A CONSERVATIVE FRONT TRACKING ALGORITHM

Vinh Tan Nguyen*, Khoo Boo Cheong*† and Jaime Peraire*‡
*Singapore-MIT Alliance
†Department of Mechanical Engineering, National University of Singapore
‡Department of Aeronautics and Astronautics, Massachusetts Institute of Technology

***Abstract*— The discontinuities in the solutions of systems of conservation laws are widely considered as one of the difficulties in numerical simulation. A numerical method is proposed for solving these partial differential equations with discontinuities in the solution. The method is able to track these sharp discontinuities or interfaces while still fully maintain the conservation property. The motion of the front is obtained by solving a Riemann problem based on the state values at its both sides which are reconstructed by using weighted essentially non oscillatory (WENO) scheme. The propagation of the front is coupled with the evaluation of "dynamic" numerical fluxes. Some numerical tests in 1D and preliminary results in 2D are presented.**

I. INTRODUCTION

It is well known that solutions to problems of conservation laws can develop discontinuities and the presence of these discontinuities have important implications for various applications. To solve these problems, the front tracking methods have been developed which treat the discontinuities as interior boundaries coupled to finite volume computation for the separated regions.

The front tracking methods use lower dimensional grids called front, to represent the discontinuities in the numerical solutions. The propagation of the fronts is obtained by requirement the jump condition to be satisfied across the boundary. Thus the propagation speed is computed to advance the front in every time step. This can essentially be done by solving the Riemann problem between the states on both sides of the fronts. To ensure the conservative property of the scheme, the propagation of the front is coupled with the "dynamics" numerical flux.

In the last few years, a number of conservative front tracking methods has been developed as found in [1], [2], [3], [5], [4] and the references cited therein. In [1] James Glimm et al. presented a scheme which tracks the discontinuities sharply while preserving the conserved quantities at a discrete level. At the discontinuities they utilized the extrapolated state values at the front. The scheme has high order of numerical accuracy in compared to many algorithms in the literature. It was further developed and modified in [2] with improved accuracy and conservation. A general problem of all the front tracking schemes is the topological handling of the front. In [2], the front is handled in a way which looks simple in one dimensional space but it becomes much more complicated in higher dimensional space. Another attempt to handle the front is presented in [4] where it is extended to unstructured mesh in the frame of finite volume method. The location, geometry and propagation of the fronts are described by the level set.

In this paper a conservative front tracking method is proposed for solving the partial differential equations with discontinuities in the solution. It will track these discontinuities or interfaces in some problems sharply and fully maintain the conservation property. The motion of the front is obtained by solving a Riemann problem. The propagation of the front is coupled with the evaluation of "dynamic" numerical fluxes. The grid moves with interface and therefore it has to be remeshed to align with the propagation of the front. Mesh regeneration is rather simple in one dimensional problem, and it is done by using a moving mesh generator available in higher dimensional space. The Riemann problem at the interface is solved exactly based on the state values at its both sides which are reconstructed by using weighted essentially non oscillatory (WENO) scheme. The present paper is organized as follows. In section 2, a conservative tracking scheme is introduced, the concept of dynamic numerical fluxes, propagation of the front and Riemann problem in one dimensional space are discussed. The scheme in higher dimensional space is presented in section 3 including mesh regenerator and interface propagation. Finally the conservative front tracking is illustrated in Section 4 by a number of numerical applications including shock tube problems, multi-fluid flow examples in one dimension and some preliminary results for problem in two dimensions.

II. THE CONSERVATIVE FRONT TRACKING

Consider a conservation law equations as follows

$$\frac{\partial w}{\partial t} + \nabla \cdot f(w) = 0 \tag{1}$$

The space integral form of the above conservation law for a cell with moving boundary is written as

$$\frac{\partial}{\partial t}\int_V w dV + \oint_S (f_n(w) - w v_n) dS = 0 \tag{2}$$

where S is the boundary of the cell V, $f_n(w)$ and v_n are the component of $f(w)$ and velocity v in the direction of outward normal to S, respectively. $(f_n(w) - w v_n)$ is called dynamic numerical flux to separate from the normal flux which is not related to the moving velocity of the boundary. The Rankine-Hugoniot condition

$$[f_n(w) - v_n w]_L^R = 0 \tag{3}$$

is applied across the cell boundary.

For simplicity we will discuss the algorithm based in one dimensional space. The conservative front tracking propagates the interface by solving the Riemann problem at the

interface. Using the reconstructed states from the left and right of the front which are obtained by weighted essentially non-oscillatory (WENO) reconstruction [7][8], the Riemann solution will give the propagation speed of the interface. In one dimensional space, the position of interface is described by $\alpha(t)$ at time t. Given a uniformly underlying grid, we mesh the computational domain in a grid that take the front as a grid point. For example, the front $\alpha(t)$ is between the cell centers of element (i) and $(i+1)$ in the underlying grid, the computational grid is formed exactly as the underlying grid except for cell (i) and $(i+1)$ where the interface position becomes a grid point. We denote

$$W_i = \frac{1}{|\Delta x_i|}\int_{\Delta x_i} w(x)dx \tag{4}$$

and

$$F^{n+1/2} = \frac{1}{\Delta t}\int_{t_n}^{t^{n+1}} (f_n(u) - v_n u)dx \tag{5}$$

as the cell average value of cell i and the flux passing through a cell interface respectively. The finite difference equation for cells connected to the front is written as

$$\Delta x_i^{n+1}\overline{W}_i^{n+1} = \Delta x_i^n \overline{W}_i^n - \Delta t[F_{i+1/2} - F_{i-1/2}] \tag{6}$$

$$\Delta x_{i+1}^{n+1}\overline{W}_{j+1}^{n+1} = \Delta x_{i+1}^n \overline{W}_{i+1}^n - \Delta t[F_{i+3/2} - F_{i+1/2}] \tag{7}$$

where $\overline{W_i}$ is the numerical approximation to W_i. To update the states, we have to solve the Riemann problem at the cell interface to get the flux as well as the interface velocity.

At the cell interfaces, the Riemann problem is as follows:

$$W = \begin{cases} W_L & if\ x < \alpha \\ W_R & if\ x > \alpha \end{cases} \tag{8}$$

where W_L and W_R are reconstructed by using WENO reconstruction. The Riemann problem can be solved approximately as in [10]. Other Riemann solvers can be found in Toro[11]. The solutions to Riemann problem consist of the speed v, the constant states on both sides of the interface and the flux going through the cell interface. Only two cells that are adjacent to the front have the moving boundary; the rest of the cells are fixed and therefore the interface velocity is equal to zero. The front speed v_α has to satisfy the Rankine-Hugoniot condition (3). Then the interface is propagated by solving the ODE

$$\frac{d\alpha}{dt} = v_\alpha. \tag{9}$$

Therefore, the interface is advanced by just simply

$$\alpha^{n+1} = \alpha^n + v_\alpha \Delta t \tag{10}$$

or this can be formulated with a higher order stencil like Runge-Kutta for example. After the propagation of the interface, there are two case scenarios. Within the time interval $[t_n, t_{n+1}]$ the interface could pass through the cell center or it may not. In the case of the interface not passing through cell center we still keep the mesh and proceed to the next time step. However mesh regeneration must be done for the case of the interface passing through cell center. If a cell is too small then we merge it with its neighbor, and we will split a large cell into two different cells if necessary. The interface is assumed to move toward its right from cell i to cell (i+1) passing through the cell center $x_{i+1/2}$ then

$$\ddot{V}_{i+2}^{n+1} = V_{i+2}^{n+1} + V_{i+1}^{n+1} \tag{11}$$

$$V_i^{n+1} = \ddot{V}_i^{n+1} + \ddot{V}_{i+1}^{n+1} \tag{12}$$

where $V_i^m = \Delta x_i W_i^m$, $m = n, n+1$. In this case, cell (i+1) and (i+2) are merged to form cell i+2 and cell i is split into cell i and (i+1) correspondingly. In splitting cell i to cell i and (i+1), a constant interpolation is employed in this current work,

$$\ddot{u}_i^{n+1} = \ddot{u}_{i+1}^{n+1} = V_i^{n+1}/\Delta_i^{n+1}x \tag{13}$$

Basically the front tracking algorithm is proceeded as follows:

- Reconstructing the states at the cell interfaces $W_{i+1/2}$ by using WENO reconstruction on irregular grid formed in the corresponding interface position.
- Solving the Riemann problems at the cell interfaces in which an approximate Riemann solver, e.g. Roe's solver, can be used at the normal cells and an exact Riemann solver has been used at the interface. Solutions to Riemann problem consist of the speed s, the constant states on both sides of the interface and the flux going through the cell interface.
- Advancing in time for both interface position and states.

III. Conservative tracking in 2D

Consider the system of conservation laws (1) in two dimensional space

$$\frac{\partial W}{\partial t} + \nabla \cdot F(W) = 0 \tag{14}$$

defined by triangulation Ω. The interface is a curve described by a set of grid points. For a triangle Δ_i in Ω, the conservation law is written as

$$|\tilde{\Delta}_i|\tilde{W}_i = |\Delta_i|W_i - \int_{\partial\Delta_i}(F_n(W) - Wv_n)dS \tag{15}$$

where W_i is the cell average value of $W(x,y)$ over triangle Δ_i, $\tilde{\Delta}_i$ is the triangle i after a time step Δt and its associated cell average value $\tilde{W}_i$.

The line integral in (15) is the dynamic numerical flux in 2D and it can be obtained by using q-point Gaussian quadrature formula

$$\int_{\partial\Delta_i}(F_n(W) - Wv_n)dS = |\partial\Delta_i|\sum_{j=1}^{q}\omega_j(F_n(W(G_j)) - W(G_j)v_n(G_j)) \tag{16}$$

in which the values $W(G_j)$ can be reconstructed from W_i using WENO reconstruction, in this case a WENO reconstruction for irregular mesh is used [9].

A. Numerical fluxes

The flux $F(W(G_j))$ is approximated by a numerical flux, either Lax-Friedrichs or Roe or any monotone flux [14] in the normal direction to the boundary. In this work, Lax-Friedrichs and Roe's flux are used. One disadvantage of Roe's linearization is that the resulting approximate Riemann solution consists of only discontinuities, with no rarefaction waves which can lead to a violation of the entropy condition. Therefore an entropy fix is needed to be applied to ensure an entropy satisfied solution [14]. Similarly as in one dimensional case, only the elements adjacent to the front have the moving boundary, the rest is considered as fixed boundary elements.

B. Interface velocity

The velocity at interface nodal points has to be specified so that the interface could be updated step by step. Given an unstructured mesh, on each edge, the values of velocity can be interpolated by using WENO scheme. In general, a WENO reconstruction is used to interpolate the velocity from the left and right side of G, u_L^G and u_R^G respectively. The moving velocity can be obtained by applying the Rankine-Hugoniot condition as the jump condition between the left (L) and right (R) states at G:

$$[\overrightarrow{F}(U) - \overrightarrow{v}U]_L^R.\overrightarrow{n} = 0 \tag{17}$$

where $\overrightarrow{v}$ and $\overrightarrow{n}$ are the vectors of interface velocity at G and normal vector respectively. The above equation (17) can be rewritten in the following form:

$$[F_n(U) - v_n U]_L^R = 0 \tag{18}$$

Therefore the normal velocity of the interface at G, v_n can be specified as:

$$v_n = \frac{F_n(U_R) - F_n(U_L)}{U_R - U_L} \tag{19}$$

While the interface velocity at nodal points is explicitly known in the linear problems, it has to be specified by interpolated from the velocity of other points on the interface in the nonlinear case. In the case of a nodal point is an intersection of edges in the mesh as shown in the Figure (1), the value of velocity at node i $\overrightarrow{v}_i$ is constructed from the left and right (v_{n_1} and v_{n_2}) so that:

$$\overrightarrow{v}_i.\overrightarrow{n}_1 = v_{n_1} \tag{20}$$

$$\overrightarrow{v}_i.\overrightarrow{n}_2 = v_{n_2} \tag{21}$$

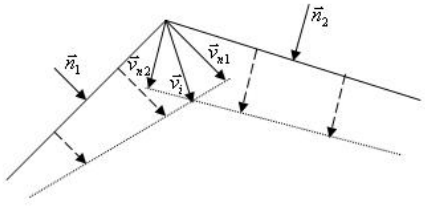

Fig. 1. Velocity at nodal points on the interface

C. Moving mesh generation

A mesh is generated using the iterative mesh generation technique in which all the nodes of the interface is marked and stored in the set INTERFACE. Now, the interface is moved by a distance $\delta(x, y)$ and we have to re-mesh the domain correspondingly. In [12][13], a mesh generator is introduced to create a mesh in which the geometry is described implicitly by its distance function. The node positions are obtained by solving for equilibrium in a truss structure and using Delaunay triangulation to reset the topology in every step. All the interface nodes are considered as fixed nodes at the new locations and we use the mesh generator to regenerate the grid. To improve the mesh after the interface moving, we assign forces in the mesh edges and solve for force equilibrium at the nodes. The force in an edge is proportional to the difference between the actual length l of the edge and its desired length l_0 which is set by the mesh size $h(x, y)$ evaluated at the mid point of the edge. There are some alternatives for the force function $f(l, l_0)$ in each edge and a model of linear spring is used to describe the force function as the repulsive forces [16]. To solve for the force equilibrium, we sum the forces for all the nodes $p(x, y)$ to get $F(p)$ and obtain a nonlinear system of equations as $F(p) = 0$. We can find the equilibrium positions by solving for the stationary solution of the system of ODEs

$$\frac{dp}{dt} = F(p), \quad t \geq 0 \tag{22}$$

using forward Euler. After each Euler step if there is any point which moves outside the geometry, it is projected back to the boundary by applying a reaction force normal to the boundary.

During the iterations, we always want to maintain a good connectivity by updating the triangulation. In the original mesh generator this is done by Delaunay triangulation, but it could be complicated and inefficient. A more robust and efficient triangulation based on the topology changes is implemented. The iteration is terminated when the mesh of sufficiently high quality is obtained.

1) Edge Flips: Instead of using Delaunay triangulation, we use the edge flipping technique [17] to ensure the good connectivity in the mesh that a circumcircle of any triangle should not contain any other triangles in the mesh. As in the Figure (2), triangle (b) is in circumcircle of triangle (a), thus we do the edge flip between two triangles and update the values of velocity accordingly. For simplicity, we apply a constant interpolation to update u, average values of velocity at the corresponding triangles.

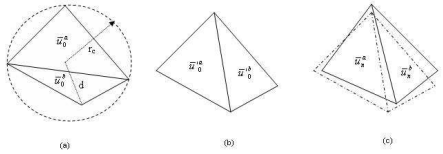

Fig. 2. Edge flip: (a) Initial triangles a and b with average values u_0^a and u_0^b, (b) After flipping and updating of u_0^a and u_0^b, (c) Final mesh with updated u_n^a and u_n^b

While the grid is regenerated, there are some circumstances in which we need to add or remove mesh points. Based on the qualitative measures for triangles [15], all the bad or degenerate triangles must be removed from the mesh one or another way. For example, if an edge is too long compared to the desired value then we need to split it into two edges by adding one point or we merge it with a neighboring edge if it is too short.

2) Splitting: It is sometimes necessary to add points to the mesh if there is an edge which is too long. We check for all the meshes. If there is any edge which is too long compared to the desired value based on value of $h(x, y)$ at its midpoint then the midpoint becomes a mesh point and the element is split into two elements with the same average values of u as the initial element.

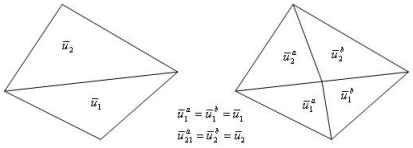

Fig. 3. Splitting

3) Merging: If an edge is too short in comparison to the expected value, we will merge it with a neighboring edge as in Figure (4). Considering all the edges connected to the short one, we merge it with the one that make the largest angle (the most obtuse) to it. We have to delete the merged node and move all the edges connected to that node to its neighboring node. The collapsing elements which take the merged node and its neighbor as two vertices are removed as well. It is more complicated to update the average values of velocity in this case than the others. A constant interpolation may be an option but it may not work well under all circumstances.

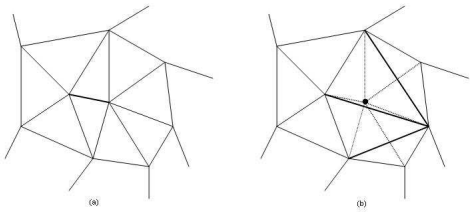

Fig. 4. Merging: (a) Initial mesh; (b) After merging

There also the case whereby a node in the interior domain approaches the boundary and makes the element to be 'flat'. To prevent a node from moving to the boundary we will add a linear string to the nodes that are closed to the boundary. The repulsive forces are produced to keep them separated from the boundary. During the meshing iterations, if the distance of a node from the boundary as measured by the distance function is small, for example $0.65h_i$ in our present algorithm, a repulsive force is added at that node in the normal direction to the boundary, i.e.

$$f_B = (d(x, y) - 0.65h_i), \tag{23}$$

where $h_i = h_0.h(x, y)$. The the gradient of distance function at a point $\nabla d(x, y)$ give the negative direction to the closest boundary and it could be computed numerically as

$$\nabla d(x, y) = (\frac{d(x + d_\varepsilon, y) - d(x, y)}{d_\varepsilon}, \frac{d(x, y + d_\varepsilon) - d(x, y)}{d_\varepsilon}) \tag{24}$$

where $d_\varepsilon = 1.0e-12$. Alternatively, the quality of the triangle can be specified by using other criteria such as aspect ratio, smallest angle and others in [17].

D. Conservative solution-updating

After the remeshing, it can be considered that there is only node movement in the whole region and we have to update the values conservatively. In [6], a conservative interpolation is proposed to update the state values in the new grid from the original one.

Assume that u_i is the cell average of $u(x, y)$ over the triangle Δ_i in the grid Ω

$$u_i = \frac{1}{|\Delta_i|} \int_{\Delta_i} u(x, y)dxdy. \tag{25}$$

In the new grid $\tilde{\Omega}$, all the points (x,y) in triangle Δ_i move to $(\tilde{x}, \tilde{y})$ in triangle $\tilde{\Delta}_i$ and $\tilde{u}_i$ is the cell average of $u(\tilde{x}, \tilde{y})$ over $\tilde{\Delta}_i$,

$$\tilde{u}_i = \frac{1}{|\tilde{\Delta}_i|} \int_{\tilde{\Delta}_i} u(\tilde{x}, \tilde{y})d\tilde{x}d\tilde{y}. \tag{26}$$

With small disturbances $(c^x(x, y), c^y(x, y)$,

$$(\tilde{x}, \tilde{y}) = \Im(x, y) = (x - c^x(x, y), y - c^y(x, y)) \tag{27}$$

The conservative interpolation is given as

$$|\tilde{\Delta}_i|\tilde{u}_i = |\Delta_i|u_i - |\partial\Delta_i| \sum_{j=1}^{q} \omega_j c_n^j u(G_j) \tag{28}$$

where $c_n = c^x n_x + c^y n_y$ with (n_x, n_y) as the unit normal vector. The above interpolation is conservative in the sense that

$$\sum_j |\tilde{\Delta}_j|\tilde{u}_j = \sum_j |\Delta_j|u_j. \tag{29}$$

IV. Numerical applications

The front tracking method has been developed without any preference of specific kind of discontinuities, in this section a number of examples are shown to demonstrate the scheme including shock wave, material interface and others. Some preliminary results of convection problems in 2D are also shown.

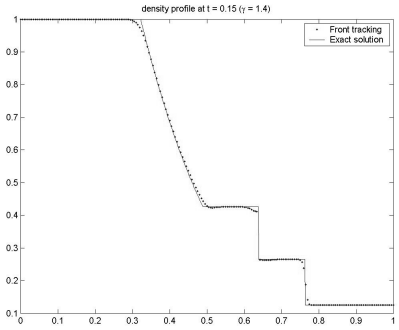

Fig. 5. Sod's problem, density profile at t=0.15s, using 3^{rd} order WENO reconstruction and 2^{nd} order Runge-Kutta, 200 grid points

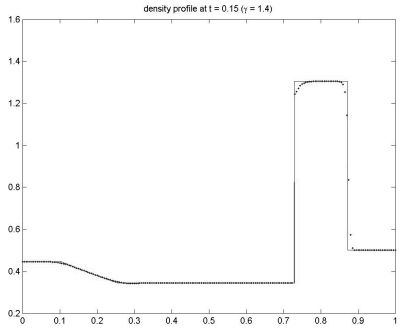

Fig. 6. Lax's problem, density profile at t=0.15s, using 3^{rd} order WENO reconstruction and 2^{nd} order Runge-Kutta, 200 grid points

A. Euler equations in one dimensional space

The compressible flow is described by the Euler equations

$$\partial W/\partial t + F(U)_x = 0 \tag{30}$$

where W= $\begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix}$, F(W) = $\begin{pmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{pmatrix}$ expressing the conservation of mass, momentum and energy of the fluid. The thermodynamics properties are given by the equation of state (EOS). For ideal gas the equation of state is simply written as follows

$$\rho e = \frac{p}{\gamma - 1} \tag{31}$$

In solving a problem with the presence of water, a more general EOS for stiff fluids has been used:

$$\rho e = \frac{p + \gamma p_\infty}{\gamma - 1} \tag{32}$$

where p_∞ is the stiffness parameter. The above stiff EOS could be considered as simplification of the more complex $Mie - Grüneisen$ EOS:

$$p = p_\infty(\rho) + \gamma(\rho e - e_\infty(\rho)) \tag{33}$$

where $p_\infty = 0$ for ideal gas.

1) Contact discontinuity tracking: In these examples a contact discontinuity is tracked by using the above conservative front tracking algorithm. Let's consider the two well-known shock-tube problems normally called Sod'problem and Lax's problem. In the computational domain [0,1] the discontinuity is initially at 0.5, $p_\infty = 0.0$ and $\gamma_L = \gamma_R = 1.4$. The left and right states are given as $\rho_L = 1.0, p_L = 1.0, u_L = 0.0$; $\rho_R = 0.125, p_R = 0.1, u_R = 0.0$ for Sod's problem and $\rho_L = 0.445, p_L = 3.528, u_L = 0.698$; $\rho_R = 0.5, p_R = 0.571, u_R = 0.0$ for Lax's problem. The results of density at time t = 0.15 are shown in Fig 5 and Fig. 6 with the exact solution as the solid line. The 3th order WENO reconstruction and 2nd order Runge-Kutta have been used.

2) Material interface tracking: The front tracking method is now used to track the material interface between two fluids. This problem is taken from [18]. Consider a gas-gas problem with different γ and $p_\infty = 0$ as a pure shock initially at 0.05 passing through the interface located at 0.5. The left, middle and right states are described as $\gamma_L = 1.4, \gamma_M = 1.4, \gamma_R = 1.67$;$\rho_L = 1.3333, \rho_M = 1.0, \rho_R = 0.139$;$p_L = 1.5 \times 10^5, p_M = p_R = 1.0 \times 10^5$;$u_L = 0.3535\sqrt{10^5}$ and $u_M = u_R = 0.0$. The result of density profile at t=0.0012 is plotted in Figure 7.

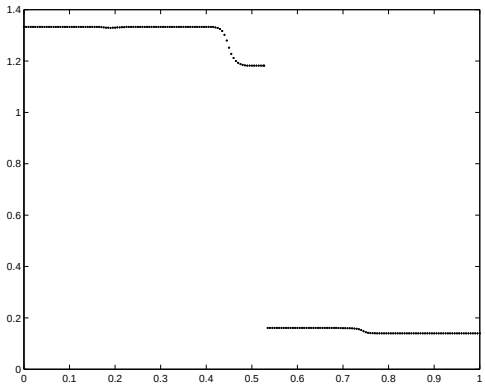

Fig. 7. Gas-gas problem, density profile at t=0.0012s, using 3^{rd} order WENO reconstruction and 2^{nd} order Runge-Kutta, 200 grid points

Considering a problem of shock impacting a water interface. The strength of the gas shock is p_L/p_M = 100.0. The shock is initially at 0.96 where the state behind the shock is $p_L = 10^7$, $\rho_L = 8.266055$, $u_L = 2494.97$ and ahead of the shock $p_L = 10^5$, $\rho_L = 1.0$, $u_M = 0.0$, $\gamma = 1.25, p_\infty = 0.0$. The interface is initially located at 5.0 with the water on its right where $p_R = 10^5$, $\rho_R = 10^3$, $u_M = 0.0$, $\gamma = 7.15, p_\infty = 3.30910^8$. The density profile is shown in Fig. 8 at time t=0.003, in which the 3rd order WENO reconstruction has been used. It can be seen from the result that there is no oscillation near the interface which is captured very well and sharply.

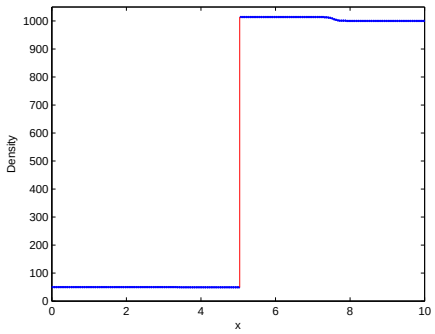

Fig. 8. Gas-water problem, density profile at t=0.003, conservative front tracking using 3rd order WENO and 2nd order RK

B. Convection problems in 2D

Consider a convection problem

$$\frac{\partial \Phi}{\partial t} + U \nabla \Phi = 0 \tag{34}$$

where $U = (-y, -x)$ and $(x, y) \in [-1; 1] \times [-1; 1]$ with initial condition as follows

$$\Phi_0(x, y) = \begin{cases} 1.0 & if\ x^2 + y^2 \leq 0.4 \\ 0.0 & otherwise \end{cases} \tag{35}$$

Initial condition and result at t=0.6 are shown in Figure (9) where the initial circle is deformed to become an ellipse. It can be seen that the mesh generator can maintain the quality of the grid in this case and the conservation property is preserved very well.

Consider another convection problem where $U = (y, -x)$ and $(x, y) \in [-1; 1] \times [-1; 1]$ with initial condition as follows

$$\Phi_0(x, y) = \begin{cases} 1.0 & if\ (x - 0.3)^2 + (y - 0.3)^2 \leq 0.4 \\ 0.0 & otherwise \end{cases} \tag{36}$$

This problem is similar to the rotation of a circle about the origin. The result at t=2.0 is plotted in the Fig. 10 together with the initial condition. While the circle is moving, the mesh is regenerated with preserved good quality and the velocity is updated conservatively.

V. Conclusion

In this paper, a fully conservative front tracking algorithm is presented for solving the equations of conservation laws with discontinuities. The scheme has been tested with several numerical examples in one dimension where the contact between gas-gas and gas-water evolves in the presence of shock waves. We have applied the scheme to two dimensions and some computed results were presented. It has been shown that the method works well and is able to capture the front very sharply. More importantly the scheme is conservative. The work is currently extended to handle the nonlinear, Euler's equations in two dimensions. Developing the scheme to three dimensions is the future extension.

References

[1] James Glimm et al., *Conservative front tracking and level set algorithms*, 2001. Proceedings of the National Academy of Sciences (PNAS), Vol. 98, pp. 14198-14201.

[2] James Glimm et al., *Conservative front tracking with improved accuracy*, 2003. SIAM J. Numer. Anal, Vol. 41, No. 5, pp. 1926-1947.

[3] James Glimm et al., *Simple front tracking*, 1999. Comtemporary Mathematics, Vol. 238, pp. 133-149.

[4] O. Gloth,D. Hanel,L. Tran,R. Vilsmeier, *A front tracking method on unstructured grids*, 2003. Computers and Fluids, Vol. 32, pp. 547-570.

[5] Mao De-Kang, *Toward front tracking based on conservation in two dimensional space*, 2000. SIAM J. Sci. Comput., Vol. 22, No. 1, pp. 113-151.

[6] Huazhong Tang and Tao Tang, *Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws*, 2003. SIAM J. Numer. Anal, Vol. 41, No. 2, pp. 487-515.

[7] C.W. Shu, *Essentially non-oscilatory and weighted essentially non-oscilatory schemes for hyperbolic conservation laws.* in Adavnced Numerical Approximation of Nonlinear Hyperbolic Equations, editted by A.Quateroni,Editor,Lecture Notes in Mathematics, CIME subseries (Springer-Verlag, Berlin/New York); ICASE Report 97-65.

[8] C.W. Shu and S. Osher, *Efficient implimentation of essentially non-oscilatory shock capturing schemes*, 1989. J. Comput. Phys., Vol. 83, pp. 32-58.

[9] Changqing Hu and Chi-Wang Shu, *Weighted essentially non-oscilatory schemes on triangle meshes*, 1999. J. Comput. Phys., Vol. 150, pp. 97-127.

[10] Roe PL, *Approximate Riemann solvers, parametter vectors and difference schemes*, 1981. J. Comput. Phys., Vol. 43, pp. 357-372.

[11] Toro E.F., *Riemann Solvers and Numerical Methods for Fluid Dynamics*, 1999. Springer-Verlag. Second Edition, 624 pages.

[12] P.-O. Persson, G. Strang, *A Simple Mesh Generator in Matlab*, 2004. SIAM Review, Vol. 46 (2), pp. 329-345.

[13] P.-O. Persson, G. Strang, *Circuit Simulation and Moving Mesh Generation*, 2004. Proc. of Int. Symp. on Comm. and Inform. Tech. 2004 (ISCIT 2004).

[14] Randall J. Leveque, *Numerical Methods for Conservation Laws*, 1992. Lectures in Mathematics, Birkhauser.

[15] David A. Field, *Qualitative measures for initial meshes*, 2000. Int. J. Numer. Meth. Engng., Vol. 47, pp. 887-906.

[16] C. Wang et al. , *Elastic mesh technique for 3D BIM simulation with an application to underwater explosion bubble dynamics*, 2003. Computers & Fluids, Vol. 32, pp. 1195-1212.

[17] Herbert Edelsbrunner, *Geometry and Topology for Mesh Generation*, 2001. Cambridge University Press.

[18] Ronald P. Fedkiw, Tariq Aslam, Barry Merriman, Stanley Osher, *A non-oscillatory Eulerian Approach to interfaces in multimaterials flows (the ghost fluid method)*, 1999. J. Comput. Phys, Vol. 152, pp. 457-492.

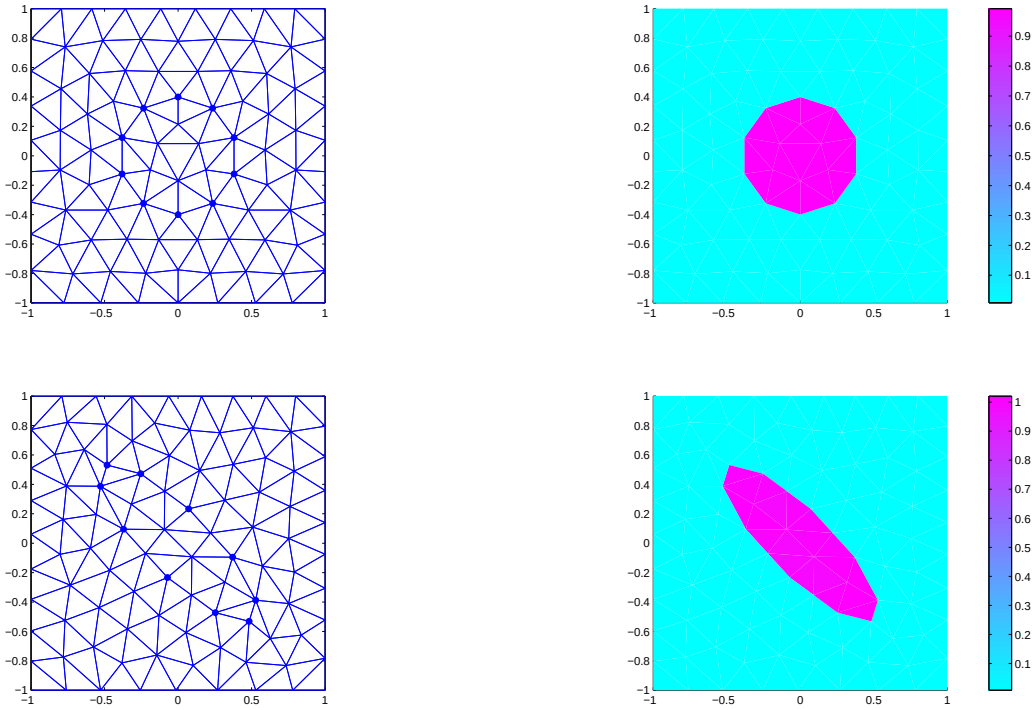

Fig. 9. Mesh on the left with the interface as dotted points and velocity on the right. From top to bottom, result at t=0.0,0.6

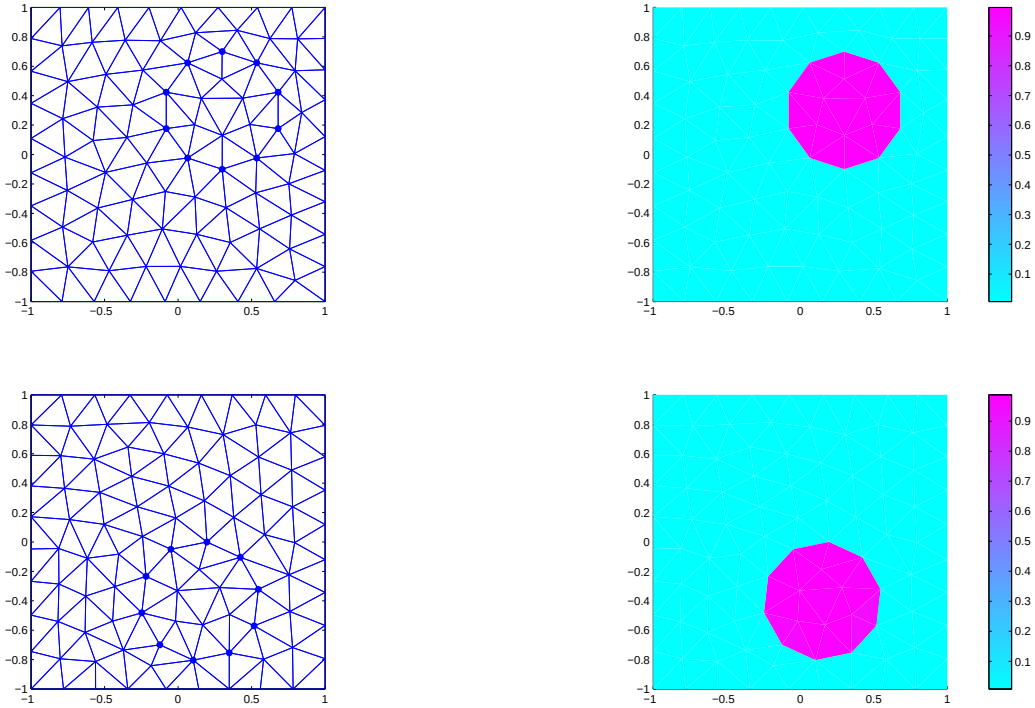

Fig. 10. Rotation problem: mesh on the left with the interface as dotted points and velocity on the right. From top to bottom, result at t=0.0,2.0