Entropy-based Mesh Refinement, II: A New Approach to Mesh Movement

Daniel W. Zaide[*] and Philip L. Roe[†]

Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109

This work presents an entropy-based monitor function for mesh movement with an application to the Euler Equations. The entropy variables provide an alternate description of the system and are used in conjunction with the conserved variables to develop a monitor function which is shown to be an efficient method of detecting waves. Two moving mesh algorithms are examined: a Gauss-Seidel iterative method and an Arbitrary Lagrangian-Eulerian method. Numerical results in 1D and 2D are used to demonstrate their ability.

I. Introduction

Adaptive meshing is a powerful method to make the most efficient use of computational resources, and indeed, in many situations, may be the only viable approach. The are numerous possible strategies, each of which begins by evaluating some kind of refinement indicator or monitor function. In response to the indicator, some local action is taken, for example mesh enrichment (h-refinement), locally improved order of accuracy (p-refinement), mesh reorientation, and mesh movement. In the matter of indicators, two contrasting approaches have been taken in the past. One is empirical, and relies on intuitive methods for detecting "flow features" that are thought to be important. This is usually not expensive, but weaknesses are the need to supply and tune multiple parameters, and the absence of an incentive to refine regions that may appear insignificant but may be important for the proper formation of the desired features. A more theoretically-based approach begins by formulating an optimisation problem. For example, we can seek a mesh that will minimise the error in some predicted scalar quantity J, such as lift, and this can be done by solving an adjoint problem to obtain the sensitivity of J to local errors in the solution. Drawbacks to this approach are the expense of the adjoint solution, and the need to focus on a specific objective J whose choice may not be obvious.

In a companion paper,[1] it is shown that taking the objective J to be the net flux of entropy through the domain boundary has some remarkable properties, and can be viewed as a form of automatic feature detection. It seems that any unresolved aspect of the solution, whether flow-based or geometrical, generates numerical entropy, and that the net entropy flux will therefore be sensitive to the resolution of that feature, and also to any region that influences it. In[1] it is found that geometrical features, shocks, boundary layers and trailing vortices are all picked up. Moreover, the adjoint solution that provides the sensitivities for this choice of objective is available at virtually no cost, because it is simply the transformation of the flow solution into entropy variables

$$\mathbf{v}(\mathbf{u}) = \partial_{\mathbf{u}}(-\rho \mathbf{u} S) \tag{1}$$

where S is the thermodynamic entropy $S = \ln(p) - \gamma \ln \rho$.

In this paper, we discuss mesh adaptation for unsteady flow problems. In principle, the adjoint approach could still be applied. The objective J could now be the net entropy flux through a control volume consisting of the entire space-time domain, but this would of course be very expensive, being essentially global in time ($0 \leq t \leq T$). Suppose that we have fixed resources that we want to utilise efficiently in time, for example by allocating them between the first half of the computation ($0 \leq t \leq T/2$) and the second half. But of course during the first half we have no idea how complicated and demanding the second half will be.

We could apply an adjoint approach over some fixed block of timesteps, or even over a single timestep, but we have not pursued this in any detail. Instead, we present a feature-based method that derives some

[*]Graduate Student

[†]Professor, AIAA Fellow

inspiration and encouragement from the adjoint approach. This employs a feature detector that is related to the entropy variables, which we call the entropy distance, and use this to drive a moving mesh,[2–5] in which mesh points migrate to the important areas of the flow. As observed above, this is merely one of the possible mesh adaptation strategies, and not necessarily the most efficient. Our object is to show that the entropy distance works as an effective driver that replaces a variety of empirical drivers, appearing to serve as a universal "feature detector". Although our presentation is chiefly in terms of the Euler equations, the entropy distance is defined for any set of conservation laws equipped with an entropy function.

We also make some observations about computing conservation laws on a moving mesh that may have more general application. In particular, we present a generalization to moving grids of the MUSCL-Hancock method that is fully explicit and second-order accurate. This is more direct and accurate than methods that rely on interpolating from one mesh to another. We also make some observations on the so-called Moving Mesh Partial Differential Equation, or MMPDE, providing a more efficient hyperbolic reformulation of what is usually presented as a parabolic problem.

II. Governing Equations

The governing equations are the Euler Equations, however, this methodology also applies to any hyperbolic system of PDEs with an entropy, such as the shallow water, the Navier-Stokes, and the Magnetohydrodynamics equations.[6] All of these can be written in vector form as

$$\mathbf{u}_t + \mathbf{f}^i_{x_i} = \mathbf{u}_t + \mathbf{f}^i_{\mathbf{u}}\mathbf{u}_{x_i} = \mathbf{u}_t + \mathbf{A}^i\mathbf{u}_{x_i} = 0 \tag{2}$$

or, for the Euler equations, expanded as

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho \\ \rho\mathbf{V} \\ E\end{bmatrix} + \nabla\cdot\begin{bmatrix}\rho\mathbf{V} \\ \rho\mathbf{V}\mathbf{V}+\mathbf{I}p \\ \mathbf{V}(E+p)\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}, \tag{3}$$

with the equation of state $p = p(\rho, i)$. For an ideal gas as

$$p = (\gamma-1)\rho i, \qquad \rho i = \left(E - \frac{1}{2}\rho\mathbf{V}^T\mathbf{V}\right). \tag{4}$$

The hyperbolicity of the system implies $\mathbf{A}^i = \mathbf{f}^i_{\mathbf{u}} = \mathbf{R}^i\mathbf{\Lambda}\mathbf{L}^i$.

III. Entropy Variables

Define the physical entropy, of an ideal gas as

$$S = \ln\left(p\rho^{-\gamma}\right). \tag{5}$$

The entropy inequality can then be written as

$$S_t + \mathbf{V}\cdot\nabla S \geq 0, \tag{6}$$

or in generalized form for a convex entropy function $g(S)$, $g' > 0$ and $g''/g' < \gamma^{-1}$ as[7]

$$g_t + \mathbf{V}\cdot\nabla g \geq 0. \tag{7}$$

The entropy variables, $\mathbf{v}$, then come from

$$U(\mathbf{u}) = -\rho g(S), \qquad \mathbf{v} = U^T_{\mathbf{u}}. \tag{8}$$

Following,[8] the change of variable Jacobians $\mathbf{u}_{\mathbf{v}}$ and $\mathbf{v}_{\mathbf{u}}$, are both symmetric and positive definite since $\mathbf{v}_{\mathbf{u}} = U_{\mathbf{uu}} = \mathbf{u}_{\mathbf{v}}{}^{-1}$. We can now write (2) as

$$\mathbf{u}_{\mathbf{v}}\mathbf{v}_t + \mathbf{f}^i_{\mathbf{v}}\mathbf{v}_{x_i} = 0, \tag{9}$$

where the matrices $\mathbf{f_v}$ are also symmetric. Barth[8] shows that the change of variable Jacobian is then necessarily of the form

$$\mathbf{u_v} = \mathbf{R}\mathbf{C}\mathbf{C}^T\mathbf{R}^T = \tilde{\mathbf{R}}\tilde{\mathbf{R}}^T \tag{10}$$

for a diagonal eigenvector scaling matrix $\mathbf{C}$. $\tilde{\mathbf{R}}$ contains the right eigenvectors of $\mathbf{A}$. Because the Euler equations are rotationally invariant, it is not important which of the $\mathbf{A}^i$ is chosen. It could be any one one of them, or some linear combination arising from a coordinate rotation. The scaling, however, is important, and is what allows the monitor function to be defined. Defining $\tilde{\mathbf{L}} = \tilde{\mathbf{R}}^{-1}$ we also have the inverse relationship

$$\mathbf{v_u} = (\tilde{\mathbf{R}}\tilde{\mathbf{R}}^T)^{-1} = \tilde{\mathbf{L}}^{\mathbf{T}}\tilde{\mathbf{L}}. \tag{11}$$

III.A. An Euler Equation Example in 1D

First, as in[9,10] define the entropy variables using $g(S) = \frac{S}{\gamma-1}$, so that the entropy function is given by

$$U(\mathbf{u}) = -\frac{\rho S}{\gamma - 1} = -\frac{\rho}{\gamma - 1}\ln(p\rho^{-\gamma}), \tag{12}$$

which gives the entropy variables as

$$\mathbf{v} = \begin{bmatrix} -\frac{S}{\gamma-1} + \frac{\gamma+1}{\gamma-1} - \frac{E}{p} \\ \frac{\rho u}{p} \\ -\frac{\rho}{p} \end{bmatrix}. \tag{13}$$

The change of variable Jacobians are then

$$\mathbf{u_v} = \begin{bmatrix} \rho & \rho u & E \\ \rho u & \rho u^2 + p & \rho u H \\ E & \rho u H & \rho H^2 - \frac{a^2 p}{\gamma-1} \end{bmatrix} \tag{14}$$

and

$$\mathbf{v_u} = \frac{\gamma - 1}{p^2}\begin{bmatrix} \frac{\gamma p^2}{(\gamma-1)^2\rho} + \frac{1}{4}\rho u^4 & -\frac{1}{2}\rho u^3 & -\frac{p}{\gamma-1} + \frac{1}{2}\rho u^2 \\ -\frac{1}{2}\rho u^3 & \frac{p}{\gamma-1} + \rho u^2 & \rho u \\ -\frac{p}{\gamma-1} + \frac{1}{2}\rho u^2 & -\rho u & \rho \end{bmatrix} \tag{15}$$

with $a^2 = \frac{\gamma p}{\rho}$ and $H = \frac{a^2}{\gamma-1} + \frac{u^2}{2}$. We write the right and left eigenvectors in the forms that are usually given, such that the matrices $\mathbf{R}, \mathbf{L}$ are mutually inverse, but not otherwise scaled;

$$\mathbf{R} = \left[\begin{array}{c|c|c} 1 & 1 & 1 \\ u - a & u & u + a \\ \frac{a^2}{\gamma-1} - ua + \frac{1}{2}u^2 & \frac{1}{2}u^2 & \frac{a^2}{\gamma-1} - ua + \frac{1}{2}u^2 \end{array}\right] \tag{16}$$

$$\mathbf{L} = \frac{1}{2a^2}\left[\begin{array}{c|c|c} \frac{\gamma-1}{2}u^2 + ua & -(\gamma-1)u - a & (\gamma-1) \\ 2a^2 - (\gamma-1)u^2 & 2(\gamma-1)u & -2(\gamma-1) \\ \frac{\gamma-1}{2}u^2 - ua & -(\gamma-1)u + a & (\gamma-1) \end{array}\right]. \tag{17}$$

The scaling matrix is then

$$\mathbf{C}^2 = \mathbf{R}^{-1}\mathbf{u_v}\mathbf{R}^{-T} = \mathbf{L}\mathbf{u_v}\mathbf{L}^T = \frac{\rho}{\gamma}\begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \gamma - 1 & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix}. \tag{18}$$

The scaled eigenvectors are

$$\tilde{\mathbf{R}} = \mathbf{R}\mathbf{C}, \qquad \tilde{\mathbf{L}} = \mathbf{C}^{-1}\mathbf{L}. \tag{19}$$

III.B. The Entropy Distance

Let two states $\mathbf{u}_1, \mathbf{u}_2$ be given, and let the associated entropy variables be $\mathbf{v}_1, \mathbf{v}_2$ in any number of dimensions. Consider the scalar quantity

$$z^2 = (\mathbf{v}_1 - \mathbf{v}_2)^T(\mathbf{u}_1 - \mathbf{u}_2) = \Delta\mathbf{v}^T\Delta\mathbf{u}. \tag{20}$$

For infinitesimal changes, we have

$$z^2 = \mathrm{d}\mathbf{v}^T\mathrm{d}\mathbf{u} = \mathrm{d}\mathbf{u}^T\mathbf{v}_\mathbf{u}^T\mathrm{d}\mathbf{u} = \mathrm{d}\mathbf{u}^T(\tilde{\mathbf{L}}^T\tilde{\mathbf{L}})\mathrm{d}\mathbf{u} = (\tilde{\mathbf{L}}\mathrm{d}\mathbf{u})^T(\tilde{\mathbf{L}}\mathrm{d}\mathbf{u}). \tag{21}$$

Now $\mathbf{L}\mathrm{d}\mathbf{u}$ gives the amplitude of the wavestrengths in a Riemann problem having $\mathbf{u}_1, \mathbf{u}_2$ as data, so that z^2 represents the sum of the squares of the wavestrengths.

Consider space divided into two halves by a plane; one half containing fluid in the state $\mathbf{u}_1$, and other in the state $\mathbf{u}_2$. We have the remarkable result that the sum of the squares of the wavestrengths exchanged between the two states does not depend on the orientation of the plane. Therefore z^2 is a measure of distance in state space, and is independent of location in physical space.

For infinitesimal changes there is an alternative form,

$$z^2 = \mathrm{d}\mathbf{v}^T\mathrm{d}\mathbf{u} = \mathrm{d}\mathbf{v}^T\mathbf{u}_\mathbf{v}\mathrm{d}\mathbf{v} = \mathrm{d}\mathbf{v}^T(\tilde{\mathbf{R}}\tilde{\mathbf{R}}^T)\mathrm{d}\mathbf{u} = (\tilde{\mathbf{R}}^T\mathrm{d}\mathbf{v})^T(\tilde{\mathbf{R}}^T\mathrm{d}\mathbf{v}) \tag{22}$$

which is valid because $\tilde{\mathbf{R}}^T\mathrm{d}\mathbf{v} = \tilde{\mathbf{R}}^T\mathbf{v}_\mathbf{u}\mathrm{d}\mathbf{u} = \tilde{\mathbf{R}}^T(\tilde{\mathbf{L}}^T\tilde{\mathbf{L}})\mathrm{d}\mathbf{u} = \tilde{\mathbf{L}}\mathrm{d}\mathbf{u}$ is an alternative expression for the wavestrength.

If the states are not close, we can follow a proof in[7] that considers the path $\mathbf{v} = \mathbf{v}_1 + \theta(\mathbf{v}_2 - \mathbf{v}_1)$ for $\theta \in [0, 1]$ and then writes

$$z^2 = \Delta\mathbf{v}^T\Delta\mathbf{u} = \Delta\mathbf{v}\int_{\mathbf{u}_1}^{\mathbf{u}_2}\mathbf{u}_\mathbf{v}\mathrm{d}\mathbf{v} = \Delta\mathbf{v}^T\int_0^1\mathbf{u}_\mathbf{v}\mathrm{d}\theta\ \ \Delta\mathbf{v}. \tag{23}$$

Because $\mathbf{u}_\mathbf{v}$ is symmetric, so is its integral, which represents a positive definite matrix. Therefore the quantity z^2 is positive for all pairs of states $\mathbf{u}_1, \mathbf{u}_2$. Note that it has the dimensions of density.

The definition of entropy, and hence of entropy distance, is unaffected by considering the Navier-Stokes equations. It offers a natural measure of distance that requires no adjustable parameters. Based on the analysis leading to (21) it is a detector of local activity that is equally sensitive to all types of wave motion.

IV. Monitor Function

The heuristic that motivates many mesh-movement algorithms is that "important" features require fine mesh spacing. In one dimension, we might take $w(\mathbf{u})\Delta x$ to be a constant, where $w(\mathbf{u})$ is a "monitor function" that measures importance. Traditionally, the monitor function is a gradient-based function.[3] The simple choice $w(\mathbf{u}) = |\partial\mathbf{u}/\partial x|$ is inadequate, because it would lead to infinite spacings in zero gradients. Instead, a common alternative is

$$w(\mathbf{u}) = \sqrt{1 + \beta^2(\partial\mathbf{w}/\partial x)^2} \tag{24}$$

for some constant β and some choice of the dependent variables $\mathbf{w}$. When discretized, this takes the form

$$w(\mathbf{u}) = \sqrt{1 + \beta^2(\Delta\mathbf{w}/\Delta x)^2}. \tag{25}$$

This monitor function has several flaws. First, it is not dimensionally consistent, and will change with the choice of measuring units. Second, there seems to be no universality about it. We propose a monitor function

$$w(\mathbf{u}) = \sqrt{1 + \beta^2 z^2/\bar{\rho}} \tag{26}$$

for some density $\bar{\rho}$ included to make w dimensionless. The free parameter β seems necessary. It controls the "aggression" with which the mesh adapts. If $\beta = 0$, no adaptation takes place, but if β is very large, all mesh points will be drawn into the most prominent feature.

It should be acknowledged that the quantity $\Delta\mathbf{v}^T\Delta\mathbf{u}$ was used long ago[11] to trigger a different kind of mesh movement scheme. This involved a kind of shock tracking in one dimension. It exploited the ability of the monitor to track all kinds of discontinuity, but did not exploit its isotropic character. Also, properties of the product $\Delta\mathbf{v}^T\Delta\mathbf{u}$ were employed in[7] to reconstruct a contact wave within the HLL flux model.

V. Mesh Redistribution

As a desirable heuristic, we may propose that all elements of a mesh are equally important. This very loose definition begs several questions. What do we mean by "element" (node, edge, face, volume?); what do we mean by "important"?, and important to what end?. The adjoint approach interprets the principle as meaning that all cells should contribute equally to the error in J. In[1] the policy is implemented by refining those cells that contribute much more than the average to that error. A strict equidistribution is not enforced and might not be practical.

Our method borrows from that brach of the moving mesh literature that expresses a desirable property of the mesh through deriving a PDE to be satisfied by the mesh. We introduce a computational space $\boldsymbol{\xi}$ in which the mesh is uniform, and a mapping $\mathbf{x}(\boldsymbol{\xi})$ that defines the actual mesh.

Geometric properties of the actual mesh can be represented by elements of the transformation matrix $\mathbf{x}_{\boldsymbol{\xi}}$. A common approach is to minimise the integral of some scalar functional of the mapping, and the Euler-Lagrange equations then provide a differential equation to be satisfied by the mesh coordinates. This is the moving mesh PDE.[2,4,5] Devising the functional to be minimized is somewhat empirical, but we follow here a proposal by Ceniceros and Hou[12] (see also[3]), to minimise

$$E = \frac{1}{2}\int w((\mathrm{grad}_{\boldsymbol{\xi}} x)^2 + (\mathrm{grad}_{\boldsymbol{\xi}} y)^2) d\boldsymbol{\xi}. \tag{27}$$

If the weighting function w is constant, then by a well-known result, both x and y would satisfy Laplace's equation. If w is not constant, then the procedure seeks to place grids that are both fine and smooth in those regions where it is large.

The Euler-Lagrange equations turn out to be

$$\nabla_{\boldsymbol{\xi}} \cdot (w\ \nabla_{\boldsymbol{\xi}} \mathbf{x}) = 0. \tag{28}$$

For $w > 0$ this is an elliptic equation, a desirable property that prevents mesh tangling.

An extreme form of equidistribution would be to insist that this equation be satisfied everywhere at all times, but this would be very expensive. Moreover, the equation merely expresses an heuristic principle, and there is little point in seeking an exact solution. It has been usual to convert equations such as (28) into a time-dependent equation with a pseudo-time τ;

$$\mathbf{x}_\tau = \kappa \nabla_{\boldsymbol{\xi}} \cdot (w\ \nabla_{\boldsymbol{\xi}} \mathbf{x})) = 0 \tag{29}$$

and to run a few iterations of this mesh problem at each timestep of the physical problem.

This amounts, of course, to a simple FTCS method for the parabolic problem, which has the drawback of requiring very small timesteps when the mesh is fine. To work around this, Tang and Tang[3] advocate a Gauss-Seidel method which doubles the allowable steps. In Section VI.B we report a further acceleration (in principle by orders of magnitude) by transforming to a hyperbolic formulation.

VI. Mesh Update and Finite Volume Method

To implement a moving mesh algorithm efficiently, we need to interweave the mesh movement and solution update steps. For compressible flows, this needs to be done conservatively. Tang and Tang[3] accomplish this as follows.

1. At the beginning of each timestep, with the current solution $\mathbf{u}$ on the current grid G, perform one iteration of the grid update, and make a conservation interpolation of $\mathbf{u}$ onto the new grid,
2. Iterate the previous step "3-5 times",
3. Update the solution on the new grid, using a two-step second-order Runge-Kutta scheme.

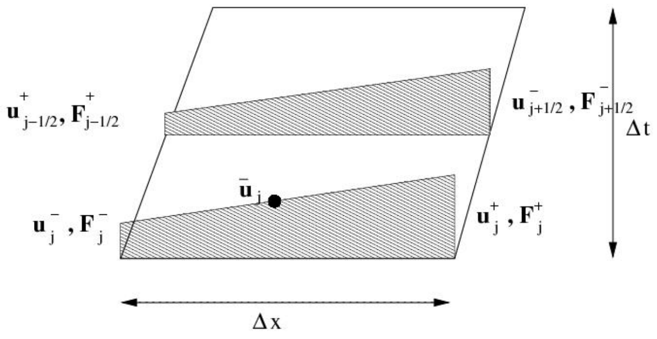
Even with only one iteration, this method contains an interpolation step and two update steps. We now describe a version that eliminates the interpolation step, and so removes one source of error, while also requiring only one Riemann solution per interface and retaining second order accuracy.

VI.A. Arbitrary Lagrangian-Eulerian (ALE) Approach

The interpolation step can be avoided by computing the new solution on a moving grid as detailed below. Numerous authors have documented the formulation of the Euler equations in a time-dependent coordinate system, and the well-known software package CLAWPACK[13] offers this as an option. Stockie et. al.,[4] reported a method that exploited this. They used Crank-Nicholson rather than Gauss-Seidel to update the mesh, and used CLAWPACK to update the solution. They experienced difficulty maintaining stability, and devised a rather elaborate procedure to overcome this.

We have taken a more straightforward approach, based on a generalization to moving meshes of the Hancock scheme.[14] The Hancock scheme is second-order, monotone, and fully discrete. It is the subject of a recent analysis by Berthon[15] and has been extended to third-order by Suzuki.[16] Here we make a straightforward generalization of the second-order version, and have found it very satisfactory. Although this is a two-step scheme, it requires only a single call to a Riemann solver for each timestep per interface.

VI.A.1. One dimension



$\mathbf{u}^+_{j-1/2}, \mathbf{F}^+_{j-1/2}$
$\mathbf{u}^-_{j+1/2}, \mathbf{F}^-_{j+1/2}$
Δt
$\bar{\mathbf{u}}_j$
$\mathbf{u}^-_j, \mathbf{F}^-_j$
$\mathbf{u}^+_j, \mathbf{F}^+_j$
Δx

Figure 1. A finite volume cell in a one-dimensional moving mesh.

Define the new nodal positions $x_i^{n+1} = x_i^n + \Delta t \dot{x}_i^n$ with an interface speed $\dot{x}_i^n$. By integrating the conservation law (2) around the control volume we obtain

$$\mathbf{u}_j^{n+1}(x^{n+1}_{i+\frac{1}{2}} - x^{n+1}_{i-\frac{1}{2}}) = \mathbf{u}_j^n(x^n_{i+\frac{1}{2}} - x^n_{i-\frac{1}{2}}) - \Delta t\left(\left(\mathbf{f}_{i+\frac{1}{2}} - \dot{x}_{i+\frac{1}{2}}\mathbf{u}^+_{i+\frac{1}{2}}\right) - \left(\mathbf{f}_{i-\frac{1}{2}} - \dot{x}_{i-\frac{1}{2}}\mathbf{u}^-_{i-\frac{1}{2}}\right)\right). \tag{30}$$

The flux on the interface is now $\mathbf{f} - \dot{x}\mathbf{u}$ and it is easy to show that this quantity may be obtained from Roe's Riemann Solver as

$$\mathbf{f} - \dot{x}\mathbf{u} = \frac{1}{2}(\mathbf{f}_L + \mathbf{f}_R - \dot{x}(\mathbf{u}_L + \mathbf{u}_R)) - \sum_k \alpha_k |\lambda_k - \dot{x}|\mathbf{r}_k. \tag{31}$$

In Hancock's method we take the initial data in cell i be

$$\mathbf{u}_i(x,0) = \bar{\mathbf{u}}_i + \tilde{\mathbf{S}}_i(x - \bar{x}_i), \tag{32}$$

where the bar denotes cell averaged values and the slopes $\tilde{\mathbf{S}}_i$ are limited. Evaluate $\mathbf{u}_i$ at the left and right cell edges, giving $\mathbf{u}_i^-$ and $\mathbf{u}_i^+$. From these states, evaluate $\mathbf{f}_i^-$ and $\mathbf{f}_i^+$. Then, halfway through the timestep evaluate

$$\mathbf{u}^+_{i-\frac{1}{2}} = \bar{\mathbf{u}}_i - \tilde{\mathbf{S}}\left(\frac{1}{2}\Delta x_i - \frac{1}{2}\Delta t \dot{x}_{i-\frac{1}{2}}\right) - \frac{1}{2}\frac{\Delta t}{\Delta x_i}(\mathbf{f}_i^+ - \mathbf{f}_i^-) \tag{33}$$

$$\mathbf{u}^-_{i+\frac{1}{2}} = \bar{\mathbf{u}}_i + \tilde{\mathbf{S}}\left(\frac{1}{2}\Delta x_i + \frac{1}{2}\Delta t \dot{x}_{i+\frac{1}{2}}\right) - \frac{1}{2}\frac{\Delta t}{\Delta x_i}(\mathbf{f}_i^+ - \mathbf{f}_i^-). \tag{34}$$

These are the states to use in the Riemann problem in Equation (30). The additional terms involving $\dot{x}$ impose only a small overhead.

Again, we took only one step to determine the new location of x_i, so that

$$\dot{x}_i = \frac{1}{\Delta t}(x_i^{n+1} - x_i^n). \tag{35}$$

VI.A.2. Two dimensions

Define a cell j with edges i and linear conserved variable distribution

$$\mathbf{u}_j = \overline{\mathbf{u}}_j + \tilde{\nabla}\mathbf{u}_j \cdot (\mathbf{r}_i - \mathbf{r}_j) \tag{36}$$

for cell center $\mathbf{r}_j = (x_j, y_j)$, edge midpoint $\mathbf{r}_i = (x_i, y_i)$ and limited gradient $\tilde{\nabla}\mathbf{u}_j$. Define an edge midpoint velocity vector as $\dot{\mathbf{r}}_i = [\dot{x}_i, \dot{y}_i]^T$ with $\dot{x}_i = \frac{1}{\Delta t}(x^{n+1} - x^n)$ and $\dot{y}_i$ defined correspondingly. Let A_j^n and A_j^{n+1} be current and new cell areas respectively. The Hancock update is then

$$\mathbf{u}_j^{n+1} = \mathbf{u}_j^n \frac{A^n}{A^{n+1}} - \frac{\Delta t}{A^{n+1}} \sum_i \left(\mathbf{f}_i^{n+\frac{1}{2}} \cdot \mathbf{n}_i^{n+\frac{1}{2}} - \mathbf{u}_i^{n+\frac{1}{2}} (\dot{\mathbf{r}}_i \cdot \mathbf{n}_i^{n+\frac{1}{2}}) \right) \tag{37}$$

where $\mathbf{n}_i = [n_x, n_y]^T$ is the scaled outward edge normal and the flux is calculated via a Riemann solver. The conserved variables at $t^{n+\frac{1}{2}}$ at edge i are

$$\mathbf{u}_i^{n+\frac{1}{2}} = \overline{\mathbf{u}}_j^n + \tilde{\nabla}\mathbf{u}_j^n \cdot (\mathbf{r}_i^n - \mathbf{r}_j^n) - \frac{\Delta t}{2A^n} \sum_i \mathbf{f}(\overline{\mathbf{u}}_j^n + \tilde{\nabla}\mathbf{u}_j^n \cdot (\mathbf{r}_i^n - \mathbf{r}_j^n)) \cdot \mathbf{n}_i^n \tag{38}$$

and within Roe's Riemann Solver we define a normal mesh velocity as $\dot{r}_n = \frac{1}{\sqrt{n_x^2 + n_y^2}}(\dot{x} n_x + \dot{y} n_y)$. The flux is then calculated from

$$\mathbf{f} = \frac{1}{2} \left(\mathbf{f}_L + \mathbf{f}_R - \dot{r}_n (\mathbf{u}_L + \mathbf{u}_R) - \sum \alpha_k |\lambda_k - \dot{r}| \mathbf{r}_k \right). \tag{39}$$

VI.B. A hyperbolic approach to mesh movement

The mesh distribution equation (29) is elliptic. To satisfy it to a sufficient approximation, we have followed other workers by embedding it in the parabolic problem,

$$\mathbf{x}_t = \kappa[(w\mathbf{x}_\xi)_\xi + (w\mathbf{x}_\eta)_\eta]. \tag{40}$$

An alternative is to embed it in the following 6×6 hyperbolic relaxation system;

$$\mathbf{x}_t = \kappa(\mathbf{p}_\xi + \mathbf{q}_\eta) \tag{41a}$$

$$\mathbf{p}_t = \frac{(w\mathbf{x})_\xi - \mathbf{p}}{\tau} \tag{41b}$$

$$\mathbf{q}_t = \frac{(w\mathbf{x})_\eta - \mathbf{q}}{\tau} \tag{41c}$$

where κ and τ are for the present arbitrary parameters. The calculation

$$\begin{aligned} \mathbf{x}_{tt} &= \kappa(\mathbf{p}_{\xi,t} + \mathbf{q}_{\eta,t}) && \text{(42a)} \\ &= \frac{\kappa}{\tau}[((w\mathbf{x})_\xi)_\xi - \mathbf{p}_\xi + ((w\mathbf{x})_\eta)_\eta - \mathbf{q}_\eta] && \text{(42b)} \\ &= \frac{\kappa}{\tau}[((w\mathbf{x})_\xi)_\xi + ((w\mathbf{x})_\eta)_\eta - \mathbf{x}_t/\kappa] && \text{(42c)} \\ & && \text{(42d)} \end{aligned}$$

shows that the hyperbolic system converges to same limiting solution as the parabolic system, which is the solution to the elliptic problem. This is of course true for any values of the parameters, but these can be chosen, as pointed out by Nishikawa,[17] to allow large stable timesteps. The wave propagation speeds for this system are $\pm\sqrt{\kappa w/\tau}$. By taking the relaxation time $\tau \propto w$ the wavespeed, and therefore the Courant number can be held constant over the domain. The equal magnitude of left- and right-going wavespeeds makes for an easy implementation of upwinding. Further details will be presented elsewhere.

VI.B.1. Discretization

In one dimension, the system is

$$x_t = \kappa p_\xi \tag{43}$$

$$p_t = \frac{w x_\xi - p}{\tau}. \tag{44}$$

These equations can be discretized on a staggered grid, with p stored in cell centers and x in its natural location at vertices. As a result of a partial one-dimensional stability analysis, the scheme

$$x_j^{n+1} = x_j^n + \frac{\kappa \Delta t}{\Delta \xi}(p_{j+\frac{1}{2}}^n - p_{j-\frac{1}{2}}^n) \tag{45}$$

$$p_j^{n+1} = p_j^n + \frac{3}{8}(p_{j-1}^n - 2p_j^n + p_{j+1}^n) + \frac{\Delta t}{\tau}\left(w_j^n \frac{(x_{j+\frac{1}{2}}^n - x_{j-\frac{1}{2}}^n)}{\Delta \xi} - p_j^n \right) \tag{46}$$

was employed, with $\Delta t = \tau$ and $\kappa = 3/(4 \max w)$. It is possible that better parameters exist.

In two dimensions, a similar scheme and staggered grid was used. A major advantage to this split system is the ability to take two timesteps in x without updating the monitor function.

VII. Results

VII.A. One dimension

Sod's problem[18] is a shocktube with $\mathbf{u}_L = [1.0, 0.0, 1.0]^T$ and $\mathbf{u}_R = [0.125, 0.0, 0.1]^T$. To preserve monotonicity, the harmonic limiter was used on conserved variable slopes. For the monitor function, equation (26) in these numerical tests, differences between cell states were taken from cell centers, leaving w defined on cell edges. To obtain w in cell centers we took averages $w_i = \frac{1}{2}(w_{i+1/2} + w_{i-1/2})$. This leaves the mesh uncoupled, so some slight smoothing was added as in previous work.[3,5] We will explore alternative procedures.

We measure errors in the computed solutions by comparing them with cell averages for the exact solution, calculated using a five-point Gaussian quadrature, and accounting for the locations of the discontinuities. Using $\beta = 30$ in the monitor function to adjust the amount of wave movement and a constant timestep of $\Delta t = \frac{\Delta x}{4}$ we can see a definite error reduction as shown in Figure 2. The percentage error reduction

$$e_{\text{red}} = \frac{100(e_{\text{uniform}} - e_{\text{mm}})}{e_{\text{uniform}}} \tag{47}$$

measures by how much the error has been reduced by switching from a uniform mesh to a non-uniform mesh from the moving mesh algorithm. One other monitor function is compared, devised by Stockie et. al.[4] and recommended by Tang and Tang.[3]

$$w = \sqrt{1 + 20\left(\frac{u_x}{\max|u_x|}\right)^2 + 100\left(\frac{s_x}{\max|s_x|}\right)^2} \tag{48}$$

with $s = p/\rho^\gamma$. This type of monitor function requires a different indicator for each type of wave in the system.

Table 1 below displays the reduction in error achived by combining each monitor with one iteration of the process "remesh + interpolate + update". This directly compares the monitor functions and shows substantial benefit from using the entropy distance. Next, in Tables 2 and 3, we look at the effect of eliminating the interpolation step to yield the process "remesh + moving update". In all cases, the direct update yield similar results at a reduced cost. Figures 3 and 4 show the accuracy, measured in the L_1 norm for both monitors.

The order of accuracy for all schemes can be measured as very close to 1.0. This is quite typical for nominally second-order schemes on this problem whose solution is mostly comprised of jumps and constant states. For smoother problems we have found that our code is in fact second-order accurate.

The solutions at T = 0.125 are plotted in Figure 2. All moving mesh methods outperform the uniform mesh and better resolve all three waves. The mesh movement, shown in Figures 5-7, follow all three waves and results in cells surrounding the waves, as determined by the monitor function.

N	50	100	200	400
$\beta = 30$	36.7	32.2	29.1	18.8
Tang	10.8	8.7	12.3	10.0

Table 1. Percentage error reduction with interpolation followed by update.

N	50	100	200	400
$\beta = 30$	41.0	32.8	28.1	32.4
Tang	18.5	19.4	14.6	15.7

Table 2. Percentage error reduction with Gauss-Seidel direct update on moving mesh.

N	50	100	200	400
$\beta = 30$	36.4	30.6	27.6	26.0
Tang	25.7	27.4	14.5	24.2

Table 3. Percentage error reduction with hyperbolic relaxation direct update on moving mesh.

VII.B. Two dimensions

We study Woodward and Colella's double Mach reflection problem[19] with states $\mathbf{u}_L = [8, 57.16, -33.00, 563.54]^T$ and $\mathbf{u}_R = [1.4, 0.0, 0.0, 2.5]^T$ initially separated by a shock at $x = \frac{1}{6} + \frac{y}{\sqrt{3}}$ in a domain $\Omega = [0, 4] \times [0, 1]$.

We start with an initially uniform grid of $\Delta x = \Delta y = 1/24$. Here, the monitor function defined on each cell interface is taken to be $w = \sqrt{1 + 30z^2/\overline{p}}$. Slight smoothing is then applied to suppress the odd-even decoupling. Timesteps of $\Delta t = 0.001$ were taken in each case. To avoid a directional bias when taking one iterative step, Jacobi Iteration was used in place of Gauss-Seidel. Looking at density at 0.2 seconds in Figures 8-10, both moving meshes do a better job capturing the features of the flow. In particular they resolve the oblique shocks more sharply by creating smooth patterns of diamond-shaped cells. The singularity at the start of the wedge is also much better resolved with a moving mesh.

Density contours are shown in Figures 11-13. The density contours are evenly spaced and equal in each situation. For both of the moving grids, there is a clear superiority in resolving the shockwave close to its fixed origin, but little or no advantage in resolving features on the right of the diagram. We think that this is because the features at the left are rather stable, and the mesh has evolved to fit them quite nicely, whereas the features on the right are newly arrived, and the mesh does not yet fit them well.

The resolution of the moving oblique shock is interesting. It is noticeably sharper on the moving grids than on the fixed grid, but because it is unsteady it does not have time to draw the mesh to itself in the way that it does at the origin. The shock attains its best resolution in those places where it is most long-established. However, once the shock has passed, the grid reverts to uniformity, which would be desirable if some other feature were to arrive later.

VIII. Conclusions and Future Work

The monitor function (26) was very successful in detecting both acoustic and convective features.

In one dimension, the gain in accuracy obtained by moving the mesh points using our preferred combination of the entropy monitor and ALE update amounted consistently to about 45%, compared with about 15-20% for the procedure in.[3] Given that our method showed only roughly first-order accuracy on the Sod problem, the same gain could have been achieved by doubling the number of mesh points, which quadruples the cost, given that time steps also have to be reduced.

In two dimensions, it was difficult to quantify the improvements, but probably the resolution of the discontinuities was again roughly doubled. The cost of doubling the grid in two dimensions is a factor of eight, and in three dimensions is sixteen. We found that our two-dimensional code spent about 30% of its time in mesh management. The ALE implementation of the moving mesh incurs only a light overhead relative to a fixed mesh, but in general, because some cells are reduced in size the time step is reduced and more timesteps have to be taken. Although the present results are merely preliminary, it seems that there could be situations where the moving mesh algorithm would pay off quite handsomely. There are numerous implementational details that are quite empirical at present, and we intend to explore the alternatives, preferably putting them on a more rational basis.

One benefit in comparison with other refinement strategies is that the mesh retains its original topology, so that no indirect addressing is required. If indirect addressing is acceptable, then some form of AMR may be preferred. The advantage of AMR is that points can be added directly to the active areas, whereas with moving meshes they must be brought from a distance. However, the entropy distance should still be a powerful monitor function.

Acknowledgements

The authors would like to thank Dr. Krzysztof Fidkowski and Professor Bram van Leer for valuable comments on this work. This work was partially supported by the DOE NNSA under the Predictive Science Academic Alliances Program by grant DE-FC52-08NA28616. The second author is grateful for hospitality provided by Dr. Nikos Nikiforakis at the Department of Applied Mathematics and Theoretical Physics, University of Cambridge, UK, and the financial support of a William Penney Fellowship from the UK Ministry of Defence.

References

[1]Fidkowski, K. J. and Roe, P. L., "Entropy-based Mesh Refinement, I: The entropy adjoint approach, AIAA paper 2009-3790, 19th AIAA CFD Meeting, San Antonio, 2009," .

[2]Huang, W. and Russell, R. D., "Adaptive mesh movement the MMPDE approach and its applications," Journal of Computational and Applied Mathematics, Vol. 128, No. 1-2, 2001, pp. 383–398.

[3]Tang, H. and Tang, T., "Adaptive Mesh Methods for One- and Two-Dimensional Hyperbolic Conservation Laws," SIAM J. Numer. Anal., Vol. 41, No. 2, 2003, pp. 487–515.

[4]Stockie, J. M., Mackenzie, J. A., and Russell, R. D., "A Moving Mesh Method for One-dimensional Hyperbolic Conservation Laws," SIAM J. Sci. Comput., Vol. 22, No. 5, 2000, pp. 1791–1813.

[5]van Dam, A. and Zegeling, P. A., "A robust moving mesh finite volume method applied to 1D hyperbolic conservation laws from magnetohydrodynamics," J. Comput. Phys., Vol. 216, No. 2, 2006, pp. 526–546.

[6]Serre, D., Systems of conservation laws, Vol. I, Cambridge University Press, 1999.

[7]Harten, A., Lax, P., and van Leer, B., "On upstream differencing and Godunov-type schemes for hyperbolic conservation laws," SIAM Rev., Vol. 25, 1983, pp. 35–61.

[8]Barth, T. J., An Introduction to Recent Developments in Theory and Numerics for Conservation Laws: Proceedings of the International School, Freiburg/Littenweiler, Germany, October 20-24, chap. Numerical Methods for gasdynamic systems on unstructured meshes, Springer-Verlag, 1997, p. 195.

[9]Roe, P., "Affordable, Entropy-consistent, Euler Flux Functions: I. Analytical Results," To Appear in JCP.

[10]Tadmor, E. and Zhong, W., "Entropy Stable Approximations of Navier-Stokes Equations with no Artificial Numerical Viscosity," Journal of Hyperbolic Differential Equations, Vol. 3, No. 3, 2006, pp. 529–560.

[11]Harten, A. and Hyman, J. M., "Self-adjusting grid methods for for one-dimensional hyperbolic conservation laws," J. Comput. Phys., Vol. 50, 1983, pp. 235–269.

[12]Ceniceros, H. and Hou, T., "An efficient dynamically adaptive mesh for potentially singular solutions," J. Comput. Phys., Vol. 172, No. 2, 2001, pp. 609–639.

[13]Leveque, R., "CLAWPACK: A software package for solving multi-dimensional conservation laws," 1997.

[14]van Leer, B., "On the relationship between the upwind-differencing schemes of Godunov, Engquist-Osher, and Roe," SIAM J. Sci. Stat. Comput., Vol. 5, No. 1, 1984, pp. 1–20.

[15]Berthon, C., "Why the MUSCL-Hancock scheme is L1 -stable," Numerische Mathematik, Vol. 104, 2006, pp. 27–46.

[16]Suzuki, Y., Discontinuous Galerkin methods for extended hydrodynamics, Ph.D. thesis, University of Michigan, 2008.

[17]Nishikawa, H., "A first-order system approach for diffusion equation. I: Second-order residual-distribution schemes," J. Comput. Phys., Vol. 227, 2007, pp. 315–352.

[18]Sod, G., "A survey of several difference methods for hyperbolic systems of nonlinear conservation laws," J. Comput. Phys., Vol. 27, 1978, pp. 1–31.

[19]Woodward, P. and Colella, P., "The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks," J. Comput. Phys., Vol. 54, 1984, pp. 115–173.

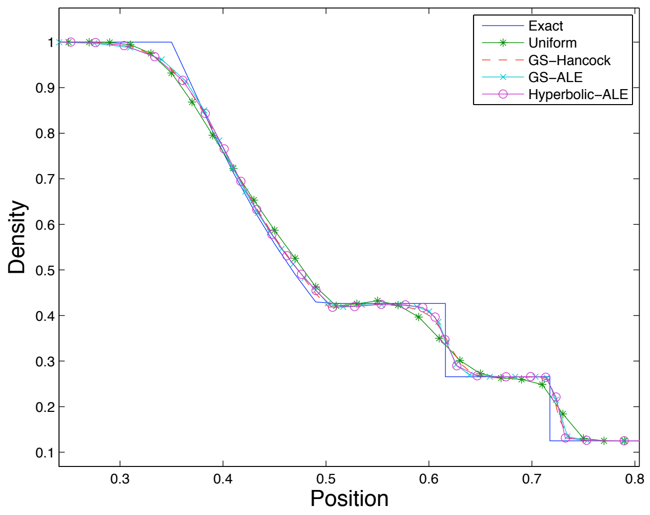


Figure 2. Comparison of Density for Solutions on a uniform mesh, GS moving mesh, GS+ALE moving mesh, HR+ALE moving mesh, and Exact solution on a 50 Cell Mesh at 0.125 seconds.

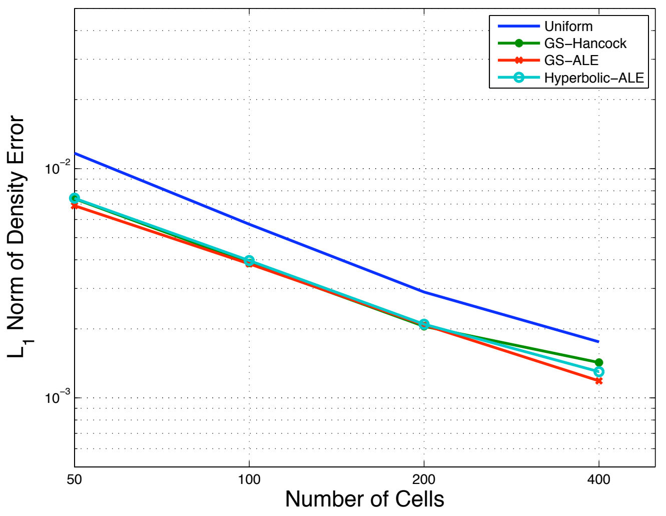


Figure 3. Density Error in the L_1 norm versus number of mesh cells using entropy distance monitor function.

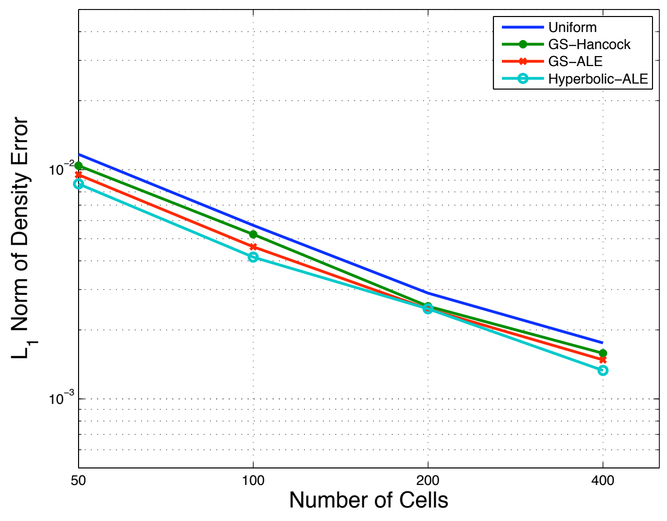


Figure 4. Density Error in the L_1 norm versus number of mesh cells using Tang and Tang's monitor function (48).

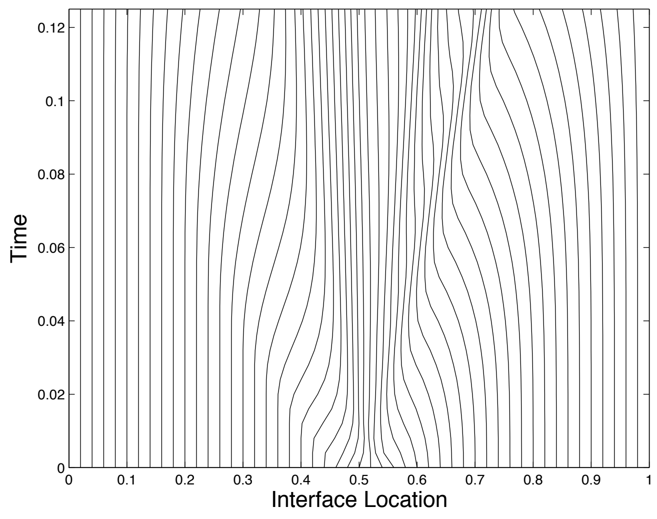


Figure 5. Mesh Movement with 50 Cells for $\beta = 30$ using "Gauss-Seidel remesh+interpolate+update" method.

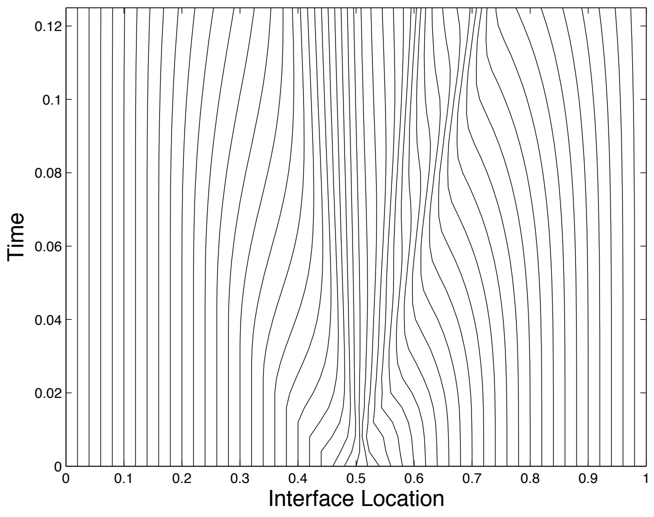


Figure 6. Mesh Movement with 50 Cells for $\beta = 30$ using "Gauss-Seidel remesh+direct update" method.

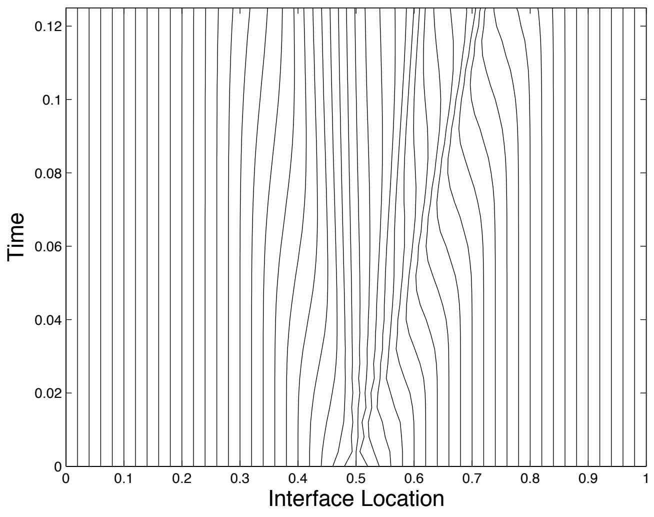


Figure 7. Mesh Movement with 50 Cells for $\beta = 30$ using "hyperbolic relaxation remesh+direct update" method.

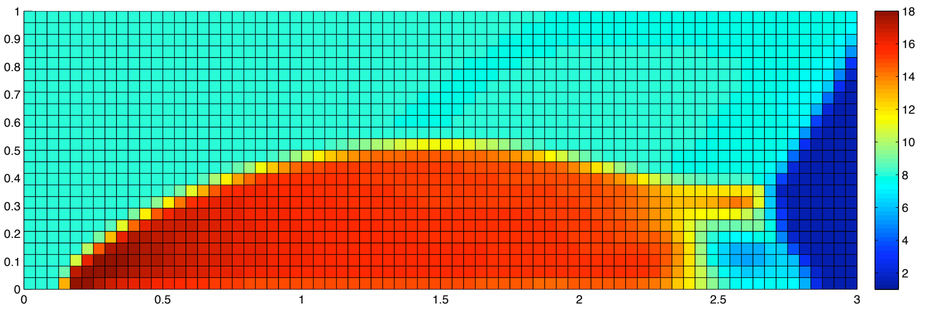


Figure 8. Density of double Mach reflection on a uniform mesh at t = 0.2 seconds.

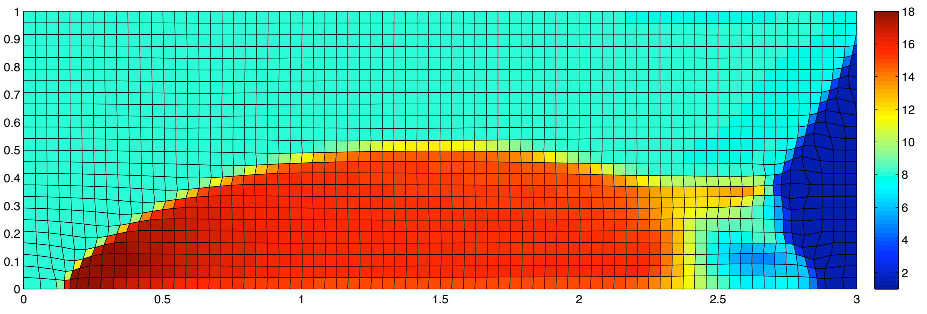


Figure 9. Density of double Mach reflection on a moving mesh using hyperbolic relaxation at t = 0.2 seconds.

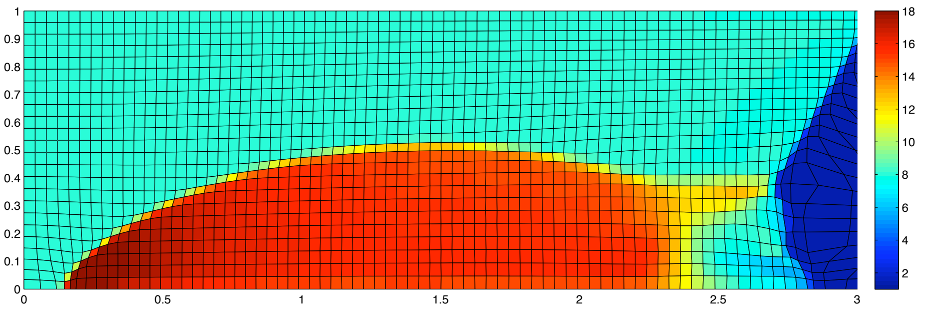


Figure 10. Density of double Mach reflection on a moving mesh using Jacobi iteration at t = 0.2 seconds.

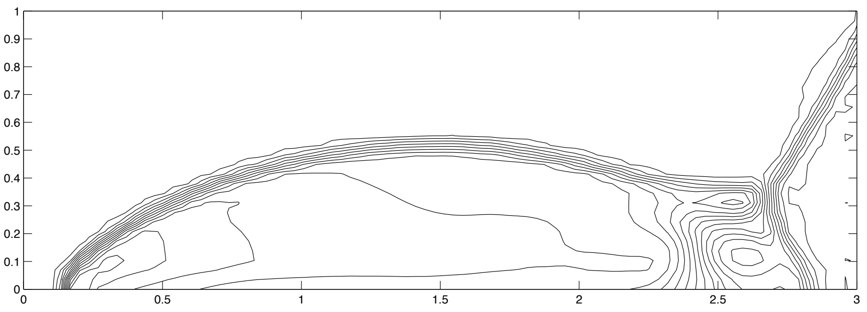


Figure 11. Twenty evenly spaced density contours of double Mach reflection on a uniform mesh at t = 0.2 seconds.

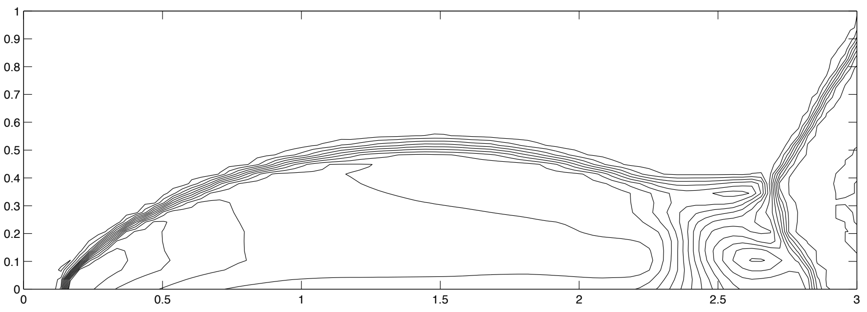


Figure 12. Twenty evenly spaced density contours of double Mach reflection on a moving mesh with hyperbolic relaxation at t = 0.2 seconds.

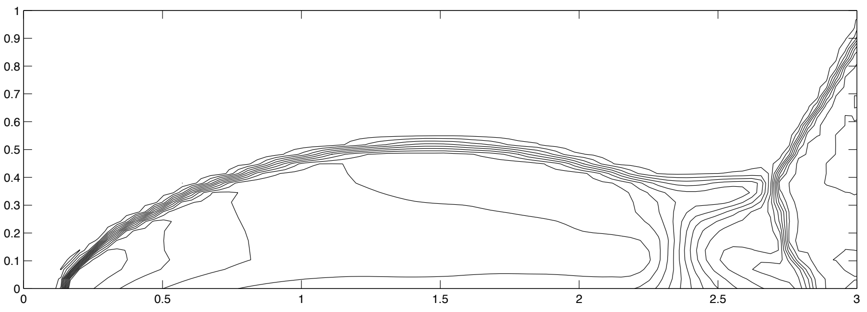


Figure 13. Twenty evenly spaced density contours of double Mach reflection on a moving mesh with Jacobi iteration at t = 0.2 seconds.