6th European Conference on Computational Mechanics (ECCM 6)
7th European Conference on Computational Fluid Dynamics (ECFD 7)
11-15 June 2018, Glasgow, UK

Adaptation of 2D unstructured mesh based on solution gradients

R. Vijay Ram[1*], Santanu Ghosh[2], Shashank Subramanian[3], Deepak Kandasamy[4]

[1] IIT Madras, Dept. of Aerospace Engineering IITM, vijayrram.ae@gmail.com
[2] IIT Madras, Dept. of Aerospace Engineering IITM, sghosh1@iitm.ac.in
[3] IIT Madras, Dept. of Aerospace Engineering IITM, shashank1693@gmail.com
[4] IIT Madras, Dept. of Aerospace Engineering IITM, deepak.vk.1994@gmail.com

The work presented in this paper attempts to improve the grid (and consequently solution) for compressible-flow simulations performed with a 2D finite-volume Euler solver for unstructured grids, using mesh adaptation based on gradients of flow parameters, (pressure and pseudo-entropy, and grid geometry (cell areas, nodal distances etc.). The procedure requires solution (primitive variables) reconstruction at grid nodes, which are interpolated using linear polynomials in 2 dimensions or inverse distance based methods, and cell-averaged gradients, which are computed using the Green-Gauss method. The adaption is terminated if the global maximum displacement of any node is less than ϵ, where ϵ is a small user defined length scale. Results indicate that the method is capable of clustering the grid near an oblique shock and a contact wave, which results in sharper resolution of the discontinuities.

1 Introduction

With the rapid development of high performance computers, computational simulation has become an essential tool for analysis of engineering problems. A core element, typical of most computational analysis, is the design of meshes, structured or unstructured, on which the governing equations that are to be analyzed are discretized. Use of such discretized meshes results in undesirable discretization errors. These errors can be reduced by a uniform mesh refinement (using finer meshes), which can result in significant increase in computational costs. An alternative, and generally adopted approach is targeted refinement, wherein the mesh is selectively refined in regions (of interest) where the solution is expected to vary rapidly in space. In certain domains (and flow simulations), it is not possible to know a priori where such regions of interest will be, which makes it difficult to design a suitably refined grid. Thus there is a requirement for techniques that adapt the mesh based on an initial solution to produces new meshes that are better at resolving the solutions.

For structured grids, targeted adaptation is done by clustering grid lines in and around regions of interest. However, in the case of unstructured grids, the lack of such grid lines does not allow a simple method of adaptation in the domain. Standard practices involve grid refinement by repeated subdivision and re-meshing of (part or whole of) the

domain. This is usually done using largest-edge-length splitting or centroid splitting and applying the Delauney edge flipping algorithm [5]. These methods, though fast, change the topology of the grid and prevent fast integration with the solver since the solver needs to read the mesh again for every cycle of adaptation.

Thus point-movement based methods for adapting unstructured grids are more efficient since they do not change the topology of the grid.

Such methods are typically based upon equidistribution of certain flow parameters [6], for instance the gradient of a flow property. This paper uses the gradients of pressure/pseudo-entropy to move the points and hence adapt the grid. The primary formulation of point-movement presented in this work derive from the adaptive techniques discussed in Eiseman [1], Vilsmeier and Hänel [3], and Jahangirian and Shoraka [7].

While a combination of the two gradients could be used, this paper deals with two different test cases: supersonic flow past a compression corner, wherein pressure gradients are used for grid adaptation, and a shock-tube setup with a stationary contact discontinuity, wherein pseudo-entropy gradients are used for grid adaptation. In principle, the method should be able to adapt the grid near both types of discontinuities: compression shocks (which involve changes in both pressure and entropy across the discontinuity) and contact waves (which involve changes in entropy and not pressure across the discontinuity). As such, for flows involving both of these discontinuities, a combination of the gradients in pressure and pseudo-entropy may be required for adapting the grid.

2 Methodology

The design of the grid adapter takes into account multiple factors: formulation for node movement, use of data structures, data interpolation to nodes from cell-averaged values, to (new) nodes after node movement and to (new) cell-centers from (interpolated) nodal data, and certain other design constraints. These are elaborated upon in the following sub-sections.

2.1 Formulation

In this work, the mesh adaptation is done by the movement of grid nodes due to the combined of effects of a flow-dependant component, G, a global grid-size based (scalar) component, S, and a local grid-topology based component (vector) $\hat{X}$. The formula thus obtained for the motion of a grid node, denoted by $\Delta\vec{x}_i$, is given in equation 1. The minimum function ensures that only cell contractions due to gradient differences are allowed.

$$\Delta\vec{x}_i = S \times \sum_{j=1}^{n_i} \min\{G_j, 0\} \times \hat{X}_{ij} \tag{1}$$

Where,

$$S = s \times L$$

$$G_j = \frac{\Phi - \phi_j}{\phi_{max}}$$

$$\Phi = \frac{\sum_{k=1}^{N} A_k \phi_k}{\sum_{k=1}^{N} A_k}$$

$$\phi_{max} = \max_{k=1}^{N} \{\phi_k\}$$

$$\hat{X}_{ij} = \frac{\vec{x}_i - \vec{c}_j}{\|\vec{x}_i - \vec{c}_j\|}$$

$$\phi_k = \|\nabla P_k\| \text{ or } \|\nabla E_k\| \text{ (user choice)}$$

and,

i is the index of the node whose movement is in question

j is the index used to iterate the cells surrounding the i^{th} node

n_i is the number of elements around the i^{th} node

$\vec{x}$ is the position vector of a node

s is the numerical scaling value; chosen to be 0.4

L is the length scale; chosen to be the length of the shortest edge in the grid

ϕ is the parameter (gradient) based on which adaptation is done

Φ is the area weighted average of ϕ

ϕ_{max} is the maximum magnitude of ϕ

N is the total number of elements in the grid

$\vec{c_j}$ is the position vector of the cell centroid of the j^{th} element around node i

2.2 Data Structure

Two data structures were tested to encode the grid for the purpose of adaptation. The following subsections discuss them.

2.2.1 Vertex-Element-Face Data Structure

The Vertex-Element-Face data structure [11] is easy to encode. However, it is considerably more difficult to enable the addition and deletion of grid members such as points or cells, due to high amount of reflexive references between them. Apart from this, the excessive references to neighbours causes the data structure to be bulked in terms of memory usage.

2.2.2 Half-Edge Data Structure (Doubly Connected Edge List)

The encoding of the Half-Edge data structure [9] is more involved, compared to the Vertex-Element data structure. However, it enables much easier routines for the addition and deletion of grid members. Also, since all the vertices and elements have only one half-edge to keep track of and all connectivity of nodes and cells are dealt with by only the half-edges. Thus the memory requirement to store the data is considerably reduced.

While the present work does not involve mesh refinement (addition/deletion of mesh elements), the half-edge data structure was preferred keeping in mind potential extension of the present work to include adaptive mesh refinement in the future.

2.3 Limiters

The following sections describes the parameters incorporated in the design of the algorithm to prevent issues that can arise due to the movement of points. They arise are typically due to round-off errors in the data, crossover of points due to large magnitude in movement, which causes flipping of triangle orientations, and endless movement scenarios where the points keep moving by small magnitudes, either locally stable or not, that can result in endless cycles of adaptation.

2.3.1 Minimum Length Threshold

A scaled value of the shortest edge length, wherein the scaling factor is defined by the user, is used as a minimum threshold for the motion of points. A node is moved if and only if the magnitude of its movement is greater than this threshold. is set to 0.1% of the minimum edge length for the results presented in this paper. This filter is used to prevent extremely small motions of points preventing endless motion. This inhibition saves up on the associated costs due to data interpolation on these nodes and the neighbouring cells.

2.3.2 Maximum Length Threshold

Another scaled value of the shortest edge length is used to prevent infinite loops of the grid adapter. This scale is defined to be 1% of the minimum edge length for the results presented in this paper. If the all the magnitudes of node movements in an iteration are less than this threshold, then the adaptation loop is exited. In other words, only as long as a point moves by a magnitude greater than 1% of the shortest edge length, then adaptation continues. This prevents locally stable dynamic movements resulting in an infinite adaptive loop.

2.3.3 Minimum Angle Threshold

In order to prevent the change in orientation of the triangles, and to enable adaptation with better grid quality, a minimum angle threshold is used. If in an adaptation step, the interior angle of any triangle falls below this threshold, the nodes that form that triangle

are frozen, and prevented from further movement henceforth. That adaptation step is discarded and a new adaptation step is initiated to maintain the motion of other points. Various values of this threshold are used in this paper to study the effects it has on the grid quality.

2.3.4 Scaling Value

The scaling value is a scalar used to limit the motion of the points given by s in the term S in Equation 1. For the case of one-dimensional adaptation, the maximum value that can be allotted to the scaling value without having to worry about crossover, is 0.25. But, for two-dimensional adaptation, making such an estimate is difficult since the number of cells that surround a point is not fixed.

Since the minimum angle threshold is sufficient to prevent crossover, the scaling value can thus take any value. Smaller values are preferred because they maintain a smoother movement of points and also allow for wider motion before the points have to be frozen in place. And, larger values allow for larger steps in the movement of points. A value of 0.4 was chosen for the scaling value in this paper.

2.4 Initial Nodal Interpolation

The solution data obtained from the solver is given at the cell centers. This has to be interpolated to the nodes to enable the interpolation of the data in adaptation steps. This interpolation is called initial nodal interpolation to differentiate it from later interpolation steps (discussed in Section 2.5).

The interpolation of data to a give node is done using all the cells that surround it including ghost cells. All the scalars and the individual vector components are separately interpolated. The following subsections describe the interpolation methods studied.

2.4.1 Area Weighted Interpolation

The reasoning for using area weighted interpolation was that the data obtained from the solver is cell averaged data, i.e., the net value of any data obtained from a cell would be the cell data times its area. Thus an interpolation based on this method possibly gives the best approximation to the interpolated variable.

2.4.2 Inverse Distance Weighted Interpolation

Interpolation using inverse distance weighting is a common technique and was used to compare with the results obtained using the area weighted interpolation. It was found that the final outcome of both the interpolation methods were similar. However, since the inverse distance weighted interpolation method is computationally less taxing, it was chosen as the initial method of data interpolation from cell centers to nodes.

2.5 Medial Nodal Interpolation

After a step of adaptation, some nodes move into neighbouring cells. The data for these nodes needs to be interpolated from the old positions of the nodes that surround that cell. This interpolation of data from old nodes to new nodes is called medial nodal interpolation. Two interpolation methods were studied to interpolate data from the old nodes to the new. They are discussed in the following subsections.

2.5.1 Inverse Distance Weighted Interpolation

This interpolation was done to keep the same interpolation routine for the nodes before and after adaptation. The cell containing the new position of the node is found. The data is interpolated from the old data of the three vertices that define the cell. However, it was observed that the data post interpolation on the nodes were not having the smooth distribution which was preferred. Thus an alternate method of data interpolation needed to be implemented.

2.5.2 Polynomial Interpolation

Each scalar/vector component is assumed to be linear in x and y. Thus they are of the form $\varphi = ax + by + c$, where φ describes the data being interpolated. First, the cell that contains the current node is found. Then using the old data of the three vertices of the cell, we can obtain three equations for the variables a, b and c. These equations are then solved by matrix inversion and the parameter is reconstructed at the new node location.

It was observed that unlike the inverse distance weighted interpolation method, the polynomial interpolation method provided smoother interpolation of data. Hence the polynomial interpolation method was chosen to interpolate the data from cell centers to moved nodes in every step of adaptation.

Note that for any point that had not moved in an adaptation step, no interpolation is done to preserve the data at the position of that node. This also reduces the computational costs of data interpolation across the grid.

2.6 Cell Interpolation

The interpolation of data from new nodes to new cell centers in each step of the adapter was tested again using two interpolation methods, viz., interpolation using inverse distance weighting and polynomial interpolation. The are described in the following subsections.

Note that for any cell that did not have any of its neighbouring points moved in a particular step of adaptation, data interpolation was not done to the particular cell to preserve the initial data, and also to reduce computational costs.

2.6.1 Inverse Distance Weighted Interpolation

Each scalar/vector component is individually interpolated from the surrounding nodes. This method was tested because of its simplicity in implementation, particularly for cells with more than three neighbouring nodes. It was observed again that the interpolation of the data was not smooth and as expected.

2.6.2 Polynomial Interpolation

Just as described in the Section 2.5.2, each scalar/vector component is interpolated using the three vertices of a cell post movement. It was found that polynomial interpolation, i.e., the linear approximation was more accurate in depicting the parameters as opposed to the inverse distance weighted interpolation. And, since all the grids studied in this paper have only triangular cells, polynomial interpolation was chosen as the method of data interpolation to the cell centers at each step of adaptation.

3 Solver

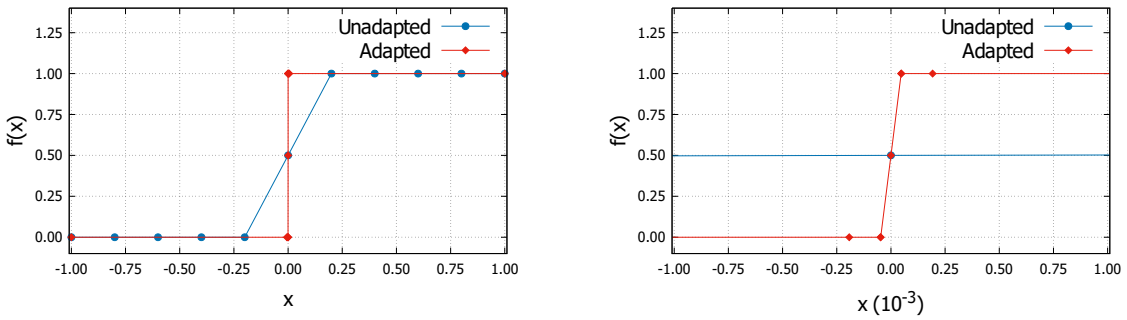
The solver is an unstructured (triangular) grid solver that uses a cell-centered finite volume method for solving compressible inviscid flows [11]. The solver uses the AUSM scheme proposed by Liou and Steffen for computing the inviscid fluxes [2]. The primitive variables are reconstructed at the interfaces using a second order reconstruction (based on gradients) which are then limited using the Venkatakrishnan limiter [4]. Temporal discretization is done using the fourth order Runge-Kutta method.

4 Flow Cases and Results

A 1D version of the adaptation formula (Formula 1) was tested on the the Heaviside function, $H(x)$. In 2D the adapter was tested for an oblique shock grid and a stationary contact wave using various minimum angle thresholds. Some of these results obtained are described below.

4.1 Heaviside step function

The Heaviside function $H(x)$ is discretized on the domain $x \in [-1, 1]$ using 11 points. The results obtained are shown in Figure 1. All points clustered about $x = 0$ as expected.



(a) Piecewise linear representations

(b) Zoomed image

Figure 1: 1D adaptation of the Heaviside function

4.2 Oblique Shock

The grid and flow details for the oblique shock as taken from Ghia et. al. [8] are given below in Table 1.

Domain length	:	1.965926 m	**Domain height**	:	1.000000 m
Turn angle	:	15^o	**Turn location**	:	(0, 0)
Number of nodes	:	7381	**Number of cells**	:	14460
Total pressure	:	101325 Pa	**Total density**	:	1.224 kg/m^3
Specific heat ratio	:	1.4	**Mach number**	:	3.0

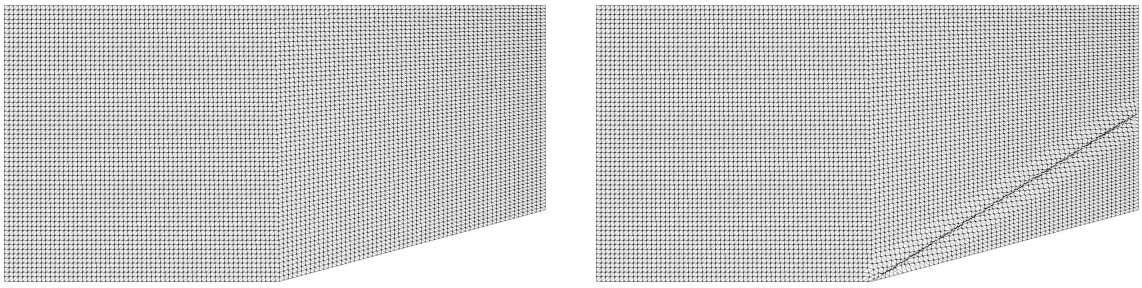
Table 1: Grid and Boundary Conditions for the Oblique Shock

The adapter was run using pressure gradient (Equation 1) for the minimum angle thresholds of 1^o, 5^o, 10^o, 15^o, 25^o, 30^o and 35^o. Some of the unadapted and adapted grids are given in Figure 2.

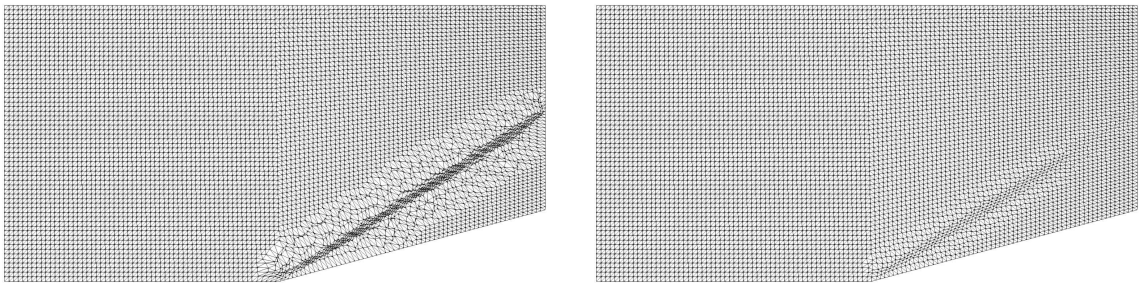
When a smaller minimum angle threshold is used, there is higher clustering of the cells about the shock. Also, the grid quality is lost for most of the cases with smaller threshold angles. The exception to this trend is the grid obtained using $A = 1^o$. This is due to the fact that the grid points close to the shock region get frozen before they can include movements in the neighbouring points due to interpolation. The points on the boundaries are prevented from moving in order to preserve the domain shape. This causes part of the grid in the shock region to not get adapted for $A = 35^o$.

Figure 3 shows the zoomed-in view contours of pressure for the solutions generated on the different adapted grids. It is observed that although the $A = 1^o$ constraint generates the thinnest shock layer, the resolution of the shock does not happen across same number of cells along the shock. The shock layer progressively become thicker as A is increased, but

it also results in the shock being resolved in a more uniform manner (based on number of cells across the shock) along the shock length. Based on these observations, it can be argued that intermediate values of A (15^o, 20^o) are more suitable for the adaptation, as they provide both sharp (thinner shock compared to unadapted grid) and uniform resolution of the shock (as opposed to A of 1^o, 5^o).

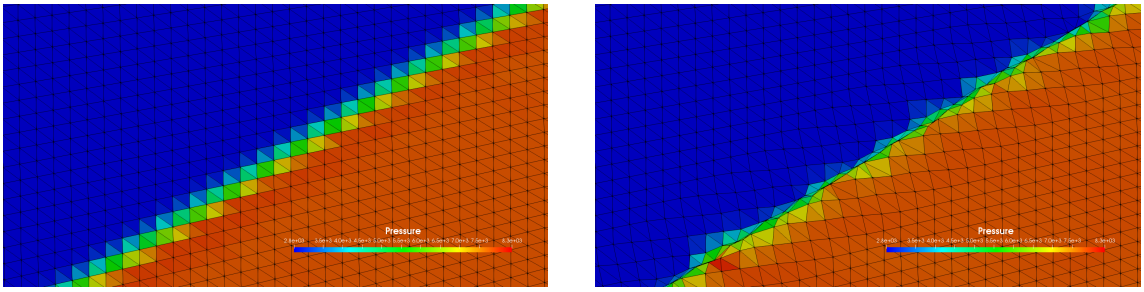


(a) Unadapted (b) A = 01^o

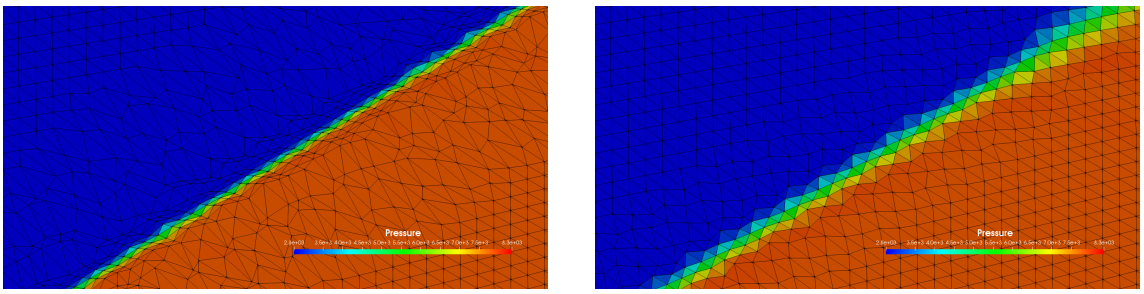


(c) A = 20^o (d) A = 35^o

Figure 2: Oblique Shock [Grid]



(a) Unadapted (b) A = 01^o



(c) A = 20^o (d) A = 35^o

Figure 3: Oblique Shock [Pressure, Zoomed]

4.3 Shock Tube

A shock-tube setup was used for the adaptation of the grid used for the simulation of a stationary contact wave. The extents of the domain were from $x = 0.0m$ to $x = 1.0m$ with the discontinuity located at $x = 0.5m$ Details of the domain and the initial conditions are provided in Table 2. The grid and the initial conditions were designed to capture the discontinuity perfectly. Thus the case was run more to see how the points near the discontinuity would move due to the adaptation rather than to see an improvement in the solution.

Domain length	:	1.0 m	**Domain height**	:	0.2 m
Number of nodes	:	8241	**Number of cells**	:	16000
Pressure (Left)	:	1.0 Pa	**Pressure (Right)**	:	1.0 Pa
Density (Left)	:	0.5 kg/m^3	**Density (Right)**	:	1.0 kg/m^3
Mach Number (Left)	:	0.0	**Mach Number (Right)**	:	0.0

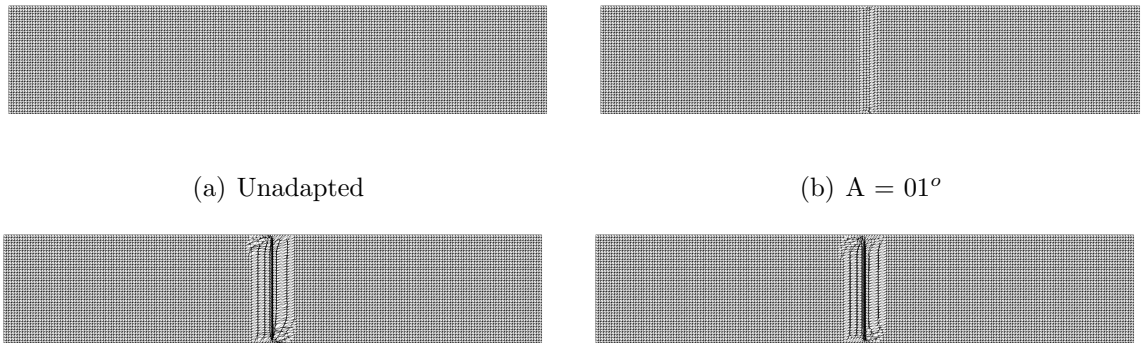
Table 2: Grid and Boundary Conditions for the Shock Tube

The adapter was run using the entropy gradients (Equation 1) for the minimum angle thresholds of 1^o, 5^o, 10^o, 15^o, 25^o, 30^o and 35^o. It was seen that for the minimum angle criteria of 30^o and 35^o, the grid did not undergo any adaptation. Some of the unadapted and adapted grids are given in Figure 4.

When a smaller minimum angle threshold is used, there is higher clustering of cells. However, the cells drift a bit to the left. This is assumed to be due to having only the contraction switch being used. Also, due to the boundary points being fixed, the grid quality is lost. These problems are somewhat alleviated when using larger threshold values (A of 10^o or 15^o).

Figure 5 shows the zoomed in view of the contours of density of the solutions generated on the adapted grids. In the case of the shock tube, due to the leftward drifting of points, the adapted grids for low threshold angles ($A = 1^o$, 5^o) don't capture the discontinuity properly. For larger angles ($A = 10^o$, 15^o), while the discontinuity is not captured perfectly as the unadapted grid, the clustering of points show promising results. Hence, intermediate values of A (10^o, 15^o) are found to be suitable for adaptation again.

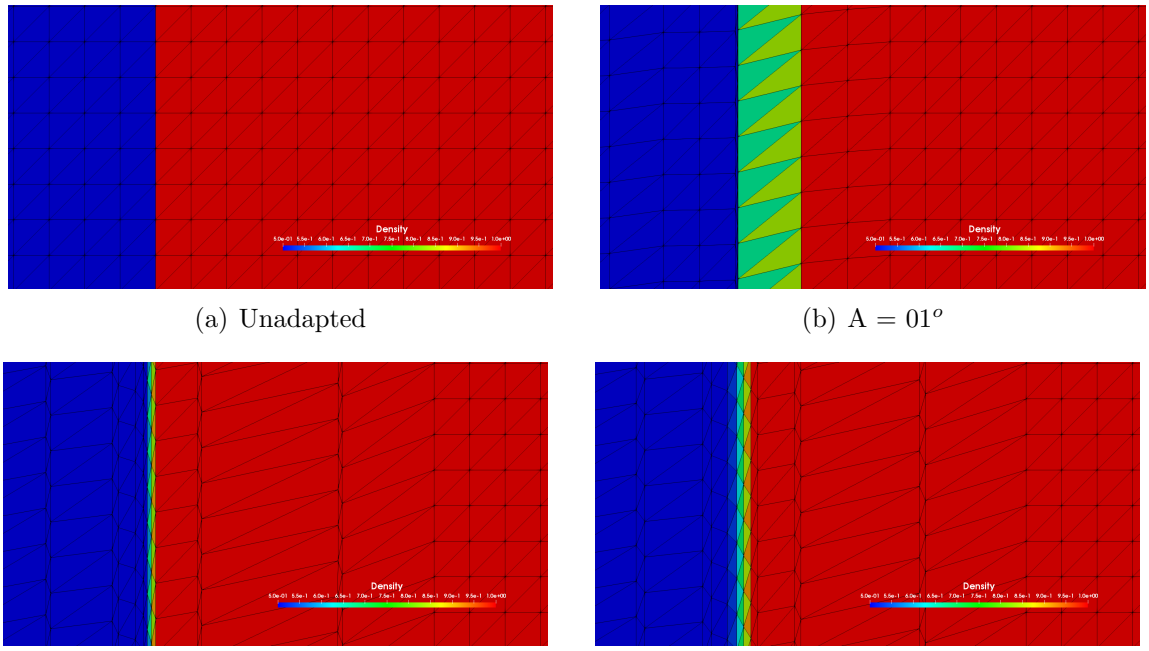
Furthermore, this example shows that if the initial, unadapted grid is optimized to represent the discontinuity, the immobilization of grid points on the boundary and the limitation to cell contraction causes the adapted grids to not be as accurate as one would hope.



(a) Unadapted (b) A = 01^o

(c) A = 10^o (d) A = 15^o

Figure 4: Shock Tube [Grid]



(a) Unadapted (b) A = 01^o

(c) A = 10^o (d) A = 15^o

Figure 5: Shock Tube [Density, Zoomed]

5 Conclusion

An algorithm that adapts triangular unstructured grids based on gradients of pressure and/or entropy was developed and tested using two different flows, viz., an oblique shock and a stationary contact wave discontinuity, for the use of the two adaptation switches separately. It was seen that the adaptation algorithm provided considerable adaptation in the regions of discontinuities when smaller angles were allowed, at the cost of grid quality. The adaptation at higher minimum angle criteria was sufficient to markedly reduce the shock thickness. The points were seen to drift upstream in the case of the stationary contact wave. This was assumed to be caused due to the immobilization of the boundary points and using only contraction switch. This drifting in turn caused the discontinuity to not be captured properly in certain cases. It was observed that freezing of boundary

points to preserve the domain shape also caused issues in adaptation due to interpolation.

REFERENCES

[1] Eiseman, P. R., Adaptive Grid Generation, *Computer Methods in Applied Mechanics and Engineering*, **64**, 321-376, 1987.

[2] Liou, M. and Steffen, Jr., C. J., A new flux splitting scheme. *Journal of Computational Physics*, **107**, 23-39, 1993.

[3] Vilsmeier, R. and Hänel, D., Adaptive Methods on Unstructured Grids for Euler and Navier Stokes Equations, *Computers Fluids*, **22**, 485-499, 1993.

[4] Venkatakrishnan, V., Convergence to steady state solutions of the Euler equations on unstructured grids with limiters, *Journal of Computational Physics*, **118**, 120-130, 1995.

[5] Löhner, R., Automatic Unstructured Grid Generators, *Finite Elements in Analysis and Design*, **25**, 111-134, 1997.

[6] Baines, M. J., Grid Adaptation via Node Movement, *Applied Numerical Mathematics*, **26**, 77-96, 1998.

[7] Jahangirian, A. and Shoraka, Y., Adaptive unstructured grid generation for engineering computation of aerodynamic flows, *Mathematics and Computers in Simulation*, **78**, pp. 627-644, 2008.

[8] Ghia, U., Bayyuk, S., Habchi, S., Roy, C., Shih, T., Conlisk, T., Hirsch, C. and Powers, J. M., The AIAA Code Verificaiton Project - Test cases for CFD Code Verification, *American Institute of Aeronautics and Astronautics Aerospace Sciences Meeting*, **48**, 2010.

[9] Sieger, D. and Botsh, M., Design, Implementation, and Evaluation of the Surface_Mesh Data Structure, *International Meshing Roundtable*, **20**, 533-550, 2011.

[10] Bahrainian, S. S. and Dezfuli, A. D., A Geometry based Adaptive Unstrucutred Grid Generation Algorithm for Complex Geological Media, *Computers and Geosciences*, **68**, 31-37, 2014.

[11] Subramanian, S., Compressible Viscous Flow Solver on Unstrucutred Grids using Finite Volume Method, *Indian Institute of Technology*, 2016.