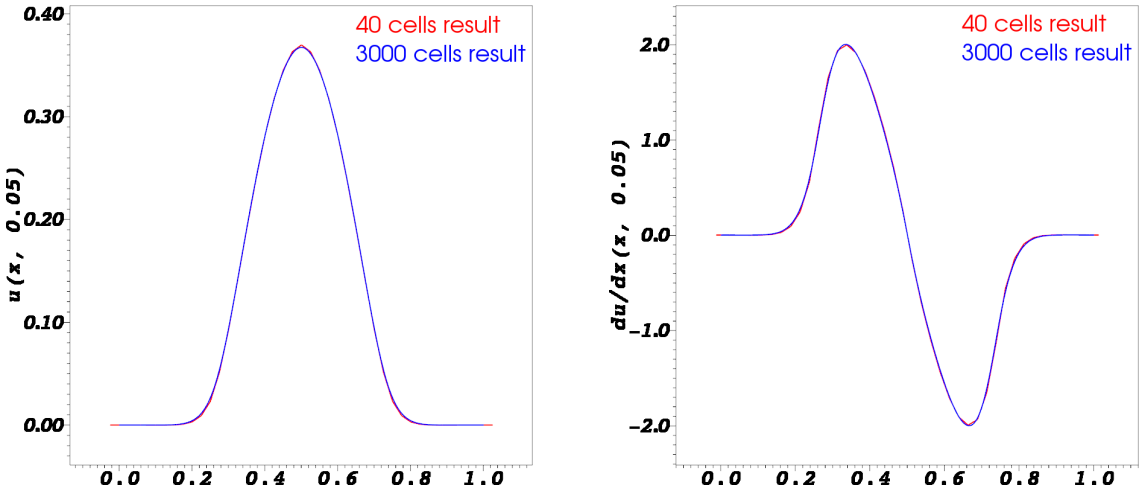

Figure 2.4: Solution for u (on the left side) and $\frac{\partial u}{\partial x}$ (on the right side) computed on 40 and 3000 cells for Test Problem 2.

	m errors				u errors					
# cells	L^∞	rate	L^1	rate	L^∞	rate	L^1	rate	mass error	rate
40	2.75E-1		1.74E-2		3.43E-3		9.40E-4		3.74E-7	
80	6.66E-2	2.05	4.28E-3	2.02	7.93E-4	2.11	2.51E-4	1.90	1.47E-7	1.34
160	1.64E-2	2.01	1.04E-3	2.04	1.98E-4	1.99	6.42E-5	1.97	6.29E-8	1.22
320	4.13E-3	1.99	2.57E-4	2.01	4.96E-5	2.00	1.60E-5	2.00	3.03E-8	1.05
640	1.01E-3	2.03	6.34E-5	2.02	1.25E-5	1.97	3.90E-6	2.04	1.52E-8	0.99
1280	2.41E-4	2.06	1.47E-5	2.10	3.43E-6	1.87	8.94E-7	2.12	7.66E-9	0.98

Table 2.1: L^∞ and L^1 errors, differences between initial and final mass, and convergence rates with respect to a reference solution computed on 3000 cells for Test Problem 2.

2.5.3 Strong Scaling Test

Both schemes (2.11), (2.20) admit easy parallelization. Our algorithm is developed on C++ with OpenMP threads. In this subsection we report execution times and make a strong scaling test.

The problem we consider is Test Problem 2 on 6000 cells with all other parameters as in Subsection 2.5.2. We make one iteration of both schemes (2.11), (2.20) that consists of 112 500 time steps for each equation. The execution times and the scaling

result are displayed on Figure 2.5. We observe that good scaling is achieved when we have at least 500 cells per processor. Since the parallelism is in space and not in time, our code is faster for cases when the ratio between cells in space versus steps in time is bigger i.e. for smaller σ values.

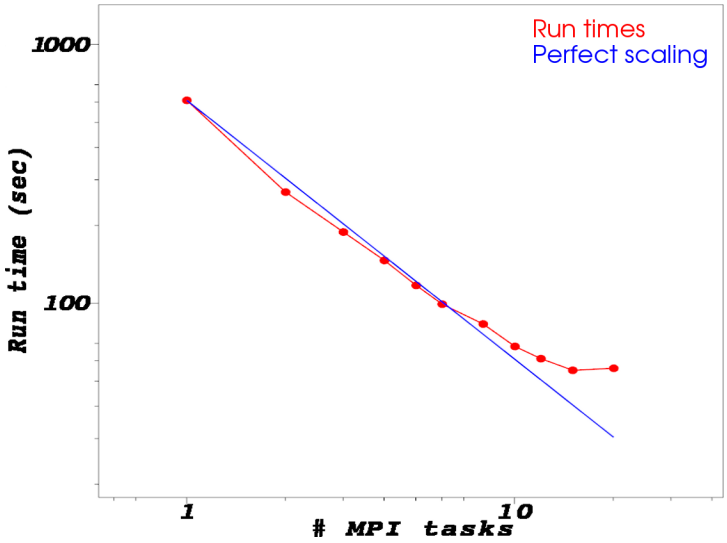

Figure 2.5: Strong scaling test on 6000 cells for Test Problem 2.

2.6 Related Work

In this subsection we describe some already existing algorithms related to the MFG equations (1.1), (1.2).

In [2], Achdou and Capuzzo-Dolcetta propose implicit finite difference methods for the stationary case, the time-dependent case where both MFG equations progress forward in time, and the case of (1.1), (1.2). The authors present detailed proofs of existence and uniqueness for the discrete problems, and provide bounds on the solutions. The paper contains results of numerical simulations for 2-dimensional test cases where both equations go forward in time. The simulations make use of a long

time approximation strategy of the stationary problem. The tests confirm that the used approach is robust when $\sigma \to 0$, and the results suggest linear convergence.

In [1], Achdou, Camilli and Capuzzo-Dolcetta study the mean field planning problem (MFGP), which puts an initial condition on $m(x, 0)$ instead of the one on $u(x, 0)$, and the penalized mean field planning problem (MFGPP), which is in the same form as (1.1), (1.2). The authors present semi-implicit finite difference schemes and prove existence and uniqueness of the solution by exploiting a connection between the discrete formulations and a minimization problem. Results for the MFGP discrete equations are obtained by solving the MFGPP discrete equations and passing to the limit of a penalization parameter. The forward - backward MFGPP finite difference scheme is solved by a Newton method. The presented numerical results show correct behavior for small σ and first order convergence. The Newton method converges slower for smaller values of σ.

In [24], Gueant examines the MFG equations (1.1), (1.2) for the special case of $H(\frac{\partial u}{\partial x}) = \frac{1}{2}(\frac{\partial u}{\partial x})^2$. The author uses a change of variables which produces two coupled heat equations with source terms. Under some assumptions on $f(x, m)$, existence and uniqueness of weak solutions for the new system are proved. Each equation is approximated in space-time, so that $m_T(x), u_0(x)$ appear as boundary conditions, by implicit finite difference schemes. The author proves existence and uniqueness for both schemes. The discrete equations are solved recursively until fixed point is reached, a Newton method is applied inside each step. The presented numerical results show first order convergence and increasing number of Newton iterations for smaller values of σ.

Alternative to these and to our approach can also be found in [3, 39].

3. ENTROPY VISCOSITY METHOD

In this section we propose a new finite element method for solving the Euler equations of compressible gas dynamics in Lagrangian frame of reference. Our method combines the following features:

- The equations are regularized in a way that provides control over oscillations around contact discontinuities as well as oscillations in shock regions. The added diffusion terms are in agreement with the generalized entropy inequalities (1.19) and the minimum principle on the specific entropy (1.35).

- The method produces high order convergence rates for smooth solutions even with active viscosity terms. This is achieved by using viscosity coefficients that clearly distinguish between smooth and singular regions, and finite element spaces of high polynomial degree for all dependent variables.

- The proposed diffusion terms are in agreement with the general requirements for artificial tensor viscosities stated in [36].

This work is motivated and influenced mostly by the idea of entropy production based artificial viscosity coefficients introduced in [27] and previously used in [61], [26], and the application of high order finite elements in Lagrangian hydrodynamics presented in [20]. The general goal of this project is similar to that of Dobrev, Kolev and Rieben in [20], however, our approach is based on different viscous regularization, viscosity coefficients, and finite element spaces.

The starting point in the derivation of our viscous regularization is the idea of adding mass and thermal viscosity, in addition to the standard momentum viscosity, in order to control density and energy oscillations in contact regions. This approach

is also useful in non-ideal gas simulations, when initial contacts transform into composite waves. The extra viscosity terms are introduced in a way so that the resulting system is still compatible with all generalized entropy inequalities from [37] and the minimum principle on the specific energy from [58]. The detailed derivation of our viscosity terms is presented by Guermond and Popov in [28].

Another significant difference between our method and the one in [20] is that we use entropy residual based viscosity coefficients which are not functions of the velocity gradient. These coefficients are zero on continuous level for smooth solutions even in regions of rotation and compression. This results in the method's high order convergence rates even with active viscosity terms.

We use the same notions of mesh representation and mesh motion as in [20], namely position is a continuous finite element function of higher polynomial degree, resulting in a curvilinear mesh. However we use continuous FE functions for density and internal energy, since the extra viscosity terms would make a discontinuous Galerkin formulation difficult to derive and compute.

3.1 Viscous Regularization

A common way to regularize (1.20)-(1.22) is to add diffusion terms which are similar to the viscosity and thermal diffusion terms in the Navier-Stokes equations. This approach, however, is in agreement with the minimum principle of the specific entropy (1.35), only if the thermal diffusion is zero, see Theorem 8.2.3 in [55]. If the thermal diffusion is removed, the Navier-Stokes regularization would consist of an artificial viscous force in the momentum equation and a corresponding term in the total energy equation. The problem with this is that the resulting viscous terms don't see contact discontinuities. In contact regions the velocity is constant, there is no compression, therefore viscosity that only depends on velocity gradients would

not be active. In Eulerian frame, as these contact discontinuities move through the computational mesh, one would see uncontrolled oscillations resulting from the Gibbs effect. This problem can be concealed in Lagrangian frame by aligning initial contact discontinuities with the cell boundaries and using discontinuous spaces, but if contact regions form in time, the above problem would appear.

Our goal is to satisfy the entropy inequality (1.19) and minimum principle on the specific entropy (1.35). It is shown in [28] that one needs to add mass and thermal viscosity, in addition to the standard momentum viscosity. This makes our method more diffusive than methods using the Navier-Stokes regularization approach, but it gives us a tool for removing oscillation in contact regions as well as the ones in shocks. A regularization that takes into account all the above considerations is described in [28] and it has the form

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = \nabla \cdot \boldsymbol{f}, \tag{3.1}$$

$$\frac{\partial}{\partial t}(\rho \boldsymbol{u}) + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) + \nabla p = \nabla \cdot \mathbb{g}, \tag{3.2}$$

$$\frac{\partial}{\partial t}(\rho E) + \nabla \cdot (\rho E \boldsymbol{u} + p \boldsymbol{u}) = \nabla \cdot (\boldsymbol{h} + \mathbb{g} \cdot \boldsymbol{u}), \tag{3.3}$$

where the viscous terms are:

$$\boldsymbol{f} = \lambda \nabla \rho, \qquad \mathbb{g} = \nu \rho \nabla \boldsymbol{u} + \boldsymbol{f} \otimes \boldsymbol{u}, \qquad \boldsymbol{h} = \lambda \nabla (\rho e) - \frac{\boldsymbol{u}^2}{2} \boldsymbol{f}.$$

Here λ and ν are coefficients that control the amount of added diffusion. These coefficients must have units of (speed $\times$ distance).

3.2 Lagrangian Formulation

We want to solve (3.1)-(3.3) in Lagrangian frame of reference. We can think of our medium as a set of particles having original positions $\boldsymbol{x_0}$. In the Lagrangian

setting these particles move with the fluid velocity, namely

$$\frac{d}{dt}\boldsymbol{x}(\boldsymbol{x_0}, t) := \boldsymbol{u}(\boldsymbol{x}, t).$$

Then the material derivative (also called total, Lagrangian, convective, etc.) of a scalar / vector function $\beta = \beta(\boldsymbol{x}(\boldsymbol{x_0}, t), t)$ is

$$\frac{d}{dt}\beta(\boldsymbol{x}(\boldsymbol{x_0}, t), t) = \frac{\partial \beta(\boldsymbol{x}, t)}{\partial t} + \boldsymbol{u}(\boldsymbol{x}, t) \cdot \nabla_{\boldsymbol{x}} \beta(\boldsymbol{x}, t).$$

Now we express equations (3.1) - (3.3) in terms of their total derivatives:

Density - (3.1) becomes

$$\frac{\partial \rho}{\partial t} + \rho \nabla \cdot \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \rho = \nabla \cdot (\lambda \nabla \rho)$$

$$\Rightarrow \frac{d\rho}{dt} = -\rho \nabla \cdot \boldsymbol{u} + \nabla \cdot (\lambda \nabla \rho).$$

Velocity - (3.2) becomes

$$\frac{\partial \rho}{\partial t}\boldsymbol{u} + \rho \frac{\partial \boldsymbol{u}}{\partial t} + \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \boldsymbol{u} \nabla \cdot (\rho \boldsymbol{u}) + \nabla p = \nabla \cdot \mathbb{g},$$

$$\rho \frac{d\boldsymbol{u}}{dt} = -\boldsymbol{u} \nabla \cdot \boldsymbol{f} - \nabla p + \nabla \cdot \mathbb{g}, \;\; \nabla \cdot \mathbb{g} = \nabla \cdot (\nu \rho \nabla \boldsymbol{u}) + \boldsymbol{f} \cdot (\nabla \boldsymbol{u}) + \boldsymbol{u} \nabla \cdot \boldsymbol{f}$$

$$\Rightarrow \rho \frac{d\boldsymbol{u}}{dt} = -\nabla p + \nabla \cdot (\nu \rho \nabla \boldsymbol{u}) + (\lambda \nabla \rho) \cdot (\nabla \boldsymbol{u}) \tag{3.4}$$

Total energy - (3.3) becomes

$$\frac{\partial \rho}{\partial t} E + \rho \frac{\partial E}{\partial t} + E \boldsymbol{u} \cdot \nabla \rho + \rho \boldsymbol{u} \cdot \nabla E + \rho E \nabla \cdot \boldsymbol{u} + \nabla \cdot (p \boldsymbol{u}) = \nabla \cdot (\boldsymbol{h} + \mathbb{g} \cdot \boldsymbol{u}),$$

$$\rho \frac{dE}{dt} = -E \nabla \cdot \boldsymbol{f} - \nabla \cdot (p \boldsymbol{u}) + \nabla \cdot (\boldsymbol{h} + \mathbb{g} \cdot \boldsymbol{u}) - \boldsymbol{f} \cdot \nabla E + \boldsymbol{f} \cdot \nabla E,$$

$$\rho\frac{dE}{dt} = -\nabla\cdot(p\boldsymbol{u}) + \nabla\cdot\Bigg(- e\boldsymbol{f} - \cancel{\frac{\boldsymbol{u}^2}{2}\boldsymbol{f}} + \lambda\nabla(\rho e) \\ \cancel{-\frac{\boldsymbol{u}^2}{2}\boldsymbol{f}} + \nu\rho\nabla\boldsymbol{u}\cdot\boldsymbol{u} + \cancel{\boldsymbol{u}^2\boldsymbol{f}}\Bigg) + \boldsymbol{f}\cdot\nabla E$$

$$\Rightarrow \rho\frac{dE}{dt} = -\nabla\cdot(p\boldsymbol{u}) + \nabla\cdot(\lambda\rho\nabla e + \nu\rho\nabla\boldsymbol{u}\cdot\boldsymbol{u}) + (\lambda\nabla\rho)\cdot\nabla E. \tag{3.5}$$

Working with an equation for the internal energy instead of (3.5) is more convenient for Lagrangian codes. We obtain it by taking a dot product of (3.4) with $\boldsymbol{u}$ and subtracting that from (3.5):

$$\rho\frac{de}{dt} = -p\nabla\cdot\boldsymbol{u} + \nabla\cdot(\lambda\rho\nabla e) + \nu\rho\nabla\boldsymbol{u} : \nabla\boldsymbol{u} + \cancel{(\nabla\cdot(\nu\rho\nabla\boldsymbol{u}))\cdot\boldsymbol{u}} \\ + (\nabla E)\cdot(\lambda\nabla\rho) - \cancel{(\nabla\cdot(\nu\rho\nabla\boldsymbol{u}))\cdot\boldsymbol{u}} - ((\lambda\nabla\rho)\cdot\nabla\boldsymbol{u})\cdot\boldsymbol{u}$$

$$\Rightarrow \rho\frac{de}{dt} = -p\nabla\cdot\boldsymbol{u} + \nabla\cdot(\lambda\rho\nabla e) + \nu\rho\nabla\boldsymbol{u} : \nabla\boldsymbol{u} + (\nabla e)\cdot(\lambda\nabla\rho).$$

Then the final Lagrangian frame system we propose is:

$$\frac{d}{dt}\boldsymbol{x}(\boldsymbol{x_0}, t) = \boldsymbol{u}(\boldsymbol{x}, t), \tag{3.6}$$

$$\frac{d\rho}{dt} = -\rho\nabla\cdot\boldsymbol{u} + \nabla\cdot(\lambda\nabla\rho), \tag{3.7}$$

$$\rho\frac{d\boldsymbol{u}}{dt} = -\nabla p + \nabla\cdot(\nu\rho\nabla\boldsymbol{u}) + (\lambda\nabla\rho)\cdot(\nabla\boldsymbol{u}), \tag{3.8}$$

$$\rho\frac{de}{dt} = -p\nabla\cdot\boldsymbol{u} + \nabla\cdot(\lambda\rho\nabla e) + \nu\rho\nabla\boldsymbol{u} : \nabla\boldsymbol{u} + (\nabla e)\cdot(\lambda\nabla\rho). \tag{3.9}$$

We solve with respect to position $\boldsymbol{x}(\boldsymbol{x_0}, t)$, density $\rho(\boldsymbol{x}, t)$, velocity $\boldsymbol{u}(\boldsymbol{x}, t)$, internal energy per mass $e(\boldsymbol{x}, t)$. The equation of state (1.23) is still $p = (\gamma - 1)\rho e$.

3.3 Discretization Details

In this subsection we derive a fully-discrete finite element method for the system (3.6) - (3.9). We propose a semi-discrete form with high-order continuous finite element spaces and discuss the notions of mesh representation, mappings, length scale, viscosity coefficients, and time discretization.

3.3.1 Notation

Let Ω_0 be our domain at time 0, $\Omega(t)$ is the domain at time t. Then $\{\mathcal{K}_h\}_{h>0}$ is a mesh family with no hanging nodes that discretizes Ω_0. As we progress in time, the initial mesh changes. By K we refer to a mesh cell that corresponds to an initial cell K_0, and the reference cell is denoted by $\hat{K}$. The mappings between them are defined as $\Phi : \hat{K} \to K, \Phi_0 : \hat{K} \to K_0$ and $\Phi \circ \Phi_0^{-1} : K_0 \to K$ where Φ and K are time-dependent (but we skip the time index). For positions $\boldsymbol{x}$ in K, we have the corresponding positions $\boldsymbol{x_0}$ and $\hat{\boldsymbol{x}}$ in K_0 and $\hat{K}$.

We use a scalar-valued nodal finite element space which is defined with respect to the starting mesh as

$$Q_k = \{v \in C^0(\Omega_0); v|_{K_0} \circ \Phi_0 \in \mathbb{Q}_k, \forall K_0 \in \mathcal{K}_h\}.$$

where $\mathbb{Q}_k$ is the set of multivariate polynomials of degree at most k. The number of nodes is denoted by N, $\varphi_1(\boldsymbol{x_0})...\varphi_N(\boldsymbol{x_0})$ are the standard nodal shape functions of Q_k. Since the mesh nodes move, we define the basis functions' time-dependent behavior as

$$\varphi(\boldsymbol{x}, t) := \varphi(\boldsymbol{x_0}) \quad (\text{equivalent to } \varphi(\boldsymbol{x}, 0) := \varphi(\boldsymbol{x_0}), \frac{d}{dt}\varphi(\boldsymbol{x}, t) := 0), \tag{3.10}$$

where by the usual convention $\boldsymbol{x} = \Phi \circ \Phi_0^{-1}(\boldsymbol{x_0})$. Examples of Q_4 basis functions on original and perturbed meshes are given on Figure 3.1 and Figure 3.2. Note that approximation using such functions stays H^1 conforming in time . Shape functions on the reference cell are denoted by $\hat{\varphi}_i(\hat{\boldsymbol{x}}), i = 1..\hat{N}$.

From this point forward, by $\rho, \boldsymbol{u}, e, \boldsymbol{x}, p, S$ we refer to the variables' discrete versions in Q_k. Quantities that only depend on $\boldsymbol{x_0}$ are taken at initial time, for example $\rho(\boldsymbol{x_0})$ is density given by initial conditions.

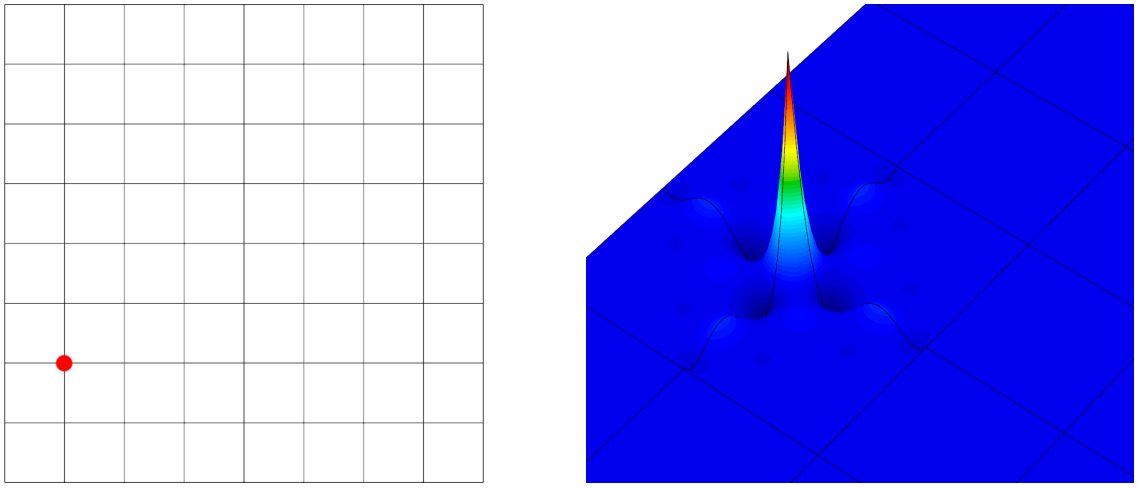

Figure 3.1: Example of a Q_4 basis function on an unperturbed mesh.

3.3.2 Semi-Discrete Form

We get a semi-discrete form by multiplying every equation (3.7) - (3.9) with a test function and integrating over $\Omega(t)$. We use spaces of same polynomial degree for all dependent variables, namely we seek $(\rho, \boldsymbol{u}, e, \boldsymbol{x}) \in (Q_k \times Q_k^d \times Q_k \times Q_k^d)$. Our tests didn't reveal any benefits in using different polynomial degree for the kinematic and thermodynamic spaces.

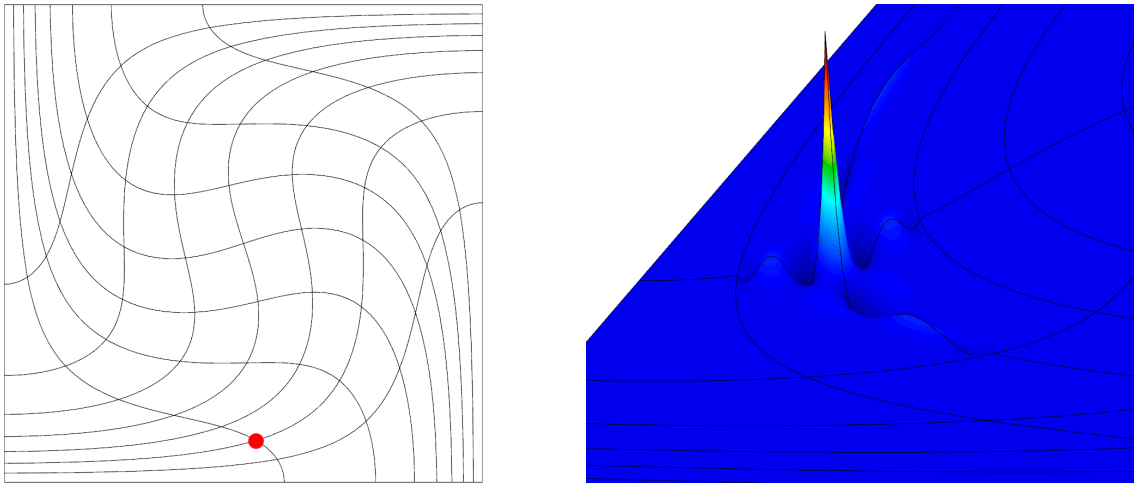

Figure 3.2: Example of a Q_4 basis function on a perturbed mesh resulting from the Taylor-Green vortex problem.

- Density - for every $j = 1...N$ we have

$$\begin{aligned}\int_{\Omega(t)} \frac{d\rho(\boldsymbol{x},t)}{dt}\varphi_j(\boldsymbol{x},t)\ d\boldsymbol{x} &= -\int_{\Omega(t)} \rho(\boldsymbol{x},t)\nabla\cdot\boldsymbol{u}(\boldsymbol{x},t)\varphi_j(\boldsymbol{x},t)d\boldsymbol{x} \\ -\int_{\Omega(t)} \lambda\nabla\rho(\boldsymbol{x},t)\cdot\nabla\varphi_j(\boldsymbol{x},t)\ d\boldsymbol{x} &+ \int_{\partial\Omega(t)} \lambda\nabla\rho(\boldsymbol{x},t)\cdot\boldsymbol{n}\varphi_j(\boldsymbol{x},t)d\boldsymbol{x}.\end{aligned} \tag{3.11}$$

- Velocity - for every dimension $m = 1...d$ and every $j = 1...N$ we have

$$\begin{aligned}&\int_{\Omega(t)} \rho(\boldsymbol{x},t)\frac{du_m(\boldsymbol{x},t)}{dt}\varphi_j(\boldsymbol{x},t)\ d\boldsymbol{x} = \int_{\Omega(t)} p(\boldsymbol{x},t)\frac{\partial\varphi_j(\boldsymbol{x},t)}{\partial x_m}\ d\boldsymbol{x} \\ &-\int_{\partial\Omega(t)} n_m p(\boldsymbol{x},t)\varphi_j(\boldsymbol{x},t)\ d\boldsymbol{x} - \int_{\Omega(t)} \nu\rho\nabla u_m(\boldsymbol{x},t)\cdot\nabla\varphi_j(\boldsymbol{x},t)\ d\boldsymbol{x} \\ &+\int_{\partial\Omega(t)} \nu\rho\nabla u_m(\boldsymbol{x},t)\cdot\boldsymbol{n}\varphi_j(\boldsymbol{x},t)\ d\boldsymbol{x} \\ &+\int_{\Omega(t)} \lambda\nabla u_m(\boldsymbol{x},t)\cdot\nabla\rho(\boldsymbol{x},t)\varphi_j(\boldsymbol{x},t)\ d\boldsymbol{x},\end{aligned} \tag{3.12}$$

  where $\boldsymbol{n}$ is the boundary's outward unit normal.

- Internal energy - for every $j = 1...N$ we have

$$\begin{aligned}
&\int_{\Omega(t)} \rho(\boldsymbol{x},t)\frac{de(\boldsymbol{x},t)}{dt}\varphi_j(\boldsymbol{x},t)\ d\boldsymbol{x} = -\int_{\Omega(t)} p(\boldsymbol{x},t)\nabla\cdot\boldsymbol{u}(\boldsymbol{x},t)\varphi_j(\boldsymbol{x},t)\ d\boldsymbol{x}\\
&-\int_{\Omega(t)} \lambda\rho(\boldsymbol{x},t)\nabla e(\boldsymbol{x},t)\cdot\nabla\varphi_j(\boldsymbol{x},t)\ d\boldsymbol{x}\\
&+\int_{\partial\Omega(t)} \lambda\rho(\boldsymbol{x},t)\nabla e(\boldsymbol{x},t)\cdot\boldsymbol{n}\varphi_j(\boldsymbol{x},t)\ d\boldsymbol{x}\\
&+\int_{\Omega(t)} \nu\rho(\boldsymbol{x},t)\nabla\boldsymbol{u}(\boldsymbol{x},t):\nabla\boldsymbol{u}(\boldsymbol{x},t)\varphi_j(\boldsymbol{x},t)\ d\boldsymbol{x}\\
&+\int_{\Omega(t)} \lambda\nabla e(\boldsymbol{x},t)\cdot\nabla\rho(\boldsymbol{x},t)\varphi_j(\boldsymbol{x},t)\ d\boldsymbol{x}.
\end{aligned} \tag{3.13}$$

And the position function $\boldsymbol{x}(\boldsymbol{x_0},t)$ is simply evolved by the chosen time integrator for the mesh motion equation, namely $\frac{d}{dt}\boldsymbol{x}(\boldsymbol{x_0},t) = \boldsymbol{u}(\boldsymbol{x},t)$.

Remark A drawback of adding mass viscosity is that the resulting mass matrices are time-dependent. For example, the left-hand side of (3.13) and the corresponding matrix are

$$\int_{\Omega(t)} \rho(\boldsymbol{x},t)\frac{d}{dt}\left(\sum_{i=1}^{N} e_i(t)\varphi_i(\boldsymbol{x},t)\right)\varphi_j(\boldsymbol{x},t)d\boldsymbol{x} = \sum_{i=1}^{N}\frac{de_i(t)}{dt}\int_{\Omega(t)}\rho(\boldsymbol{x},t)\varphi_i(\boldsymbol{x},t)\varphi_j(\boldsymbol{x},t)d\boldsymbol{x},$$

$$(M_e)_{ij} = \int_{\Omega_0}\rho(\boldsymbol{x}(\boldsymbol{x_0},t),t)\varphi_i(\boldsymbol{x_0})\varphi_j(\boldsymbol{x_0})|\det J_{\boldsymbol{x_0}\to\boldsymbol{x}}|d\boldsymbol{x_0}.$$

Here $\rho(\boldsymbol{x}(\boldsymbol{x_0},t),t)|\det J_{\boldsymbol{x_0}\to\boldsymbol{x}}| = \rho(\boldsymbol{x_0})$ is not true, which would be the case if we don't have additional terms in the mass equation, see [20].

Remark In most cases, the boundary integrals can be dropped. Let's consider the one containing p in (3.12). For a boundary that is parallel to a coordinate axis and we have $\boldsymbol{u}\cdot\boldsymbol{n} = 0$ as a boundary condition, the integral must be zero, because

this condition is enforced by eliminating exactly those entries from the linear system which correspond to the shape functions involved in that integral. The integral must also be zero when we have $e = 0$ on the boundary. Now consider the boundary integrals containing a viscosity coefficient λ or ν. Model test cases usually assume smooth regions around the boundary, hence one can say the viscosity coefficients there should be zero. For cases of a shock wave interacting with the boundary, however, these boundary integrals must be taken into account.

3.3.3 Mesh Representation and Position Mappings

Mesh position is discretized by a finite element function that represents each degree of freedom node's position at time t. A particle with original position $\boldsymbol{x_0}$ is moved to a new position $\boldsymbol{x}(\boldsymbol{x_0}, t)$ given by the standard finite element expansion

$$\boldsymbol{x}(\boldsymbol{x_0}, t) = \sum_{j=1}^{N} \boldsymbol{X_j}(t)\varphi_j(\boldsymbol{x_0}), \tag{3.14}$$

where $\boldsymbol{X_j}(t)$ is the position of the node associated with basis function φ_j at time t. The use of high-order polynomial basis functions implies the nodal positions are interpolated by high-order polynomials, hence they are connected by curves and the mesh is curvilinear. Notice that in order to obtain the position of any point in our computational mesh, we only need the positions of the finite element nodal points and the original basis functions. This approach is very efficient since it doesn't involve any complicated curve reconstructions.

The formula (3.14) gives us a straightforward way to define the time-dependent position mapping $\Phi : \hat{\boldsymbol{x}} \to x$ from the reference cell $\hat{K}$ to an actual cell of interest K:

$$\boldsymbol{x}(\hat{\boldsymbol{x}}, t) = \sum_{i=1}^{\hat{N}} \boldsymbol{X_j}(t)\hat{\varphi}_i(\hat{\boldsymbol{x}}), \tag{3.15}$$

where $\hat{\varphi}_i$ is a basis function on the reference cell, $\hat{N}$ is the number of degrees of freedom on the reference cell, and $j \in \{1..N\}$ is the node index corresponding to the reference node $i \in \{1..\hat{N}\}$ (local to global DOFS mapping). An example of such mapping, which uses Q_2 spaces, is shown on Figure 3.3. Looking at this figure, the formula (3.15) tells us that we can obtain the right side's black points' positions only by using the positions of the right side's red points (which are the moved Q_2 FE nodes) and the basis functions on the reference cell.

The time-dependent Jacobian matrix of this mapping is then simply

$$J := \frac{\partial \boldsymbol{x}}{\partial \hat{\boldsymbol{x}}} = \sum_{i=1}^{\hat{N}} \boldsymbol{X_j}(t) \otimes \nabla \hat{\varphi}_i(\hat{\boldsymbol{x}}). \tag{3.16}$$

Note that with this definition, J has the usual form:

$$J_{ij} = \frac{\partial x_i}{\partial x_j}, \qquad i, j = 1..d. \tag{3.17}$$

At $t = 0$, we have $\Phi_0 : \hat{\boldsymbol{x}} \to \boldsymbol{x_0}$ with Jacobian J_0 given by (3.16). Later in the text, we also use the mapping $\Phi \circ \Phi_0^{-1} : \boldsymbol{x_0} \to \boldsymbol{x}$ with Jacobian $J J_0^{-1}$.

3.3.4 Length Scales

Artificial viscosity coefficients (λ, ν) must scale like speed times distance. The usual approach is to define (1) a mesh dependent length scale and (2) a shock-capturing quantity, for example the entropy production, so that both (1) and (2) form the final viscosity coefficient. As our tests indicated, the correct approach is to consider the combination of (1) and (2) instead of taking them as independent notions. In this subsection we define three different length scales and we match them with specific coefficients in the next subsection. All scales are defined pointwise in order to match the usage of high order polynomial spaces.

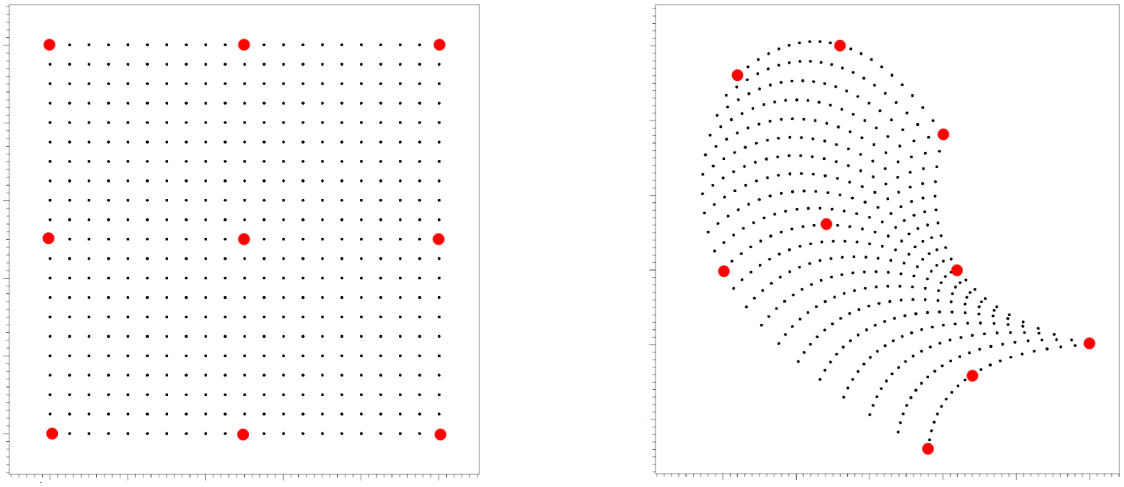

Figure 3.3: Example of a Q_2 mapping between reference and actual coordinates.

We start by defining a smooth initial mesh length scale function $h_0(\boldsymbol{x})$:

- On each initial cell K_0, for $\boldsymbol{x} \in K_0$ define $h^*(\boldsymbol{x}) = \frac{1}{k}|K_0|^{1/d}$.
- If $h^*(\boldsymbol{x})$ has no jumps, then $h_0 = h^*$. Otherwise h_0 is computed by a smoothing procedure with some smoothing constant ε:

$$\int_{\Omega_0} h_0(\boldsymbol{x})\varphi(\boldsymbol{x})\ d\boldsymbol{x} + \varepsilon(h^*_{min})^2 \int_{\Omega_0} \nabla h_0(\boldsymbol{x}) \cdot \nabla\varphi(\boldsymbol{x})\ d\boldsymbol{x} = \int_{\Omega_0} h^*(\boldsymbol{x})\varphi(\boldsymbol{x})\ d\boldsymbol{x}. \tag{3.18}$$

We usually take h_0 in the same finite element space as our dependent variables.

Once $h_0(\boldsymbol{x})$ is known, we define our three length scales by the mapping $\Phi \circ \Phi_0^{-1} : \boldsymbol{x_0} \to \boldsymbol{x}$ with Jacobian JJ_0^{-1} from initial to actual coordinates:

1. $h_1(\boldsymbol{x})$ is defined as a perturbation of the initial mesh in the direction of current motion $\boldsymbol{u}(\boldsymbol{x})$:

$$h_1(\boldsymbol{x}(\boldsymbol{x_0}, t)) = h_0(\boldsymbol{x_0})\frac{|JJ_0^{-1}(\boldsymbol{x_0})\boldsymbol{u}(\boldsymbol{x})|}{|\boldsymbol{u}(\boldsymbol{x})|}. \tag{3.19}$$

2. $h_2(\boldsymbol{x})$ is defined as in [20]. That is, it's the perturbation of the initial mesh in the direction of maximal compression:

$$h_2(\boldsymbol{x}(\boldsymbol{x_0}, t)) = h_0(\boldsymbol{x_0}) \frac{|JJ_0^{-1}(\boldsymbol{x_0})\boldsymbol{s}(\boldsymbol{x})|}{|\boldsymbol{s}(\boldsymbol{x})|}. \tag{3.20}$$

where $\boldsymbol{s}(\boldsymbol{x})$ is the eigenvector that corresponds to the smallest eigenvalue $\mu(\boldsymbol{x})$ of $\nabla^s \boldsymbol{u}(\boldsymbol{x})$. Here $\mu(\boldsymbol{x})$ is a measure of maximal compression (or minimal expansion if the value is positive), and $\boldsymbol{s}(\boldsymbol{x})$ is the direction in which this compression occurs.

3. $h_3(\boldsymbol{x})$ is defined to be the measure of the volume change at $\boldsymbol{x}$:

$$h_3(\boldsymbol{x}(\boldsymbol{x_0}, t)) = h_0(\boldsymbol{x_0}) |\det(JJ_0^{-1}(\boldsymbol{x_0}))|. \tag{3.21}$$

Notice that h_1 and h_2 are direction-dependent, while h_3 is not. They are not finite element functions, their values are computed whenever needed (usually at quadrature points).

3.3.5 Viscosity Coefficients

As discussed in the previous subsection, the amounts of artificial viscosity (λ, ν) must contain an appropriate combination of length scale plus a shock detector, and should be computed at each quadrature point. We first define "first-order" viscous coefficients. We call them "first-order" or "linear", because their shock detector part will not go to zero under mesh refinement, hence they can produce at most first order convergence for a smooth solution. Let $\boldsymbol{x}^n$ be the position of the quadrature point of interest at time t_n, and c_{visc} is a tunable constant. We have two options corresponding to h_1 and h_2:

- Option 1:

$$\begin{aligned} \lambda_1^{visc} &:= c_{visc} h_1(\boldsymbol{x}^n) |\boldsymbol{u}^n(\boldsymbol{x}^n)|, \\ \nu_1^{visc} &:= c_{visc} h_1(\boldsymbol{x}^n) \left(\sqrt{\gamma T^n(\boldsymbol{x}^n)} + |\boldsymbol{u}^n(\boldsymbol{x}^n)| \right). \end{aligned} \tag{3.22}$$

  This quantity is not a real shock detector since it will diffuse the solution everywhere we have some kind of motion.

- Option 2 (which is used in [20]):

$$\begin{aligned} \lambda_2^{visc} &:= c_{visc} h_2^2(\boldsymbol{x}^n) |\mu(\boldsymbol{x}^n)|, \\ \nu_2^{visc} &:= \begin{cases} c_{visc} h_2(\boldsymbol{x}^n) \left(\sqrt{\gamma T^n(\boldsymbol{x}^n)} + h_2(\boldsymbol{x}^n) |\mu(\boldsymbol{x}^n)| \right) & \mu(\boldsymbol{x}^n) \leq 0, \\ c_{visc} h_2^2(\boldsymbol{x}^n) |\mu(\boldsymbol{x}^n)| & \text{otherwise.} \end{cases} \end{aligned} \tag{3.23}$$

  where $\mu(\boldsymbol{x}) \leq 0$ corresponds to compression regions. Notice that this coefficient is sharper than the first one, but it could be active for smooth solutions as well (as long as they admit compression). In contact regions, however, this coefficient will be inactive, since the velocity is constant.

Then we define a "non-linear" coefficient, which is based on the fact that the entropy production is zero (on continuous level) for smooth solutions and non-zero in singular regions. In Lagrangian frame, equation (1.31) becomes

$$\frac{dS}{dt} + S \nabla \cdot \boldsymbol{u} = 0.$$

Then on discrete level, we define

$$D := \frac{S^n - S^{n-1}}{t_n - t_{n-1}} + S^n \nabla \cdot \boldsymbol{u}^n(\boldsymbol{x}^n), \tag{3.24}$$

where $S^n = S(\rho^n, e^n)$ is the entropy functional at t_n. If we have an energy source as in (1.34) then (3.24) becomes

$$D := \frac{S^n - S^{n-1}}{t_n - t_{n-1}} + S^n \nabla \cdot \boldsymbol{u}^n(\boldsymbol{x}^n) + \frac{Q}{f'(s^n)} e^n, \tag{3.25}$$

where $s^n = s(\rho^n, e^n)$ is the specific entropy at t_n and Q is the source contribution at the quadrature point of interest. The total time derivative of S can be approximated by higher order backward differencing, we do that when we want to achieve higher order convergence for smooth test cases. Notice that the above formula uses multiple consecutive meshes. The resulting coefficient is

$$\nu^{entr} := c_{entr} h_3^2(\boldsymbol{x}^n) \frac{|D|}{|S^n - \overline{S^n}|_{\infty,\Omega(t_n)}}, \qquad \overline{S^n} := \int_{\Omega(t_n)} S^n dx, \tag{3.26}$$

where c_{entr} is a tunable constant. Taking the non-linear coefficient into account, our two viscosity options become:

- Option 1:

$$\lambda_1^n := \min(\lambda_1^{visc}, \nu^{entr}), \qquad \nu_1^n := \min(\nu_1^{visc}, \nu^{entr}). \tag{3.27}$$

- Option 2:

$$\lambda_2^n := \min(\lambda_2^{visc}, \nu^{entr}), \qquad \nu_2^n := \min(\nu_2^{visc}, \nu^{entr}). \tag{3.28}$$

We expect the first-order coefficients to be the active part in neighborhoods of shocks and contacts, while the entropy coefficient to provide vanishing viscosity in smooth regions.

Using appropriate combinations of length scale and shock detectors is essential for avoiding incorrect mesh behavior. Notice the connection between the used length scales and the shock detector definitions for all coefficients (3.22), (3.23), (3.26). The

first-order shock detectors are direction dependent and their length scale corresponds to their directions. On the other hand, the entropy production coefficient is direction-independent and so is its length scale, and this works for most test cases. In general if we use the Option 1 coefficients, then the length scale in (3.26) should be h_1 or h_3, and if we use the Option 2 coefficients, then the length scale for D should be h_2 or h_3. The appropriate choice however is problem-specific. Example of an incorrect combination between the Option 2 coefficients and h_1 in (3.26) is shown on Figure 3.4.

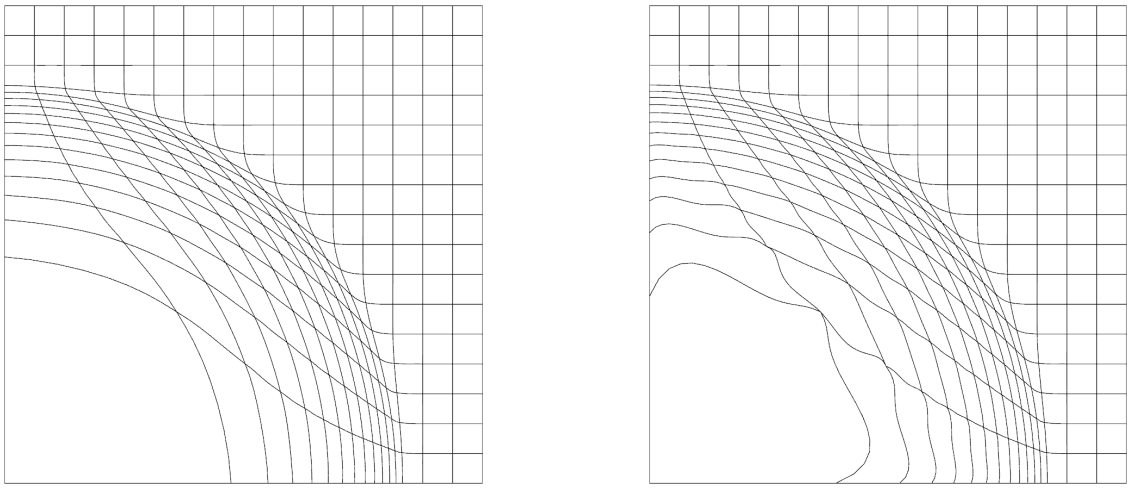

Figure 3.4: Resulting meshes from applying the Option 2 coefficients to a Q_4 position function with h_3 (on the left side) and h_1 (on the right side) in equation (3.26).

3.3.6 Consistency with General Viscosity Requirements

Here we comment on the compatibility of our viscosity tensors with the general requirements for artificial tensor viscosities stated in [56] and [36]. We use the formulation from [36]:

- The viscous terms must be invariant under orthogonal transformations of $\boldsymbol{x}$

and $\boldsymbol{u}$: our method satisfies this requirement on discrete level, the details are given after the list of requirements.

- For the specific entropy s, see (1.29), we must have $\frac{ds}{dt} \geq 0$: our regularization satisfies a weaker notion of this requirement, namely the minimum principle on the specific entropy (1.35) on continuous level. This result is established in [28] (Theorem 3.5).

- The regularized system must be Galilean invariant: our system in Lagrangian form (3.7) - (3.9) is Galilean invariant. All viscosity coefficients in option 2, the entropy production D, h_2, h_3 satisfy the requirement on discrete level. Option 1 and h_1 are not Galilean invariant since they depend explicitly on $\boldsymbol{u}$.

- The artificial viscosity must preserve radial symmetry: we can achieve radial symmetry only on continuous level, since all length scale definitions depend on directions and initial cells sizes. This causes violation of radial symmetry for non-radial or non-uniform meshes. Those differences decrease under mesh refinement. Radial symmetry on discrete level can be achieved in special cases, for example meshes consisting of curved radial cells obtained from each other by orthogonal transformation.

- The viscous force in the momentum equation must be zero for linear velocity (uniform contractions, rigid rotations) and all artificial viscosity terms must be zero in regions of expansion: on continuous level, these requirements are handled by the entropy coefficient (3.26), since the entropy production (1.31) goes to zero in smooth regions.

Now we show that on discrete level our regularization terms are invariant under orthogonal transformation.

Proposition 3.3.1 *Under orthogonal transformation of* $\boldsymbol{x}$ *and* $\boldsymbol{u}$*, the scalar diffusion terms in the mass equation* (3.7) *and the energy equation* (3.9) *stay the same, while the artificial force term in the momentum equation* (3.8) *is transformed the same way as* $\boldsymbol{x}$ *and* $\boldsymbol{u}$*.*

Proof Since this argument is on fully discrete level, we introduce some new notation:

$$\begin{aligned}
\rho(\boldsymbol{x},t) &= \sum_{j=1}^{N} \rho_j(t)\varphi_j(\boldsymbol{x},t), \qquad e(\boldsymbol{x},t) = \sum_{j=1}^{N} e_j(t)\varphi_j(\boldsymbol{x},t),\\
u_m(\boldsymbol{x},t) &= \sum_{j=1}^{N} U_{jm}(t)\varphi_j(\boldsymbol{x},t), \qquad m = 1..d, \quad \boldsymbol{U_j} = (U_{j1}...U_{jd})^T,\\
x_m(\boldsymbol{x_0},t) &= \sum_{j=1}^{N} X_{jm}(t)\varphi_j(\boldsymbol{x_0}), \qquad m = 1..d, \quad \boldsymbol{X_j} = (X_{j1}...X_{jd})^T,
\end{aligned}$$

where the mesh representation details can be found in equations (3.14) - (3.17). The orthogonal transformation is

$$\boldsymbol{X_j^*} = A\boldsymbol{X_j}, \quad \boldsymbol{U_j^*} = A\boldsymbol{U_j}, \quad \text{where } A^T = A^{-1}.$$

and all other quantities used with upper index $*$ are defined in the transformed frame. The Jacobian J^* of the transformation $\hat{\boldsymbol{x}} \to \boldsymbol{x^*}$ can be expressed by the Jacobian of $\hat{\boldsymbol{x}} \to \boldsymbol{x} \to \boldsymbol{x^*}$:

$$J^* = AJ, \quad \det(J^*) = \det(J).$$

Then for the finite element shape functions' gradients we have

$$\nabla\varphi^* = (J^*)^{-T}\nabla\hat{\varphi} = AJ^{-T}\nabla\hat{\varphi},$$

and the discrete gradient of $\boldsymbol{u^*}$ is

$$\nabla \boldsymbol{u^*} = \sum_{j=1}^{N} \boldsymbol{U_j^*} \otimes \nabla \varphi_j^* = \sum_{j=1}^{N} (A\boldsymbol{U_j}) \otimes (AJ^{-T}\nabla\hat{\varphi}_j) = \sum_{j=1}^{N} A\boldsymbol{U_j}(\nabla\varphi_j)^T A^T = A\nabla\boldsymbol{u}A^T.$$

Lets examine the artificial viscosity contribution to a node j in the mass equation (3.11). On fully discrete level, after moving to the reference frame and applying a quadrature rule with Q points, the contribution from an arbitrary cell K^* is computed by the quantity

$$-\sum_{q=1}^{Q}\left(w_q|\det(J^*)|\lambda^*\sum_{i=1}^{N}\rho_i(AJ^{-T}\nabla\hat{\varphi}_i)\cdot(AJ^{-T}\nabla\hat{\varphi}_j)\right),$$

where all quadrature weights, gradients, Jacobians, and viscosity coefficient depend on the quadrature points. But this is the same as

$$-\sum_{q=1}^{Q}\left(w_q|\det(J)|\lambda^*\sum_{i=1}^{N}\rho_i(J^{-T}\nabla\hat{\varphi}_i)\cdot(J^{-T}\nabla\hat{\varphi}_j)\right),$$

where the latter is equal to the contribution from the corresponding cell K if $\lambda^* = \lambda$. Now we look at the artificial force term in the momentum equation (3.12). The contribution to a node j for dimension m from a cell K^* is

$$\begin{aligned}
&\sum_{q=1}^{Q} w_q|\det(J^*)|\Bigg(-\nu^*\hat{\rho}\sum_{i=1}^{N}U_{im}^*(AJ^{-T}\nabla\hat{\varphi}_i)\cdot(AJ^{-T}\nabla\hat{\varphi}_j)\\
&\qquad\qquad+\lambda^*\hat{\varphi}_j\sum_{k=1}^{N}\sum_{i=1}^{N}U_{km}^*\rho_i(AJ^{-T}\nabla\hat{\varphi}_k)\cdot(AJ^{-T}\nabla\hat{\varphi}_i)\Bigg)=\\
&\sum_{q=1}^{Q} w_q|\det(J)|\Bigg(-\nu^*\hat{\rho}\sum_{i=1}^{N}A_m\cdot\boldsymbol{U_i}(J^{-T}\nabla\hat{\varphi}_i)\cdot(J^{-T}\nabla\hat{\varphi}_j)\\
&\qquad\qquad+\lambda^*\hat{\varphi}_j\sum_{k=1}^{N}\sum_{i=1}^{N}A_m\cdot\boldsymbol{U_k}\rho_i(J^{-T}\nabla\hat{\varphi}_k)\cdot(J^{-T}\nabla\hat{\varphi}_i)\Bigg),
\end{aligned}$$

where A_m is the m-th row of A. Hence the artificial force is transformed the same way as $\boldsymbol{u}$ and $\boldsymbol{x}$ if $\lambda^* = \lambda$ and $\nu^* = \nu$. The extra terms in the internal energy equation (3.13) can be written in a similar to the above way and we reach the same conclusion. The only interesting term there is $\nabla \boldsymbol{u^*} : \nabla \boldsymbol{u^*} = \nabla \boldsymbol{u} : \nabla \boldsymbol{u}$ since Frobenius norms are invariant under orthogonal transformations. Hence the orthogonal transformation invariance holds if $\lambda^* = \lambda$ and $\nu^* = \nu$. Similar arguments apply for all boundary integrals as well.

Now we consider the shock detectors. For the entropy production (3.24) we have $D^* = D$ since it depends only on scalars and $\nabla \cdot \boldsymbol{u^*}$:

$$\begin{aligned}\nabla \cdot \boldsymbol{u^*} &= \sum_{j=1}^{N} \nabla \varphi_j^* \cdot \boldsymbol{U_j^*} = \sum_{j=1}^{N} (AJ^{-T} \nabla \hat{\varphi}_j) \cdot (A\boldsymbol{U_j}) = \\ &= \sum_{j=1}^{N} (A \nabla \varphi_j) \cdot (A\boldsymbol{U_j}) = \sum_{j=1}^{N} \nabla \varphi_j \cdot \boldsymbol{U_j} = \nabla \cdot \boldsymbol{u}.\end{aligned}$$

Option 1 shock detectors in (3.22) are also equal since $|\boldsymbol{u^*}| = |A\boldsymbol{u}| = |\boldsymbol{u}|$. For Option 2, see (3.23), we have $\nabla^s \boldsymbol{u^*} = A \nabla^s \boldsymbol{u} A^T$, hence $\nabla^s \boldsymbol{u^*}$ and $\nabla^s \boldsymbol{u}$ are similar matrices. Then we have $\mu^* = \mu$ and $\boldsymbol{s^*} = A\boldsymbol{s}$. Hence all shock detectors are invariant under orthogonal transformation.

When we consider the length scales, we need to look at the Jacobian of the transformation $\boldsymbol{x_0^*} \to \boldsymbol{x^*}$. That is the Jacobian of $\boldsymbol{x_0^*} \to \boldsymbol{x_0} \to \boldsymbol{x} \to \boldsymbol{x^*}$ which is $AJJ_0^{-1}A^T$. For the initial scales we have $h_0^* = h_0$ since orthogonal transformations keep lengths and angles. Then we have

$$\det(A) = 1 \Rightarrow \det(AJJ_0^{-1}A^T) = \det(JJ_0^{-1}) \Rightarrow h_3^* = h_3,$$

$$\frac{|AJJ_0^{-1}A^T\boldsymbol{u^*}|}{|\boldsymbol{u^*}|} = \frac{|AJJ_0^{-1}\boldsymbol{u}|}{|A\boldsymbol{u}|} = \frac{|JJ_0^{-1}\boldsymbol{u}|}{|\boldsymbol{u}|} \Rightarrow h_1^* = h_1,$$

$$\frac{|AJJ_0^{-1}A^T\boldsymbol{s}^*|}{|\boldsymbol{s}^*|} = \frac{|AJJ_0^{-1}\boldsymbol{s}|}{|A\boldsymbol{s}|} = \frac{|JJ_0^{-1}\boldsymbol{s}|}{|\boldsymbol{s}|} \Rightarrow h_2^* = h_2.$$

This implies $\lambda^* = \lambda, \nu^* = \nu$ for all of our options. $\square$

3.3.7 Time Discretization

We discretize the time derivatives of $(\rho, \boldsymbol{u}, e, \boldsymbol{x})$ by standard explicit Runge-Kutta methods. Such method of order r is defined by the lower triangular table given in Table 3.1. Let our solution at time t be $\boldsymbol{v} = (\rho, \boldsymbol{u}, e, \boldsymbol{x})$. We define the operators

a_1					
a_2	b_{21}				
a_3	b_{31}	b_{32}			
...	...				
a_r	b_{r1}	b_{r2}	...	$b_{r,r-1}$	
	c_1	c_2	...	c_{r-1}	c_r

Table 3.1: Runge-Kutta lower triangular table for a method of order r.

$F_\rho, \boldsymbol{F_u}, F_e, \boldsymbol{F_x}$ corresponding to the weak form (3.11) - (3.13) by

$$\int_{\Omega(t)} F_\rho(\boldsymbol{v},t)\varphi \, d\boldsymbol{x} = \int_{\Omega(t)} \Big(-\rho\nabla\cdot\boldsymbol{u}\varphi - (\lambda\nabla\rho)\cdot\nabla\varphi \Big)\, d\boldsymbol{x},$$

$$\int_{\Omega(t)} \rho[F_{\boldsymbol{u}}(\boldsymbol{v},t)]_m\varphi \, d\boldsymbol{x} = \int_{\Omega(t)} \left(p\frac{\partial\varphi}{\partial x_j} - \nu\rho\nabla u_m\cdot\nabla\varphi + \lambda\nabla u_m\cdot\nabla\rho\varphi \right) d\boldsymbol{x}, \ \forall m = 1...d,$$

$$\int_{\Omega(t)} \rho F_e(\boldsymbol{v},t)\varphi \, d\boldsymbol{x} = \int_{\Omega(t)} \Big(-p\nabla\cdot\boldsymbol{u}\varphi + (\nu\rho\nabla\boldsymbol{u} : \nabla\boldsymbol{u})\varphi$$
$$- \lambda\rho\nabla e\cdot\nabla\varphi + (\lambda\nabla e\cdot\nabla\rho)\varphi \Big)\, d\boldsymbol{x},$$

$$\boldsymbol{F_x}(\boldsymbol{v},t) = \boldsymbol{u}.$$

Then if $\boldsymbol{F}(\boldsymbol{v},t) := (F_\rho, \boldsymbol{F}_{\boldsymbol{u}}, F_e, \boldsymbol{F}_{\boldsymbol{x}})$, the solution $\boldsymbol{v}$ is evolved by

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^n + \Delta t \sum_{i=1}^{r} c_i \boldsymbol{k}_i,$$

$$\boldsymbol{k}_i(\rho, u, e, x) := \boldsymbol{F}\left(\boldsymbol{v}^n + \Delta t \sum_{j=1}^{r} b_{ij}\boldsymbol{k}_j, t_n + a_i \Delta t\right).$$

3.3.8 Time Step Control

Because of finite speed of propagation, time step for explicit methods is expected to restrict propagation of information up to one cell, hence it's defined as length scale divided by maximum propagation speed. Although adding viscosity terms in general changes the hyperbolic nature of the problem, we use the same time step scaling, since those terms are degenerate and active only in shock regions of limited thickness. However increasing the artificial viscosity constants decreases the time step. We also use the usual idea of repeating time steps that are suspected to produce oscillations. The way to guess that is by comparing consequent time step sizes:

1. Having $(\rho^n, \boldsymbol{u}^n, e^n, \boldsymbol{x}^n)$, compute:

$$\Delta t = \min_{\boldsymbol{x}\in\Omega_n} \frac{ch_2(\boldsymbol{x})}{\sqrt{\gamma T^n(\boldsymbol{x})} + |\boldsymbol{u}^n(\boldsymbol{x})|}. \tag{3.29}$$

2. Using Δt, evolve the solution to $(\rho^{n+1}, \boldsymbol{u}^{n+1}, e^{n+1}, \boldsymbol{x}^{n+1})$ and then compute:

$$\Delta t^* = \min_{\boldsymbol{x}\in\Omega_{n+1}} \frac{ch_2(\boldsymbol{x})}{\sqrt{\gamma T^{n+1}(\boldsymbol{x})} + |\boldsymbol{u}^{n+1}(\boldsymbol{x})|}.$$

3. If $\Delta t^* < \Delta t \Rightarrow \Delta t = 0.9\Delta t$, go to (2) (repeat the step)

4. Else: $n = n+1, t = t + \Delta t$, if $\Delta t^* > 1.25\Delta t \Rightarrow \Delta t = 1.02\Delta t$.

go to (2).

Where the CFL constant c decreases if the amounts of artificial viscosity are increased. One can also use different constants in steps 3 and 4 (as long as those constants are reasonable).

3.4 Numerical Tests

In this subsection we demonstrate the behavior of our method on standard Lagrangian hydrodynamics test cases with known exact solutions. First we use a smooth solution to show the high-order convergence properties of the method. Then, we use a 1D Riemann problem to demonstrate the shock-capturing properties of the entropy production-based viscosity coefficients and the methods non-oscillating behavior in contact regions. Finally, we turn to 2D shock problems which test the methods symmetry preservation and mesh behavior.

For all test cases, we solve the resulting linear system using a conjugate gradient algorithm with a diagonal Jacobi preconditioner.

Our method is developed by using the parallel finite element methods library MFEM [19]. The obtained results are visualized through the OpenGL visualization tool GLVis [18].

3.4.1 2D Taylor-Green Vortex

The goal of this test case is to demonstrate that for smooth solutions our method achieves high order convergence rates. All simulations are done by keeping the viscosity terms active (we do not set them to zero explicitly) in equations (3.11) - (3.13). This confirms the convergence to zero of the entropy production-based viscosity coefficients on discrete level.

In the 2D Taylor-Green vortex, a smooth solution for the Euler equations is manufactured by designing particular initial conditions and introducing an internal

energy source Q that keeps all variables, except mesh position, at steady state:

$$\rho_t = \boldsymbol{u}_t = e_t = 0.$$

This means that all variables stay constant while the mesh moves. The above is equivalent to

$$\nabla\rho = 0, \quad \nabla \cdot \boldsymbol{u} = 0, \quad \rho\boldsymbol{u} \cdot \nabla\boldsymbol{u} = -\nabla p, \quad Q = \boldsymbol{u} \cdot \nabla e.$$

The source Q changes the Lagrangian frame energy equation to

$$\rho\frac{de}{dt} = -p\nabla \cdot \boldsymbol{u} + Q,$$

and the equations for density and velocity stay the same. The particular initial conditions and source that give the above relations are

$$\begin{aligned}
&\rho(\boldsymbol{x_0}) = 1, \;\; \boldsymbol{u}(\boldsymbol{x_0}) = (\sin(\pi x_0)\cos(\pi y_0), -\cos(\pi x_0)\sin(\pi y_0)), \\
&p(\boldsymbol{x_0}) = \frac{\rho}{4}\left(\cos(2\pi x_0) + \cos(2\pi y_0) + 1\right), \;\; \gamma = \frac{5}{3}, \\
&\qquad Q = \frac{3\pi}{8}\left(\cos(3\pi x)\cos(\pi y) - \cos(\pi x)\cos(3\pi y)\right).
\end{aligned}$$

For smooth solutions, the order of the method depends on the polynomial degree of the used finite element spaces, and, more importantly, how fast the artificial viscosity terms converge to zero. In order to optimize the latter, we use backward differencing with 2 points for the time derivative in (3.25). This, together with the h_3^2 scaling in (3.26) produces at most 4th order convergence to zero for all artificial terms added in the equations. One can use backward differencing with more points in (3.25) in order to achieve orders higher than 4.

We run the problem to $t = 0.5$ and we put $\boldsymbol{u} \cdot \boldsymbol{n} = 0$ on the boundary. Initial and final mesh and velocity magnitudes with Q_4 finite element spaces are shown on Figure 3.5. Comparison between Q_1 and Q_4 simulations with similar number of degrees of freedom are shown on Figure 3.6. The sub-zonal high order resolution of the Q_4 space makes it superior compared to the Q_1 space. In Table 3.2 we show L^1 convergence rates for velocity computed with finite element spaces of different order. We observe the expected high order convergence rates.

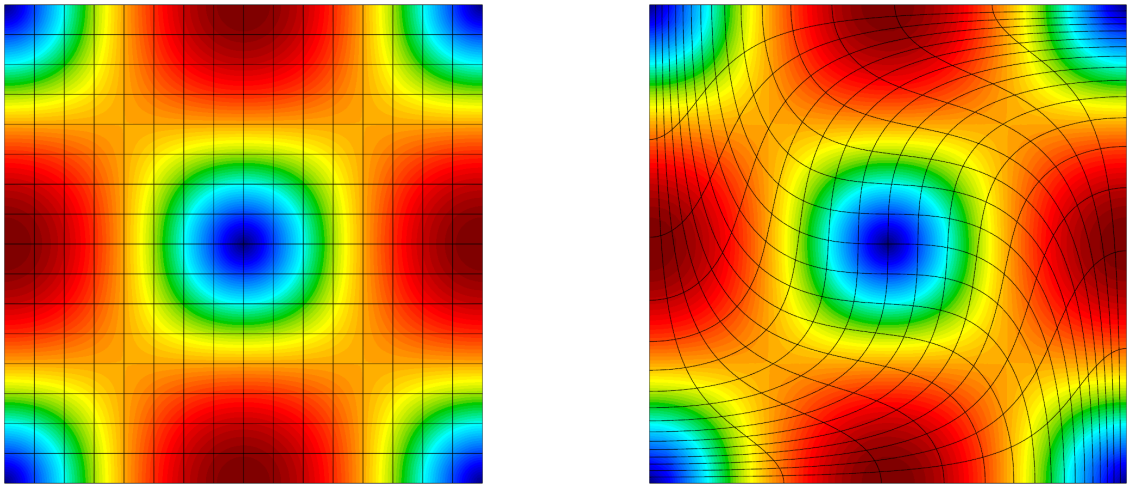

Figure 3.5: Velocity magnitude on the initial mesh (on the left side), and on the final mesh (on the right side) computed by Q_4 FE spaces on 16×16 cells for the 2D Taylor-Green vortex problem.

3.4.2 1D Sod Tube

The goal of this test case is to demonstrate the method's shock and contact capturing properties. This is a 1D Riemann problem in $[0, 1]$ that develops a rarefaction,

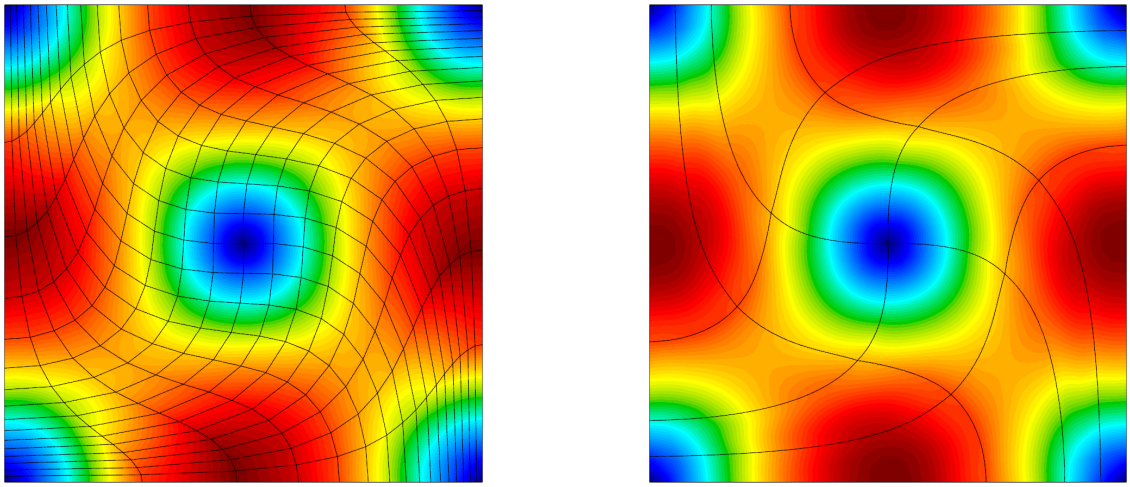

Figure 3.6: Final mesh and velocity magnitude for a Q_1 simulation on 16×16 cells (on the left side), and for a Q_4 simulation on 4×4 cells (on the right side) for the 2D Taylor-Green vortex problem.

	Q_1		Q_2		Q_3		Q_4	
h_0	L^1 error	rate	L^1 error	rate	L^1 error	rate	L^1 error	rate
1/4	3.48e-1		3.62e-2		1.87e-2		6.25e-3	
1/8	7.18e-2	2.27	6.02e-3	2.58	2.51e-3	2.89	2.64e-4	4.56
1/16	1.07e-2	2.74	1.54e-3	1.96	1.71e-4	3.87	1.17e-5	4.49
1/32	1.95e-3	2.45	4.00e-4	1.94	6.70e-6	4.67	8.60e-6	3.76
1/64	4.26e-4	2.19	1.23e-4	1.70	4.54e-7	3.88	6.66e-7	3.69

Table 3.2: L^1 velocity errors and convergence rates for the 2D Taylor-Green vortex problem.

a contact and a shock wave. The initial conditions are:

$$\rho(x_0) = \begin{cases} 1.0 & x_0 \leq 0.5, \\ 0.125 & \text{otherwise}, \end{cases}, \quad \boldsymbol{u}(x_0) = 0, \quad p(x_0) = \begin{cases} 1.0 & x_0 \leq 0.5, \\ 0.1 & \text{otherwise}, \end{cases}, \quad \gamma = 1.4.$$

The final time is 0.2 and we put $\boldsymbol{u} \cdot \boldsymbol{n} = 0$ on the boundary.

We first discuss the effects of using the Option 2 viscosity coefficients (3.23).

On Figure 3.7 we show density field (on the left figure) and the magnitude of the compression measure $\mu(\boldsymbol{x})$ from equation (3.23) at final time. The simulation uses Q_1 FE spaces on a mesh composed of 128 cells. We don't take the minimum (3.28) for this simulation. We observe that the Option 2 coefficient provides sufficient diffusion in the shock region, but almost none in the contact. The reason for this is that the compression measure $\mu(\boldsymbol{x})$ is based only on velocity gradients, but the velocity is constant in the contact. The oscillations in density, pressure and energy around the contact cause some small velocity gradients, but the generated compression is too small to provide diffusion for those oscillations.

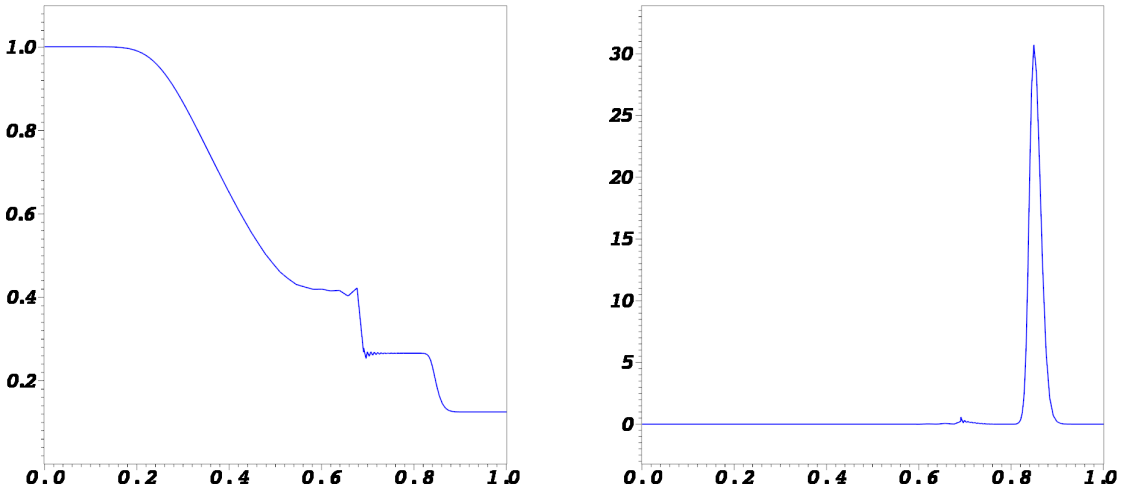

Figure 3.7: Density field (on the left side), and compression measure (on the right side) resulting from using the Option 2 viscosity coefficients for the 1D Sod tube problem.

Next we present results obtained by using the Option 1 viscosity coefficients. On the left side of Figure 3.8 we show simulations that only use the first order coefficients (3.22) without taking the minimum (3.27). We observe that for both Q_1 and Q_4 FE spaces those coefficients provide sufficient diffusion in all regions. Then on the right

side of Figure 3.8 we show the corresponding simulations that take into account the entropy production (3.26) by taking the minimum (3.27). The results are sharper and there are still no oscillations in the contact region, meaning that the entropy production provides sufficient diffusion in those regions. On the left figure, where we use the first order coefficients, we observe that the results for Q_1 and Q_4 FE spaces are essentially the same, but on the right side, where we use the non-linear coefficients, the higher-order simulation is more oscillatory. The presented simulations use 256 cells for Q_4 and 1024 cells for Q_1. In the case of Q_1 FE spaces with non-linear coefficients, the contact region is diffused in 18 cells, and the shock is diffused in 7 cells. In the case of Q_4 FE spaces with non-linear coefficients, the contact is diffused in 5 cells, and the shock is diffused in 2 cells. On the left side of Figure 3.9 we compare the L^2 projections of the first order coefficient (3.22), which was used to generate the left side of Figure 3.8, and the entropy production coefficient (3.26) that was used on the right side of Figure 3.8. We observe that in smooth regions the active part of the minimum (3.27) is the entropy production coefficient, and in shocks the active part is the first order coefficient. On the right of Figure 3.9 we show comparison between pressures obtained by the Option 1 entropy production based coefficient.

In Table 3.3 we show L^1 convergence rates for density computed with Q_1, Q_2 and Q_4 spaces. The table is aligned so that the rates in each row are computed by approximately the same number of degrees of freedom. We observe that the errors and rates for all finite element spaces are similar. We are close to the optimal rate of 1 for discontinuous solutions.

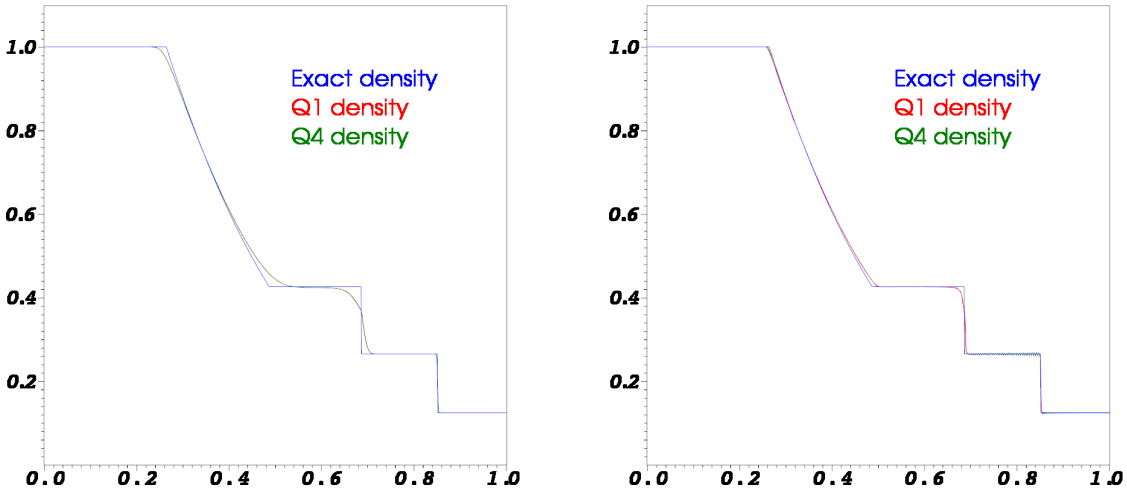

Figure 3.8: Density fields computed with the Option 1 first order coefficients (on the left side), and Option 1 combined with the entropy production based coefficients (on the right side) for the 1D Sod tube problem.

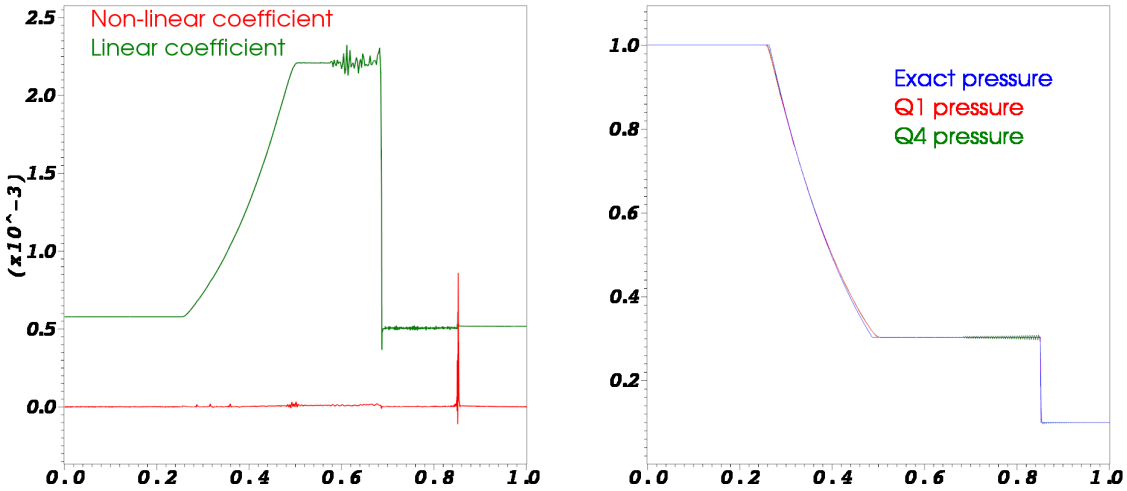

Figure 3.9: Comparison between the L^2 projections of the Option 1 linear and non-linear viscosity coefficients (on the left side), and pressure fields computed by the Option 1 non-linear coefficient (on the right side) for the 1D Sod tube problem.

3.4.3 2D Sedov Explosion

The Sedov explosion, introduced in [54], is a standard problem used for testing shock propagation symmetry. The initial conditions are

Q_1			Q_2			Q_4		
h_0	L^1 error	rate	h_0	L^1 error	rate	h_0	L^1 error	rate
1/64	0.02194		1/32	0.02216		1/16	0.02190	
1/128	0.01319	0.73	1/64	0.01335	0.73	1/32	0.01264	0.79
1/256	0.00723	0.86	1/128	0.00747	0.83	1/64	0.00701	0.85
1/512	0.00382	0.92	1/256	0.00394	0.92	1/128	0.00383	0.87
1/1024	0.00199	0.94	1/512	0.00204	0.95	1/256	0.00207	0.88

Table 3.3: L^1 density errors and convergence rates for the 1D Sod tube problem.

$$\rho(\boldsymbol{x_0}) = 1,\ \boldsymbol{u}(\boldsymbol{x_0}) = 0,\ e(\boldsymbol{x_0}) = \delta(0),\ \int_{\Omega_0} e(\boldsymbol{x_0}) d\boldsymbol{x_0} = E_{tot},\ \gamma = 1.4.$$

The energy deposited as a delta function at the origin converts from internal into kinetic, creating an expanding shock wave. The problem is usually run to $t = 1.0$.

First we discuss results obtained by using Q_2 FE spaces on a Cartesian uniform mesh for $\Omega_0 = [0, 1.2] \times [0, 1.2]$ with 64×64 cells. In this case the initial energy is $E_{tot} = 0.25$. We put $\boldsymbol{u} \cdot \boldsymbol{n} = 0$ on the left and bottom boundaries.

On the right side of Figure 3.10 we make a comparison between the exact density and densities obtained by using the Option 1, see (3.27), and Option 2, see (3.28), viscosity coefficients. We observe that for this example Option 2 is sharper. The left side of Figure 3.10 shows the final density and mesh obtained by using the Option 2 viscosity coefficient (3.28). We observe correct mesh motion and proper shock capturing.

On Figure 3.11 we show the entropy production (3.26) without the scaling with h_3^2, and comparison with a simulation that only uses the first order viscosity (3.23) without taking the minimum (3.28). We see taking into account the entropy based viscosity coefficient in (3.28), our solution becomes sharper.

On Figure 3.12 we show the final velocity magnitude and pressure obtained by

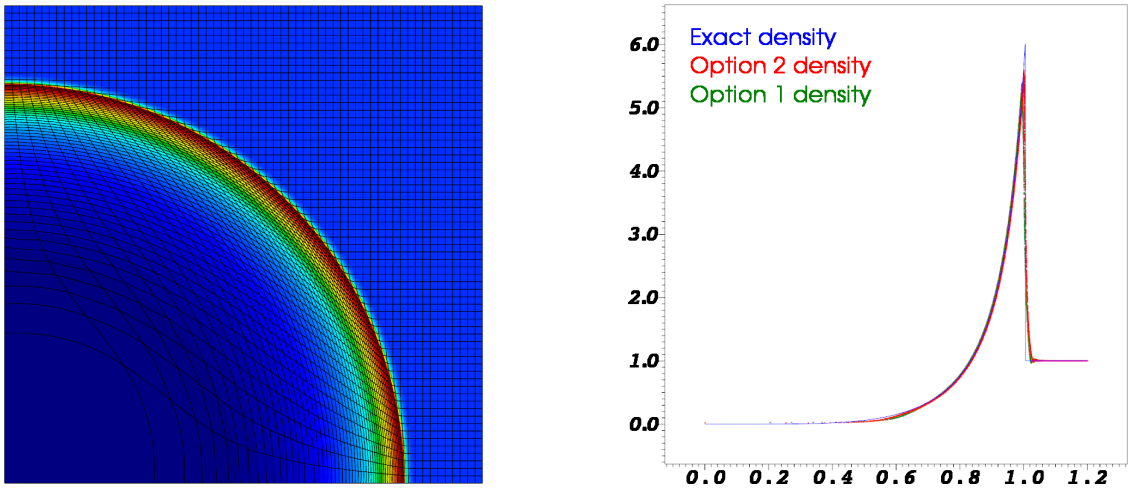

Figure 3.10: Final mesh and density (on the left side), and density vs. radius comparison between exact density and densities obtained by Option 1 and Option 2 viscosity coefficients (on the right side) for the 2D Sedov explosion problem.

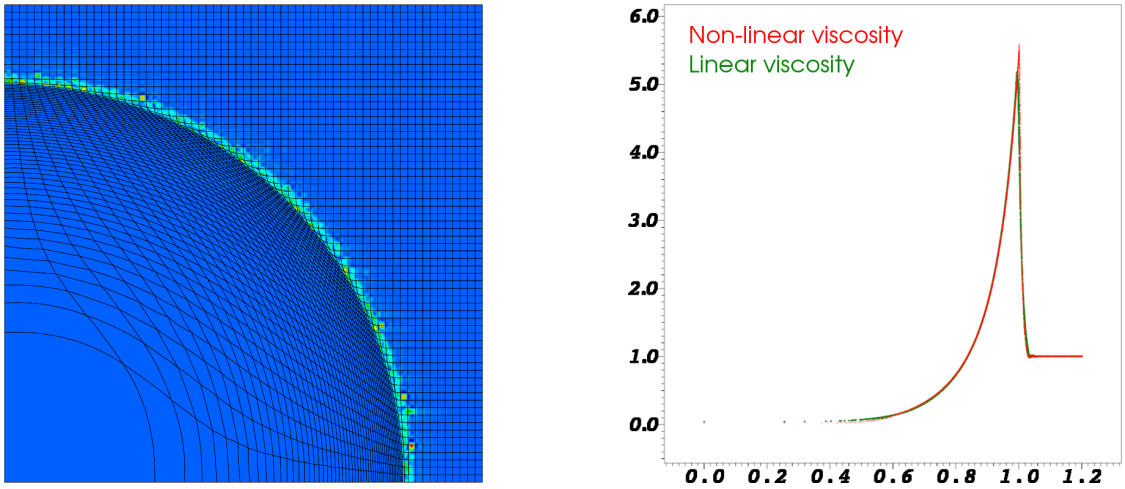

Figure 3.11: Entropy production (on the left side), and density vs. radius comparison between first order and entropy viscosity results (on the right side) for the 2D Sedov explosion problem.

the Option 2 viscosity (3.28).

Next we consider a Cartesian non-uniform mesh for $\Omega_0 = [-1.2, 1.2] \times [-1.2, 1.2]$. In this case the initial energy is $E_{tot} = 1.0$. We have 32×64 cells in quadrant #1,

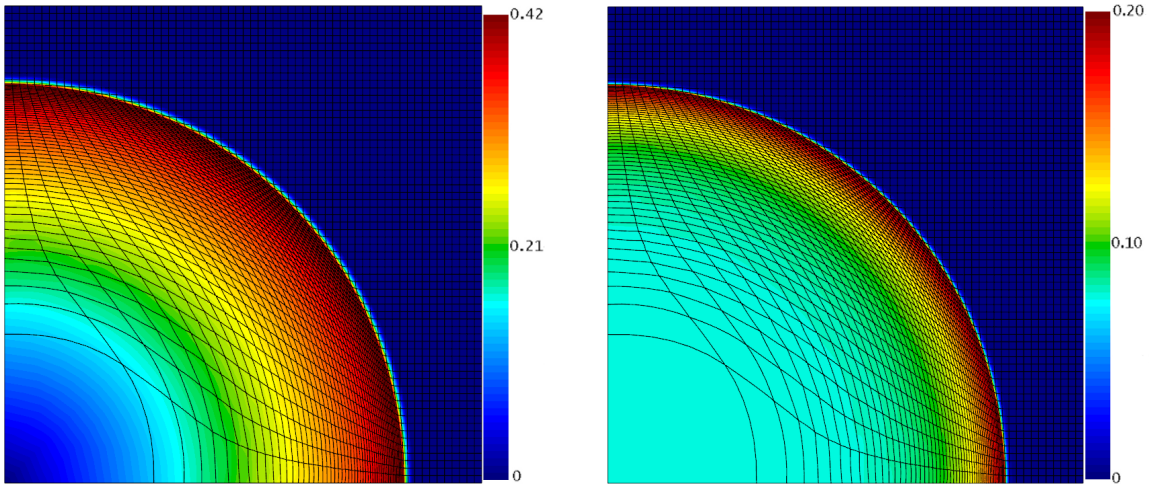

Figure 3.12: Final velocity magnitude (on the left side), and pressure (on the right side) for the 2D Sedov explosion problem.

64×64 in #2, 64×32 in #3, and 32×32 in #4. The purpose of this mesh is to test preservation of radial symmetry and correct mesh evolution. On Figure 3.13 we show final density and mesh, and comparison between exact solution and densities obtained in each quadrant. This simulations uses Q_2 FE spaces. The scatter plot only uses values on the lines $|x| = |y|$. We observe good radial symmetry preservation with respect to the shock location.

3.4.4 2D Noh Implosion

The Noh implosion, introduced in [52], is another problem for testing shock propagation symmetry. The initial conditions are

$$\rho(\boldsymbol{x_0}) = 1, \;\; \boldsymbol{u}(\boldsymbol{x_0}) = \frac{\vec{r}}{||\vec{r}||}, \;\; e(\boldsymbol{x_0}) = 0, \;\; \gamma = \frac{5}{3}.$$

The initial velocity generates an outward traveling shock wave. The problem is usually run to $t = 0.6$.

First we discuss results obtained by using Q_2 FE spaces on a Cartesian uniform

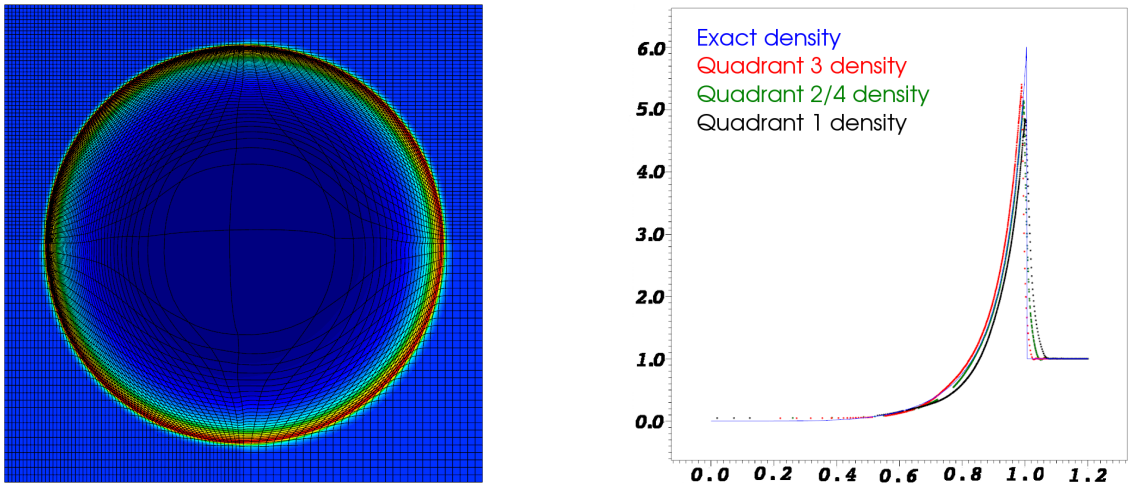

Figure 3.13: Final density and mesh (on the left side), and density vs. radius comparison to exact solution for each quadrant (on the right side), on a non-uniform mesh for the 2D Sedov explosion problem.

mesh for $\Omega_0 = [0, 1.2] \times [0, 1.2]$ with 64×64 cells. We put $\boldsymbol{u} \cdot \boldsymbol{n} = 0$ on the left and bottom boundaries. All simulations presented for this test case use the Option 1 viscosity coefficients (3.22), since the results obtained by the Option 2 coefficients (3.28) are in general more oscillatory.

On the right side of Figure 3.14 we show comparison between the exact density, density obtained by using the Option 1 first order viscosity (3.22) without taking the minimum (3.27), and density obtained by using the full Option 1 coefficients (3.27). By using the non-linear viscosity the result becomes much sharper, but this comes for the cost of some additional oscillations. The left side of Figure 3.14 presents the final density and mesh obtained by using the full Option 1 viscosity (3.27). On Figure 3.15 we show the final forms of the entropy production coefficient (3.26) and the Option 1 first order coefficient (3.22). Again we observe that in smooth regions the active part of the minimum (3.27) is the entropy production coefficient, and in shocks the active part is the first order coefficient.

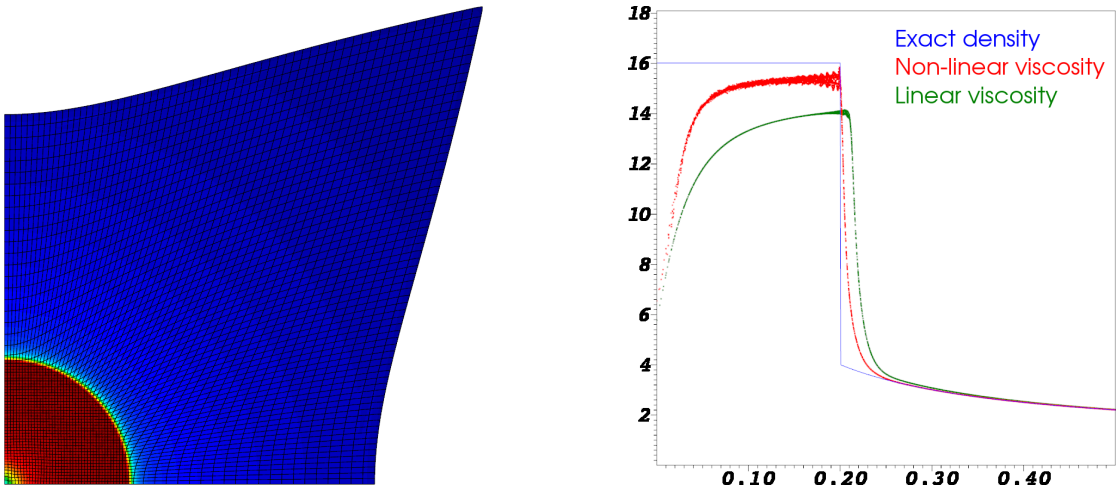

Figure 3.14: Final density and mesh (on the left side), and density vs. radius comparison between exact density, first order and entropy viscosity results (on the right side) for the 2D Noh implosion problem.

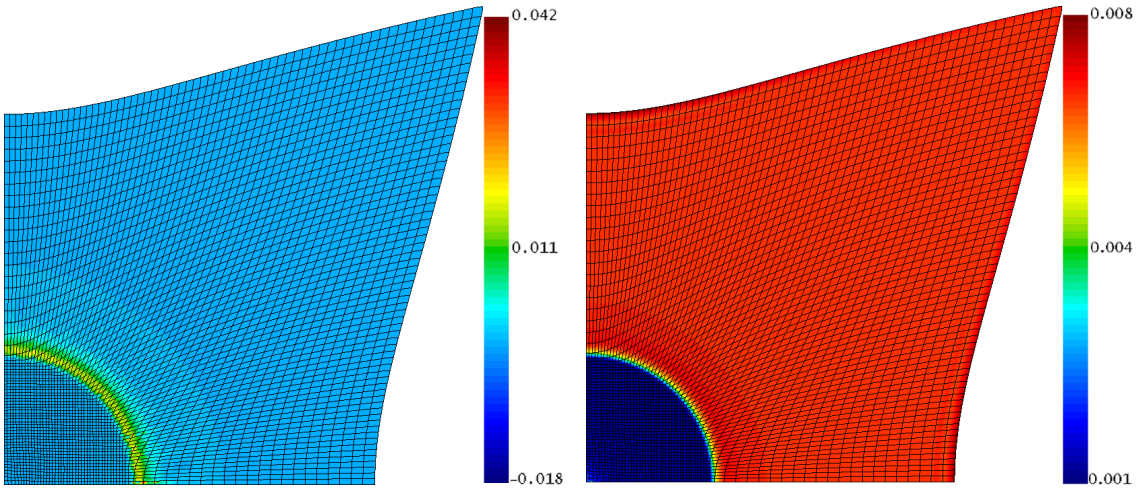

Figure 3.15: Entropy production based viscosity coefficient (on the left side), and the Option 1 first order viscosity coefficient (on the right side) for the 2D Noh implosion problem.

On Figure 3.16 we show the final velocity magnitude and pressure obtained by the Option 1 viscosity (3.27).

Next we consider a Cartesian non-uniform initial mesh for $\Omega_0 = [-1.2, 1.2] \times$

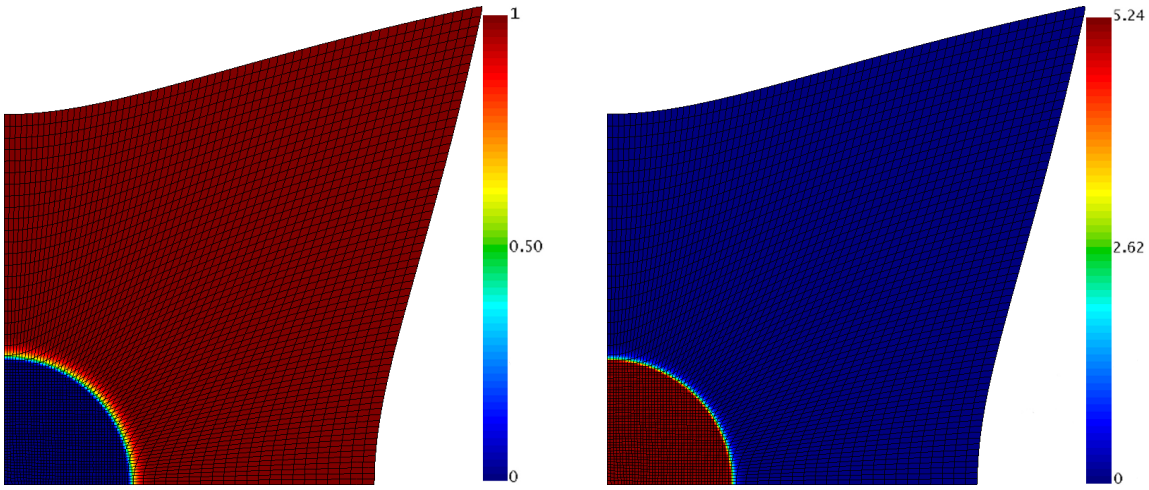

Figure 3.16: Final velocity magnitude (on the left side), and pressure (on the right side) for the 2D Noh implosion problem.

$[-1.2, 1.2]$. We have 32×64 cells in quadrant #1, 64×64 in #2, 64×32 in #3, and 32×32 in #4. The purpose of this mesh is to test preservation of radial symmetry and correct mesh evolution. On Figure 3.17 we present the final density and mesh, and comparison between exact solution and densities obtained in each quadrant. This simulation uses Q_2 FE spaces. We observe some motion in the middle of the domain. This problem is caused by the combination of wall heating, see [52], and having different cells sizes in the middle of the domain. These result in different density values for the different quadrants in the middle, which causes the more dense material to push the less dense material. A way to improve the above situation is to use a smooth initial length scale h_0. The initial length scale we use is shown on Figure 3.18. Although we have applied a smoothing procedure with the very big smoothing constant $\varepsilon = 1000.0$ in (3.18), we still see that the big h_0 gradient in the middle of the domain causes mesh deformation. Using a uniform initial length scale is also not appropriate, since it would cause adding too big (or too small) artificial viscosity to one of the quadrants. The right side of Figure 3.18 shows the final pressure field

where we can see the already mentioned wall heating in the middle of the domain.

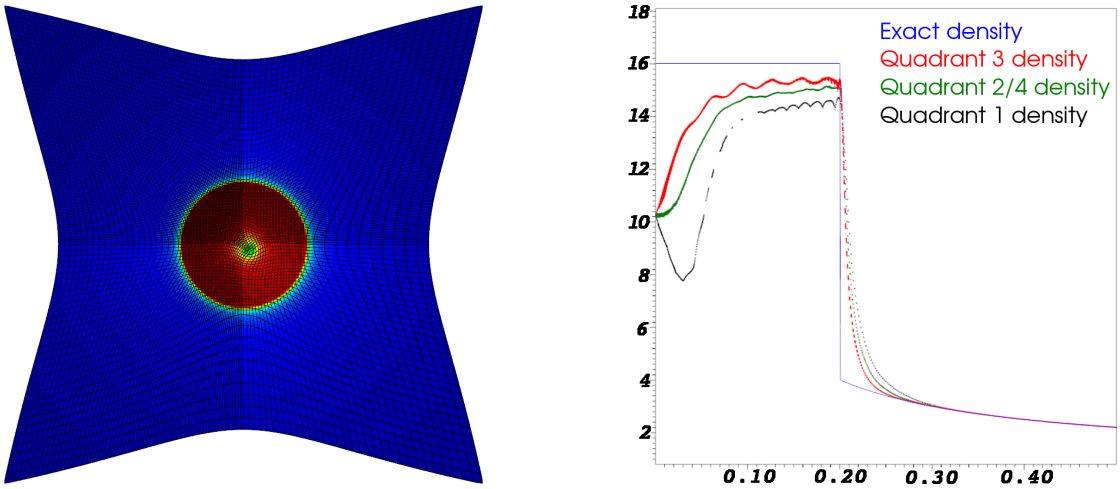

Figure 3.17: Final density and mesh (on the left side), and density vs. radius comparison to exact solution for each quadrant (on the right side), on a non-uniform mesh for the 2D Noh implosion problem.

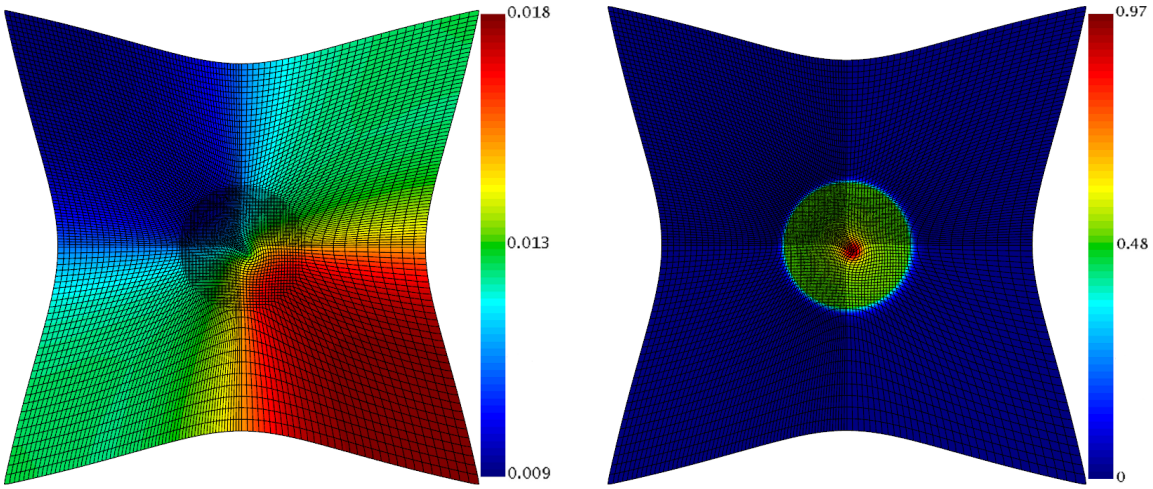

Figure 3.18: Initial length scale at final time (on the left side), and pressure (on the right side), on a non-uniform mesh for the 2D Noh implosion problem

3.4.5 3D Noh Implosion and Parallel Performance

Here we demonstrate our method's ability to perform parallel 3D calculations. The initial conditions are the same as in the 2D case, namely

$$\rho(\boldsymbol{x_0}) = 1,\ \boldsymbol{u}(\boldsymbol{x_0}) = \frac{\vec{r}}{||\vec{r}||},\ e(\boldsymbol{x_0}) = 0,\ \gamma = \frac{5}{3},$$

where all quantities are three-dimensional. Again we run to $t = 0.6$ with Option 1 viscosity coefficients.

On Figure 3.19 we show final density and mesh, and the mesh division between 64 parallel tasks for a simulation with Q_2 spaces on a Cartesian uniform mesh with 32 cells in each space direction. The initial domain is $\Omega_0 = [-1.2, -1.2, -1.2] \times [1.2, 1.2, 1.2]$.

We test the parallel performance of our algorithm by running the above test case to time $t = 0.1$ with Q_2 finite element spaces on a Cartesian mesh consisting of 16 cells in each direction (4096 cells in total). This simulation ends after 62 time steps, the steps are evolved by the RK4 time integrator. Strong scaling results are presented on Figure 3.20. The scaling rates deteriorate when we have less than 128 cells per MPI task. Communication between different MPI tasks occurs for the following procedures:

- Computation of the time step (3.29).
- Computation of the global entropy production normalization constant used in the denominator of equation (3.26).
- Assembly of global matrices for density, velocity and specific internal energy at each Runge-Kutta sub-stage.

- Solving the global linear system for density, velocity and specific internal energy, and distribution of the new solution to the different MPI tasks at each Runge-Kutta sub-stage.

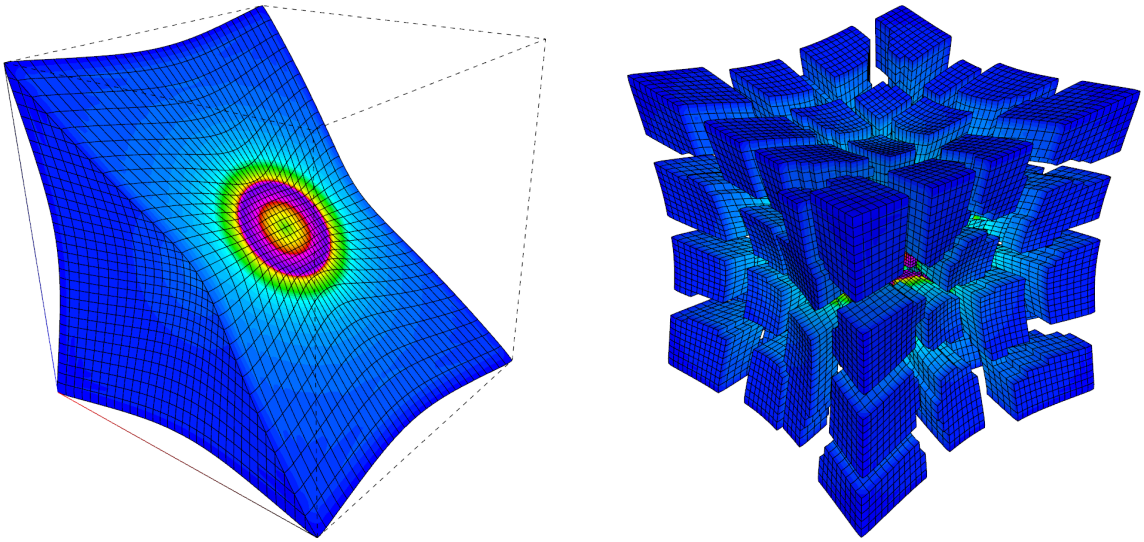

Figure 3.19: Final density (on the left side), and 64 MPI tasks division (on the right side) on 32 cells in each direction for the 3D Noh implosion problem.

3.5 Extensions of the Method

3.5.1 New Viscosity Coefficients

The main idea in this subsection is to define viscosity coefficients that don't depend on any constants and don't use explicit length scales, but the resulting viscous tensor has the correct scaling. The motivation for this approach is to avoid the complications that arise with explicit definitions of length scales and tuning constants that are generally not universal for every test case. This approach has been applied successfully to scalar conservation laws in [25]. First we switch from scalar viscous coefficients to viscous tensors: $\lambda \rightarrow \lambda J J^T, \nu \rightarrow \nu J J^T$ where J is still the Jacobian of

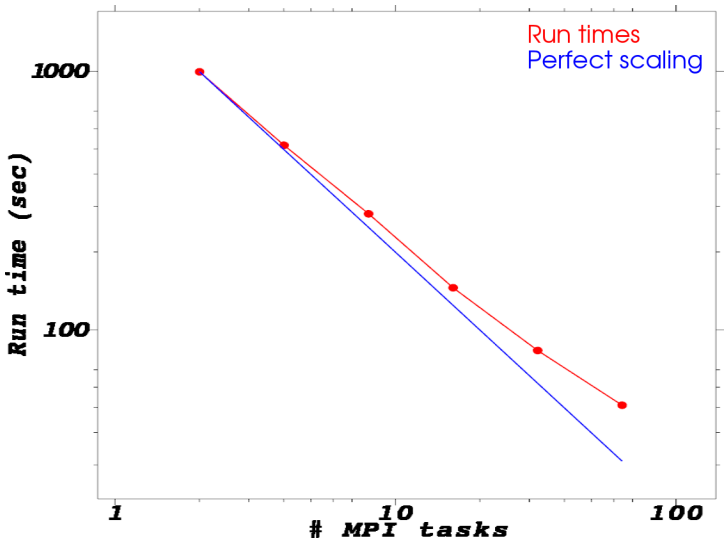

Figure 3.20: Strong scaling test for the 3D Noh implosion problem. Run times with 2^k MPI tasks, $k = 1...6$, compared to perfect scaling.

the transformation $\hat{\boldsymbol{x}} \to \boldsymbol{x}$. Notice that in these new terms the coefficients λ and ν must scale like (speed/distance), which would be the scaling of the entropy viscosity coefficient (3.26) before the multiplication by h_3^2. Then the density equation becomes

$$\frac{d\rho}{dt} = -\rho\nabla\cdot\boldsymbol{u} + \nabla\cdot(\lambda JJ^T\nabla\rho).$$

The equation for the m-th velocity component u_m becomes

$$\rho\frac{du_m}{dt} = -\frac{\partial p}{\partial x_m} + \nabla\cdot(\nu\rho[JJ^T\nabla\boldsymbol{u}]_m) + \nabla u_m\cdot(\lambda JJ^T\nabla\rho),$$

where A_m is the m-th column vector of a matrix A. Notice that

$$[JJ^T\nabla\boldsymbol{u}]_m = (JJ^T)[\nabla\boldsymbol{u}]_m = JJ^T\nabla u_m.$$

The last term of the velocity equations can be written as

$$\nabla u_m \cdot (\lambda J J^T \nabla \rho) = \lambda (\nabla u_m)^T J (J^T \nabla \rho) = \lambda (J^T \nabla u_m)^T (J^T \nabla \rho) = \lambda (J^T \nabla u_m) \cdot (J^T \nabla \rho).$$

So finally the m-th velocity equation becomes

$$\rho \frac{du_m}{dt} = -\frac{\partial p}{\partial x_m} + \nabla \cdot (\nu \rho J J^T \nabla u_m) + \lambda (J^T \nabla u_m) \cdot (J^T \nabla \rho).$$

The internal energy equation is

$$\rho \frac{de}{dt} = -p \nabla \cdot \boldsymbol{u} + \nabla \cdot (\lambda \rho J J^T \nabla e) + \nu \rho (J J^T \nabla \boldsymbol{u}) : \nabla \boldsymbol{u} + \nabla e \cdot (\lambda J J^T \nabla \rho).$$

Here notice that

$$\begin{aligned} \nu \rho (J J^T \nabla \boldsymbol{u}) : \nabla \boldsymbol{u} &= \nu \rho \sum_{m=1}^{d} [J J^T \nabla \boldsymbol{u}]_m \cdot \nabla u_m \\ &= \nu \rho \sum_{m=1}^{d} (J J^T \nabla u_m) \cdot \nabla u_m = \nu \rho \sum_{m=1}^{d} (J^T \nabla u_m)^2. \end{aligned}$$

Then the internal energy equation becomes

$$\rho \frac{de}{dt} = -p \nabla \cdot \boldsymbol{u} + \nabla \cdot (\lambda \rho J J^T \nabla e) + \nu \rho \sum_{m=1}^{d} (J^T \nabla u_m)^2 + \lambda (J^T \nabla e) \cdot (J^T \nabla \rho).$$

The final system is:

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \boldsymbol{u} + \nabla \cdot (\lambda J J^T \nabla \rho), \tag{3.30}$$

$$\rho \frac{du_m}{dt} = -\frac{\partial p}{\partial x_m} + \nabla \cdot (\nu \rho J J^T \nabla u_m) + \lambda (J^T \nabla u_m) \cdot (J^T \nabla \rho), \quad m = 1..d, \tag{3.31}$$

$$\rho \frac{de}{dt} = -p \nabla \cdot \boldsymbol{u} + \nabla \cdot (\lambda \rho J J^T \nabla e) + \nu \rho \sum_{m=1}^{d} (J^T \nabla u_m)^2 + \lambda (J^T \nabla e) \cdot (J^T \nabla \rho). \tag{3.32}$$

3.5.1.1 Mass Viscosity Coefficient $\boldsymbol{\lambda}$

The coefficient λ is obtained by imposing positivity of density on discrete level. We examine a continuous, piecewise linear finite element approximation of the density equation (3.30). The i-th equation for time t_{k+1} is

$$\sum_{j=1}^{N} \frac{\rho_j^{k+1} - \rho_j^k}{\Delta t} \int_{K \subset S_i} \varphi_i \varphi_j \, d\boldsymbol{x} = - \sum_{K \subset S_i} \int_K \left(\lambda_K (J^T \nabla \rho^k)(J^T \varphi_i) + \rho^k \nabla \cdot \boldsymbol{u}^k \varphi_i \right) d\boldsymbol{x},$$

where S_i is the support of the basis function φ_i. We lump the mass matrix and the left-hand side becomes

$$\sum_{j=1}^{N} \frac{\rho_j^{k+1} - \rho_j^k}{\Delta t} \int_{K \subset S_i} \varphi_i \varphi_j \, d\boldsymbol{x} \to \sum_{j=1}^{N} \frac{\rho_i^{k+1} - \rho_i^k}{\Delta t} \int_{K \subset S_i} \varphi_i \varphi_j \, d\boldsymbol{x} =$$

$$= \frac{\rho_i^{k+1} - \rho_i^k}{\Delta t} \int_{K \subset S_i} \left(\varphi_i \sum_{j=1}^{N} \varphi_j \right) d\boldsymbol{x} == \frac{\rho_i^{k+1} - \rho_i^k}{\Delta t} \int_{K \subset S_i} \varphi_i \, d\boldsymbol{x} =: \frac{\rho_i^{k+1} - \rho_i^k}{\Delta t} m_i.$$

Then the i-th equation is

$$\rho_i^{k+1} = \rho_i^k - \frac{\Delta t}{m_i} \sum_{K \subset S_i} \int_K \left(\lambda_K (J^T \nabla \rho^k) \cdot (J^T \varphi_i) + \rho^k \nabla \cdot \boldsymbol{u}^k \varphi_i \right) d\boldsymbol{x},$$

$$\begin{aligned} \rho_i^{k+1} =& \rho_i^k \left(1 - \frac{\Delta t}{m_i} \sum_{K \subset S_i} \int_K \left(\lambda_K (J^T \nabla \varphi_i)^2 + \nabla \cdot \boldsymbol{u}^k \varphi_i^2 \right) \right) d\boldsymbol{x} \\ & - \sum_{j \in I(S_i), j \neq i} \rho_j^k \frac{\Delta t}{m_i} \sum_{K \subset S_{ij}} \int_K \left(\lambda_K (J^T \nabla \varphi_j) \cdot (J^T \nabla \varphi_i) + \varphi_j \nabla \cdot \boldsymbol{u}^k \varphi_i \right) d\boldsymbol{x}, \end{aligned}$$

where S_{ij} is the common support of the basis functions φ_i, φ_j. The density will stay positive, if the coefficients of $\rho_j^k, j = 1..N$ are all positive. The coefficient of ρ_i^k is handled by the CFL condition on the time step Δt. This time step is expected to be small enough in order to keep the multiplier of ρ_i^k positive. However we are more

interested in the other term of the above expression. The coefficients of $\rho_j^k, j \neq i$ can be kept positive by setting the viscosity constant λ to

$$\lambda_K = \max_{j \neq i \in I(K)} \frac{\left| \int_{S_{ij}} \nabla \cdot \boldsymbol{u}^k \varphi_i \varphi_j \, d\boldsymbol{x} \right|}{-\int_{S_{ij}} (J^T \nabla \varphi_j) \cdot (J^T \nabla \varphi_i) \, d\boldsymbol{x}}. \tag{3.33}$$

Here $I(K)$ is the set of indices j so that φ_j has support in K. Another option that is more convenient for parallel computations is

$$\lambda_K = \max_{j \neq i \in I(K)} \frac{\left| \int_{K} \nabla \cdot \boldsymbol{u}^k \varphi_i \varphi_j \, d\boldsymbol{x} \right|}{-\int_{K} (J^T \nabla \varphi_j) \cdot (J^T \nabla \varphi_i) \, d\boldsymbol{x}}. \tag{3.34}$$

Notice that λ_K scales like speed over distance and the Frobenius norm of $\lambda_K J J^T$ scales like speed times distance as expected.

3.5.1.2 Momentum Viscosity Coefficient $\boldsymbol{\nu}$

In this subsection we try to derive a length-scale independent definition for our other viscosity coefficient, ν. One way to derive ν_K is to use a Prandtl number connection:

$$\mathcal{P} = \frac{\lambda_K}{\nu_K}, \tag{3.35}$$

but then the question what is the appropriate $\mathcal{P}$ remains open.

Here we show a derivation of ν by imposing the $\frac{ds}{dt} \geq 0$ on discrete level. The final result, however, turns out to be unusable for numerical computations. As before, we obtain the specific entropy equation by multiplying (3.30) by ρs_ρ, (3.32) by s_e and adding the two resulting equations:

$$\rho \left(s_\rho \frac{d\rho}{dt} + s_e \frac{de}{dt} \right) + \nabla \cdot \boldsymbol{u} \cancel{\left(\rho^2 s_\rho + p s_e \right)} = \rho s_\rho \nabla \cdot (\lambda J J^T \nabla \rho) + s_e(...),$$

$$\rho\frac{ds}{dt} = \rho s_\rho \nabla\cdot(\lambda J J^T\nabla\rho) + s_e\left(\nabla\cdot(\lambda\rho J J^T\nabla e) + \nu\rho\sum_{m=1}^{d}(J^T\nabla u_m)^2 + \lambda(J^T\nabla e)\cdot(J^T\nabla\rho)\right). \tag{3.36}$$

The specific entropy will not decrease, if the right-hand side is non-negative. Although, in our setting, s is not a finite element function, we still introduce a weak form of (3.36), because we want to avoid computing second derivatives of ρ and e. As before, we introduce a piecewise linear finite element approximation with lumped the mass matrix. Then the i-th equation becomes

$$s_i^{k+1} = s_i^k + \frac{\Delta t}{m_i}\sum_{K\subset S_i}\int_K\left(\lambda_K\rho^k s_\rho(J^T\nabla\rho^k)\cdot(J^T\nabla\varphi_i) + \lambda_K\rho^k s_e(J^T\nabla e^k)\cdot(J^T\nabla\varphi_i) + \nu_K\rho^k s_e\sum_{m=1}^{d}(J^T\nabla u_m)^2\varphi_i + \lambda_K s_e(J^T\nabla e^k)\cdot(J^T\nabla\rho^k)\varphi_i\right)d\boldsymbol{x}.$$

Then ν_K is defined by

$$\nu_K = \frac{\lambda_K}{\int_K\rho^k s_e\sum_{m=1}^{d}(J^T\nabla\boldsymbol{u}_m^k)^2\varphi_i\,d\boldsymbol{x}}\max_i\left(\left|\int_K\rho^k s_\rho(J^T\nabla\rho^k)\cdot(J^T\nabla\varphi_i) + \rho^k s_e(J^T\nabla e^k)\cdot(J^T\nabla\varphi_i) + s_e(J^T\nabla e^k)\cdot(J^T\nabla\rho^k)\varphi_i\,d\boldsymbol{x}\right|\right) \tag{3.37}$$

However this expression is not applicable for discrete computations since it is not well-defined in cases of constant velocity (and this is the usual initial condition for most standard benchmark problems), zero density, or zero internal energy. Because of that we have no numerical data for (3.37).

Remark As we saw above, one might consider evolving s, instead of e, as a finite element variable.

3.5.1.3 Combination with Entropy Production

Taking the entropy production into account is trivial, since the entropy production D defined in (3.24) has the correct scaling:

$$\begin{aligned} \lambda &= \min\left(\lambda, c_{entr}\frac{|D|}{|S^n - \overline{S^n}|_{\infty,\Omega(t_k)}}\right), \\ \nu &= \min\left(\nu, c_{entr}\frac{|D|}{|S^n - \overline{S^n}|_{\infty,\Omega(t_k)}}\right). \end{aligned} \tag{3.38}$$

The entropy coefficients, however, still depend on the tunable constant c_{entr}.

3.5.1.4 Preliminary Numerical Results

Here we show the application of the new length-scale independent coefficients to the Sod tube and Sedov problems. We use Q_1 finite element spaces. We don't lump the mass matrices since it causes blow-ups. The results are obtained by using the Prandtl number connection (3.35) with $\mathcal{P} = 0.1$.

On Figure 3.21 we show the new coefficients produce the expected result for the 1D Sod tube problem. Again we observe no oscillations in the contact region and precise shock capturing.

A more complicated example, the 2D Sedov problem, is shown on Figure 3.22. Unfortunately our results show mesh tangling. Mesh motion can be improved by increasing the Prandtl number, but this comes for the price of increased diffusion.

3.5.1.5 Summary of the Approach and Open Problems

The new definitions (3.33), (3.35) of the viscosity coefficients (λ, ν) remove the requirement for explicit definition of length scale, they provide positivity of density on discrete level, and the coefficients are easy to combine with the entropy production by equation (3.38). By using these new coefficient we obtain encouraging initial

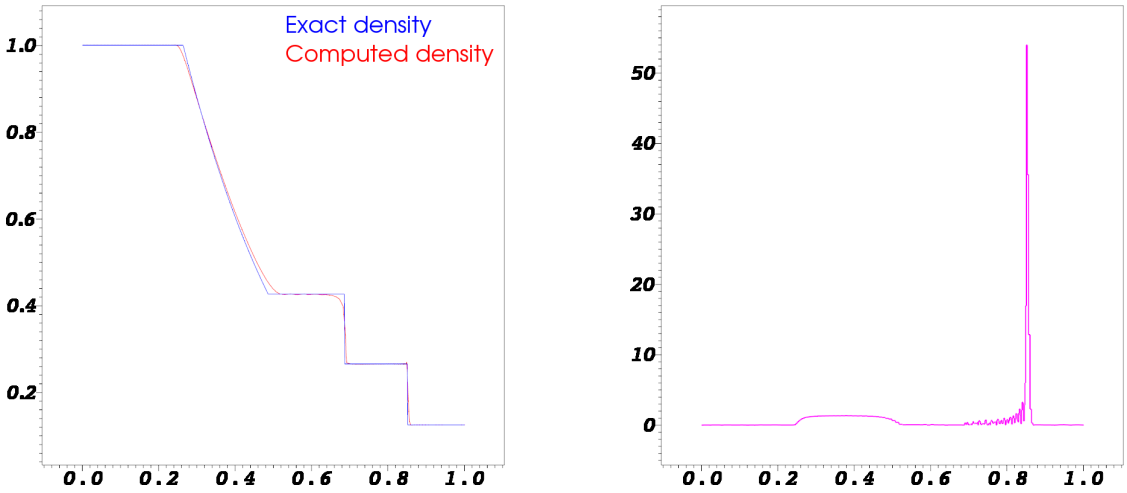

Figure 3.21: Final density and exact solution (on the left side), and L^2 projection of the final piecewise constant coefficient λ (on the right side) for the 1D Sod tube problem.

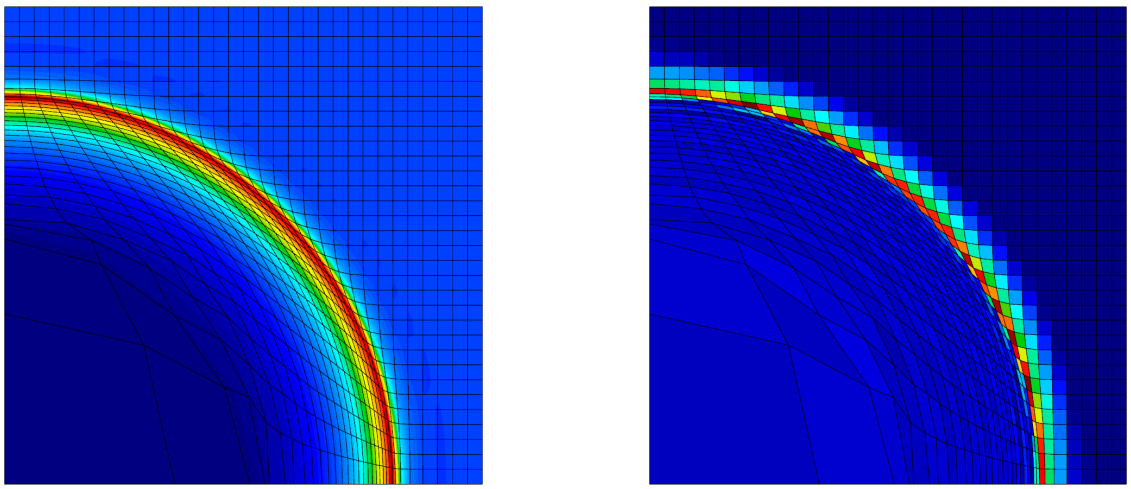

Figure 3.22: Final density (on the left side) and the piecewise constant coefficient λ (on the right side) for the 2D Sedov explosion problem

numerical results. However there are still some problems that must be addressed:

- The definition of the artificial force coefficient ν for the momentum equation is not clear as discussed after the derivation of equation (3.37).

- Using tensor viscosity coefficients of the form λJJ^T results in losing radial symmetry on continuous level, see Remark 4 in [36]. The mesh tangling we observe in the simulation of the 2D Sedov problem is most likely caused by loss of radial symmetry.

- The discussed approach assumes the use of Q_1 spaces. Extensions to spaces of higher degree are not established. There are ongoing efforts involving the use of sub-grids and special mass lumpings.

- The discussed approach expects lumped matrices. While this is not an issue for scalar conservation laws, how to lump matrices for the Euler equations and obtain stable solutions is not clear at present time.

- For 3D simulations, the denominator of (3.33) may become negative. In order to avoid that, additional approximations are necessary, see [25].

3.5.2 The Brenner Model

The regularization we present by equations (3.1)-(3.3) is easy to transform into a model of fluid dynamics proposed by Brenner in [7]. In this subsection we derive this model and discuss its properties. Our ultimate goal is to achieve pointwise mass conservation and mass matrices that are constant in time, while retaining all the properties of our regularization.

The main idea is to introduce additional velocity variable $\bar{\boldsymbol{u}}$ which is used to move the computational mesh. This new velocity is derived by incorporating the mass diffusion into the mesh motion. In Eulerian frame, our regularized density equation is

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = \nabla \cdot (\lambda \nabla \rho).$$

This equation can be written as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \left[\rho \left(\boldsymbol{u} - \lambda \frac{\nabla \rho}{\rho} \right) \right] = 0.$$

Motivated by this equation, we define our mesh velocity $\bar{\boldsymbol{u}}$ and use it to move our positions $\boldsymbol{x}$:

$$\bar{\boldsymbol{u}} := \boldsymbol{u} - \lambda \frac{\nabla \rho}{\rho}, \tag{3.39}$$

$$\frac{d\boldsymbol{x}(\boldsymbol{x_0}, t))}{dt} := \bar{\boldsymbol{u}}(\boldsymbol{x}, t). \tag{3.40}$$

Then the material derivative of a scalar / vector function $\beta = \beta(\boldsymbol{x}(\boldsymbol{x_0}, t), t)$ is

$$\frac{d}{dt}\beta(\boldsymbol{x}(\boldsymbol{x_0}, t), t) = \frac{\partial \beta(\boldsymbol{x}, t)}{\partial t} + \bar{\boldsymbol{u}}(\boldsymbol{x}, t) \cdot \nabla_{\boldsymbol{x}} \beta(\boldsymbol{x}, t).$$

Using the new notion of material derivative, the Lagrangian frame density equation becomes

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \bar{\boldsymbol{u}}. \tag{3.41}$$

The above procedure moved the diffusion term into the mesh motion. A hopeful guess would be that the new equation (3.41) also diffuses the density field by expanding zones in which we have big density gradients. This, however, is not always the case. Depending on the signs of $\boldsymbol{u}$ and $\nabla \rho$, by looking at (3.39), one can easily come up with an example so that we get $\nabla \cdot \bar{\boldsymbol{u}} < \nabla \cdot \boldsymbol{u}$, meaning that the new mesh velocity $\bar{\boldsymbol{u}}$ actually causes more compression than the old velocity $\boldsymbol{u}$. A way to resolve this is to introduce a switch in the sign of viscosity coefficient λ, but one should not forget that the coefficient λ is also used in the thermal diffusion term, where we must always have a positive sign.

3.5.2.1 Pointwise Mass Conservation

Equation (3.41) implies pointwise mass conservation. For completeness we show the derivation of this statement. By the Reynolds transport theorem and (3.41) we get

$$\begin{aligned}\frac{d}{dt}\int_{\Omega(t)} \rho(\boldsymbol{x},t)\ d\boldsymbol{x} &= \int_{\Omega(t)} \frac{\partial \rho(\boldsymbol{x},t)}{\partial t} + \nabla\cdot(\rho(\boldsymbol{x},t)\bar{\boldsymbol{u}}(\boldsymbol{x},t))\, d\boldsymbol{x} \\ &= \int_{\Omega(t)} \frac{d\rho(\boldsymbol{x},t)}{dt} + \rho\nabla\cdot\bar{\boldsymbol{u}}\ d\boldsymbol{x} = 0,\end{aligned}$$

hence for any domain $\Omega(t)$ that was originally Ω_0, we have the equality

$$\int_{\Omega(t)} \rho(\boldsymbol{x},t)d\boldsymbol{x} = \int_{\Omega_0} \rho_0(\boldsymbol{x_0})d\boldsymbol{x_0}.$$

Here we use the mapping $\Phi\circ\Phi_0^{-1} : \boldsymbol{x_0} \to \boldsymbol{x}$ with Jacobian JJ_0^{-1} from initial to actual coordinates to obtain

$$\int_{\Omega_0} \rho(\boldsymbol{x},t)\left|JJ_0^{-1}(\boldsymbol{x_0},t)\right| d\boldsymbol{x_0} = \int_{\Omega_0} \rho_0(\boldsymbol{x_0})d\boldsymbol{x_0}.$$

Since Ω_0 was an arbitrary control volume, at every point $\boldsymbol{x}$ that originates from $\boldsymbol{x_0}$, we have the pointwise mass conservation:

$$\rho(\boldsymbol{x},t)\left|JJ_0^{-1}(\boldsymbol{x_0},t)\right| = \rho_0(\boldsymbol{x_0}). \tag{3.42}$$

3.5.2.2 New Form of the System

Here we state the final Lagrangian system after taking into account the new mesh motion (3.40). We start with the velocity equation (3.8) and rewrite it as

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} - \lambda\frac{\nabla\rho}{\rho}\cdot\nabla\boldsymbol{u}\right) = -\nabla p + \nabla\cdot(\nu\rho\nabla\boldsymbol{u}),$$

$$\Rightarrow \rho\frac{d\boldsymbol{u}}{dt} = -\nabla p + \nabla\cdot(\nu\rho\nabla\boldsymbol{u}).$$

The same procedure applies for the specific internal energy equation (3.9):

$$\rho\left(\frac{\partial e}{\partial t} + \boldsymbol{u}\cdot\nabla e - \lambda\frac{\nabla\rho}{\rho}\cdot\nabla e\right) = -p\nabla\cdot\boldsymbol{u} + \nabla\cdot(\lambda\rho\nabla e) + \nu\rho\nabla\boldsymbol{u} : \nabla\boldsymbol{u},$$

$$\Rightarrow \rho\frac{de}{dt} = -p\nabla\cdot\boldsymbol{u} + \nabla\cdot(\lambda\rho\nabla e) + \nu\rho\nabla\boldsymbol{u} : \nabla\boldsymbol{u}.$$

Then the new Lagrangian frame system takes the following form:

$$\frac{d}{dt}\boldsymbol{x}(\boldsymbol{x_0}, t) = \bar{\boldsymbol{u}}(\boldsymbol{x}, t), \quad \text{where} \quad \bar{\boldsymbol{u}} := \boldsymbol{u} - \lambda\frac{\nabla\rho}{\rho}, \tag{3.43}$$

$$\rho(\boldsymbol{x}, t)\left|JJ_0^{-1}(\boldsymbol{x_0}, t)\right| = \rho_0(\boldsymbol{x_0}), \tag{3.44}$$

$$\rho\frac{d\boldsymbol{u}}{dt} = -\nabla p + \nabla\cdot(\nu\rho\nabla\boldsymbol{u}), \tag{3.45}$$

$$\rho\frac{de}{dt} = -p\nabla\cdot\boldsymbol{u} + \nabla\cdot(\lambda\rho\nabla e) + \nu\rho\nabla\boldsymbol{u} : \nabla\boldsymbol{u}. \tag{3.46}$$

We solve with respect to position $\boldsymbol{x}(\boldsymbol{x_0}, t)$, density $\rho(\boldsymbol{x}, t)$, velocity $\boldsymbol{u}(\boldsymbol{x}, t)$, internal energy per mass $e(\boldsymbol{x}, t)$. The equation of state (1.23) is still $p = (\gamma - 1)\rho e$. Note that the density ρ can be computed pointwise, it doesn't have to be a finite element function. Also note that we need to introduce a weak form of the equation for the mesh coordinates $\boldsymbol{x}$, since the expression for $\bar{\boldsymbol{u}}$ is not in the used finite element space.

The weak form for equation (3.43) is the following:

For every dimension $m = 1...d$ and every $j = 1...N$ we have

$$\int_{\Omega(t)} \frac{dx_m(\boldsymbol{x_0}, t)}{dt} \varphi_j(\boldsymbol{x}, t)\ d\boldsymbol{x} = \int_{\Omega(t)} u_m(\boldsymbol{x}, t)\varphi_j(\boldsymbol{x}, t)\ d\boldsymbol{x}$$

$$+ \int_{\Omega(t)} \lambda(\ln \rho) \frac{\partial \varphi_j(\boldsymbol{x}, t)}{\partial x_m}\ d\boldsymbol{x} - \int_{\partial\Omega(t)} \lambda(\ln \rho)\varphi_j(\boldsymbol{x}, t) n_d\ d\boldsymbol{x}.$$

The weak forms for velocity $\boldsymbol{u}$ and specific internal energy e are the ones from (3.12), (3.13) by removing the terms which are moved in the material motion. All other details of the discrete method are applicable to this new problem.

3.5.2.3 Constant Mass Matrices

Another advantage of this approach is that all resulting mass matrices are constant in time. For example, the left-hand side of (3.13) and the corresponding matrix are

$$\int_{\Omega(t)} \rho(\boldsymbol{x}, t) \frac{d}{dt} \left(\sum_{i=1}^{N} e_i(t)\varphi_i(\boldsymbol{x}, t) \right) \varphi_j(\boldsymbol{x}, t) d\boldsymbol{x} = \sum_{i=1}^{N} \frac{de_i(t)}{dt} \int_{\Omega(t)} \rho(\boldsymbol{x}, t)\varphi_i(\boldsymbol{x}, t)\varphi_j(\boldsymbol{x}, t) d\boldsymbol{x},$$

$$(M_e)_{ij} = \int_{\Omega_0} \rho(\boldsymbol{x}, t)\varphi_i(\boldsymbol{x_0})\varphi_j(\boldsymbol{x_0}) |J J_0^{-1}(\boldsymbol{x_0}, t)| d\boldsymbol{x_0}.$$

Here $\rho(\boldsymbol{x}, t) |\det J J_0^{-1}(\boldsymbol{x_0}, t)| = \rho(\boldsymbol{x_0})$ holds by (3.42), hence the mass matrix is constant in time and can be assembled only at initial time.

3.5.2.4 Summary of the Approach and Open Problems

Introducing the mesh velocity $\bar{\boldsymbol{u}}$ enables us to derive a system that conserves mass pointwise and produces time-independent mass matrices. All regularization

properties are also valid for this new system. We don't need to solve a finite element weak form for the density ρ, since it can be evolved pointwise by using the Jacobian matrix of the mapping $\boldsymbol{x_0} \rightarrow \boldsymbol{x}$, but we need to solve a weak form for the mesh positions $\boldsymbol{x}$. All other details of the discrete method can be used for the numerical solution of the new system. The real problem of the approach is how to define the coefficient λ, so that the mesh velocity given by (3.39) always results in diffusion of the density field, and at the same time the definition of λ is appropriate in the specific internal energy equation (3.46).

4. ALE HYDRODYNAMICS IN BLAST

This section presents approaches to problems I faced as part of the BLAST team in Lawrence Livermore National Laboratory during the summers of 2012 and 2013. BLAST [41] is a high-order finite element hydrodynamics research code. Its goal is to improve the accuracy of Lagrangian and Arbitrary Lagrangian-Eulerian (ALE) simulations for compressible Euler equations, and to provide a viable path to extreme parallel computing and exascale architectures. Related publications and additional details about BLAST can be found in [41, 20, 16, 21, 17]. My work was concentrated in improving the ALE capabilities of BLAST, namely the research and development of algorithms for parallel mesh relaxation, parallel multi-field advection remap, and multi-material simulations.

The main differences between the Lagrangian phase in BLAST, and the Entropy Viscosity method from Section 3, are the viscous regularization, the form of the viscosity coefficients, and the used finite element spaces. BLAST uses the Navier-Stokes regularization approach, compression based viscosity coefficients, discontinuous finite element spaces for the thermodynamic variables, and continuous FE spaces for the kinematic variables. The details of these are discussed in Subsection 3.2 where we make a general overview of the Lagrangian phase in BLAST.

In this section we introduce the stages of the ALE approach in the context of BLAST and discuss the connections with the Entropy Viscosity method. The goal of ALE is to overcome mesh tangling, mesh imprinting, and too small time steps that usually result from the Lagrangian phase. ALE methods extend the Lagrangian ones by three additional stages:

1. Mesh optimization - once the mesh quality during the Lagrangian phase deteri-

orates, one introduces a better mesh according to some quantities that measure mesh quality.

2. Solution remap - solution defined on the old mesh is mapped to a solution defined on the new mesh. The remap algorithms must preserve mass, momentum, kinetic and internal energy, and the functions' maximum and minimum values.

3. Multi-material zone treatment - the solution remap step introduces mixed zones, where a single cell contains multiple materials. In those zones one needs to come up with appropriate mixing of equations of state or reconstructing an exact interface in order to compute pressure, sound speed, etc.

Approaches to the above steps are discussed in Subsections 4.2, 4.3 and 4.4, respectively. Throughout this section we stick to the notation established throughout Section 3 and in particular Subsection 3.3.1.

4.1 Overview of the Lagrangian Phase in BLAST

In this subsection we make a brief overview of the Lagrangian phase in BLAST. We focus on its differences with the Entropy Viscosity method which is discussed in Section 3, while concepts that overlap in both methods are omitted. Complete description of the Lagrangian phase in BLAST can be found in [20].

4.1.1 Viscous Regularization

The Lagrangian phase of BLAST uses the Navier-Stokes regularization approach, namely the regularized system in Lagrangian frame has the form

$$\frac{d}{dt}\boldsymbol{x}(\boldsymbol{x_0}, t) = \boldsymbol{u}(\boldsymbol{x}, t), \tag{4.1}$$

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \boldsymbol{u}, \tag{4.2}$$

$$\rho \frac{d\boldsymbol{u}}{dt} = -\nabla p + \nabla \cdot \sigma_a, \tag{4.3}$$

$$\rho \frac{de}{dt} = -p \nabla \cdot \boldsymbol{u} + \sigma_a : \nabla \boldsymbol{u}, \tag{4.4}$$

where σ_a is an artificial stress tensor that depends only on velocity gradients. The corresponding Lagrangian frame regularization used in the Entropy Viscosity method is presented in Subsection 3.2. Advantages and disadvantages of both approaches are already discussed in Subsection 3.1.

BLAST supports four distinct artificial stress types. The symmetric velocity gradient can be decomposed in the form

$$\nabla^s \boldsymbol{u} = \sum_{m=1}^{d} \mu_m \boldsymbol{s_m} \otimes \boldsymbol{s_m}, \qquad \boldsymbol{s_i} \cdot \boldsymbol{s_j} = 0, \qquad \mu_1 \leq ... \leq \mu_d,$$

where μ_k and $\boldsymbol{s_k}$ are its eigenvalues and eigenvectors, respectively, sorted from smallest to largest eigenvalue. If we define a measure of compression in some arbitrary direction $\boldsymbol{s}$ as

$$\Delta_{\boldsymbol{s}} \boldsymbol{u} := \lim_{\alpha \to 0} \frac{\boldsymbol{u}(\boldsymbol{x} + \alpha \boldsymbol{s})}{\alpha |\boldsymbol{s}|} \cdot \frac{\boldsymbol{s}}{|\boldsymbol{s}|} = \frac{\boldsymbol{s} \cdot \nabla \boldsymbol{u} \cdot \boldsymbol{s}}{\boldsymbol{s}^2} = \frac{\boldsymbol{s} \cdot \nabla^s \boldsymbol{u} \cdot \boldsymbol{s}}{\boldsymbol{s}^2}, \tag{4.5}$$

then one can derive that $\boldsymbol{s_1}$ is the direction of maximal compression, while μ_1 is the value of the compression measure (4.5) in the direction $\boldsymbol{s_1}$. Having this in mind, the four artificial stress types are

$$\begin{aligned} \sigma_{a_1} &= \kappa_{\boldsymbol{s_1}} \nabla \boldsymbol{u}, & \sigma_{a_2} &= \kappa_{\boldsymbol{s_1}} \nabla^s \boldsymbol{u}, \\ \sigma_{a_3} &= \kappa_{\boldsymbol{s_1}} \mu_1 \boldsymbol{s_1} \otimes \boldsymbol{s_1}, & \sigma_{a_4} &= \sum_{m=1}^{d} \kappa_{\boldsymbol{s_m}} \mu_m \boldsymbol{s_m} \otimes \boldsymbol{s_m}, \end{aligned} \tag{4.6}$$

where $\kappa_{\boldsymbol{s}}$ is a direction dependent viscosity coefficient which is explained later. The tensors σ_{a_1} and σ_{a_2} are standard, while σ_{a_3} is a purely one dimensional viscous stress that only takes into account the direction of maximal compression. The tensor σ_{a_4} can be viewed as a generalization of σ_{a_2} and σ_{a_3}, since it can handle interactions of multiple shocks by taking into account multiple directions of strong compression. Further details about the above artificial tensor stress types and their application to test problems can be found in [20].

4.1.2 Semi-Discrete Form

BLAST uses the continuous finite element space Q_k^d for the kinematic variables (position, velocity), and a discontinuous FE space Q_k^* for the specific internal energy:

$$Q_k = \{v \in C^0(\Omega_0); v|_{K_0} \circ \Phi_0 \in \mathbb{Q}_k, \forall K_0 \in \mathcal{K}_h\},$$
$$Q_k^* = \{v \in L^2(\Omega_0); v|_{K_0} \circ \Phi_0 \in \mathbb{Q}_k, \forall K_0 \in \mathcal{K}_h\}.$$

The optimal combination for the numerical experiments in BLAST is to take the discontinuous FE space of order one less than the kinematic space, namely pairs of the type Q_k-Q_{k-1}^*. If $\{\boldsymbol{\phi_j}\}_{j=1}^N$ and $\{w_j\}_{i=1}^{N^*}$ are the basis functions of Q_k^d and Q_{k-1}^*, respectively, then the semi-discrete form that corresponds to the system (4.1)-(4.4) is to find $\rho \in Q_{k-1}^*, \boldsymbol{u} \in Q_k^d, e \in Q_{k-1}^*, \boldsymbol{x} \in Q_k^d$ so that the following equations hold for each $j = 1...N$ and $i = 1...N^*$:

$$\rho(\boldsymbol{x}, t) \left| J J_0^{-1}(\boldsymbol{x_0}, t) \right| = \rho_0(\boldsymbol{x_0}), \tag{4.7}$$

$$\int_{\Omega(t)} \rho(\boldsymbol{x},t)\frac{d\boldsymbol{u}(\boldsymbol{x},t)}{dt}\cdot\boldsymbol{\phi_j}(\boldsymbol{x},t)\ d\boldsymbol{x} =$$
$$\int_{\Omega(t)} p(\boldsymbol{x},t)\nabla\cdot\boldsymbol{\phi_j}(\boldsymbol{x},t)\ d\boldsymbol{x} - \int_{\Omega(t)} \sigma_a : \nabla\boldsymbol{\phi_j}(\boldsymbol{x},t)\ d\boldsymbol{x} \tag{4.8}$$
$$-\int_{\partial\Omega(t)} p(\boldsymbol{x},t)\boldsymbol{\phi_j}(\boldsymbol{x},t)\cdot\boldsymbol{n}\ d\boldsymbol{x} + \int_{\partial\Omega(t)} \boldsymbol{n}\cdot\sigma_a\cdot\boldsymbol{\phi_j}(\boldsymbol{x},t)\ d\boldsymbol{x}$$

$$\int_{\Omega(t)} \rho(\boldsymbol{x},t)\frac{de(\boldsymbol{x},t)}{dt}w_i(\boldsymbol{x},t)\ d\boldsymbol{x} =$$
$$\int_{\Omega(t)} -p(\boldsymbol{x},t)\nabla\cdot\boldsymbol{u}(\boldsymbol{x},t)w_i(\boldsymbol{x},t)\ d\boldsymbol{x} + \int_{\Omega(t)} \sigma_a : \nabla\boldsymbol{u}(\boldsymbol{x},t)w_i(\boldsymbol{x},t)\ d\boldsymbol{x} \tag{4.9}$$

$$\frac{d}{dt}\boldsymbol{x}(\boldsymbol{x_0},t) = \boldsymbol{u}(\boldsymbol{x},t), \tag{4.10}$$

Notice that the density ρ doesn't have to be a finite element function since it's evolved pointwise, and all resulting mass matrices are constant in time (see Subsection 3.5.2). Boundary integrals were discussed in Subsection 3.3.2, the same comments are valid for the boundary integrals in equation (4.8). Equation (4.9) is solved locally on each cell since the FE space for e is discontinuous and the equation doesn't contain numerical fluxes across the cell boundaries. The semi-discrete form (4.7)-(4.10) guarantees exact mass, momentum and total energy conservation, see [20]. The corresponding semi-discrete form used in the Entropy Viscosity method is presented in Subsection 3.3.2.

4.1.3 Viscosity Coefficients

The viscosity coefficient $\kappa_{\boldsymbol{s}}$ in the definition of the artificial stress tensor σ_a (4.6) has the form

$$\kappa_{\boldsymbol{s}}(\boldsymbol{x}) := \rho\left(q_2 h_{\boldsymbol{s}}^2|\Delta_{\boldsymbol{s}}\boldsymbol{u}| + q_1\zeta_0\zeta_1 h_{\boldsymbol{s}}c_s\right), \tag{4.11}$$

where q_1 and q_2 are linear and quadratic scaling coefficients, respectively; c_s is the speed of sound at $\boldsymbol{x}$; $h_{\boldsymbol{s}}$ is length scale that measures the perturbation of the initial

mesh in the direction $\boldsymbol{s}$, namely

$$h_{\boldsymbol{s}}(\boldsymbol{x}(\boldsymbol{x_0},t)) = h_0(\boldsymbol{x_0})\frac{|JJ_0^{-1}(\boldsymbol{x_0})\boldsymbol{s}(\boldsymbol{x})|}{|\boldsymbol{s}(\boldsymbol{x})|}; \tag{4.12}$$

the quantity ζ_0 is used to suppress the linear term at points where vorticity dominates the flow

$$\zeta_0 := \frac{|\nabla \cdot \boldsymbol{u}|}{||\nabla \boldsymbol{u}||}$$

and ζ_1 is a compression switch that makes the linear term inactive at expansion points:

$$\zeta_1 := \begin{cases} 1 & \text{if } \Delta_{\boldsymbol{s}}\boldsymbol{u} < 0, \\ 0 & \text{if } \Delta_{\boldsymbol{s}}\boldsymbol{u} \geq 0, \end{cases}$$

The coefficient (4.11) is very similar to the Option 2 first order viscosity coefficient used in the Entropy Viscosity method (see Subsection 3.3.5), comparisons with other viscosity coefficients can be found in Subsection 3.5.

4.1.4 Application of the Entropy Production Based Coefficient

We can apply the entropy production based coefficient (2.50) in BLAST. One way to do this is to alter (4.11) in the following way:

$$\kappa_{\boldsymbol{s}}(\boldsymbol{x}) := \rho \min\left(q_2 h_{\boldsymbol{s}}^2 |\Delta_{\boldsymbol{s}}\boldsymbol{u}| + q_1\zeta_0\zeta_1 h_{\boldsymbol{s}} c_s, \nu^{entr}\right), \tag{4.13}$$

where ν^{entr} is defined in Subsection 2.3.5, equation (3.26). By doing this we make the artificial viscosity go to zero in smooth regions, which is not a property of the coefficient (4.11). We validate this statement by running the 2D Taylor-Green vortex problem (see Subsection 3.4.1) with active viscosity that is scaled by the coefficient (4.13). The simulations use type 1 artificial tensor σ_{a_1} and RK4 time integrator. The

results are given in Table 4.1. We observe high order convergence rates even with active viscosity terms.

	Q_2-Q_1^*		Q_3-Q_2^*		Q_4-Q_3^*	
h_0	L^1 error	rate	L^1 error	rate	L^1 error	rate
1/4	6.28E-2		1.65E-2		5.90E-3	
1/8	9.10E-3	2.78	1.65E-3	3.32	4.60E-4	3.68
1/16	2.46E-3	1.88	1.96E-4	3.07	1.71E-4	4.74
1/32	6.52E-4	1.91	2.32E-5	3.04	1.69E-6	3.33

Table 4.1: L^1 velocity errors and convergence rates resulting from using the entropy based viscosity coefficient for the 2D Taylor-Green problem.

Next we verify that the coefficient (4.13) is appropriate for simulations with shocks. We run the 2D Sedov explosion (see Subsection 3.4.3) on a 20×20 mesh with type 1 artificial tensor σ_{a_1} and the pair Q_2-Q_1^* FE spaces. Final density field and viscosity coefficient are presented on Figure 4.1. We observe correct mesh motion and precise shock capturing.

4.2 Mesh Relaxation

The goal of mesh relaxation is to increase the CFL time step and to avoid mesh tangling by changing the current computational mesh $\tilde{\Omega}$ to a new mesh Ω. In ALE methods, the mesh relaxation step is usually performed after some fixed number of Lagrangian steps.

Following the notation from Subsection 3.3.1, let $\boldsymbol{x_1}^0 ... \boldsymbol{x_N}^0$ be the positions of our finite element nodes that correspond to the position function $\tilde{\boldsymbol{x}}$ of $\tilde{\Omega}$. The relaxation procedure we propose replaces each interior node's position with a weighted average

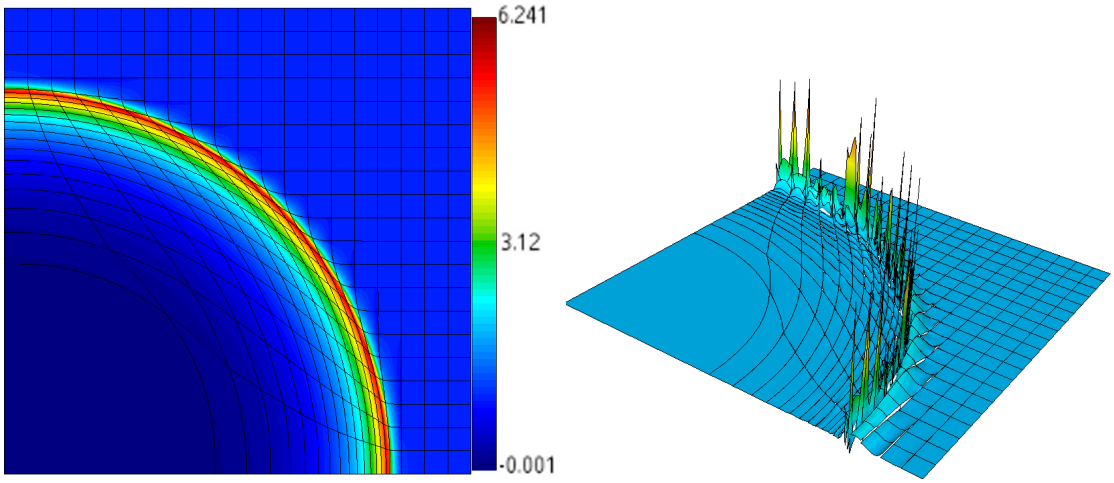

Figure 4.1: Final density field (on the left side), and entropy viscosity coefficient (on the right side) for the 2D Sedov explosion problem.

of its neighbors by the formula

$$\boldsymbol{x_i}^{n+1} = \sum_{j \in N_i} c_{ij} \boldsymbol{x_j}^n, \tag{4.14}$$

where N_i is the set of neighbors of node i, c_{ij} are some weights so that $c_{ij} \in (0,1), \sum_{j \in N_i} c_{ij} = 1$. The formula (4.14) can be generalized by introducing the $N \times N$ mesh Laplacian matrix L so that

$$L_{ii} = 1, \qquad L_{ij} = -c_{ij}, \qquad \sum_{j=1}^{N} L_{ij} = 0.$$

Then solving the system

$$L\boldsymbol{x} = 0$$

with initial guess $\boldsymbol{x}^0$ and eliminated boundary nodes (keeping their positions constant) is equivalent to taking the limit of the iteration (4.14) as $N \to \infty$. A precon-

ditioner can be applied to obtain the general method

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + P(0 - L\boldsymbol{x}^n), \tag{4.15}$$

where P is a preconditioner for L. The formula (4.15) is general with respect to the definition of L. We focus on two choices of L:

1. L_1 connects neighboring nodes with equal weights. Neighboring nodes are determined by the finite element sparsity, namely two nodes are neighbors if they share a cell. Applying L_1 to high-order FE spaces is not a good idea due to the differences between nodes associated with elements, faces, edges and vertices.

2. L_2 connects neighboring nodes by assembling the stiffness matrix on the reference cell.

Remark For dimensions $d > 1$, in (4.15) the mesh Laplacian L is block-diagonal, having d equal $N \times N$ blocks. The position vector $\boldsymbol{x}$ is ordered by dimension components first.

4.2.1 Numerical Tests

On Figure 4.2 we show initial and relaxed Q_2 mesh in 2D after 5 iterations of (4.15) with mesh Laplacian L_1 and no preconditioner. On Figure 4.3 we show cross-sections of initial and relaxed Q_3 mesh in 3D after 3 iterations of (4.15) with mesh Laplacian L_2 and no preconditioner.

The mesh relaxation algorithms need to be parallel in order to be compatible with the BLAST framework. Examples of mesh distribution to different MPI tasks can be seen on Figure 4.4.

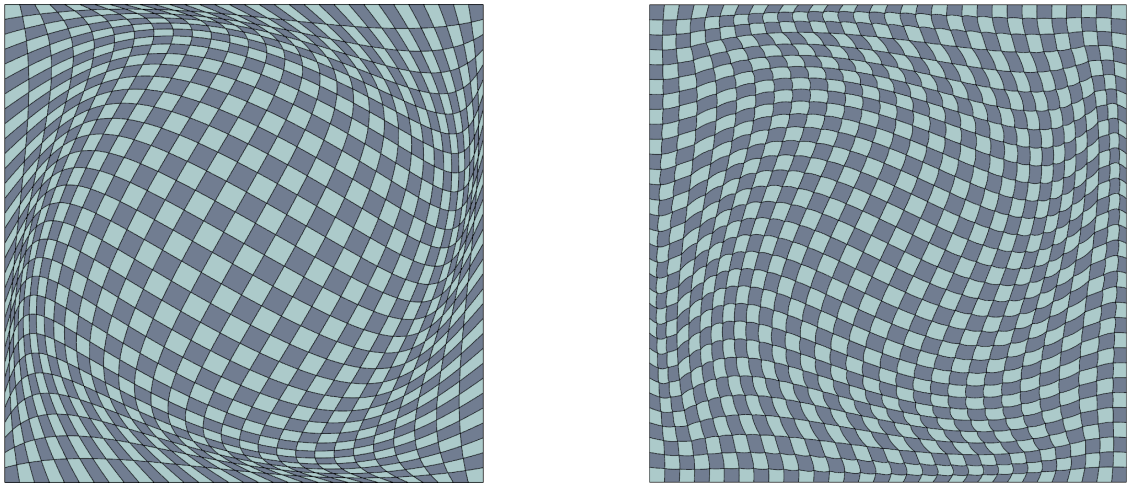

Figure 4.2: Perturbed 2D mesh(on the left side), and the corresponding relaxed mesh (on the right side) computed by 5 steps of the L_1 smoother.

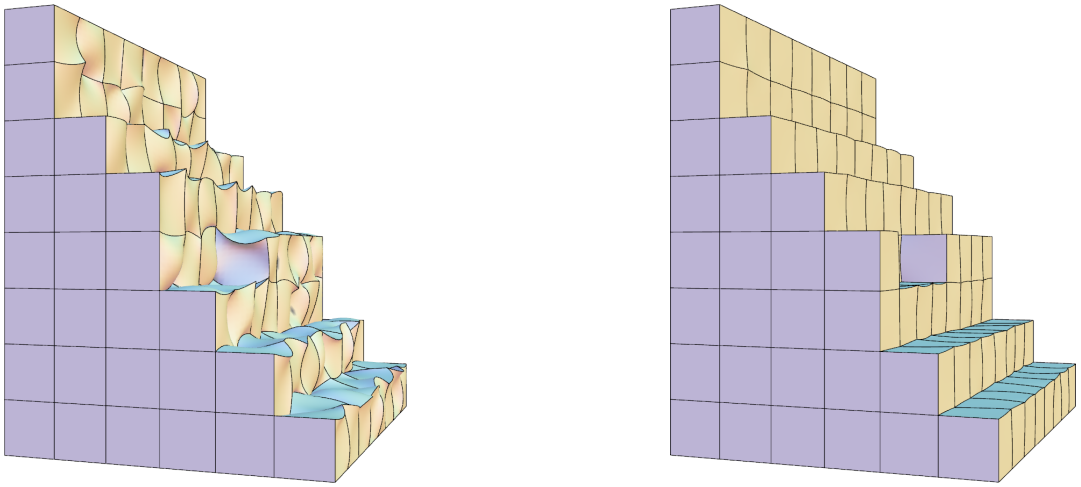

Figure 4.3: Cross-sections of perturbed (on the left side), and relaxed 3D meshes (on the right side) computed by 3 steps of the L_2 smoother.

4.3 Solution Remap

Let $\tilde{\Omega}$ be our starting mesh, and let Ω be the new mesh obtained by relaxing $\tilde{\Omega}$. Having a scalar finite element function $\tilde{v}$ defined on $\tilde{\Omega}$, the goal of the remap step is to to generate a new FE function v on Ω. The new function v must be in the same

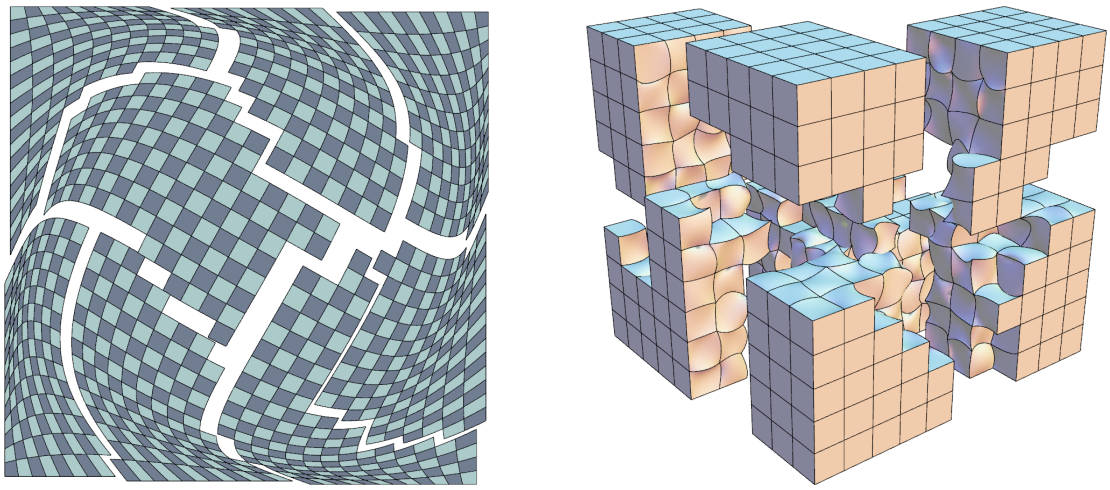

Figure 4.4: Mesh distribution to different MPI tasks for 2D (on the left side) and 3D (on the right side) high order meshes.

FE space as $\tilde{v}$. The remap procedure must satisfy the following properties:

1. Accuracy, i.e. polynomials up to some degree must be remapped with no error.
2. Conservation, namely $\int_\Omega v = \int_{\tilde{\Omega}} \tilde{v}$.
3. Monotonicity, i.e. the remap procedure must not generate new extrema.

A straightforward way to define a remap would be a simple interpolation of $\tilde{v}$ on the FE nodes $\boldsymbol{x_1}...\boldsymbol{x_N}$ of Ω, namely

$$v(\boldsymbol{x_j}) = \tilde{v}(\boldsymbol{x_j}), \quad \forall j = 1...N,$$

but this approach does not satisfy any of the required properties. A better approach is to consider a projection operation, namely

$$\begin{aligned} &\int_\Omega v\psi_j \, d\boldsymbol{x} = \int_\Omega \tilde{v}\psi_j \, d\boldsymbol{x}, \ \forall j = 1...N \ \Rightarrow \\ &\int_\Omega \sum_{i=1}^{n} v_i\psi_i\psi_j \, d\boldsymbol{x} = \int_\Omega \sum_{i=1}^{n} \tilde{v}_i\tilde{\psi}_i\psi_j \, d\boldsymbol{x}, \ \forall j = 1...N, \end{aligned}$$

where ψ and $\tilde{\psi}$ are the basis functions on Ω and $\tilde{\Omega}$, respectively. In order to compute the above integrals, however, one must consider the overlaps between the corresponding old and new cells in Ω and $\tilde{\Omega}$. Such computations become difficult for the case of high-order, curved meshes.

4.3.1 Advection Remap

In this subsection we consider advection based remap. We introduce "pseudo-time" $\tau \in [0, 1]$, and we slightly change notation so that

$$\Omega = \Omega(\tau), \qquad \boldsymbol{x} = \boldsymbol{x}(\tilde{\boldsymbol{x}}, \tau)$$

represent intermediate domains and positions, respectively, where $\Omega(0) = \tilde{\Omega}, \boldsymbol{x}(\tilde{\boldsymbol{x}}, 0) = \tilde{\boldsymbol{x}}$, and $\Omega(1), \boldsymbol{x}(\tilde{\boldsymbol{x}}, 1)$ is the final configuration after the remesh step. We choose to define intermediate positions (i.e. the mesh motion) by

$$\boldsymbol{x}(\tilde{\boldsymbol{x}}, \tau) := \tilde{\boldsymbol{x}} + \tau(\boldsymbol{x}(\tilde{\boldsymbol{x}}, 1) - \tilde{\boldsymbol{x}}). \tag{4.16}$$

Then, in pseudo-time, we have the mesh velocity

$$\bar{\boldsymbol{u}}(\boldsymbol{x}) = \frac{\partial \boldsymbol{x}}{\partial \tau} = \boldsymbol{x}(\tilde{\boldsymbol{x}}, 1) - \tilde{\boldsymbol{x}}. \tag{4.17}$$

Notice that the definition of mesh motion (4.16) infers that $\bar{\boldsymbol{u}}$ is independent of τ. We think of our unknown υ as

$$\upsilon = \upsilon(\boldsymbol{x}(\tilde{\boldsymbol{x}}, \tau), \tau), \qquad \upsilon(\boldsymbol{x}(\tilde{\boldsymbol{x}}, 0), 0) = \tilde{\upsilon}(\tilde{\boldsymbol{x}}),$$

and its material derivative in pseudo-time is

$$\frac{d\upsilon}{d\tau} = \frac{\partial \upsilon}{\partial \tau} + \bar{\boldsymbol{u}} \cdot \nabla \upsilon.$$

Ideally, the remap procedure must provide $\frac{\partial \upsilon}{\partial \tau} = 0$, meaning that the unknown function stays constant while the mesh transitions from $\tilde{\Omega}$ to $\Omega(1)$. Therefore, our remap problem reduces to solving the advection equation

$$\frac{d\upsilon}{d\tau} = \bar{\boldsymbol{u}} \cdot \nabla \upsilon, \qquad \upsilon(\boldsymbol{x}(\tilde{\boldsymbol{x}}, 0), 0) = \tilde{\upsilon}(\tilde{\boldsymbol{x}}), \tag{4.18}$$

with mesh motion defined in (4.16), and corresponding mesh velocity $\bar{\boldsymbol{u}}$ defined in (4.17).

4.3.2 Multi-Field Remap

Now we go back to the BLAST framework and state the corresponding advection remap equations for density, momentum and internal energy. Each velocity component is remapped independently as a continuous field. For simplicity we focus on the remap of one component denoted by u. The resulting problems are

$$\frac{d\rho}{d\tau} = \bar{\boldsymbol{u}} \cdot \nabla \rho, \qquad \frac{d}{dt}(\rho u) = \bar{\boldsymbol{u}} \cdot \nabla (\rho u), \qquad \frac{d}{dt}(\rho e) = \bar{\boldsymbol{u}} \cdot \nabla (\rho e). \tag{4.19}$$

This is a system of equations with respect to ρ, u, e, where the initial conditions for those variables are given by the solution state before the remesh procedure. Since the FE space Q_k^* that is used for ρ and e is discontinuous, we propose discontinuous Galerkin weak forms of the density and internal energy equations:

- Density - we seek $\rho \in Q_k^*$, so that for every $j = 1...N^*$ we have

$$\begin{aligned} \int_{\Omega(\tau)} \frac{d\rho}{dt} w_j \; d\boldsymbol{x} &= \sum_{K \in \Omega(\tau)} \int_K (\bar{\boldsymbol{u}} \cdot \nabla \rho) w_j \; d\boldsymbol{x} \\ - \sum_{f \in \mathcal{F}_i(\tau)} \int_f (\bar{\boldsymbol{u}} \cdot \boldsymbol{n_f}) [\![\rho]\!] \{w_j\} \; d\boldsymbol{x} &- \frac{1}{2} \sum_{f \in \mathcal{F}_i(\tau)} \int_f |\bar{\boldsymbol{u}} \cdot \boldsymbol{n_f}| [\![\rho]\!] [\![w_j]\!] \; d\boldsymbol{x}. \end{aligned} \tag{4.20}$$

- Internal energy - we seek $e \in Q_k^*$, so that for every $j = 1...N^*$ we have

$$\begin{aligned} \int_{\Omega(\tau)} \rho \frac{de}{dt} w_j \; d\boldsymbol{x} &= \sum_{K \in \Omega(\tau)} \int_K \rho (\bar{\boldsymbol{u}} \cdot \nabla e) w_j \; d\boldsymbol{x} \\ - \sum_{f \in \mathcal{F}_i(\tau)} \int_f \bar{\rho} (\bar{\boldsymbol{u}} \cdot \boldsymbol{n_f}) [\![e]\!] \{w_j\} \; d\boldsymbol{x} &- \frac{1}{2} \sum_{f \in \mathcal{F}_i(\tau)} \int_f \bar{\rho} |\bar{\boldsymbol{u}} \cdot \boldsymbol{n_f}| [\![e]\!] [\![w_j]\!] \; d\boldsymbol{x}. \end{aligned} \tag{4.21}$$

Here $\mathcal{F}_i(\tau)$ is the set of internal faces, $[\![\psi]\!] = \psi_i - \psi_e$ is face jump, and $\{\psi\} = \frac{1}{2}(\psi_i + \psi_e)$ is face average, and $\bar{\rho} = \{\rho\} - \frac{1}{2}\text{sgn}(\bar{\boldsymbol{u}} \cdot \boldsymbol{n_f})[\![\rho]\!]$ is the upwind value. Since the FE space Q_k that is used for u is continuous, we propose the following continuous weak form for the velocity equation:

- Velocity components - we seek $u \in Q_k$, so that for every $j = 1...N$ we have

$$\int_{\Omega(\tau)} \rho \frac{du}{dt} \varphi_j \; d\boldsymbol{x} = \int_{\Omega(\tau)} \rho (\bar{\boldsymbol{u}} \cdot \nabla u) \varphi_j \; d\boldsymbol{x}. \tag{4.22}$$

In order to handle discontinuous functions in a monotonic way, we can introduce some diffusion procedures around the jumps or flux limiting. These approaches are not yet fully established and they are not part of this document. The semi-discrete forms (4.20), (4.21), (4.22) are discretized in pseudo-time by choosing a fixed number of time steps (we have a constant mesh velocity), and using a generic Runge-Kutta time integrator.

4.3.3 Numerical Tests

We show some preliminary results for functions without jumps. On Figure 4.5 we use the weak form (4.20) to remap the density field $\rho(\tilde{\boldsymbol{x}}) = \sin(\pi x)\sin(\pi y)$. On Figure 4.6 we use the weak form (4.22) to remap the field $\boldsymbol{u}(\tilde{\boldsymbol{x}}) = (\frac{\pi}{2} + \arctan(20(x - 0.5)), \frac{\pi}{2} + \arctan(20(y - 0.5)))$. These simulations use Q_2 FE spaces for position and velocity, and Q_2^* FE space for density on a 32×32 mesh.

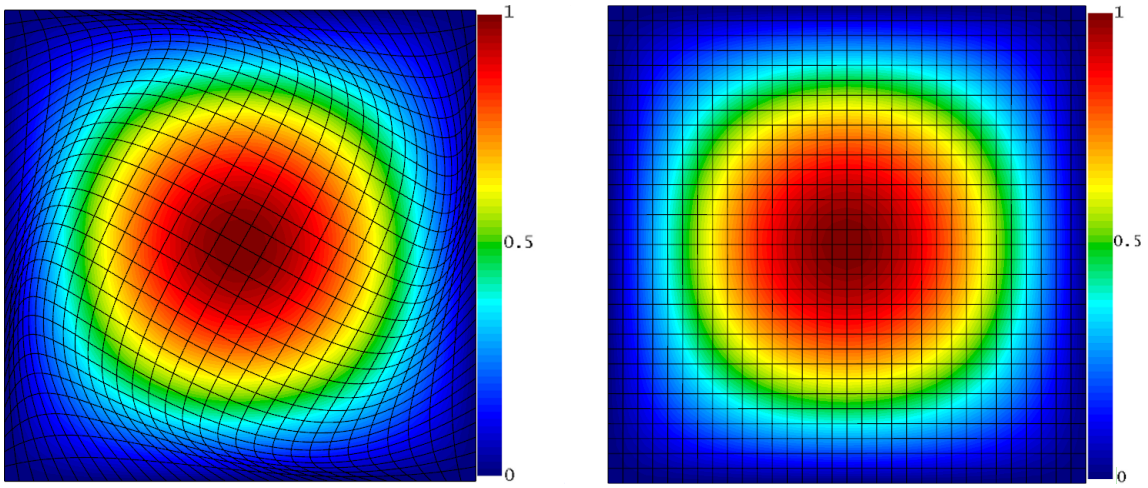

Figure 4.5: Original mesh and density (on the left side), and the corresponding relaxed mesh and remapped density (on the right side).

In Table 4.2 we show errors and convergence rates for mass, momentum, kinetic and internal energy for the above examples. The remapped specific internal energy field is $e(\tilde{\boldsymbol{x}}) = \sin(\pi x)\sin(\pi y)$. We use a standard RK4 time integrator that performs the denoted number of steps in pseudo-time. The convergence rates resolve the time integrator's accuracy, implying that the semi-discrete forms (4.20), (4.21), (4.22) provide exact conservation of all quantities.

The remap algorithms need to be parallel in order to be compatible with the

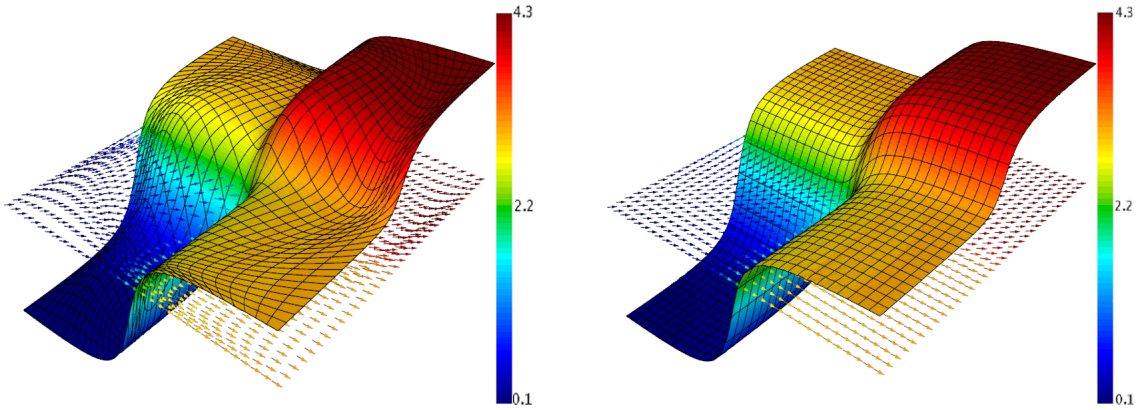

Figure 4.6: Original mesh and velocity field (on the left side), and the corresponding relaxed mesh and remapped velocity field (on the right side).

h, # steps	mass error	rate	IE error	rate	KE error	rate	MOM(x, y) error	rate
1/8, 10	1.3E-8		5.7E-7		1.7E-6		(2.0E-8, 2.0E-8)	
1/16, 20	8.9E-10	3.86	4.4E-8	3.69	1.0E-7	4.08	(1.4E-9, 1.4E-9)	3.83
1/32, 40	5.6E-11	3.99	2.0E-9	4.45	4.3E-9	4.53	(8.9E-11, 8.8E-11)	3.97
1/64, 80	3.2E-12	4.12	9.8E-11	4.35	1.6E-10	4.74	(4.9E-12, 5.3E-12)	4.18

Table 4.2: Mass, internal energy (IE), kinetic energy (KE) and momentum (MOM) errors for a remap of jump-free fields.

BLAST framework. Examples of mesh distribution to different MPI tasks can be seen on Figure 4.7.

4.4 Multi-Material Simulations

The goal of this subsection is to propose methods for computing pressure values in cells containing more than one material. Such cells result from the ALE remesh and remap steps.

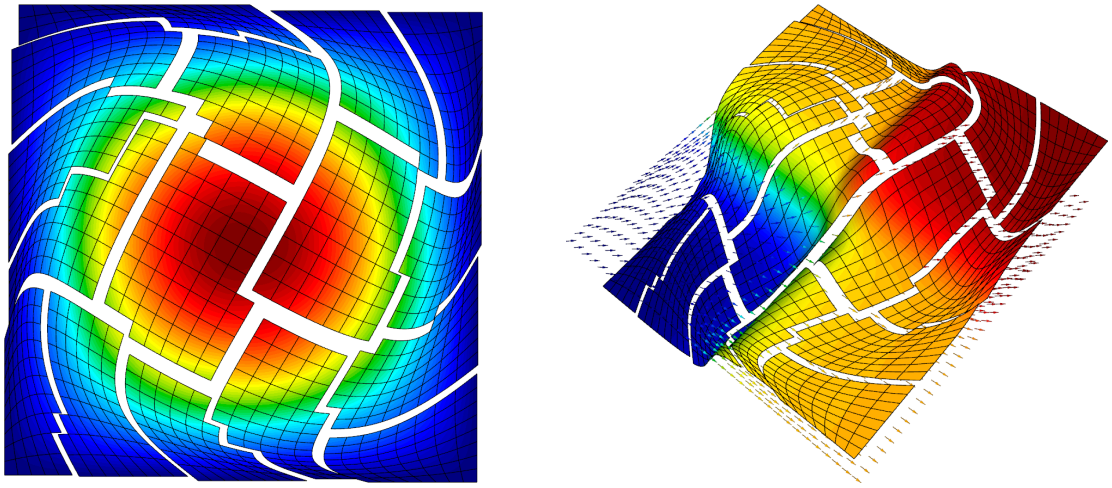

Figure 4.7: Mesh distribution to different MPI tasks for remap of jump-free fields.

4.4.1 Material Indicator Functions

We use the initial material configuration to define "material indicator functions", namely

$$\eta_r(\boldsymbol{x},0) = \begin{cases} 1 & \text{if material } r \text{ is present at } \boldsymbol{x}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.23}$$

We choose these functions to be in our thermodynamical (discontinuous) FE space Q_k^*. If we have a material interface that is initially inside a cell, we diffuse those functions by

$$\int_{\Omega_0} \eta_r^*(\boldsymbol{x},0)\varphi(\boldsymbol{x})\ d\boldsymbol{x} + \varepsilon h^2 \int_{\Omega_0} \nabla\eta_r^*(\boldsymbol{x},0)\cdot\nabla\varphi(\boldsymbol{x})\ d\boldsymbol{x} = \int_{\Omega_0} \eta_r(\boldsymbol{x},0)\varphi(\boldsymbol{x})\ d\boldsymbol{x}, \tag{4.24}$$

where φ and η_r^* are basis functions and solution in the continuous space Q_k. We project η_r^* back to Q_k^* to get the final diffused indicator function. The constant $\varepsilon > 0$ controls the amount of diffusion, note that the same constant must be used for all materials. Once we have the initial indicator functions, we evolve them in time

following the mesh motion, namely

$$\frac{d}{dt}\eta_r(\boldsymbol{x},t)=0. \tag{4.25}$$

Notice that (4.23), (4.24), (4.25) imply that at all times we have

$$\sum_r \eta_r(\boldsymbol{x},t)=1,$$

hence we can also think of these functions as material fractions.

The presence of a material interface inside a cell at initial time also requires the diffusion of the density and specific internal energy fields. However this introduces new difficulties, e.g. consider a jump in ρ and e, so that the pressure stays constant across the interface. In this case any smoothing of the ρ and e values in the interface region causes non-constant pressure, because the connection $p=p(\rho,e)$ is non-linear. We show an example by considering the 2D Triple Point Interaction problem, where the initial conditions are given on Figure 4.8, and results for material interfaces aligned with cell boundaries are presented in [20]. We use a mesh with 57×25 cells, so that all material interfaces are not aligned with cell boundaries. Diffusing the initial ρ and e values inside the mixed cells results in wrong initial pressure values, leading to a spurious wave in the solution at later times. Example of such situation is presented on Figure 4.8.

4.4.2 Material-Specific Extension Functions

We tackle the above problem by introducing density and specific internal energy "extension functions". That is, we break our original ρ and e into material-specific densities ρ_r and energies e_r, $r=1...$#materials. The idea behind these functions is to use them in order to compute correct pressure values. At initial time, they are

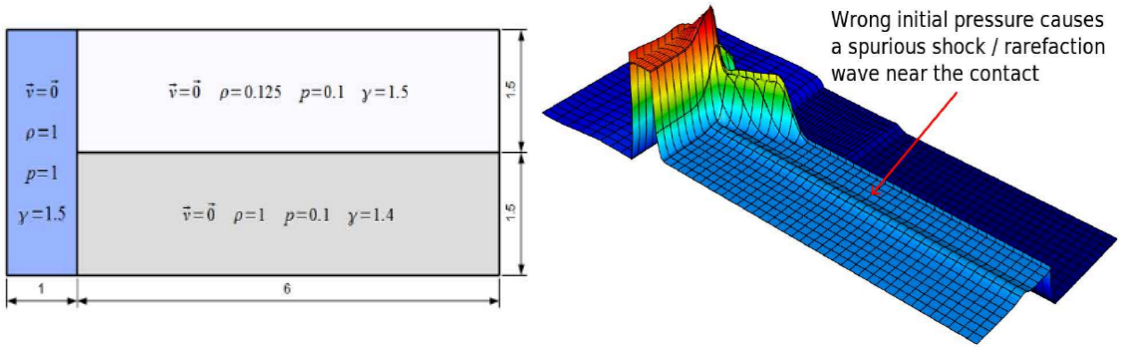

Figure 4.8: Initial conditions (on the left side), and spurious wave resulting from wrong initial pressure values (on the right side) for the Triple Point Interaction problem.

initialized by

$$\rho_r(\boldsymbol{x},0) = \begin{cases} \rho^*(\boldsymbol{x},0) & \eta_r(\boldsymbol{x},0) > 0, \\ 0 & \eta_r(\boldsymbol{x},0) = 0, \end{cases} \qquad e_r(\boldsymbol{x},0) = \begin{cases} e^*(\boldsymbol{x},0) & \eta_r(\boldsymbol{x},0) > 0, \\ 0 & \eta_r(\boldsymbol{x},0) = 0, \end{cases} \tag{4.26}$$

where ρ^*, e^* are the extended material values within the cell, e.g. if we have a two-material 1D cell $[0,1]$ with a material interface at $x = \frac{1}{2}$ and densities given by

$$\rho(x,0) = \begin{cases} \rho_L & x \leq \frac{1}{2}, \\ \rho_R & x > \frac{1}{2}, \end{cases} \qquad \Rightarrow \qquad \rho_1 = \rho_L, \rho_2 = \rho_R, \forall x \in [0,1].$$

Notice that the extension functions are not influenced by diffusion procedures and they are independent of each other. We use the equations of state for each material, together with the material-specific densities and energies, to compute material-specific pressures p_r. Then we define final pressure at a point in one of two ways:

- By material mixing:

$$p(\boldsymbol{x},t) = \sum_r \eta_r(\boldsymbol{x},t) p_r(\rho_r(\boldsymbol{x},t), e_r(\boldsymbol{x},t)). \tag{4.27}$$

- By dominant material ("exact" interface reconstruction):

$$p(\boldsymbol{x},t) = p_r(\rho_r(\boldsymbol{x},t), e_r(\boldsymbol{x},t)), \qquad r = \arg\max_i(\eta_i(\boldsymbol{x},t)). \tag{4.28}$$

Global density and global internal energy are defined as

$$\rho(\boldsymbol{x},t) = \sum_r \eta_r(\boldsymbol{x},t)\rho_r(\boldsymbol{x},t), \quad \rho(\boldsymbol{x},t)e(\boldsymbol{x},t) = \sum_r \eta_r(\boldsymbol{x},t)\rho_r(\boldsymbol{x},t)e_r(\boldsymbol{x},t). \tag{4.29}$$

4.4.3 Time Evolution

We evolve the velocity $\boldsymbol{u}$ and the material specific densities and energies ρ_k, e_l in a way that preserves mass and the total energy on semi-discrete level. The material-specific densities ρ_k are evolved in time by pointwise mass conservation, namely

$$\rho_r(\boldsymbol{x},t) \left| J J_0^{-1}(\boldsymbol{x_0},t) \right| = \rho_r(\boldsymbol{x},0).$$

The material velocity $\boldsymbol{u}$ is evolved by (4.8), where the pressure is computed by (4.28) or (4.27), and the viscosity coefficient κ from (4.11) is defined as

$$\kappa_{\boldsymbol{s}}(\boldsymbol{x}) := \sum_r \eta_r \rho_r \left(q_2 h_{\boldsymbol{s}}^2 |\Delta_{\boldsymbol{s}} \boldsymbol{u}| + q_1 \zeta_0 \zeta_1 h_{\boldsymbol{s}} (c_s)_r \right), \tag{4.30}$$

where $(c_s)_r$ is material-specific sound speed. In order to evolve each e_k in time, we replace the weak form (4.9) by a weak that takes into account the extension functions.

That is, we seek $e_r \in Q^*_{k-1}$ so that for every $i = 1...N^*$ we have

$$\begin{aligned}&\int_{\Omega(t)} \eta_r(\boldsymbol{x},t)\rho_r(\boldsymbol{x},t)\frac{de_r(\boldsymbol{x},t)}{dt}w_i(\boldsymbol{x},t)\ d\boldsymbol{x} = \\ -&\int_{\Omega(t)} \theta_r p_r(\boldsymbol{x},t)\nabla\cdot\boldsymbol{u}(\boldsymbol{x},t)w_i(\boldsymbol{x},t)\ d\boldsymbol{x} + \int_{\Omega(t)} (\sigma_a)_r : \nabla\boldsymbol{u}(\boldsymbol{x},t)w_i(\boldsymbol{x},t)\ d\boldsymbol{x},\end{aligned} \tag{4.31}$$

where θ_r is given by

$$\theta_r = \begin{cases} \eta_r(\boldsymbol{x},t) & \text{in the case of mixed pressure (4.27),} \\ \delta_{rl}, l = \arg\max_j(\eta_j(\boldsymbol{x},t)) & \text{in the case of dominant pressure (4.28),} \end{cases}$$

and $(\sigma_a)_r$ is a material-specific stress tensor obtained by plugging the viscous coefficient

$$(\kappa_{\boldsymbol{s}})_r(\boldsymbol{x}) := \eta_r\rho_r\left(q_2 h_{\boldsymbol{s}}^2|\Delta_{\boldsymbol{s}}\boldsymbol{u}| + q_1\zeta_0\zeta_1 h_{\boldsymbol{s}}(c_s)_r\right). \tag{4.32}$$

in the chosen case of (4.6).

4.4.4 Numerical Tests

Here we show results for the Triple Point Interaction problem discussed earlier in this subsection (see Figure 4.8). We define three material indicator functions and corresponding densities and energy extensions. We use a mesh with 57×25 cells, Q_3 FE spaces for $\boldsymbol{u}$ and $\boldsymbol{x}$, Q^*_2 FE spaces for e_r, η_r, material mixing pressure (4.27), and the final time is $T = 3.0$. Final density profiles are shown on Figure 4.9. Example of initial and final material indicator functions is shown Figure 4.10. Example of initial and final density extension functions is shown on Figure 4.11.

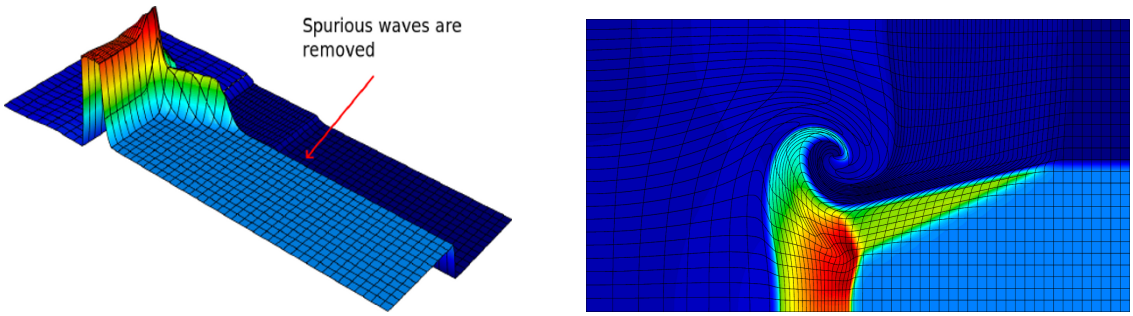

Figure 4.9: Density profiles at time 1.5 (on the left side), and at time 3.0 (on the right side) for the Triple Point Interaction problem.

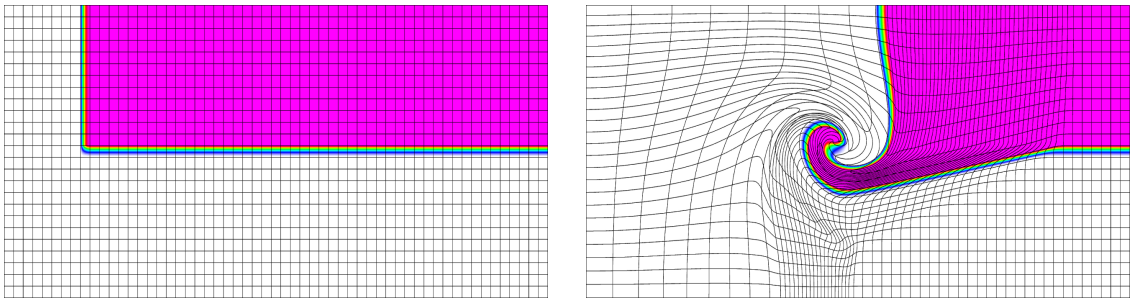

Figure 4.10: Example of a material indicator function at initial time (on the left side), and at time 3.0 (on the right side) for the Triple Point Interaction problem.

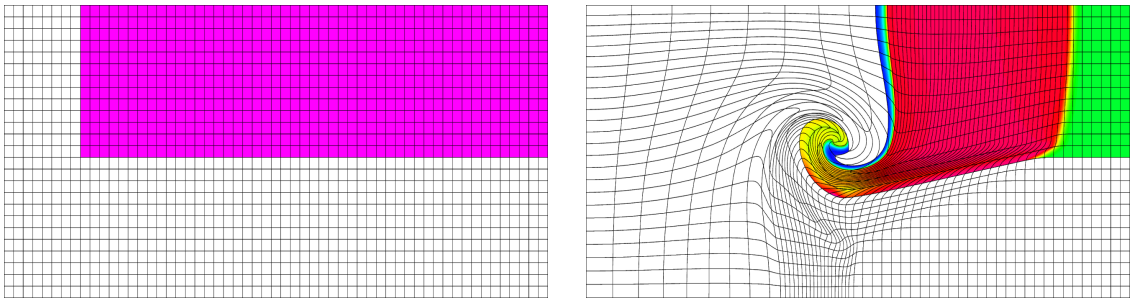

Figure 4.11: Example of a density extension function at initial time (on the left side), and at time 3.0 (on the right side) for the Triple Point Interaction problem.

5. CONCLUSION

5.1 Mean Field Games

We have presented a parallel fixed point iteration algorithm that combines a second order scheme for the forward Hamilton-Jacobi equation (1.1), and a second order scheme for the backward convection-diffusion equation (1.2). The second order accuracy of the method is confirmed numerically, and our numerical results agree with the already existing data in the field. The schemes' simplicity and the method's parallel ability allow us to use highly refined meshes. We have eliminated the memory problems arising from the combination of implicit time stepping and forward-backward coupling of the equations.

This work can be extended by introducing 2D algorithms that use the same central schemes approach. This will result in more computations inside a single time step, hence it will exploit better the parallel abilities of our numerical method. We expect to achieve similar run times as in 1D, since the 2D methods will do the same number of time steps while performing more computations per cell.

5.2 Entropy Viscosity Method

We have presented a high-order curvilinear finite element method which combines non-oscillatory behavior in contacts, sharp shock detection, compatibility with generalized entropy inequalities (1.19) and minimum principle on the specific entropy (1.35) on continuous level, and compatibility with the general tensor viscosity requirements from [36]. The method is general with respect to the polynomial degree of the used finite element spaces and the order of the time integration methods. The reported results confirm that out coefficients converge to zero for smooth solutions and we achieve convergence to exact solutions with rate close to the optimal rate of

one for standard shock wave problems. We observe proper radial symmetry preservation for uniform and non-uniform initial meshes, robust capturing of the problems' geometric features, and ability to represent details of the flow within a single zone. All these features come for the price of adding extra viscosity terms in all equations and using continuous finite element spaces for all variables, which results in a bit more dissipative behavior compared to already existing results.

We have presented some preliminary ideas how to extend our method by introducing length-scale independent artificial viscosity coefficients, and how to achieve pointwise mass conservation and time-independent mass matrices by incorporating the mass diffusion into the mesh motion. The method can also be extended by adding ALE capabilities, namely mesh relaxation, solution remap, and multi-material treatment stages.

5.3 ALE Extensions

We have presented some approaches and preliminary results for the different stages of ALE simulations. The proposed mesh relaxation uses a harmonic smoothing scheme based on a mesh Laplacian matrix. This algorithm is robust and easy to parallelize. We have discussed remap algorithms that solve Lagrangian-type advection equations in pseudo-time. These algorithms are used for all fields and they have good conservation properties. By introducing some material-specific notions we have achieved reasonable pressure values in most mixed zones, and we have fixed the pressure problem at initialization.

The presented work is based on the results obtained during two summer internships, hence it is by no means complete. Ideas about local mesh relaxation, solution-dependent mesh relaxation, and other mesh optimization algorithms are being discussed. Approaches focused on the monotonicity properties of the discussed

remap algorithms are also work in progress. The presented multi-material methods are defined only with respect to the Lagrangian part of BLAST, they must also be extended to the ALE stages.

REFERENCES

[1] Y. Achdou, F. Camilli, and I. Capuzzo-Dolcetta, *Mean field games: Numerical methods for the planning problem*, SIAM J. Control Optim., 50 (2012), pp. 77–109.

[2] Y. Achdou and I. Capuzzo-Dolcetta, *Mean field games: Numerical methods*, SIAM J. Numer. Anal., 48 (2010), pp. 1136–1162.

[3] Y. Achdou, J.-M. Lasry, P.-L. Lions, and B. Moll, *Heterogeneous Agent Models in Continuous Time*, To Appear, (2014).

[4] C. Bardos, A. Y. le Roux, and J.-C. Nédélec, *First order quasilinear equations with boundary conditions*, Comm. Partial Differential Equations, 4 (1979), pp. 1017–1034.

[5] A. Barlow, *A compatible finite element multi-material ALE hydrodynamics algorithm*, Internat. J. Numer. Methods Fluids, 56 (2008), pp. 953–964.

[6] D. Benson, *An efficient, accurate, and simple ALE method for nonlinear finite element programs*, Comput. Methods Appl. Mech. Engrg., 72 (1989), pp. 305–350.

[7] H. Brenner, *Fluid mechanics revisited*, Physica A: Statistical Mechanics and its Applications, 370 (2006), pp. 190–224.

[8] E. Caramana, D. Burton, and M. Shashkov, *The construction of compatible hydrodynamics algorithms utilizing conservation of total energy*, J. Comput. Phys., 146 (1998), pp. 227–262.

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

[10] E. Caramana, M. Shashkov, and P. Whalen, *Formulations of artificial viscosity for multi-dimensional shock wave computations*, J. Comput. Phys., 144 (1998), pp. 70–97.

[11] L. Corrias, M. Falcone, and R. Natalini, *Numerical schemes for conservation laws via Hamilton-Jacobi equations*, Math. Comp., 64 (1995), pp. 555–580.

[12] R. Darlington, T. McAbee, and G. Rodrigue, *A study of ALE simulations of Rayleigh-Taylor instability*, Comp. Phys. Comm., 135 (2001), pp. 58–73.

[13] X. Ding, G. Chen, and P. Luo, *Convergence of the Lax-Friedrichs scheme for isentropic gas dynamics I*, Acta. Math. Sci., 5 (1985), pp. 415–432.

[14] R. J. DiPerna, *Convergence of approximate solutions to conservation laws*, Arch. Rational Mech. Anal., 82 (1983), pp. 27–70.

[15] ——, *Convergence of the viscosity method for isentropic gas dynamics*, Comm. Math. Phys., 91 (1983), pp. 1–30.

[16] V. Dobrev, T. Ellis, T. Kolev, and R. Rieben, *Curvilinear finite elements for Lagrangian hydrodynamics*, Internat. J. Numer. Methods Fluids, 65 (2011), pp. 1295–1310.

[17] ——, *High-order curvilinear finite elements for axisymmetric Lagrangian hydrodynamics*, Comput. and Fluids, 83 (2013), pp. 58–69.

[18] V. Dobrev and T. Kolev, *GLVis - OpenGL visualization tool*. http://glvis.googlecode.com, 03/18/2014.

[19] ——, *MFEM - Modular finite element discretization library.* mfem.googlecode.com, 03/18/2014.

[20] V. Dobrev, T. Kolev, and R. Rieben, *High-order curvilinear finite element methods for Lagrangian hydrodynamics*, SIAM J. Sci. Comput., 34 (2012), pp. B606–B641.

[21] ——, *High order curvilinear finite elements for elasticplastic Lagrangian dynamics*, J. Comput. Phys., 257 (2014), pp. 1062–1080.

[22] J. Dukowicz and B. Meltz, *Vorticity errors in multidimensional Lagrangian codes*, J. Comput. Phys., 99 (1992), pp. 115–134.

[23] D. Flanagan and T. Belytschko, *A uniform strain hexahedron and quadrilateral with orthogonal hourglass control*, Internat. J. Numer. Methods Engrg., 17 (1981), pp. 679–706.

[24] O. Gueant, *Mean field games equations with quadratic Hamiltonian: a specific approach*, Math. Models Methods Appl. Sci., 22 (2012).

[25] J.-L. Guermond and M. Nazarov, *A maximum-principle preserving linear finite element method for scalar conservaton equations*, To appear, (2014).

[26] J.-L. Guermond, M. Nazarov, and B. Popov, *Implementation of the entropy viscosity method*, Tech. Rep. 4015, KTH, Numerical Analysis, Sweden, Stockholm, 2011. QC 20110720.

[27] J.-L. Guermond, R. Pasquetti, and B. Popov, *Entropy viscosity method for nonlinear conservation laws*, J. Comput. Phys., 230 (2011), pp. 4248–4267.

[28] J.-L. Guermond and B. Popov, *Viscous regularization of the Euler equations and entropy principles*, To appear, (2014).

[29] A. Harten, *On the symmetric form of systems of conservation laws with entropy*, J. Comput. Phys., 49 (1983), pp. 151–164.

[30] A. Harten, P. Lax, D. Levermore, and W. Morokoff, *Convex entropies and hyperbolicity for general Euler equations*, SIAM J. Numer. Anal., 35 (1998), pp. 2117–2127.

[31] A. Harten and S. Osher, *Uniformly high-order accurate nonoscillatory schemes*, SIAM J. Numer. Anal., 24 (1987), pp. 279–309.

[32] C. Hirt, A. Amsden, and J. Cook, *An arbitrary Lagrangian-Eulerian computing method for all flow speeds*, J. Comput. Phys., 135 (1997), pp. 198–216.

[33] C. Hu and C.-W. Shu, *A discontinuous Galerkin finite element method for Hamilton-Jacobi equations*, SIAM J. Sci. Comput., 21 (1999), pp. 669–690.

[34] S. Jin and Z. Xin, *Numerical passage from systems of conservation laws to Hamilton-Jacobi equations, relaxation schemes*, SIAM J. Numer. Anal., 35 (1998), pp. 2385–2404.

[35] P. Kjellgren and J. Hyvarien, *An arbitrary Lagrangian-Eulerian finite element method*, Comp. Mech., 21 (1998), pp. 81–90.

[36] T. Kolev and R. Rieben, *A tensor artificial viscosity using a finite element approach*, J. Comput. Phys., 228 (2009), pp. 8336–8366.

[37] S. Krushkov, *First order quasilinear equations with several independent variables*, Math. USSR-Sb., 10 (1970), pp. 217–243.

[38] A. Kurganov and E. Tadmor, *New high-resolution semi-discrete central schemes for Hamilton-Jacobi equations*, J. Comput. Phys., 160 (2000), pp. 720–742.

[39] A. Lachapelle, J. Salomon, and G. Turinici, *A monotonic algorithm for mean field games model in economics*, Math. Models Meth. Appl. Sci., 1 (2010), pp. 1–22.

[40] J.-M. Lasry and P.-L. Lions, *Mean Field Games*, Jpn. J. Math., 2 (2007), pp. 229–260.

[41] Lawrence Livermore National Laboratory, *BLAST - Object-oriented high-order finite element shock hydrocode*. www.llnl.gov/casc/blast, 03/18/2014.

[42] P. Lax, *Weak solutions of nonlinear hyperbolic equations and their numerical computation*, Comm. Pure Appl. Math., 7 (1954), pp. 159–193.

[43] ——, *Shock Waves and Entropy in Contributions to Nonlinear Functional Analysis*, Academic Press, New York, NY, 1971.

[44] C.-T. Lin and E. Tadmor, *High-Resolution non-oscillatory central schemes for Hamilton-Jacobi equations*, SIAM J. Sci. Comput., 21 (1999), pp. 2163–2186.

[45] ——, *L^1-stability and error estimates for approximate Hamilton-Jacobi solutions*, Numer. Math., 87 (2001), pp. 701–735.

[46] P.-L. Lions, *Cours du College de France: Theorie des jeux a champs moyens (video lectures)*. http://www.college-de-france.fr/site/pierre-louis-lions/#course, 04/05/2014.

[47] P.-L. Lions and P. Souganidis, *Convergence of MUSCL and filtered schemes for scalar conservation laws and Hamilton-Jacobi equations*, Numer. Math., 69 (1995), pp. 441–470.

[48] P.-H. Maire, *A high-order cell-centered Lagrangian scheme for two-dimensional compressible fluid flows on unstructured meshes*, J. Comput. Phys., 228 (2009), pp. 2391–2425.

[49] P.-H. Maire, R. Abgrall, J. Breil, and J. Ovadia, *A cell-centered Lagrangian scheme for two-dimensional compressible flow problems*, SIAM J. Sci. Comput., 29 (2007), pp. 1781–1824.

[50] A. Majda, *Compressible fluid flow and systems of conservation laws in several space variables*, Springer-Verlag, New York, 1984.

[51] H. Nessyahu and E. Tadmor, *Nonoscillatory central differencing for hyperbolic conservation laws*, J. Comput. Phys., 87 (1990), pp. 408–463.

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

[53] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922.

[54] L. Sedov, *Similarity and dimensional methods in mechanics*, CRC Press, Boca Raton, FL, 1993.

[55] D. Serre, *Systemes de lois de conservation I: hyperbolicite, entropies, ondes de choc*, Diderot Editeur, Paris, 1996.

[56] M. Shashkov and J. Campbell, *A tensor artificial viscosity using a mimetic finite difference algorithm*, J. Comput. Phys., 172 (2001), pp. 739–765.

[57] M. Shashkov, J. Hyman, and J. Campbell, *Mimetic finite difference operators for second-order tensors on unstructured grids*, Comput. Math. Appl., 44 (2002), pp. 157–173.

[58] E. Tadmor, *A minimum entropy principle in the gas dynamics equations*, Appl. Numer. Math., 2 (1986), pp. 211–219.

[59] J. Von Neumann and R. Richtmyer, *A method for the numerical calculation of hydrodynamic shocks*, J. Appl. Phys., 21 (1950), pp. 232–237.

[60] M. Wilkins, *Methods in computational physics*, Academic Press, San Diego, CA, 1964.

[61] V. Zingan, J.-L. Guermond, J. Morel, and B. Popov, *Implementation of the entropy viscosity method with the discontinuous Galerkin method*, Comput. Methods Appl. Mech. Engrg., 253 (2013), pp. 479–490.