Moving Meshes and Higher Order Finite Volume Reconstructions

Fredrik Svensson

Moving Meshes and Higher Order Finite Volume Reconstructions

Fredrik Svensson[†]

[†] *Centre for Mathematical Sciences, Lund University, Box 118, SE–221 00 Lund, Sweden*

fredrik.svensson@na.lu.se

Abstract

In this paper, we present and evaluate a moving mesh finite volume method for hyperbolic conservation laws. The method consists of two parts; a mesh moving scheme based on the algorithm of Tang and Tang, and a third order accurate bi-hyperbolic reconstruction which is an extension of Marquina's PHM. The resulting algorithm calculates the solution of the conservation laws directly in physical space, without any transformation of the computational grid or the hyperbolic equations. Numerical experiments in one and two space dimensions indicate high numerical accuracy of the method.

Key words : Moving mesh method, hyperbolic conservation law, finite volume method, high-order reconstruction.

1 Introduction

When solving hyperbolic PDEs

$$\begin{aligned} &u_t + f(u)_x = 0 \\ &u(x,0) = u_0(x), \end{aligned} \tag{1}$$

where $x \in \Omega_c \subset \mathbb{R}^d$, $t > 0$, the solution will typically contain smooth parts, as well as moving shocks and contact discontinuities. Therefore, a challenging task is to construct methods resolving these discontinuities. A common approach is to use a very fine, equidistant mesh together with a high-resolution method. However, this will lead to over-resolution of the solution in the smooth regions, and numerical computations may be far too costly to be feasible, especially in several space dimensions. To remedy this problem, *adaptivity* can be used to increase the solution accuracy where the errors are expected to be large (like where the solution has large gradients, e.g. around discontinuities).

One successful adaptive strategy is to use *moving mesh methods* (also called r-methods). These methods have one thing in common; the grid points are moved to those regions in the computational domain Ω_c where they are most needed, thus keeping the number of mesh points constant. Many authors have been doing research on these methods, e.g. [4, 5, 6, 7, 9, 12, 13, 15, 18, 19, 21, 29, 30, 31, 32] during the last years.

Tang and Tang [32] investigated a moving mesh method in which the finite volume time evolution of the PDE (1) was decoupled from the mesh moving part. This is an advantageous approach; thus one can choose between several different high-resolution finite volume methods for solving hyperbolic PDEs. Also, Tang and Tang developed a formula for calculating the finite volume averages on the adapted meshes from previously known data. This formula is *conservative* in the sense that it preserves the total amount of the conserved variables over the mesh from one moving mesh iteration to another.

One common drawback of moving mesh methods is that the possibly large differences in cell sizes resulting from the grid movement may impose a severe restriction on the time step size during the solution process, due to the CFL condition. Therefore, one must take great care when choosing parameters in the adaptive algorithms for the mesh movement. In [30], Tang and Tang's method was combined with local time stepping in order to deal with this disadvantage.

Another problem arises; most finite volume methods are designed for uniform meshes and therefore cannot be applied to the method from [32]. Mostly, to solve this problem one transforms the adapted grid to an equidistant one (e.g. [30]). However, this requires that a smooth grid mapping can be found such that solution accuracy will not get lost during the transformation. Also, the transformation of the equation (1) results in a more complex Riemann problem that can be difficult to solve, since the well-known, high-performing approximate Riemann solvers may be inapplicable.

In this paper, we combine the method of Tang and Tang with the third order bi-hyperbolic finite volume method for quadrilateral meshes from [25], thus avoiding the transformation of the hyperbolic equations. Also, we derive a third order extension of the conservative update formula of Tang and Tang. We show that the PHM methods LHR and LHHR originally presented by Marquina [23], the Power-PHM by Serna [27] and the locally logarithmic methods by Artebrant and Schroll [1, 2, 3] can be applied to non-uniform, adapted meshes in one and two space dimensions, yielding good computational results.

The outline of this paper is as follows. In Section 2, the moving mesh method will be discussed, followed by an investigation of the finite volume method in Section 3. The algorithm will be extended to two space dimensions in Section 4, and numerical experiments are presented in Section 5. Finally, Section 6 concludes the paper.

2 The Moving Mesh Method

2.1 The Equation Controlling the Mesh Movement

We apply the following variational approach for the mesh movement (see [20, 32] for more information on this topic). Let x be a point in the computational domain Ω_c

and ξ be a corresponding point in a logical domain Ω_l. Both Ω_c and Ω_l are subsets of $\mathbb{R}^d$ and the grid points in Ω_l are uniformly distributed. The goal is to find a map $x = x(\xi)$ from Ω_l to the physical domain Ω_c, and to use this mapping for the mesh adaption. The map is given as the Euler-Lagrange equation of the functional

$$E(\bar{\xi}) = \frac{1}{2} \sum_k \int_{\Omega_p} \nabla \xi_k^T G_k^{-1} \nabla \xi_k \, d\bar{x} \ ,$$

namely

$$\nabla \cdot (G_k^{-1} \nabla \xi_k) = 0 \ , \quad 1 \leq k \leq d \ . \tag{2}$$

The G_k are symmetric, positive definite matrices called *monitor functions*. In this paper, we always apply monitor functions of the form

$$G_k = \omega I \ , \quad 1 \leq k \leq d \ , \tag{3}$$

where I is the identity matrix of appropriate size and ω is a positive weight function depending only on the solution u of (1) and its gradients.

As can be expected from (3), the choice of ω will greatly influence the properties of the mesh movement. Typically, one chooses ω large where a dense grid is needed. See [8, 14, 29] for further discussions on the choice of different monitor functions and their properties.

2.2 Solving the Mesh Movement Equation

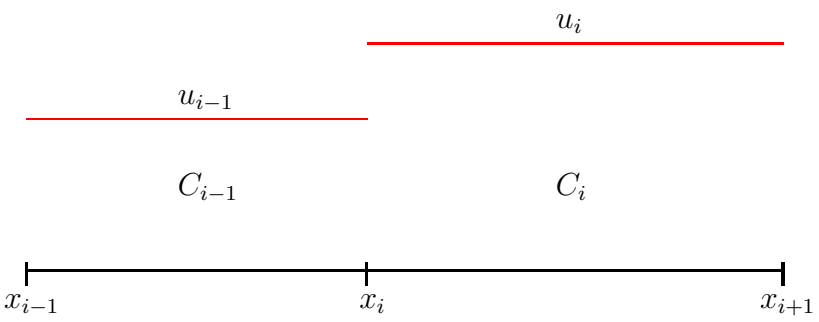

Figure 1: Grid cell notation.

In one space dimension, (2) combined with our simple monitor functions becomes $(\omega^{-1}\xi_x)_x = 0$. Thus, one can derive the equidistribution principle $(\omega x_\xi) = \text{constant}$. In practical computations, we solve the following, equivalent equation

$$(\omega x_\xi)_\xi = 0 \ , \tag{4}$$

to obtain mesh movement.

Let the problem domain Ω_c be divided into computational cells $C_i = [x_i, x_{i+1}]$, where $i = 0, \ldots, N-1$. The following Gauss-Seidel iteration is used for solving a discrete version of (4), thus moving the mesh points from x_i to $\tilde{x}_i$

$$\omega(u_i)(x_{i+1} - \tilde{x}_i) - \omega(u_{i-1})(\tilde{x}_i - \tilde{x}_{i-1}) = 0 \ . \tag{5}$$

This process will be repeated iteratively until the mesh reaches its equilibrium depending on the underlying solution u. In our numerical experiments, we have run the iterations until

$$\max_i |x_i - \tilde{x}_i| < \varepsilon \ , \quad 0 < \varepsilon \ll 1 \ ,$$

but at most five iterations, however.

After each iteration, the averages $\tilde{u}_i$ of the new mesh $\tilde{x}_i$ needs to be computed from the previously known u_i. In [32], the following formula was introduced for this average update:

$$\Delta\tilde{x}_i\tilde{u}_i = \Delta x_i u_i - ((cu)_{i+1} - (cu)_i) \ , \tag{6}$$

where $c_i = x_i - \tilde{x}_i$, and $\Delta x_i = vol(C_i) = (x_{i+1} - x_i)$. The border flux terms cu are approximated by upwinding

$$(cu)_i = \frac{c_i}{2}(u_i^+ + u_i^-) - \frac{|c_i|}{2}(u_i^+ - u_i^-) \ . \tag{7}$$

The solutions u_i^+ and u_i^- in the point x_i from the right and left side, respectively, can be reconstructed according to the procedure outlined in Section 3.1.

The formula (6) is second order accurate in c, but as long as $|c_i| \ll \min(\Delta x_{i-1}, \Delta x_i)$, we will expect that the use of (6) will not destroy the order properties of a third order finite volume method. However, following the approach of Tang and Tang, it is possible to derive a conservative, third order update formula. This is accomplished as follows.

Consider the integral denoting the unknown total amount of a conserved variable u in a cell $\tilde{C}_i$. Use the ansatz $\tilde{x} = x - c(x)$ and find

$$\int_{\tilde{x}_i}^{\tilde{x}_{i+1}} u(\tilde{x}) \ d\tilde{x} = \int_{x_i}^{x_{i+1}} u(x - c(x))(1 - c_x(x)) \ dx \ .$$

Expand u:

$$u(x - c(x)) \approx u(x) - c(x)u_x(x) + \frac{1}{2}c^2(x)u_{xx}(x) \ .$$

Then

$$u(x - c(x))(1 - c_x(x)) \approx u - cu_x + \frac{1}{2}c^2u_{xx} - c_xu + cc_xu_x - \frac{1}{2}c^2c_xu_{xx} =$$

$$= u - (cu)_x + \frac{1}{2}(c^2u_x)_x + \mathcal{O}(c^3) \ .$$

Thus,

$$\int_{\tilde{x}_i}^{\tilde{x}_{i+1}} u(\tilde{x}) \ d\tilde{x} \approx \int_{x_i}^{x_{i+1}} u(x) - (cu)_x + \frac{1}{2}(c^2u_x)_x \ dx \ ,$$

and we get the conservative update formula

$$\Delta\tilde{x}_i\tilde{u}_i = \Delta x_i u_i - ((cu)_{i+1} - (cu)_i) + \frac{1}{2}((c^2u_x)_{i+1} - (c^2u_x)_i) \ . \tag{8}$$

for finding the sought average $\tilde{u}_i$. This formula is a third order extension of (6) and (7) may be reused to determine the border fluxes. In the new, second order term an approximation for the gradient of u is needed. These gradients are approximated in second order of accuracy during the reconstruction, see Section 3.1 below.

3 The Finite Volume Method

We solve the hyperbolic conservation law

$$u(x,t)_t + f(u(x,t))_x = 0$$

by approximating the time derivative of the averages at a time t^n

$$\frac{d}{dt}u_i(t^n) = -\frac{1}{\Delta x}\left(f(u(x_i,t^n)) - f(u(x_{i+1},t^n))\right) \ , \tag{9}$$

where the averages are

$$u_i(t^n) = \frac{1}{\Delta x_i}\int_{C_i} u(x,t^n)\ dx \ , \tag{10}$$

and then integrating (9) to get the new averages at the next time point t^{n+1}. We will use the third order SSP Runge-Kutta method described in [10] for the time integration.

The fluxes $f(u(x_i,t^n))$ and $f(u(x_{i+1},t^n))$ will be replaced with some approximate solver for Riemann problems. To get third order of accuracy in the flux balance (9) and hence in the solutions u_i we need to *reconstruct* the point value border fluxes from the known cell averages f_i and f_{i+1}. This will be explained in the next subsection.

3.1 Reconstruction on Non-uniform Meshes

We use the third order hyperbolic method outlined in [25] for the reconstructions. It is an extension of the Piecewise Hyperbolic Method (PHM) by Marquina [23], which uses hyperbolas

$$r_i(x) = a_i + \frac{b_i}{(x - x_{i+1/2}) + c_i} \ , \quad x \in [x_i, x_{i+1}] \tag{11}$$

as ansatz functions. Either of the algorithms LHR, LHHR or the LHPR (by Serna [27]) determines the parameters a_i, b_i and c_i cell wise from the solution average

$$u_i = \frac{1}{\Delta x_i}\int_{C_i} u(x)\ dx = \frac{1}{\Delta x_i}\int_{C_i} r_i(x)\ dx, \tag{12}$$

and from the derivatives at the cell boundaries

$$r_i'(x_i) = d^- \ , \quad r_i'(x_{i+1}) = d^+ \ . \tag{13}$$

On uniform meshes, one gets the derivatives d^+ and d^- easily by central differencing. However, since the algorithms require that d^+ and d^- are known up to second order accuracy, this approach will not work in the general case when the cells are of different sizes. In [25], it was shown that the derivatives can be approximated as a linear combination of three adjacent cell averages

$$\alpha_i^+ u_{i+1} + \beta_i^+ u_i + \gamma_i^+ u_{i-1} = d_i^+ + \mathcal{O}(h^2) \ , \tag{14}$$

and

$$\alpha_i^- u_{i+1} + \beta_i^- u_i + \gamma_i^- u_{i-1} = d_i^- + \mathcal{O}(h^2) \ , \tag{15}$$

where $h = \max(\Delta x_{i+1}, \Delta x_i, \Delta x_{i-1})$. The coefficients α, β and γ are the solutions of the linear systems

$$\begin{pmatrix} 1 & 1 & 1 \\ \frac{\Delta x_{i+1}}{2} & -\frac{\Delta x_i}{2} & -\left(\Delta x_i + \frac{\Delta x_{i-1}}{2}\right) \\ \Delta x_{i+1}^2 & \Delta x_i^2 & 3\Delta x_i^2 + 3\Delta x_i \Delta x_{i-1} + \Delta x_{i-1}^2 \end{pmatrix} \begin{pmatrix} \alpha_i^+ \\ \beta_i^+ \\ \gamma_i^+ \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

and

$$\begin{pmatrix} 1 & 1 & 1 \\ \frac{\Delta x_{i+1}}{2} + \Delta x_i & -\frac{\Delta x_i}{2} & -\frac{\Delta x_{i-1}}{2} \\ 3\Delta x_i^2 + 3\Delta x_i \Delta x_{i+1} + \Delta x_{i+1}^2 & \Delta x_i^2 & \Delta x_{i-1}^2 \end{pmatrix} \begin{pmatrix} \alpha_i^- \\ \beta_i^- \\ \gamma_i^- \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \ .$$

In computational cells containing local extrema of the solution, so called transition cells, the order of the hyperbolic methods drop to two [23, 24, 27]. Therefore, instead of using hyperbolic functions, a successful reconstruction ansatz is logarithmic functions [1, 3]

$$r_i^L(x) = A + B\log(x + C) + D\log(x + E) \ ,$$

yielding the LDLR reconstruction procedure. It has the same requirements on average (12) and derivatives (13) as the hyperbolic reconstructions, and the linear combinations (14) and (15) can be used to obtain the cell border derivatives. The LDLR method is also a formally third order method, but its numerically observed order is higher than that for the hyperbolic methods since it can reconstruct *isolated* local extreme values within a cell. We will employ both the LDLR and the hyperbolic schemes in our numerical tests (Section 5).

4 Extension to Two Space Dimensions

In this Section, we will show how to extend the solution procedure to two space dimensions. Our goal is now to find a map $x = x(\xi, \eta)$ and $y = y(\xi, \eta)$ from the variables (ξ, η) in a logical domain, and then to apply the high-resolution finite volume method from [25] on the adapted meshes.

The two-dimensional bi-hyperbolic PDE solver from [25] works as follows. The conservation law

$$\frac{\partial U(x,y,t)}{\partial t} + \nabla \cdot F(U(x,y,t)) = 0$$

will be solved by time stepping of its average derivative for all cells C_{ij}

$$\frac{d}{dt} U_{ij}(t) = -\frac{1}{vol(C_{ij})} \sum_{p=1}^{k} \int_{\gamma_p} F_p(U(x,y,t)) \cdot \mathbf{n}_p \, dS \ . \tag{16}$$

Here, the flux integration is carried out over each of the k linear cell edges ($\partial C_{ij} = \gamma_1 + \ldots + \gamma_k$). Hence, the fluxes on all the boundaries γ_p need to be reconstructed for the solution. The bi-hyperbolic algorithm calculates six weights $\mu_{(0,k)}, \ldots, \mu_{(5,k)}$ for each of the k boundaries in every cell C_{ij} such that one gets the first order derivatives of the fluxes to be reconstructed as

$$\sum_l \mu_{(l,k)} F_l^{av} = F_k' + \mathcal{O}(h^2) \ ,$$

where F^{av} are the averages of the cell fluxes and $h = \max_{p,q \in C_{ij}} \|p-q\|$. The weights $\mu_{(l,k)}$ are computed from integral expressions depending on the mesh geometry. On moving meshes, the weights are dynamic and need to be updated after each mesh iteration. A detailed description of the procedure of calculating the weights can be found in [25].

The bi-hyperbolic reconstruction function is extended from (11) as follows:

$$r_{ij}(x,y) \quad = \quad a_{ij} + \frac{b_{ij}}{(x - x_{ij}^m) + c_{ij}} + \frac{d_{ij}}{(y - y_{ij}^m) - e_{ij}}, \ (x,y) \in C_{ij}. \qquad (17)$$

The variables x_{ij}^m and y_{ij}^m are the coordinates of the cell mid points. With this ansatz, the calculation of the coefficients $a_{ij}, \ldots, e_{ij}$ can be applied separately in x- and y-dimensions using the original one-dimensional algorithms LHR, LHHR or LHPR.

If applying the moving mesh algorithm of Tang and Tang directly, the resulting grid will be fully adapted to the underlying solution, and we will have difficulties applying the bi-hyperbolic method. Although it was designed for general meshes in two space dimensions, after every moving mesh step one must recalculate the dynamic weights $\mu_{(l,k)}$ for every computational cell C_{ij}. This is very costly when implemented numerically, and the benefit from using adaption compared to using higher order, non-adaptive methods directly on very fine, equidistant meshes is lost.

Therefore, we will simplify the calculations by restricting the adaption mapping to $x = x(\xi)$ and $y = y(\eta)$. This will move an entire line of coordinates at the same speed, thus making all cells *rectangular* (but still non-uniform). In this way, the calculation of the coefficients $\mu_{(l,k)}$ can be made much simpler, and the cell boundary integration (16) becomes trivial. One drawback is that in two space dimensions, some computational regions having smooth solutions (those being aligned with regions containing large discontinuities) will be too highly resolved. However, numerical experiments have shown that this is still an advantageous approach compared to applying the bi-hyperbolic method directly on general grids. See Section 5.2 for 2D mesh examples.

The two dimensional mesh movement is handled as follows. Let the problem domain Ω_c be divided into rectangular computational cells $C_{ij} = [x_j, x_{j+1}] \times [y_i, y_{i+1}]$, $i = 0, \ldots, N_y - 1$, $j = 0, \ldots, N_x - 1$. We employ the coordinate-wise weight functions

$$\omega_j^{(x)} = \max_i \omega^{(x)}(U_{ij})$$

and

$$\omega_i^{(y)} = \max_j \omega^{(y)}(U_{ij}).$$

The mesh movement will now be handled by applying (5) with the above weight functions separately in each coordinate direction.

From [32] we get the second order update formula for the averages in 2D

$$\iint_{\tilde{C}_{ij}} U(\tilde{x},\tilde{y})\ d\tilde{x}\tilde{y} = \iint_{C_{ij}} U(x,y)\ dxdy - \int_{\partial C_{ij}} (c^x n_x + c^y n_y) U(x,y)\ dxdy. \quad (18)$$

corresponding to (6). Here, $\mathbf{n} = (n_x, n_y)$ is the unit length normal and $(c^x, c^y) = (x - \tilde{x}, y - \tilde{y})$. All terms of order $|c|^2$ or higher have been neglected. Rewriting (18) yields

$$vol(\tilde{C}_{ij})\tilde{U}_{ij} = vol(C_{ij})U_{ij} - \int_{\partial C_{ij}} (c^x n_x + c^y n_y) U(x,y)\ dxdy.$$

Since the cells are kept rectangular during mesh adaption, the integral above is straight forward to compute exactly when using the reconstruction ansatz (17). Thus, no order reduction occurs from the integration.

In principle, it is possible to derive a third order version of (18), but it is necessary to introduce some additional constraints. Let $\mathbf{c} = (c^x, c^y)$ denote the local mesh movement and make the change of variables

$$\begin{aligned} \tilde{x} &= x - c^x \\ \tilde{y} &= y - c^y \end{aligned}$$

in the integral of the total amount of u in $\tilde{C}_{ij}$

$$\iint_{\tilde{C}_{ij}} u(\tilde{x},\tilde{y})\ d\tilde{x}d\tilde{y} = \iint_{C_{ij}} u(x - c^x, y - c^y)((1 - c^x_x)(1 - c^y_y) - c^y_x c^x_y)\ dxdy\ .$$

Expand u up to second order:

$$\begin{aligned} u(x - c^x, y - c^y) &\approx u(x,y) - u_x(x,y)c^x - u_y(x,y)c^y + \\ &+ \frac{1}{2}u_{xx}(x,y)(c^x)^2 + u_{xy}(x,y)c^x c^y + \frac{1}{2}u_{yy}(x,y)(c^y)^2 = \\ &= u - c^x u_x - c^y u_y + \frac{1}{2}(c^x)^2 u_{xx} + c^x c^y u_{xy} + \frac{1}{2}(c^y)^2 u_{yy} \end{aligned}$$

Multiplying this expansion with the determinant of the Jacobian of the change of variables $\det J$ yields

$$\begin{aligned} u(x - c^x, y - c^y) \cdot \det J = u - (c^x u)_x - (c^y u)_y + \left(\frac{1}{2}(c^x)^2 u_x\right)_x + \left(\frac{1}{2}(c^y)^2 u_y\right)_y + \\ + c^x c^y u_{xy} + c^x c^y u_y + c^y_y c^x u_x + c^x_x c^y_y u - c^y_x c^x_y u\ , \end{aligned}$$

neglecting terms of order c^3 or higher. Assuming now that $c^x = c^x(x)$ and $c^y = c^y(y)$, i.e. the mesh is rectangular, we see that

$$(c^x c^y u)_{xy} = c^x_x c^y_y u + c^x_x c^y u_y + c^x c^y_y u_x + c^x c^y u_{xy}\ ,$$

and $c_x^y c_y^x u = 0$. Thus,

$$\iint_{\tilde{C}_{ij}} u(\tilde{x},\tilde{y})\ d\tilde{x}d\tilde{y} = \iint_{C_{ij}} u\ dxdy - \iint_{C_{ij}} \nabla\cdot(\mathbf{c}u)\ dxdy +$$

$$+\frac{1}{2}\iint_{C_{ij}} \nabla\cdot\left((c^x)^2u_x,(c^y)^2u_y\right)dxdy + \iint_{C_{ij}}(c^xc^yu)_{xy}\ dxdy\ .$$

Finally, by the Divergence Theorem;

$$vol(\tilde{C}_{ij})\tilde{u}_{ij} = vol(C_{ij})u_{ij} - \int_{\partial C_{ij}}(\mathbf{c}u)\cdot\mathbf{n}\ ds +$$

$$\frac{1}{2}\int_{\partial C_{ij}}\left((c^x)^2u_x,(c^y)^2u_y\right)\cdot\mathbf{n}\ ds + \iint_{C_{ij}}(c^xc^yu)_{xy}\ dxdy\ ,$$

where $\mathbf{n}$ is the outward normal of the boundary ∂C_{ij}. The expression above is computed numerically by replacing u with r_{ij} from (17) and using the upwind formula (7) in the first integral.

5 Numerical Experiments

In this Section, we will test the combination of the hyperbolic (or logarithmic) methods [25] and the moving mesh method [32], using both second and third order average update procedures. We implement the solution algorithm by Tang and Tang with one modification: during the very first mesh generation, we do not use any of the average update formulae after moving the mesh one iteration. This is unnecessary, since before any time evolution of the equation, the solution is known from the initial data and we can simply calculate the values of $\tilde{u}_i$ or $\tilde{U}_{ij}$ by re-averaging the initial data according to the new mesh. We run a great many more iterations of (5) during the first mesh generation than during the later ones, in order to make sure that the grid manages to adjust itself to the discontinuities properly. This approach saves some computational time and helps keeping the solution errors small.

5.1 One Dimensional Problems

Example 1: We solve the linear advection problem (with periodic boundary conditions)

$$u_t + u_x = 0, \quad -1 < x < 1, \quad u(x,0) = u_0(x)$$

with

$$u_0(x) = \begin{cases} \frac{1}{6}(G(x,\beta,z-\delta)+G(x,\beta,z+\delta)+4G(x,\beta,z)) & -0.8 \le x \le -0.6, \\ \frac{1}{6}(F(x,\alpha,a-\delta)+F(x,\alpha,a+\delta)+4F(x,\alpha,a)) & 0.4 \le x \le 0.6, \\ 0 & \text{otherwise.} \end{cases}$$

Also,

$$G(x,\beta,z) = e^{-\beta(x-z)^2}$$

and

$$F(x,\alpha,a) = \sqrt{\max(1-\alpha^2(x-a)^2, 0)}\ ,$$

with parameters $a = 0.5$, $z = -0.7$, $\delta = 0.005$, $\alpha = 10$ and $\beta = \ln 2/(36\delta^2)$. This initial data is taken from Jiang and Shu [16], but we only include the Gaussian and the ellipse. We choose not to use the same problem setup as in [16], since in the original problem, the number of movable grid points in the smooth regions are too few to properly cover all the discontinuities, making mesh movement not so efficient.

We run simulations up to $t = 2$ s with CFL $= 0.3$ using all the four previously mentioned reconstruction procedures for finding the average (10), together with the well-known Lax-Friedrichs numerical flux as approximate Riemann solver. We use $N = 100$ and the monitor function $G = \omega I$, $\omega = \sqrt{1 + 0.25|u_{xx}|}$. This monitor is chosen such that one achieves a sufficient grid clustering near the corners of the solution. No discontinuities are present, so we have omitted any dependence of the first derivative of the solution in the monitor function. A monitor of similar type was used in [33] for the Hamilton-Jacobi equations, however also including the gradient.

The simulation result at $t = 0.1$ s is shown in Figure 2 with a blowup in Figure 4, and at $t = 2$ s in Figure 5. See Figures 3 for the mesh movement in time. In the Figure, we observe the clustering of mesh points near the corners, as desired.

We have employed a simple second order MUSCL-reconstruction based on the minmod-limiter for comparison against the third order methods. It uses the following estimation of u^+ and u^- in the upwinding formula (7):

$$u_i^+ = u_i + \frac{1}{2}(x_i - x_{i+1})s_i, \quad u_i^- = u_{i-1} + \frac{1}{2}(x_i - x_{i-1})s_{i-1},$$

where

$$s_i(s^+, s^-) = \text{sign}(s^-) \cdot \max(0, \min(|s^-|, \text{sign}(s^-) \cdot s^+))$$

and

$$s^+ = \frac{u_{i+1} - u_i}{0.5 \cdot (x_{i+2} + x_i)}, \quad s^- = \frac{u_i - u_{i-1}}{0.5 \cdot (x_{i+1} + x_{i-1})}.$$

See Table 1 for a comparison of errors in the numerical simulations of this problem. Obviously, adaptivity greatly increases the accuracy of all the methods. Also, we see that the difference between second and third order average updating during the mesh movement phase is not very large, but in all the simulations, the third order update performs slightly better, especially in the maximum norm. We note that although improving the simulation results, the second order adaptive MUSCL cannot compete with the third order methods.

Example 2: In this example, we solve the following problem:

$$u_t + u_x = 0, \quad x \in [-4, 4],$$

with the initial data

$$u(x,0) = 1 - e^{-\frac{1}{1+x^8}}.$$

It is easily verified that it has the analytical solution

$$u(x,t) = 1 - e^{-\frac{1}{1+(x-t)^8}},$$

	L^1-error	L^∞-error
LHR		
no adaption	0.0209	0.1827
2nd order average update	0.0052	0.1023
3rd order average update	0.0052	0.0712
LHHR		
no adaption	0.0214	0.1685
2nd order average update	0.0054	0.1195
3rd order average update	0.0053	0.0884
LHPR		
no adaption	0.0227	0.2110
2nd order average update	0.0064	0.1045
3rd order average update	0.0061	0.0533
LDLR		
no adaption	0.0208	0.1949
2nd order average update	0.0056	0.1149
3rd order average update	0.0055	0.0731
MUSCL, minmod		
no adaption	0.0270	0.1927
2nd order average update	0.0115	0.1735
3rd order average update	—	—

Table 1: Errors in the numerical solutions of Example 1 at $t = 0.1$ s.

and can therefore be used for error analysis. We run the simulations using the LHPR scheme with Powermod3-limiter up to $t = 0.5$ s with CFL $= 0.3$, $\omega = \sqrt{1 + 25|u_{xx}|}$ and different numbers of grid cells N. Again, no discontinuities are present in the solution, but it contains sharp corners. The same type of weight function as in Example 1 is therefore used.

See Figure 6 for the solution, Figure 7 for the mesh movement in time and Table 2 for the simulation errors. The notation is the same as in the previous Example. The errors in the Table are plotted against N in Figures 8 and 9. Indubitably, adaptivity gives the best simulation results, and third order updates perform somewhat better than second order updates. It is worth remarking, that despite the moderate level of adaption (the fraction between the largest and the smallest cell size is not extremely large), computational errors are significantly reduced.

Example 3. We now turn our focus to Sod's problem [28], which is the one-dimensional Euler equations (for which we determine the pressure p using $p = (\gamma - 1)(E - \rho u^2/2)$)

$$\begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix}_t + \begin{pmatrix} \rho u \\ \rho u^2 + p \\ (E + p)u \end{pmatrix}_x = 0,$$

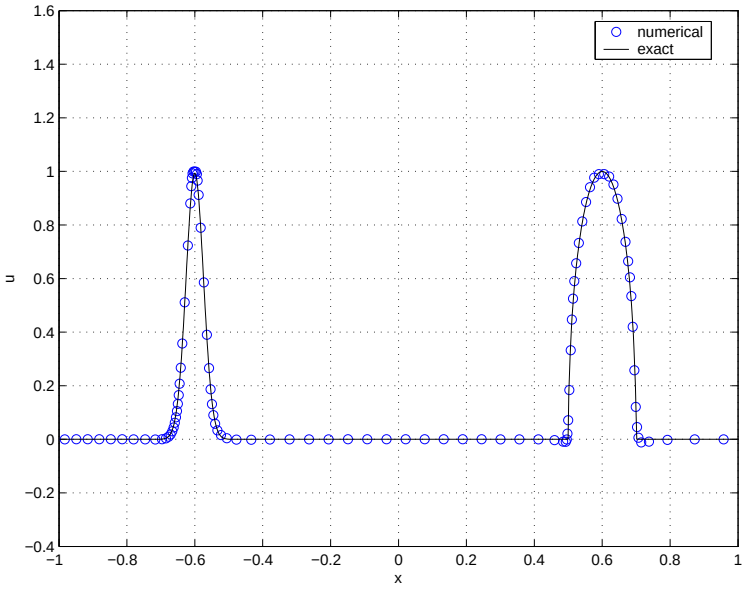

Figure 2: The advection problem (Example 1) at $t = 0.1$ s computed with LHR-reconstruction and third order average update.

	No adaption		2nd order average update		3rd order average update	
N	L^1-error	L^∞-error	L^1-error	L^∞-error	L^1-error	L^∞-error
10	0.3698	0.1400	0.2992	0.0902	0.3052	0.0961
20	0.0747	0.0488	0.0727	0.0379	0.0770	0.0410
40	0.0450	0.0421	0.0213	0.0155	0.0184	0.0122
80	0.0334	0.0342	0.0120	0.0130	0.0101	0.0108
160	0.0184	0.0160	0.0074	0.0079	0.0059	0.0067
320	0.0094	0.0082	0.0045	0.0046	0.0035	0.0036
640	0.0047	0.0040	0.0023	0.0024	0.0021	0.0019

Table 2: Solution errors at $t = 0.5$ s in Example 2.

coupled with the initial data

$$(\rho, u, p) = \begin{cases} (1, 1, 1) & 0 \le x < 0.5 \\ (0.125, 0, 0.1) & 0.5 \le x \le 1. \end{cases}$$

This challenging problem have been tested in moving mesh frameworks by several authors, see e.g. [29, 32].

In Figure 10 the result of the simulation at $t = 0.2$ s is shown. We used 3rd order average update based on the LHPR scheme with CFL $= 0.3$ and 80 grid cells. Since the solution will contain three components, one has to decide which of these variables the monitor function should depend on. It may be inappropriate to simply let ω be a function of $\|(\rho, \rho u, E)\|$ since the values of the entities are probably of different magnitudes. Reference [29] contains a further discussion on this. Still, we need to find a monitor that can adapt the mesh to the discontinuities, rarefactions

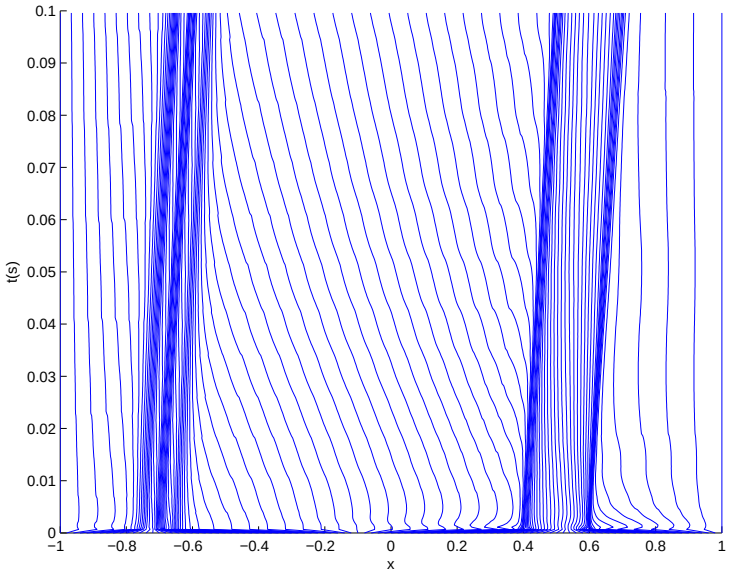

Figure 3: The mesh movement in time of Example 1 up to $t = 0.1$ s. Third order average update was used.

and other types of behaviour in all the variables. For Sod's problem we found that a weight function being dependent on the density ρ and entropy $s = p/\rho^\gamma$ accomplishes this (this choice of ω was inspired by the Sod simulation in [32]). Therefore, we use

$$\omega = \sqrt{1 + \varsigma|u_\xi| + \varsigma|s_\xi|}.$$

and $\varsigma = 25$.

We see in Figure 10 that the method proposed in this paper behaves very well also for this nonlinear problem with discontinuous initial data. In Figure 12 we compare the higher order results with those of lower order and without adaptivity. We note that third order average update makes the slopes around the shock and contact discontinuity much steeper and more accurate than the other simulations.

5.2 Two Dimensional Problems

In this Section, we will test two-dimensional version of the presented method on the Euler equations of gas dynamics

$$U_t + F(U)_x + G(U)_y = 0,$$

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \quad F(U) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix}, \quad G(U) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}.$$

with the pressure $p = (\gamma - 1)(E - \frac{\rho}{2}(u^2 + v^2))$. We run all simulations for pure air, so $\gamma = 1.4$.

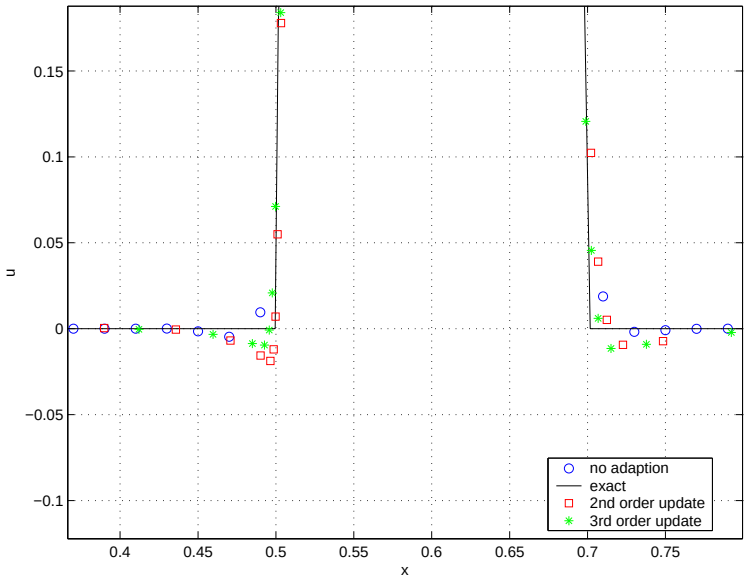

Figure 4: Blowup of solution at the bottom of the ellipse (Example 1).

EXAMPLE 4: (A rectangular explosion problem.) A rectangular section of high density is immersed in lower density air. The problem setup is

$$(\rho, u, v, p) = \begin{cases} (10, 0, 0, 5) & \text{if } 0.4 \leq x \leq 0.6 \text{ and } 0.4 \leq y \leq 0.6. \\ (1, 0, 0, 1) & \text{otherwise.} \end{cases}$$

The initial data are chosen such that both pressure and density are several times larger inside the rectangle than on the outside. The gas is at rest at $t = 0$, and thereafter it will spread out in all the four different directions yielding the nice simulation results in Figure 13. For this problem, we use

$$\omega^{(x)}(U) = \sqrt{1 + \varsigma|\rho_\xi|}, \quad \omega^{(y)}(U) = \sqrt{1 + \varsigma|\rho_\eta|}, \tag{19}$$

$\varsigma = 0.25$, and the LHPR-scheme on a mesh of size $N_x = N_y = 100$. We let CFL $= 0.5$. To detect discontinuities which characterize the solution of the nonlinear hyperbolic conservation laws like the Euler equations, we choose a weight related to gradients.

Both the second and third order average updates in 2D have been tested, and graphically we conclude that there is no significant difference between the results. We therefore only include the higher order version, see Figure 13 for the density contours of the solution at $t = 0.2$ s. Clearly, the moving mesh solution contains sharper discontinuities than the equidistant one, as expected.

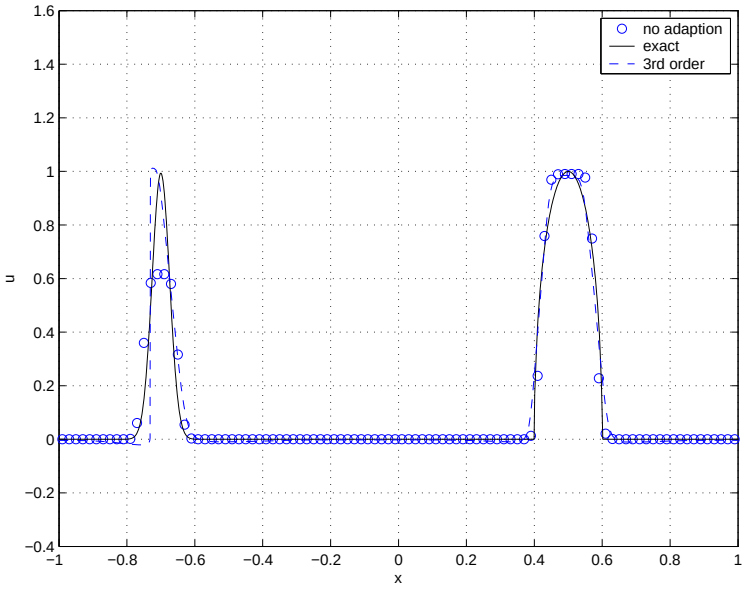

Figure 5: The advection problem (Example 1) at $t = 2$ s computed with LHHR-reconstruction.

Example 5: Riemann problem, Configuration 6 (see [17]), (also known as Configuration B [26]). This is a well-known test example with the initial data

$$(\rho, u, v, p) = \begin{cases} (1.0, 0.75, -0.5, 1) & x \geq 0.5, \quad y \geq 0.5 \\ (2.0, 1.5, 1.0, 1) & x \leq 0.5, \quad y \geq 0.5 \\ (1.0, -0.75, 0.5, 1) & x \leq 0.5, \quad y \leq 0.5 \\ (3.0, -2.25, -1.5, 1) & x \geq 0.5, \quad y \leq 0.5 \end{cases}.$$

This problem is solved with CFL $= 0.5$ up to $t = 0.3$ s with weight function (19) and $\varsigma = 5$ on a 100×100 mesh. See Figure 15 for the density contours with/without the moving mesh algorithm, and Figure 16 for the mesh at $t = 0.3$ s.

Example 6: Another Riemann problem, denoted Configuration 7 [17]. The problem setup of this experiment is

$$(\rho, u, v, p) = \begin{cases} (1, 0.1, 0.1, 1) & x \geq 0.5, \quad y \geq 0.5 \\ (0.5197, -0.6259, 0.1, 0.4) & x \leq 0.5, \quad y \geq 0.5 \\ (0.8, 0.1, 0.1, 0.4) & x \leq 0.5, \quad y \leq 0.5 \\ (0.5197, 0.1, -0.6259, 0.4) & x \geq 0.5, \quad y \leq 0.5 \end{cases}.$$

We employ the same mesh size and CFL-number as in the previous Riemann problem, but finish the computations at $t = 0.25$ s. We use the same monitor function as in the previous example. Again, we see a lot better resolution in the computations (Figure 17) using the adaptive schemes.

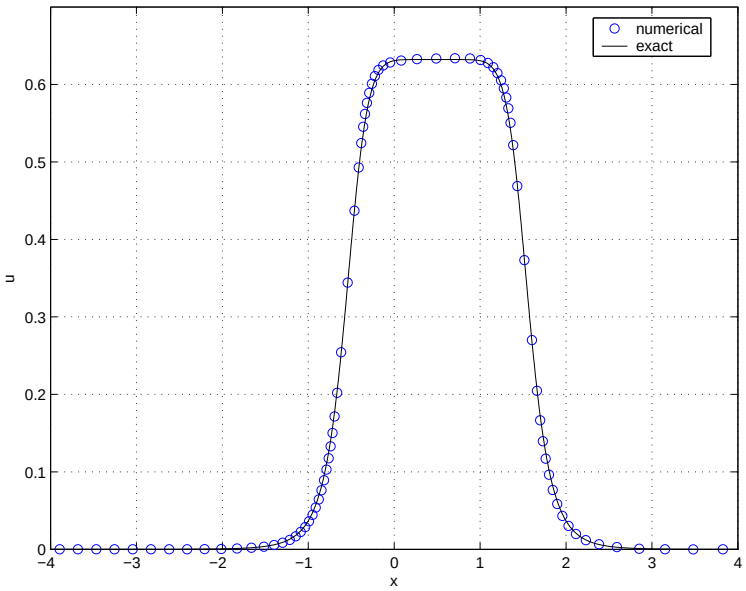

Figure 6: Solution of Example 2 ($N = 80$, adaption using third order update) at $t = 0.5$ s computed with the LHPR scheme.

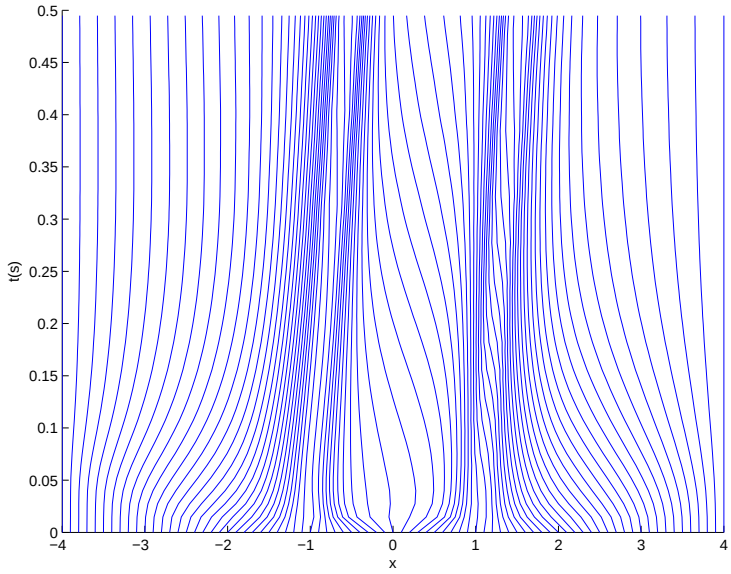

Figure 7: Time motion of the grid lines in Example 2.

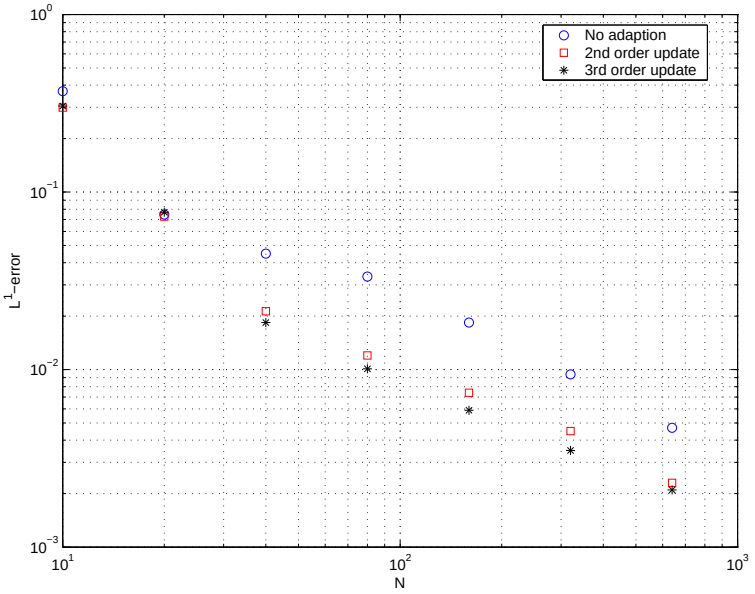

Figure 8: Solution errors in Example 2 at different mesh sizes in L^1-norm.

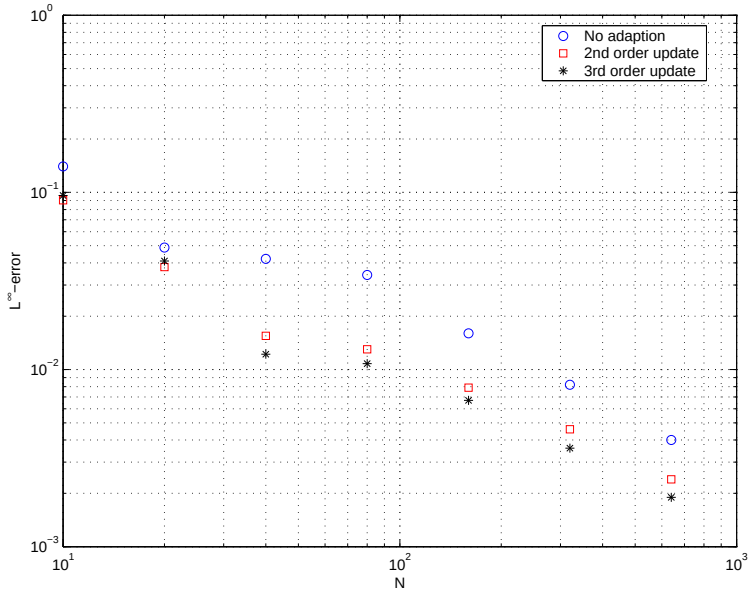

Figure 9: Solution errors in Example 2 at different mesh sizes in L^∞-norm.

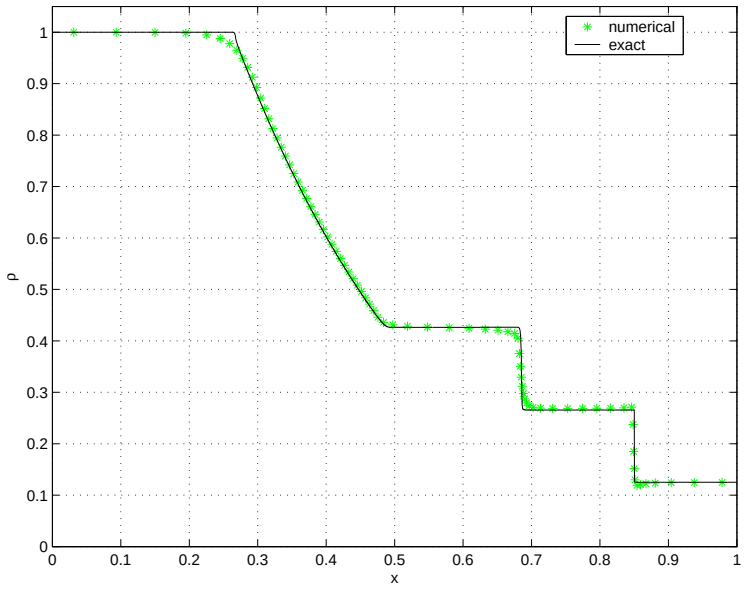

Figure 10: Density of the solution in Sod's problem computed with higher order average update and the LHPR scheme at $t = 0.2$ s.

We observe the same quality in the densities of Riemann problems 6 and 7 as those observed in [32], despite the fact that rectangular mesh adaption has been employed.

6 Conclusion

In this paper, we have combined the moving mesh method of Tang and Tang with the third order hyperbolic/logarithmic reconstruction method based on Marquina, such that all computations are performed directly in physical space, with no coordinate or equation transformation involved. We have developed third order extensions of the average update formulae introduced in [32], and shown that the new expressions yield good computational results. We note that the third order finite volume methods keep the advantage over a second order method when applied on moving meshes.

In one space dimension, we find that the third order average update formula yields slightly better simulation results than the second order one. However, in two space dimensions, the quality of the solutions obtained by either of the update formulae are the same, thus we deem that the second order average update by Tang and Tang will suffice for combination with the third order methods.

7 Acknowledgement

The author wishes to thank H.J. Schroll for his many useful comments and suggestions greatly improving this paper.

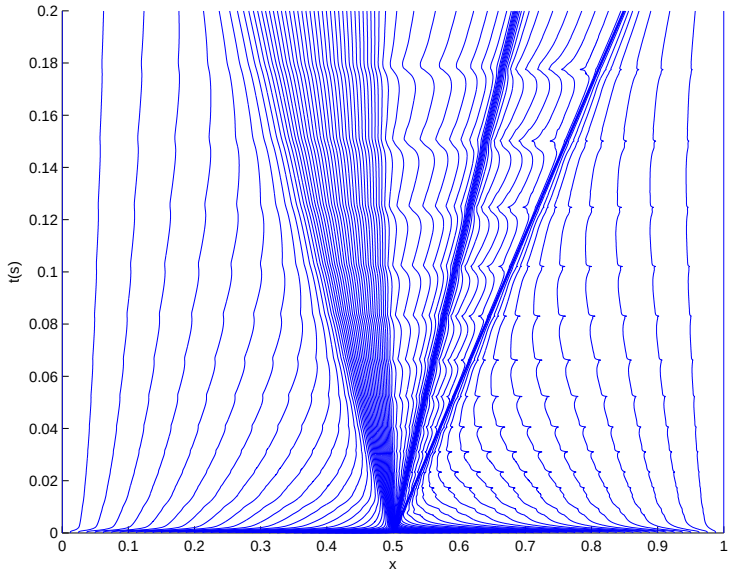

Figure 11: Mesh movement in time for Sod's problem.

References

[1] Artebrant R. (2006) *Third order accurate non-polynomial reconstruction on rectangular and triangular meshes*, J. Sci. Comp. , in press.

[2] Artebrant R. and Schroll H.J. (2005) *Conservative logarithmic reconstructions and finite volume methods*, SIAM J. Sci. Comput. **27**(1), pp 294–314.

[3] Artebrant R. and Schroll H.J. (2005) *Limiter-free third order logarithmic reconstruction*, SIAM J. Sci. Comput., to appear.

[4] Azarenok B. (2002) *Variational barrier method of adaptive grid generation in hyperbolic problems of gas dynamics*, SIAM J. Numer. Anal. **40**(2), pp 651–682.

[5] Azarenok B. and Tang T. (2005) *Second-order Godunov-type scheme for reactive flow calculations on moving meshes*, J. Comput. Phys. **206**, pp 48–80.

[6] Brackbill J.U. and Saltzmann J.S. (1982) *Adaptive zoning for singular problems in two dimensions*, J. Comput. Phys. **46**, pp 342–368.

[7] Bramkamp F., Lamby Ph. and Müller S. (2004) *An adaptive multiscale finite volume solver for unsteady and steady state flow computations*, J. Comput. Phys. **197**, pp 460–490.

[8] Cao W., Huang W. and Russell R. (2000) *A study of monitor functions for two-dimensional adaptive mesh generation*, SIAM J. Sci. Comput. **20**(6), pp 1978–1994.

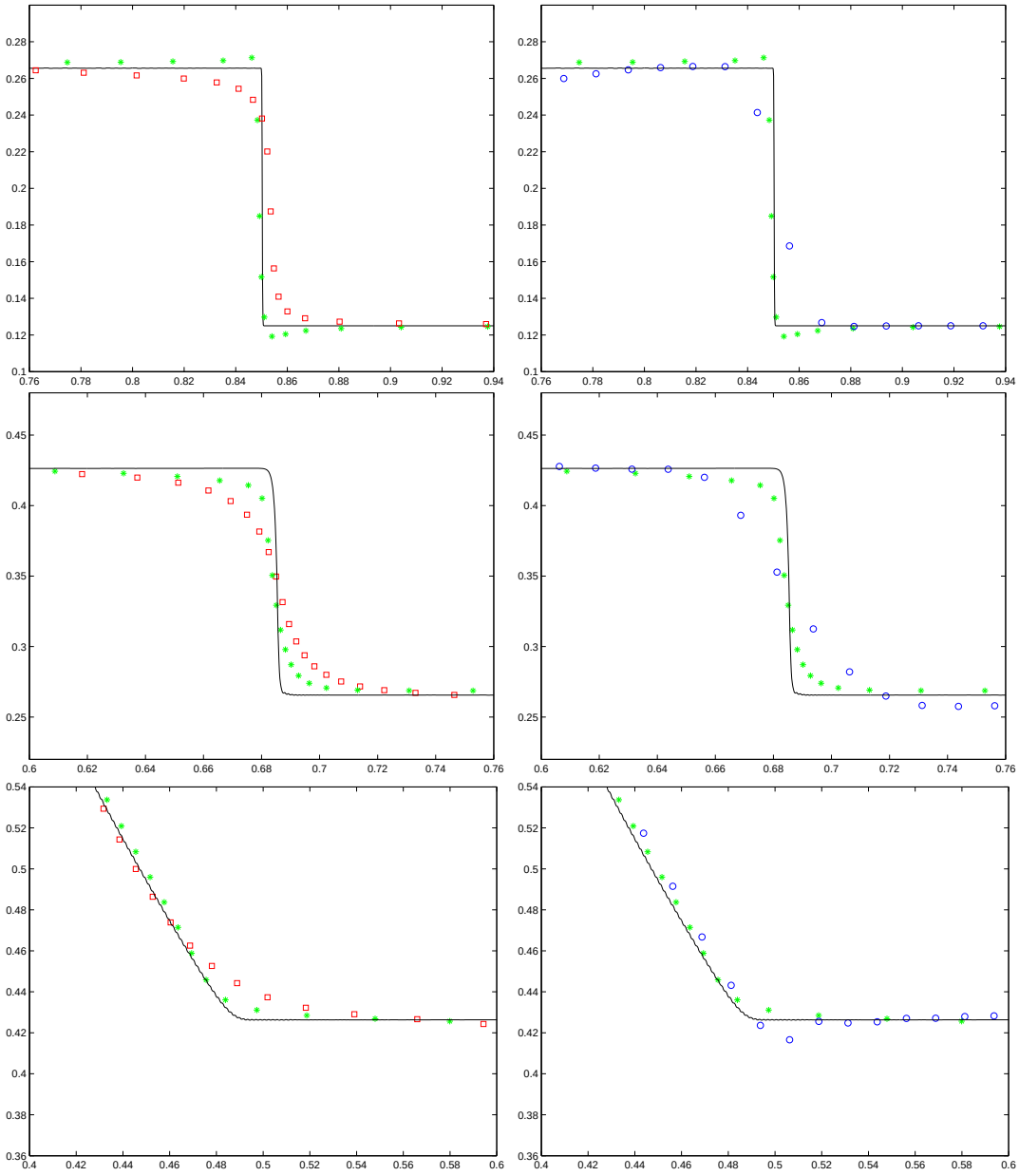

Figure 12: Blowup of the Sod problem solution densities at $t = 0.2$ s. 'o': LHPR without adaption, '□': minmod with second order average update, '*': LHPR with third order average update. Top figures: the shock, middle figures: the contact discontinuity, bottom figures: the rarefaction wave.

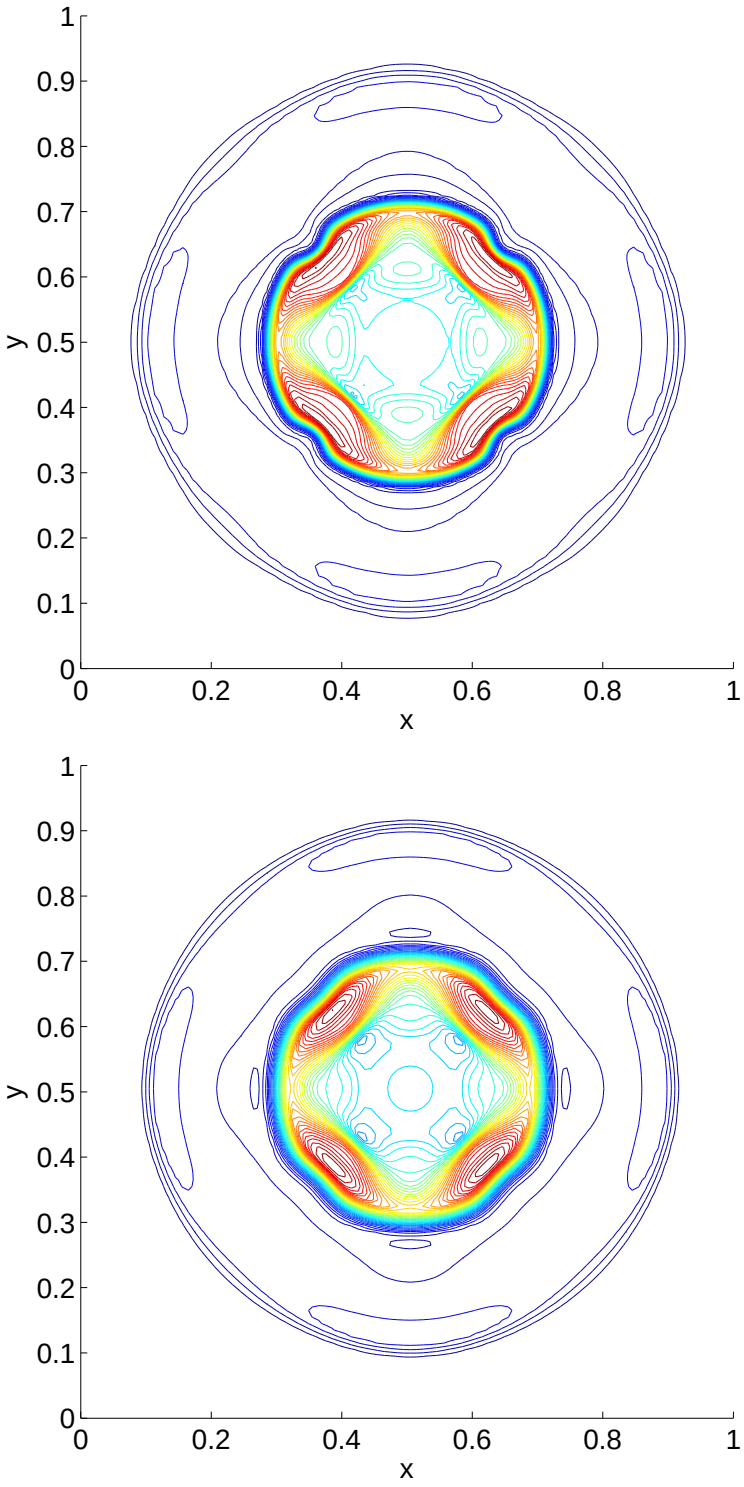

Figure 13: Density contours of the rectangle problem at $t = 0.20$ s. The result in the upper figure is computed using adaptivity (third order), in the lower without.

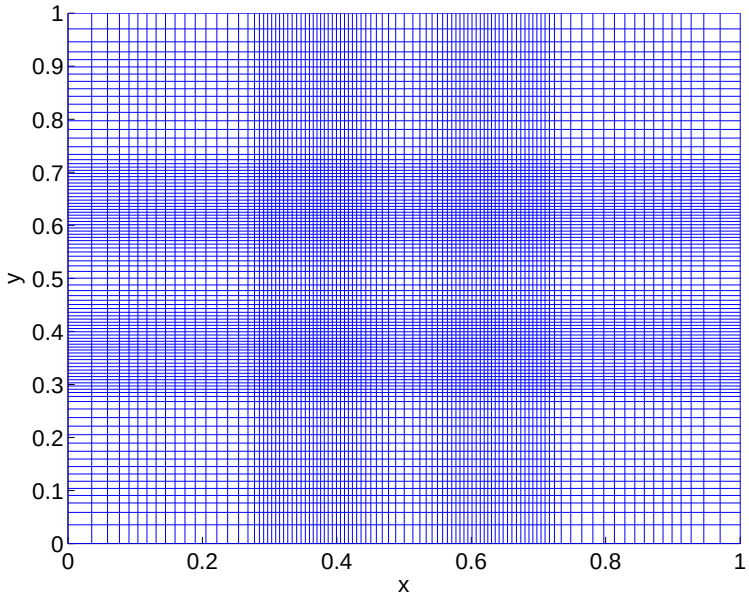

Figure 14: Adapted mesh of the rectangle problem at $t = 0.20$ s.

[9] Fazio R. and LeVeque R.J. (2003) *Moving-mesh methods for one-dimensional hyperbolic problems using CLAWPACK*, Comput. Math. Appl. **45**, pp 273–298.

[10] Gottlieb S., Shu C.W. and Tadmor E. (2001) *Strong Stability-Preserving high-order time discretization methods*, SIAM Review **43**(1), pp 89–112.

[11] Hall O., Schroll H.J. and Svensson F. (2005) *High-resolution simulation of inviscid flow in general domains*, Int. J. Numer. Meth. Fluids **47**, pp 1061–1067.

[12] Harten A. and Hyman J. (1983) *Self adjusting grid methods for one-dimensional hyperbolic conservation laws*, J. Comput. Phys. **50**, pp 235–269.

[13] Huang W., Ren Y. and Russell R. (1994) *Moving mesh methods based on moving mesh partial differential equations*, J. Comput. Phys. **113**, pp 279–290.

[14] Huang W. and Sun W. (2003) *Variational mesh adaption II: error estimates and monitor functions*, J. Comput. Phys. **184**, pp 619–648.

[15] Ivanenko A. and Azarenok B. (2002) *Application of moving adaptive grids for numerical solution of 2D nonstationary problems in gas dynamics*, Int. J. Numer. Meth. Fluids **39**, pp 1–22.

[16] Jiang G.-S. and Shu C.-W. (1996) *Efficient implementation of Weighted ENO schemes*, J. Comput. Phys. **126**, pp 202–228.

[17] Lax P. and Liu X.-D. (1998) *Solution of two-dimensional Riemann problems of gas dynamics by positive schemes*, SIAM J. Sci. Comput. **19**(2), pp 319–340.

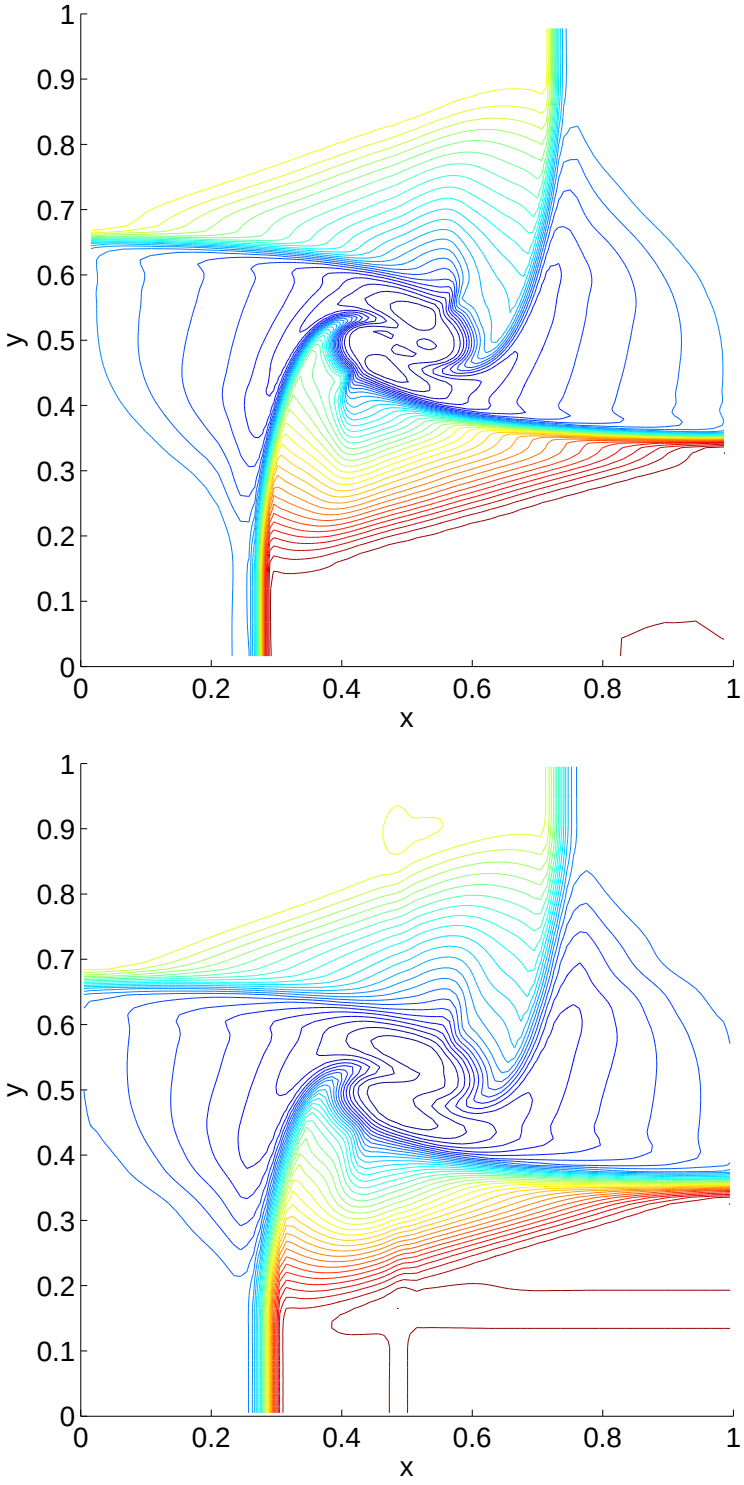

Figure 15: Density contours of Configuration 6 at $t = 0.30$ s. The upper solution is computed using adaptivity (third order), lower without.

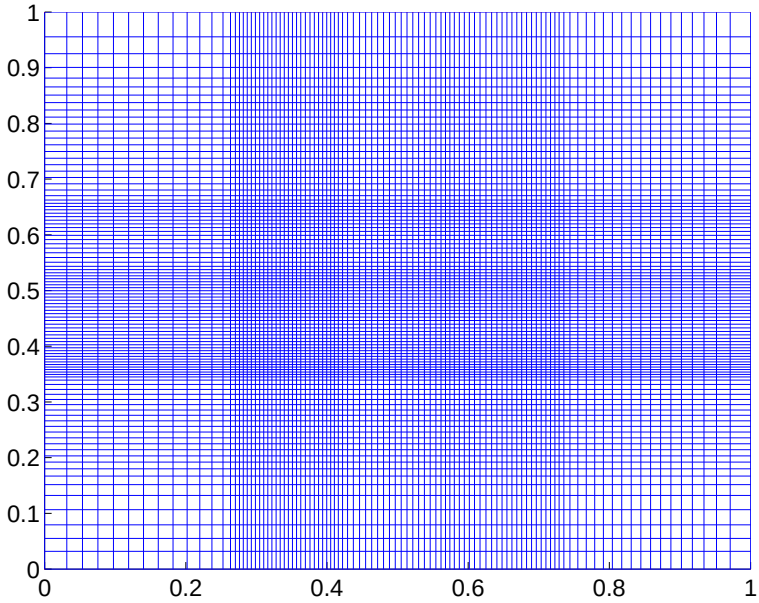

Figure 16: Mesh of Configuration 6 at $t = 0.30$ s.

[18] Li S. and Petzold L. (1997) *Moving mesh methods with upwinding schemes for time-dependent PDEs*, J. Comput. Phys. **131**, pp 368–377.

[19] Li R., Tang T. and Zhang P. (2001) *Moving mesh methods in multiple dimensions based on harmonic maps*, J. Comput. Phys. **170**, pp 562–588.

[20] Li R., Tang T. and Zhang P. (2002) *A moving mesh finite element algorithm for singular problems in two and three space dimensions*, J. Comput. Phys. **177**, pp 365–393.

[21] Lipnikov K. and Shashkov M. (2006) *The error-minimization-based strategy for moving mesh methods*, Commun. Comput. Phys. **1**(1), pp 53–80. `http://www.global-sci.com`

[22] Liu F., Ji S. and Liao G. (1998) *An adaptive grid method and its application to steady Euler flow calculations*, SIAM J. Sci. Comput. **20**(3), pp 811–825.

[23] Marquina A. (1994) *Local piecewise hyperbolic reconstruction of numerical fluxes for nonlinear scalar conservation laws*, SIAM J. Sci. Comput. **15**(4), pp 892–915.

[24] Schroll H.J. (2004) *Relaxed high resolution schemes for hyperbolic conservation laws*, J. Sci. Comp. **21**(2), pp 251–279.

[25] Schroll H.J. and Svensson F. (2005) *A bi-hyperbolic finite volume method on quadrilateral meshes*, J. Sci. Comp., in press.

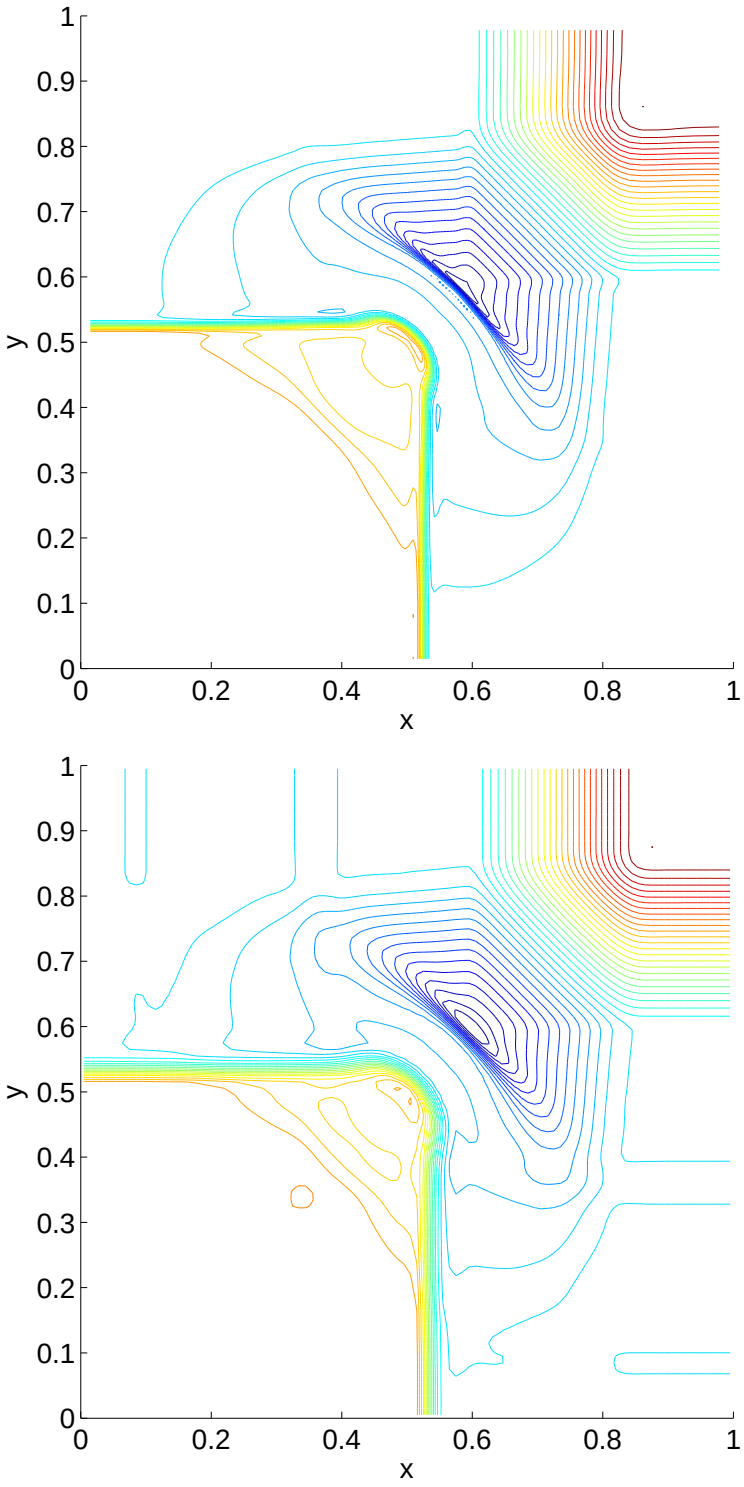

Figure 17: Density contours of Configuration 7 at $t = 0.25$ s. The upper solution is computed using adaptivity (third order average update, lower without adaptivity.

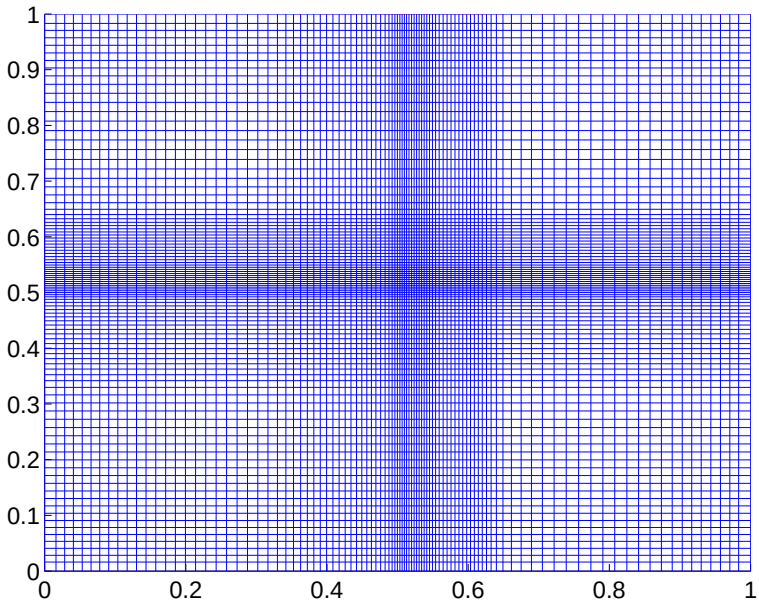

Figure 18: Adapted mesh of Configuration 7 at $t = 0.25$ s.

[26] Schulz-Rinne C.-W., Collins J. and Glaz H. (1993) *Numerical solution of the Riemann problem for two-dimensional gas dynamics*, SIAM J. Sci. Comput. **14**(6), pp 1394–1414.

[27] Serna S. (2005) *A class of extended limiters applied to piecewise hyperbolic methods*, SIAM J. Sci. Comput., to appear.

[28] Sod G.A. (1978) A survey of finite difference methods for systems of nonlinear hyperbolic conservation laws, J. Comput. Phys. **27**, pp 1–31.

[29] Stockie J., Mackenzie J. and Russel R. (2001) *A moving mesh method for one-dimensional hyperbolic conservation laws*, SIAM J. Sci. Comput. **22**(5), pp 1791–1813.

[30] Tan Z., Zhang Z., Huang Y. and Tang T. (2004) *Moving mesh methods with locally varying time steps*, J. Comput. Phys. **200**, pp 347–367.

[31] Tang T. (2005) *Moving mesh methods for computational fluid dynamics*, In Z.-C. Shi, Z. Chen, T. Tang and D. Yu, editors, Recent Adavances in Adaptive Computation, vol **383** of Comtemporary Mathematics, pp 185–218, American Mathematics Society, 2005.

[32] Tang H. and Tang T. (2003) *Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws*, SIAM J. Numer. Anal. **41**(2), pp 487–515.

[33] Tang H.-Z., Tang T. and Zhang P. (2003) *An adaptive mesh redistribution method for nonlinear Hamilton-Jacobi equations in two- and three-dimensions*, J. Comput. Phys. **188**, pp 543–572.

[34] van Leer B. (1982) *Flux-vector splitting for the Euler equations*, in Lecture Notes in Physics, E. Krause (ed.), Springer-Verlag, pp 507–512.